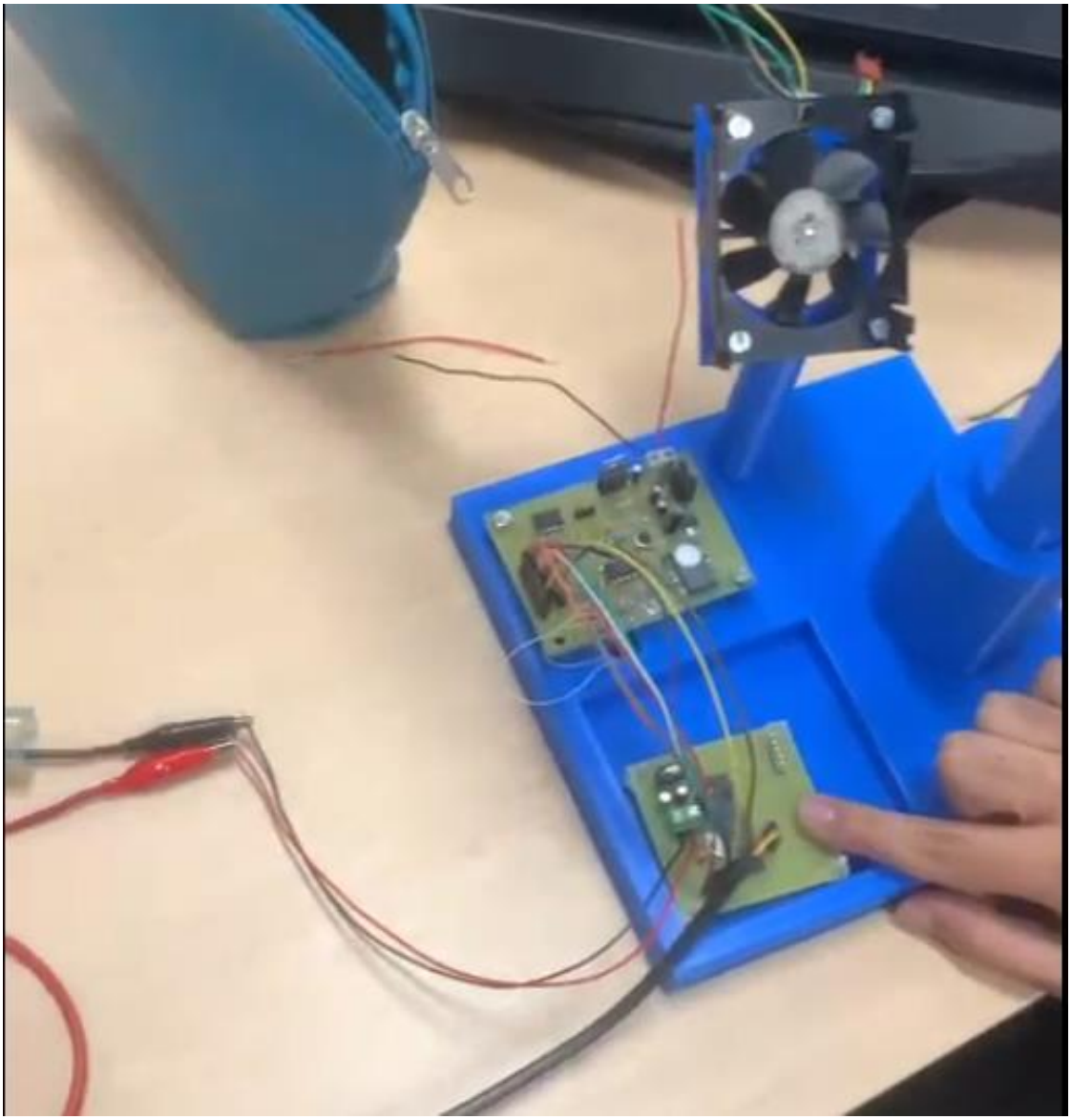
CONSTRUCCIÓN Y PRUEBAS

1-CIM41



A. Plan de Trabajo:

Como se puede observar de la entrega anterior, esta sección supone una extensión de los plazos previstos y desviaciones del documento anterior, suponiendo apenas un ligero cambio en el diagrama original.

1. **Análisis de especificación:** resumidos en el documento de especificación (el soporte y el CA)
2. **Diseño** (en espiral):
   - SW
   - HW
3. **Construcción**
4. **Pruebas**:

-Inclusión de trazas (a lo largo de todo el proyecto)

-Elaboración documentación (a lo largo de todo el proyecto).

Esta sección se centra, sin embargo, en las secciones 3 y 4 de la planificación, así como en los costes finales de todo el trabajo. En gris claro se denotan las secciones ya mostradas en el documento anterior y exclusivas de este.

1. Planificación temporal:

Semana del 10-10 (Semana 0):

1. **T-0**: Lectura y comprensión del enunciado (ya realizado).
   Tiempo: aprox. 1 hora por alumno.
2. **T-0.5** Elaboración básica del soporte de plataforma en github + creación de documentos necesarios vacíos (ya realizado).
   Tiempo: 25 minutos.
   NOTA: los códigos de las prácticas se están subiendo acá. No se contará el tiempo empleado para al elaboración de dicho código de forma detallada, pero es aproximadamente 3 horas por alumno (1 en casa + las 2 en clase por sesión).

Semana del 17-10 (Semana 1): inicio formal del proyecto.

1. **T-0.9**: Completar documento de requisitos en clase (2 horas).
2. **T-1**: Rellenar documentos de plan de Trabajo, Cuadro de tiempos invertidos de cada tarea (especificación y diseño) y estimación de costes.
   Tiempo: 2 horas 20 minutos (en grupo).
3. **T-2**: Diseño de SW a alto y bajo nivel:
   a. **T-2.1**: Elaboración Diagrama de Bloques (alto nivel). Tiempo: 25 minutos.
   b. **T-2.2**: Elaboración Diagrama de Estados (bajo nivel). Tiempo: 25 minutos.
4. **T-3.1**: Ampliación de documentación: especificación de los dos apartados anteriores en documento de diseño SW y HW, e inicio de **T-3.2:** elaboración de documento de trazas.
   Tiempo: 40 minutos.

Semanas 2 y 3:

- **T-3.3:** Elaboración y completitud de documentos y archivos de la fase de especificación y diseño (en clase y casa), incluye cuadro de desviación de tiempos invertidos de cada tarea (especificación y diseño).
- **T-4:** Inicio desarrollo del programa SW (principalmente en casa).

- **T-5.1:** Inicio de elaboración de descripción del protocolo de pruebas atendiendo al diseño.
- **T-6:** Diseño lógico y diseño físico (principalmente en clase).

Semanas 4 a 7:

- **T-3.4**: Desarrollo del documento de Incidencias HW y SW.
- **T-3.5**: Elaboración manual usuario (**OPCIONAL**).
- **T-3.6**: Cuadro de tiempos invertidos de cada tarea (construcción y pruebas) (**Semana 4**).
- **T-E1 (o T-7)**: ENTREGA DE PROYECTO (ESPECIFICACIÓN Y DISEÑO) y retoques finales (**Semana 6**). Pare entonces la **T-3.2:** Elaboración de Documento de Trazas debe haberse completado a nivel de diseño y comenzar a proceder con trazas a nivel de pruebas.
- **T-8:** Comienzo de construcción (y pequeña parte de pruebas). Inicio de inclusión de código de prácticas refinado y ajustado al proyecto (como muy tarde).
- **T-9:** Comienzo de elaboración del vídeo de construcción y pruebas.
- **T-3.7**: Actualización de desviación del tiempo invertido en cada tarea (construcción y pruebas) (hasta **semana 9**)

Semanas 8 a 10:

- La T-5.1 Descripción del protocolo de pruebas debe iniciarse como muy tarde en la **semana 8**.
- **T-10**: Pruebas y retoques a la construcción del circuito.
- El documento de Incidencias HW y SW, la desviación del tiempo invertido y el manual de usuario deben haberse terminado como muy tarde en la **semana 9**.
- **T-E2 (o T-11)**: ENTREGA DEL PROYECTO (CONSTRUCCIÓN Y PRUEBAS): **Semana 10**.

Fase/tarea del proyecto	S.0	S.1	S.2	S.3	S.4	S.5	S.6	S.7	S.8	S.9	S.10
T-0											
T-0.5											
T-0.9											
T-1											
T-2											
*** T-2.1**											
*** T-2.2**											
T-3											
*** T-3.1**											
*** T-3.2**											
*** T-3.3**											
*** T-3.4**											
*** T-3.5**					*	*	*	*	*	*	*
*** T-3.6**											
*** T-3.7**											
T-4											
T-5											
*** T-5.1**											
T-6											
T-E1											
T-8											

T-9											
T-10											
T-E2											

NOTA: Verde claro es para la entrega 1, verde oscuro para la entrega 2, zona tachada indica posible retraso permitido, en rojo oscuro se indican retrasos, en azul adelantos y en naranja se indican tareas que se detuvieron por no poderse terminar a tiempo.

2. Cuadro de tiempos invertidos, en cada tarea o labor, por trabajo individual de los participantes y por trabajo en grupo + Estimación de presupuestos

NOTA: en el tiempo (en minutos) en grupo asumimos el tiempo empleado en conjunto, que puede suponer un overlap con el conjunto de tiempos individuales; no el total de tiempos individuales por separado.

NOTA: Suponemos 40 € / hora.

Estimación T-3.3: un tiempo considerado a terminar documentación genérica + suponiendo que cada día o cada dos días cada uno individualmente actualiza sus datos de desviación y tarda unos 5 minutos el subirlo al github. 5 semanas * 7 días / 1.5 * 5 = 116 minutos adicionales

Estimación T-3.4 caso “peor”: 7 semanas, suponemos que 2 horas por semana rellenando y corrigiendo la lista de errores, 14 horas en total.

Estimación T-3.6: como para hacer la estimación de tiempos invertidos genérica hemos tardado 30 minutos, vamos a suponer que para la 3.6 se requiere un tiempo similar, al tener menos tareas por delante pero bastante más detalladas.

Estimación T-3.7: idéntico al 3.3, 3 semanas * 7 días / 1.5 * 5 = 70 minutos adicionales

Estimación T-6: como se hace principalmente en clase, tomamos 2 horas a la semana en grupo, más otra hora en casa individual en caso de incidencias. Como son 5 semanas, son 10 horas en clase y 5 individuales.

Estimación T-8: suponemos a igual parte en clase y en casa, por entonces los proyectos de otras asignaturas se habrán empezado a acumular, aunque no mucho; así que suponemos que tenemos de media 2 horas disponibles en casa individuales, 2 en clase en grupo, más otras 2 en grupo en casa para este proyecto en particular a la semana. Son 7 semanas, así que 14 individuales y 28 horas en grupo

Estimación T-9: suponiendo que cada día en clase (no nos podemos llevar el dispositivo a casa) hacemos unos vídeos de 5 minutos (en grupo) donde mostramos las pruebas, 1 clase por semana, en 7 semanas, y luego se requieren 20 minutos de post-producción (realizados por un compañero individualmente) para refinarlo en un vídeo de presentación, 35 en grupo y 20 para un particular.

Estimación T-10: de 2-3 horas semanales en clase (suponemos que en el peor caso el profesor reserva parte de la clase teórica para continuar las prácticas), suponemos 10 minutos de refinamiento del código en casa individual a la semana, como el T-10 son 3 semanas, 9 horas en grupo y 30 minutos en individual.

*Tiempo no medido en mayor detalle debido a ser opcional.

Fase/tarea del proyecto	Alejandro Serrano	Raúl Parla	Ismael Carrasco	Alejando Riñón	En grupo
T-0	60	60	60	60	5
T-0.5	25	3	0.5	0.5	5
T-0.9	15	5	5	5	120
T-1	140	20	20	0 (no pudo asistir)	150
T-2	50	50	50	24	50
*** T-2.1**	25	25	25	12	25
*** T-2.2**	25	25	25	12	25
T-3	1251	851	851	851	1140
*** T-3.1**	-	-	-	-	40
*** T-3.2**	15	15	15	15	40
*** T-3.3**	236	176	176	176	120
*** T-3.4**	840	560	560	560	840
*** T-3.5**	60*	*	*	*	*
*** T-3.6**	30	30	30	30	30
*** T-3.7**	70	70	70	70	70
T-4	320	320	480	480	480
T-5	40	40	40	40	40
*** T-5.1**	40	40	40	40	40
T-6	300	300	300	300	600
T-E1	-	-	-	-	1
T-8	840	840	840	840	1680
T-9	<-20->	<-20->	<-20->	<-20->	35
T-10	30	30	30	30	630
T-E2	-	-	-	-	1
COSTE ESTIMADO	2060.67 €	1692.67 €	1797.67 €	1767 €	2971.34€

A esto se le añade el coste de materiales adicionales no incluidos:

- Sensor I2C VEML7700 7.06 €
- Sensor LM35 3.35 €
- Sensor IAQ-core 22.56 €
- Sensor HIH-4000 11.25 €
- Tira de LEDes Adafruit SK9822 19.8 €
- PIC16F886 4.60 €
- Regulador LM78MOS-TO-220 0.75 €

Lo que da un total de costes Individuales + Coste de Grupo + Coste componentes = **10358.72 €**

3. Desviación del tiempo invertido con respecto al planificado.

Fase/tarea del proyecto	Alejandro Serrano	Raúl Parla	Ismael Carrasco	Alejando Riñón	En grupo
T-0	60	60	60	60	5
T-0.5	25	3	0.5	0.5	5
T-0.9	15	5	5	5	120
T-1	141	20	20	0 (no pudo asistir)	150

T-2	669 [2]	270	178	210	367.5
*** T-2.1**	180 [2]	65	5	5	137
*** T-2.2**	489 [4]	205 [4]	173 [4]	205 [4]	230.5 [4]
T-3	221	104	82	164	89
*** T-3.1**	12	14	12	14	14
*** T-3.2**	6	20	0	20	5
*** T-3.3**	115 [5]	70	70	70	70
*** T-3.4**	20	0	0	0	0
*** T-3.5**	46*	0*	0*	60*	60*
*** T-3.6**	11 [8]	0	0	0	0
*** T-3.7**	11 [8]	0	0	0	0
T-4	102	2	300	262	102
T-5	1	37.5	0	37.5	37.5
*** T-5.1**	1	37.5	0	37.5	37.5
T-6	390	240	240	240	240
T-E1	13	-	-	-	0
T-8	580	300	360	300	300
T-9	80 [11]	85 [11]	60 [11]	60 [11]	60 [11]
T-10	360	360	360	360	360
T-E2	-	-	-	-	15
COSTE FINAL	1771.33 €	991 €	1110.33 €	1132.66 €	1234 €

Lo que supone un coste total de **6308.39 €**, notablemente menor de lo previsto, indicando también que no pudimos completarlo a tiempo.

Abajo queda indicado el diagrama de Gantt de desviaciones cometidas.

Fase/tarea del proyecto	S.0	S.1	S.2	S.3	S.4	S.5	S.6	S.7	S.8	S.9	S.10
T-0											
T-0.5											
T-0.9											
T-1											
T-2			[1]								
*** T-2.1**			[1]								
*** T-2.2**			[1]	[4]	[4][6]						
T-3											
*** T-3.1**			[1]								
*** T-3.2**											
*** T-3.3**											
*** T-3.4**			[3]	[3]							
*** T-3.5**					*	*	*	*	*		
*** T-3.6**											
*** T-3.7**								[10]	[10]	[10]	[10]
T-4											
T-5											
*** T-5.1**											
T-6											

T-E1											
T-8					[7]						
T-9											
T-10											
T-E2											

[1] La causa del retraso se debió a que para consolidar estas fases de forma consensuada se requiere de gran parte del grupo (sino su totalidad) para planificar, los planes de cada miembro del equipo supusieron que solo se pudiera quedar un domingo por la mañana por 2 horas. Tras ello, un contratiempo en la vida real supuso que gran parte del equipo se tuviera que marchar a las 2 horas de contactar, y como parte de los borradores de dichas fases ya se habían realizado, se decidió quedar al día siguiente (lunes) para completarlo.

El lunes debido a complicaciones por falta de familiaridad con el uso del Papyrus respecto a otras herramientas como StarUML y a ser una reunión parcial del equipo (Ismael Carrasco sufrió una urgencia médica y no pudo asistir); supuso un consumo de tiempo notablemente mayor del esperado que supuso un retraso adicional para la realización del documento HW y SW y del documento de trazas y pruebas al día siguiente (Martes).

[2] El tiempo extra se debió al tiempo necesario para familiarizarse con la herramienta Papyrus.

[3] La falta de familiaridad con el SW se tuvo que trazar en las incidencias SW.

[4] El enunciado sobre el vídeo de diseño detallado SW nos hizo pensar que a lo mejor necesitábamos una descripción mucho más detallada con un diagrama de estados adicional para la rutina de tratamiento de interrupción, que al menos mencionase qué cosas se ejecutaban en el programa principal y cuáles en la rutina de tratamiento de interrupción. Esto finalmente supuso además una ligera definición de funciones auxiliares.

[5] debido a [4], una parte de lo que sería trabajo para hacer del 3.3 se ha incluido como parte del 2.2, aunque en realidad hay al menos entre 120 y 180 minutos de 2.2 que se podrían asignar a 3.3, pues durante la semana.

[6] debido a ciertas complicaciones a lo hora de concertar cita (y la falta del enunciado de entrega del vídeo de diseño detallado), se retrasó la reunión del fin de semana de S.3 del domingo al lunes de la S.4, pero aún adelantado con respecto a las clases.

[7] el retraso de [6] supuso que parte de lo que iríamos preparando para la sesión del fin de semana de S.4 se fuese adelantando un poco, decidimos emplear parte de ese tiempo en comenzar a codificar el código final a la vez que hacíamos el diagrama de estados. Es por eso por lo que el código de las funciones del diagrama de Estados adicional ya está mayoritariamente en código o pseudocódigo y no solo comentarios.

[8] supusieron un tiempo menor porque al final decidimos fusionar ambas desviaciones y planificaciones en un único documento que luego editaríamos más tarde.

[9] debido a retrasos en la semana 6 (21 de Noviembre 2022) por la imposibilidad de realizar pruebas por no poder soldarlo todo y porque el profesor enfermó de gripe, no se pudo hacer lo esperado, solo la parte más teórica que era completar la elaboración de pruebas según el diseño físico.

[10] Fuimos muy optimistas y no tuvimos en cuenta que este diagrama requeriría ser actualizado de manera tan constante hasta la semana 10, aunque ya previmos que hasta la semana 9 se iría realizando.

[11] Originariamente pensamos que debía realizarse un vídeo adicional con grabaciones del funcionamiento del SMA-LAMP, al final no hubo tal vídeo pero el esfuerzo se registra aquí.

B. Construcción lámpara SMA

Acá se deja el link de nuestro proyecto de Github donde se encuentran diversos documentos, así como el código de SMA-LAMP, SMA.COMP y diagramas empleados durante el proyecto; con la subcarpeta *Construcción y Pruebas* centrada específicamente en esta sección:

https://github.com/tardisfromtornspace/ProyectoEquipoLampara

C. Pruebas unitarias individuales (para cada tipo de actuador y sensor) y de aplicación.

Para las pruebas hemos empleado caja negra y caja gris. Para algunas de las pruebas realizadas hemos añadido vídeos y algunas imágenes (ambos accesibles desde el github).

En caso de que se detecten errores, se procedió a revisar el componente HW y SW relacionado (ver trazas) y se realizó debugging para la aplicación involucrada (SMA-LAMP, SMA-COMP o ambos) y se realizará la prueba de nuevo. Cuando se halla la solución se indica en este documento. Si la incidencia es suficientemente grande se indica en la sección de incidencias.

No fueron contadas las pruebas de Estructura Mecánica (ya que se han realizado por los de Industriales) ni del CA (salvo pruebas de interfaces, ya se diseñó por conocidos del profesor). Las pruebas entre SMA_LAMP y SMA_COMP que son comunes solo se definen en ambos lados por claridad a la hora de realizar las pruebas, centrándose en las diferencias entre ambas partes.

Para los I2C, si hubiera dado tiempo, habríamos comprobado que no hubiese problemas con las resistencias de pull-up de 4k7.

Acá quedan las trazas:

Requisito	**Componente HW**	**Componente SW**
ENC-10	SMA-LAMP, ordenador Linux	SMA-LAMP, SMA-COMP
ENC-20	Sensor Temperatura, Humedad, Ruido, CO2 y Luminosidad, SMA-LAMP, USART y ordenador Linux.	Sensor Temperatura, Humedad, Ruido, CO2 y Luminosidad; SMA-LAMP y SMA-COMP
ENC-30	EEPROM/Flash	Memoria
AP-10	SMA-LAMP (registros/mem volátil)	SMA-LAMP
AP-20	Ordenador Linux/USART	SMA-COMP
AP-30	Ordenador Linux/USART	SMA-COMP
MO-10	CAD	Sensor Ruido
MO-20	Timer	Sensor Ruido
MO-30	Timer	Sensor Temperatura, Humedad, Ruido, CO2 y Luminosidad
MO-40	I2C (C02 y Luminosidad), ADC (el resto)	Sensor Temperatura, Humedad, Ruido, CO2 y Luminosidad
AC-10	I2C, USART	Memoria
AC-20	PWM, USART	PWMVentilador

Y acá el conjunto de pruebas (indicando aquellas que no pudimos realizar a tiempo):

Dispositivo	Prueba	Resultado (Pasa/No Pasa (Éxito))
Estructura mecánica (ya construido)	N/A	N/A
Componente CA (ya construido)	(interfaces ya existentes, realizado por compañeros de Norberto)	N/A
Comprobación de malas soldaduras:	1º Al soldar se comprueba que se puede conectar el componente a la pista (probar con multímetro que la zona más distante de la pista correcta sí conecta) 2º Al soldar no se comunican pistas que no deberían estar conectados (ver con multímetro que las pistas cercanas no se conectan con el componente). Esto incluye asegurarse que los tornillos no se cargan.	Pasa (No hay problemas). Pasa (No hay problemas).
Comprobación de carga (PICKIT-3)	Se puede cargar un programa a la placa con éxito (en este caso un programa de prueba de UART empleado anteriormente)	Pasa (ver vídeo TestUART).
Sensor de luminosidad VEML7700	Comprobar si sube la luminosidad al incrementar valor (poner lámpara o taparlo con un dedo) y que no interfiera con los resultados del sensor de CO2. Esto se hace con una versión simplificada del programa final con solo los I2C, primero solo para este sensor y luego para ambos sensores.	-----NO DIO TIEMPO-----
Sensor de CO2 iAQ-Core	Comprobar si cambian valores de C02 en diferentes lugares de la universidad (o mejor, al echar vaho), y que el sensor de luminosidad no interfiere. Esto se hace con una versión simplificada del programa final con solo los I2C, primero solo para este	----- NO DIO TIEMPO -----

	sensor y luego para ambos sensores.	
Sensor de humedad relativa HIH4000	Comprobar si el valor de la humedad varia (al echar vaho). Utilizaremos una versión del programa de adc para ello.	El sensor funciona.
Sensor de temperatura LM35-LP	Comprobar si la temperatura incrementa adecuadamente (poner mano caliente o algún objeto frío). Utilizaremos una versión del programa de adc para ello.	El sensor funciona.
1. Nivel de ruido.	Comprobamos la señal analógica de ruido (haremos ruidos con objetos o la voz para probarlo) y que no interfiera con la de humedad relativa ni temperatura. Utilizaremos una versión del programa de adc para ello y luego una versión del programa final con solo lo de ADC, necesita de UART.	Adc: Pasa. Nota: posteriormente nos dejó de funcionar por motivos desconocidos, tras analizarlo por varias horas y no encontrar la causa SW se postergó a otras pruebas.
2. SCL.	El bus I2C conecta el sensor de humedad y el de CO2 y funciona a 100kHz.	----- NO DIO TIEMPO -----
3. SDA.	El bus I2C conecta el sensor de humedad y el de CO2 y funciona a 100kHz	----- NO DIO TIEMPO -----
4. Humedad relativa.	Comprobamos la señal analógica y que no interfiera con temperatura ni ruido. Utilizaremos una versión del programa de adc para ello (con) y luego una versión del programa final con solo lo de ADC.	El sistema lee, pero los valores no se relacionan con nada. Nota: posteriormente nos dejó de funcionar por motivos desconocidos, tras analizarlo por varias horas y no encontrar la causa SW se postergó a otras pruebas.
5. Temperatura	Comprobamos la señal analógica y que no interfiera con ruido ni humedad. Utilizaremos una versión del programa de adc para ello y luego una versión del programa final con solo lo de ADC.	El sistema lee, pero los valores no se relacionan con nada. Nota: posteriormente nos dejó de funcionar por motivos desconocidos, tras analizarlo por varias horas y no encontrar la causa SW se postergó a otras pruebas.
6. PWM ventilador	Comprobar si aumenta o disminuye la velocidad del ventilador (CCP1).	Pasa (ver vídeo TestPWM)

	Utilizaremos primero el programa del PWM de prácticas y luego una versión del programa final con el PWM.	*Evento: hay que darle un pequeño golpecito para que arranque a veces -> Resistencia estática muy grande o polvo en el motor. El profesor dijo que eso no es de nuestra competencia, y además luego ya se resolvió.
7. SCK. (Línea de control de un bus SPI. Emisor luz lámpara) Sugerencia, como el pin de I2C y SPI son el mismo, simulamos SPI por SW.	Comprobar nivel de luminosidad de la lampara al recibir la señal del bus SPI.	Transmitir un código de ejemplo según el datasheet con máxima iluminancia (31) no encendía ningún LED. Ejecutamos debug para comprobar por qué, descubrimos un error en el for que no hacía pasar ni una sola instrucción de estas, tras resolverlo ya sí se podían encender pero por algún motivo solo el primer led de la tira y débilmente (Fig. 1). Procedíamos a comprobar por debug de nuevo cuando se acabó el tiempo. -- NO SE COMPLETÓ POR FALTA DE TIEMPO --
8. SDO. (Línea de salida de datos hacia un bus SPI. Emisor luz lámpara). Sugerencia, como el pin de I2C y SPI son el mismo, simulamos SPI por SW.	Comprobación niveles de luz por la lampara.	Ídem del anterior -- NO SE COMPLETÓ POR FALTA DE TIEMPO --
USART	Comprobar que se pueden enviar y recibir datos (se usa el mismo programa de la usart que el de las prácticas)	Pasa (ver vídeo TestUART).
Componente SMA		
SMA-LAMP	Comprobamos que es capaz de alterar la velocidad del ventilador mediante la señal PWM, que varían los colores de la tira de leds (bus SPI) y que guarda los estados en la memoria (EEPROM/Flash) (según las trazas descritas anteriormente) y que es capaz de enviar datos por la USART	Pasa, salvo lo mencionado del SPI.

	ENC-10: SMA-LAMP se arranca correctamente	Hubo problemas con SPI (ver explicación arriba), aparte de eso todos los otros sistemas se iniciaban correctamente.
	ENC-20: se envían los datos de los sensores disponibles y los no listos a SMA-COMP	----- NO DIO TIEMPO -----
	ENC-30: al modificar la configuración de LEDes y ventilador, la siguiente vez que se arranca, sale esa misma configuración y no la de por defecto	Funciona (ver Fig. 2).
	AP-10: apagar SMA-COMP directamente no debe crashear SMA-LAMP	----- NO DIO TIEMPO -----
	AP-20: SMA-LAMP envía un pong (señal "B") a los ping ("b") de SMA_COMP por USART	----- NO DIO TIEMPO -----
	AP-30: se puede dar la orden de apagar SMA-LAMP por USART (señal "e"), debe salir de su bucle.	----- NO DIO TIEMPO -----
	MO-10: SMA-LAMP traduce todas las señales de ruido a las categorías Bajo [0, 400], intermedio (400, 900] y alto (900, infinito)	----- NO DIO TIEMPO A VERIFICAR EN EL MODELO REAL -----
	MO-20: el ruido debe monitorizarse cada 10 ms sin interferencias de los otros ADC (debe cumplirse el sample and hold) + solo debe actualizarse cada segundo dando la categoría de ruido más alta por segundo	La simulación indicaba que se ejecutaría en ese período de tiempo. No dio tiempo a verificar en el modelo real.
	MO-30: humedad, temperatura, CO2 y luminosidad deben leerse cada 5 segundos y mandar cada 5 segundos valores actualizados.	La simulación indicaba que se ejecutaría en ese período de tiempo. No dio tiempo a verificar en el modelo real.
	MO-40: en el printf el SMA-LAMP envía la cadena con los datos indicados en la categorías a/m/b (ruido), ºC (temperatura), % (humedad relativa), ppm (para CO2) y lx (para luminosidad)	Se comprobó que efectivamente puede enviarlos por minicom, pero no dio tiempo a comprobar en SMA-COMP puesto que aún no se había instalado en el computador.

		----- NO DIO TIEMPO A VERIFICAR EN EL MODELO REAL -----
	AC-10: al recibir por USART la señal de cambio de LED (señal “d”) los siguientes 4 caracteres recibidos se entenderán como el nivel de R, G, B y luminosidad, que se enviarán a los LEDes y por ENC-30, a la memoria.	----- NO DIO TIEMPO -----
	AC-20: al recibir por USART la señal de cambio de PWM (señal “c”) el siguiente carácter recibido se entenderá como el porcentaje del duty cycle del ventilador, que se enviarán al CCP1 y por ENC-30, a la memoria.	----- NO DIO TIEMPO -----
SMA-COMP	Comprobamos que la interfaz programada funciona correctamente, realizando las operaciones oportunas, siendo capaz de leer los datos de los sensores, codificar los estados de los leds y de la señal PWM y apague la lampara cuando sea necesario (según las trazas descritas anteriormente).	La interfaz programada funciona, pero no dio tiempo a montar en el sistema de pruebas.
	ENC-20: se reciben los datos de los sensores disponibles y los no listos de la SMA-LAMP y se tratan para ser legibles con el formato empleado (p.ej. en ventanas)	----- NO DIO TIEMPO -----
	AP-20: Apagar SMA-LAMP supone que SMA-COMP de un mensaje de que se ha perdido la conexión y deberá cerrarse SMA-COMP. Esto se hace por comunicación por USART donde SMA_COMP manda a la lámpara un ping periódico por USART (señal “b”), y si no responde en un tiempo determinado se apaga. Comprobar que no ocurren falsos positivos.	----- NO DIO TIEMPO -----

	AP-30: SMA-COMP debe tener un botón para mandar una orden de apagado al SMA-LAMP que envíe por la USART el comando “e”)	Efectivamente tiene el botón, pero no se puedo probar su funcionamiento por las causas mencionadas anteriormente.
	MO-10: las señales interpretables como “bajo/medio/alto” 1, 2, 3 de SMA-LAMP deben traducirse a “alto”, “medio”, “bajo” para el usuario final	Esta prueba se alteró para explicarse a nivel del manual de usuario (ver sección “MANUAL DE USUARIO”)
	MO-20: el SMA-COMP refleja las actualizaciones de ruido cada segundo.	----- NO DIO TIEMPO -----
	MO-30: humedad, temperatura, CO2 y luminosidad deben recibirse de la USART y actualizarse por pantalla cada 5 segundos	----- NO DIO TIEMPO -----
	MO-40: los datos recibidos desde SMA-LAMP debo recibirlos en categorías a/m/b (ruido), ºC (temperatura), % (humedad relativa), ppm (para CO2) y lx (para luminosidad)	Comprobado, el SMA-LAMP al final envía los datos directamente sin que SMA-COMP necesite traducirlos a unidades específicas.
	AC-10: la aplicación del ordenador debe tener un botón que permita enviar una cadena del tipo “d0000” dónde d es el indicador del comando LED y 0000 son 4 caracteres que indican el R, G, B y luminosidad, por supuesto éstos se hacen legible al usuario y a prueba de errores (no menos de 0, no más de 255 (o 31 para luminosidad)).	Efectivamente tiene el botón, pero no se puedo probar su funcionamiento por las causas mencionadas anteriormente.
	AC-20: la aplicación del ordenador debe tener un botón que permita enviar una cadena del tipo “c0” dónde c es el indicador del comando PWM y 0 es el carácter que indica porcentaje del duty cycle, por supuesto legible al usuario y a prueba de errores (no menos de 0, no más de 100).	Efectivamente tiene el botón, pero no se puedo probar su funcionamiento por las causas mencionadas anteriormente.

Ejemplo de caso de prueba modelo: encendemos el dispositivo SMA-LAMP y vemos que el ventilador está apagado y la luz es blanca, leemos sensores por SMA-COMP, tras unos segundos escribimos PWM, tras eso leemos de nuevo y comprobamos PWM y LEDes. Unos segundos después escribimos PWM, leemos datos y cerramos el SMA-COMP. Volvemos a abrir SMA-COMP y vemos que sigue funcionando. Ahora ordenamos apagar la SMA-LAMP por comando de la SMA-COMP y debemos ver como SMA-COMP nos dice que ya no recibe ping de la LAMP y se para. Ahora iniciamos de nuevo y vemos que el SMA-LAMP debe tener la configuración de PWM y LED pre-apagado.

- Resultados que pudimos obtener de este caso de prueba: desgraciadamente solo la parte inicial, que el ventilador está apagado; y que se guarda la última configuración.

Figura 1. Resultado problemático de ledes SPI

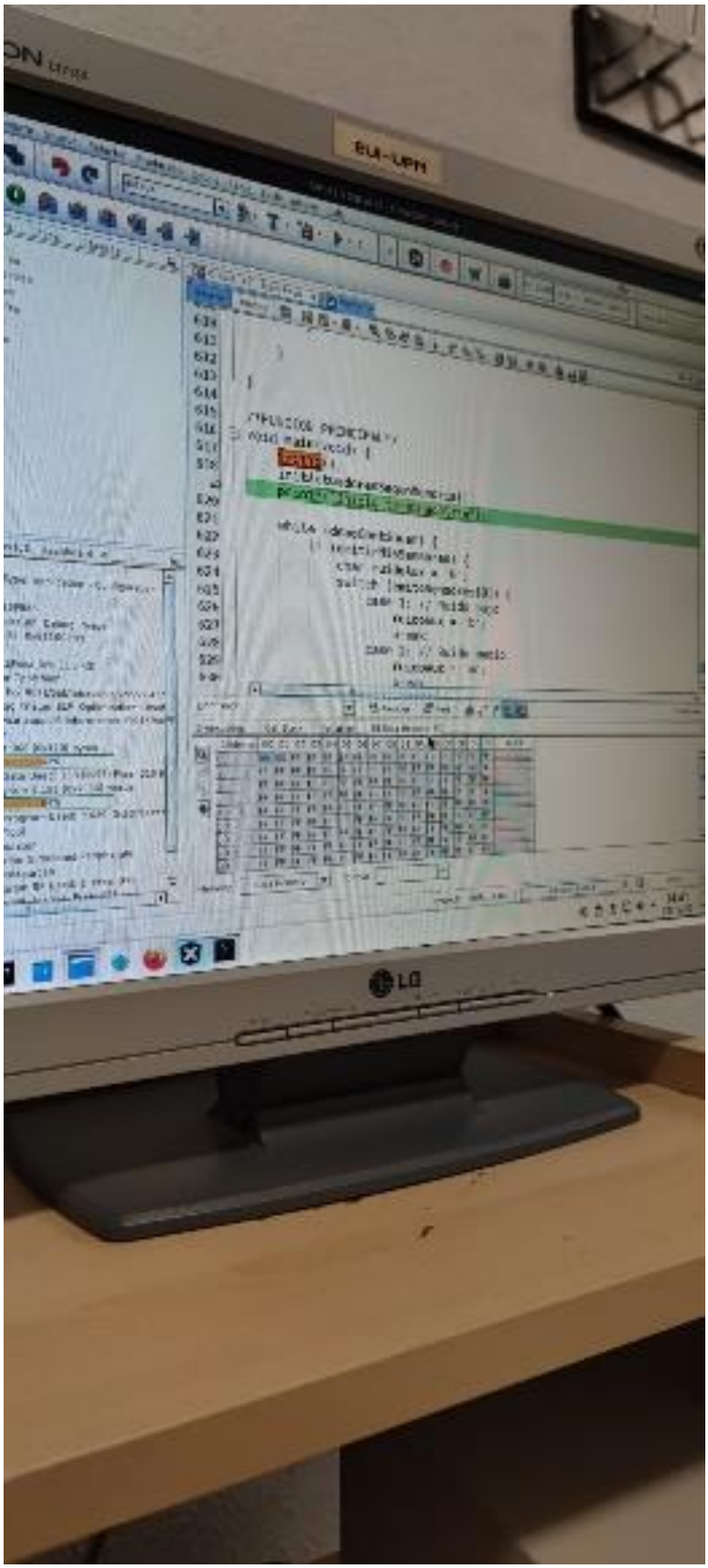


Figura 2. Almacenamiento en memoria de valores guardados. El primer bit indica si se ha escrito antes en la memoria, el segundo bit el porcentaje de PWM enviado al motor (0% por defecto), los siguientes 4 valores indican el R, G, B (de 0 a 255, 0xFFFF por defecto) y luminosidad (de 0 a 31, 0x1ff)

D. Incidencias en la construcción del componente SW del SE.

-**Incidencia SW nº 1**: Papyrus no permite escribir diagramas de bloques tan fácilmente como StarUML, es muy rígido.

* Solución: más práctica para familiarizarse con el SW.

-Incidencia SW nº 2: descubrir que a nivel de HW parte de nuestro programa no era válido pues TXREG no admite más de un char por vez.

* Solución: crear condiciones que actúen como buffer y almacenen los resultados que SMA_LAMP recibe.

-Incidencia SW nº 3: en realidad por motivo desconocido la lectura de valores del ejemplo adc-1 aplicado a nuestra placa no recibe.

* Procedimiento: se evaluó por debugging y en real. Originarimante se sospechó causa HW ya que el programa utilizado era idéntico al de la entrega del adc-1.c para las mismas entradas y ese es plenamente funcional, pero no se detectaron pistas ni conexiones incorrectas y el propio profesor comprobó por osciloscopio que aunque por algún motivo la lectura que llegaba a nuestra placa era más débil de lo usual, el sensor en sí funcionaba. Tras ser incapaces de detectar la causa del problema y ver que íbamos cortos de tiempo, decidimos postergarlo hasta poder comprobar los otros módulos primero. Finalmente no dio tiempo a resolverlo.

-Incidencia SW nº 4: problema con la EEPROM, nos lleva a bucle infinito.

* Solución: ejecutamos el debuger para detectar el problema, descubrimos que no se puede usar la interrupción que queríamos para establecer “!continuoEscribiendo” en la espera de la línea 306. Utilizamos “EECON1bits.WR == 1” como alternativa y eso lo resolvió.

E. Incidencias en la construcción del componente HW del SE.

Incidencia HW1: broca mal posicionada, causaba vibraciones.

- Solución: reposicionar y reajustar, comprobar que el primer agujero no se haya dañado (se comprobó y no estaba suficientemente dañado como para requerir medidas adicionales).

Incidencia HW2: nos pasamos al realizar agujeros de sujeción esquinas y se rompe un extremo.

- Solución: al añadir las patas de los extremos no causa problema.

Incidencia HW3: condensadores de tamaño entre pines (pitch) de casi 5mm en vez de 2.5mm.

- Solución: usar un objeto para doblar los pines hacia adentro a la posición adecuada.

F. Lámpara SE - Manual de usuario

Versión 22-12-2022

Introducción

Bienvenidos al manual de usuario de la lámpara.

Este manual pretende aclarar la correcta utilización del sistema SMA para un estudio saludable, tanto la sección de Lámpara (SMA-LAMP) como la de Interfaz gráfica de usuario (SMA-COMP). También cabe aclarar que el sistema está aún en fase de desarrollo, así que, si detectan errores, bugs o simplemente quieren añadir comentarios y sugerencias, comuníquenlas al jefe de proyecto alejandro.serranol@alumnos.upm.es. Esto también significa que este manual puede sufrir modificaciones sin previo aviso, así que por favor consulte el manual con frecuencia si actualiza el software.

SMA-LAMP

Lo que se entiende como la lámpara en sí, y es la que realmente monitoriza el entorno y controla los LEDes y el ventilador. El sistema puede soportar una alimentación de 12V y 0.6 amperios. Actualmente no existe un botón en la placa que permita el encendido y apagado manual del sistema, sino que se utiliza un interruptor externo, y en su defecto se utiliza la técnica plug-in/out (conecta o desconecta de la corriente).

Al arrancar la primera vez el sistema encenderá los LEDes a blanco brillante (lo que en línea de comandos sería “255 255 255 31” por es bus SPI) y con el ventilador apagado (intensidad “0” en el cable de CCP1) que posteriormente pueden personalizarse de forma limitada mediante la interfaz gráfica de usuario de SMA-COMP. Las siguientes veces que arranque seguirá con su configuración pre-apagado, manteniéndose los valores de ventilador y LEDes previos, gracias a que se guardan en la memoria Flash del sistema.

Posteriormente el SMA-LAMP procede a leer los sensores y comunicar los valores al SMA-COMP mediante la UART, que es el protocolo que se usa para comunicarse bidireccionalmente a la hora de enviar y recibir datos y comandos (para más detalle en los comandos, vea la sección SMA-COMP). Hay algunos sensores (como el de dióxido de carbono) que tardan su tiempo en arrancar y solo arrojan valores basura mientras tanto. Dichos sensores (así como aquellos que no pudieran ser inicializados) arrojan un valor -1 hasta entonces. Aunque la lectura de sensores se realiza a diferentes frecuencias (por ejemplo, el ruido a 10 ms, mientras que el resto cada 5 segundos) y algunas utilizan conexión por I2C en vez del envío de dichos valores se realiza cada 5 segundos (cada segundo en el caso del ruido, que además no envía su valor, sino una categoría de intensidad más alta recibida en ese período) utilizando la función printf que envía la cadena:

“Ruido = categoria %d, humedad = %d %%, temperatura = %d %c C, CO2 = %d ppm, Luminosidad = %d lx”

La categoría de ruido puede ser “b” (bajo, hasta un valor de 400), “m” (medio, hasta un nivel de 900), “a” (alto, por encima de 900) y “E” (Error); mientras que el resto de valores son numéricos y se expresan en las unidades indicadas.

Por situaciones de debug, SMA-LAMP puede también enviar al SMA-COMP diversos printf que ayudan al usuario y al programador detectar errores en la ejecución en áreas sin mucho tráfico (no se incluyen printf en las áreas con interrupciones).

SMA-COMP

El sistema SMA-LAMP responde a una serie de comandos pasados por la interfaz gráfica de usuario, que además comunica a éste los niveles de ruido, humedad, temperatura, luminosidad y CO2 detectados para que a juicio del estudiante se realicen medidas correctivas (ver sección MEDIDAS CORRECTIVAS). Esta interfaz de usuario tiene como requisitos un sistema operativo UNIX para poder arrancar, así como conectar el SMA-LAMP al puerto USB adecuado (tty/usb0) para permitir la comunicación Lámpara-usuario. Además, el SMA-COMP comprueba si SMA-LAMP está siquiera conectado antes de proceder mediante la función "ping" (en la que el SMA-COMP envía una señal "b" y SMA-LAMP responde con señal "B"), en caso contrario se apaga y no acepta más comandos hasta que se reinicie (o, puesto en términos del usuario, cerrar y reabrir el programa SMA-COMP).

El sistema por defecto tiene botones de encendido y apagado de la lámpara que actúan al ser presionados. Además, tiene botones de valores para la señal PWM y ledes. Cuando haces clic en dichos botones se abre un panel que te permite introducir texto, en este caso, introduce un valor de 0 a 100 para el porcentaje de frecuencia de rotación del motor; y una cadena de números del 0 al 255 para el color de los LEDes en codificación RGB ("Red, Green, Blue", niveles de rojo, verde y azul, respectivamente, siendo 0 total ausencia y 255 el máximo valor de cada color) y del 0 al 31 para su intensidad. Este color será asignado a todos los LEDes de la lámpara.

La interfaz la tenemos programada en Python y encontramos diferentes funciones. Cabe destacar que al arrancar esta encontramos diferentes botones que explicamos a continuación:

- *Encender*: se encarga de encender las luces de la lampara.
- *Apagar*: se encarga de apagar las luces de la lampara.
- *Enviar intensidad*: se encarga de enviar un valor entre 0-31 que hace referencia a la intensidad de la luz.
- *Apagar_V*: se encarga de apagar el ventilador.
- *Encender_V*: se encarga de encender el ventilador.
- *Enviar intensidad V*: se encarga de enviar la intensidad con la que queremos que el ventilador funcione.
- *Valor R*: se encarga de enviar un valor entre 0-255 para determinar la intensidad del color rojo de la ristra de leds.
- *Valor G*: se encarga de enviar un valor entre 0-255 para determinar la intensidad del color verde de la ristra de leds.
- *Valor B*: se encarga de enviar un valor entre 0-255 para determinar la intensidad del color azul de la ristra de leds.
- *Estado*: se encarga de enviar un carácter (S/N). Si el carácter enviado es S devolverá el estado de los sensores encontrados en nuestro proyecto. Mientras que si se envía el carácter N no devolverá nada respecto al estado de los sensores.

MEDIDAS CORRECTIVAS

Si el CO2 está por encima de 700 ppm se recomiendan abrir las ventanas. Por encima de 1000 es obligatorio para prevenir somnolencia y dolores de cabeza. Si es superior a 4000 salga de la estancia INMEDIATAMENTE puesto que cantidades superiores pueden causar daños cerebrales (detectoresco2, 2022).

Los niveles de humedad ambiental recomendados se sitúan entre el 40% y el 60%. Niveles inferiores suponen sequedad en ojos, mucosas y piel que pueden facilitar infecciones y reducen el rendimiento, mientras que niveles superiores favorecen la aparición de problemas de humedad tales como el moho y ácaros que pueden causar reacciones alérgicas (HUMESTOP, 2021). Si abrir una ventana no ayuda, se recomienda comprar humidificadores o deshumidificadores según el caso.

La temperatura óptima según la legislación se sitúa entre los 16 y 26 grados centígrados de temperatura, aunque esto obviamente varía según el uso de ropa, por ejemplo, con ropa de invierno el límite superior debería rebajarse (Aradas, 2022). Niveles superiores facilitan el sobrecalentamiento y la sudoración, la pérdida de eficiencia, la deshidratación y el desfallecimiento; mientras que niveles inferiores tampoco ayudan en la concentración y ralentizan a la persona, pudiendo sufrir hipotermia si la temperatura es suficientemente baja.

Para niveles de ruido se recomienda cambiar de ambiente si permanece en categoría “a” por más de 5 minutos, ya que una exposición continua a niveles elevados no solo suele dificultar la concentración, pero pueden provocar problemas auditivos permanentes. Para niveles de ruido medios se recomienda el uso de cascos u otras medidas que reduzcan el ruido.

Busque el ambiente luminoso que más le convenga, pero a ser posible bien iluminado. Niveles demasiado bajos de luminosidad pueden favorecer la aparición de problemas visuales como miopía y vista cansada. Una vez encuentre la iluminación óptima para usted, recuerde el valor recibido por el sensor como referencia para estudios futuros.

BIBLIOGRAFÍA

Aradas, A. (6 de 7 de 2022). *www.cuestioneslaborales.es*. Obtenido de www.cuestioneslaborales.es: https://www.cuestioneslaborales.es/temperatura-adecuada-en-el-centro-de-trabajo/

detectoresco2. (5 de 12 de 2022). *www.detectoresco2.com*. Obtenido de www.detectoresco2.com: https://www.detectoresco2.com/niveles-seguros-y-recomendados-de-co2/

HUMESTOP. (4 de 6 de 2021). *humestop.net*. Obtenido de humestop.net: https://humestop.net/niveles-optimos-humedad-ambiente