ESPECIFICACIÓN Y DISEÑO

1-CIM41

A. Especificación:

NOTA: el soporte y el CA ya están construidos, así que los pasos de Análisis, Diseño y Construcción de ambos no es aplicable a este plan de trabajo. Solo las pruebas de integración con estos.

1. **Análisis de especificación:** resumidos en el documento de especificación (el soporte y el CA
2. **Diseño** (en espiral):
   - SW
   - HW (dependiente de lo dado e clase, de la mano)
3. **Construcción**
4. **Pruebas**:

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

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

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 a la hora de realizar; área verde indica.

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 en las actividades de diseño con respecto al planificado en el periodo de especificación.

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	152	104	82	104	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**	6	0	0	0	0
*** T-3.5**	1*	0*	0*	0*	0*
*** 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	75	60	60	60	60
T-10	0	0	0	0	0
T-E2	-	-	-	-	0
COSTE FINAL	WIP	WIP	WIP	WIP	WIP

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**											
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.

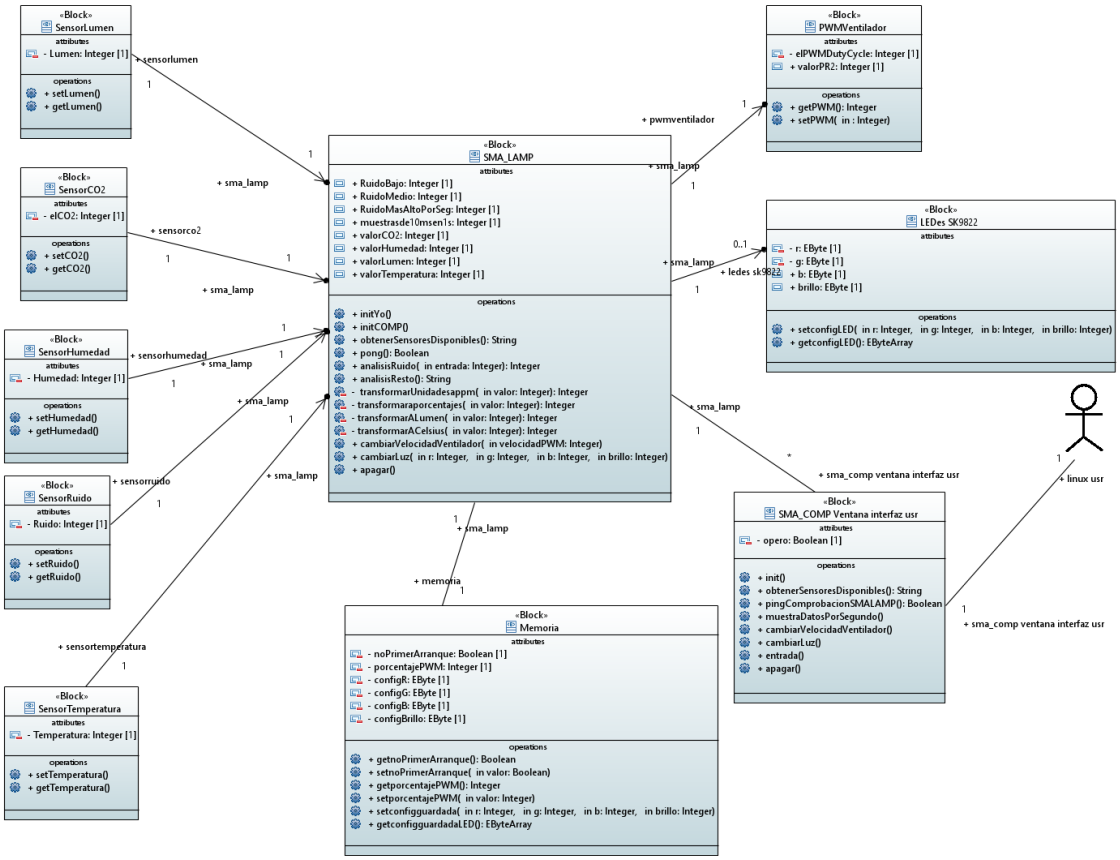
[8] supusieron un tiempo menor porque al final decidimos fusionar ambas desviaciones y planificaciones en un único documento que luego editaríamos más tarde.

[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 eldiseño físico.

B. Diseño:

1. Diseño software de alto nivel. Se planteará el diseño software de alto nivel (solo grandes bloques del sistema) utilizando para ello un diagrama de bloques (SysML).

Nos hemos basado en una estructura de arquitectura sensores-controlador-actuadores, en las que los sensores entregan los datos a la unidad SMA_LAMP de control que será la que actúe sobre los LEDes y motor PWM. Nótese que se incluyen dos bloques adicionales, uno que permite la comunicación en doble sentido con un usuario Linux, y otro que almacenará las características de los actuadores PWM y LEDes en memoria flash cada vez que se modifiquen.



2. Diseño lógico hardware. Debe elaborarse el diseño lógico hardware (o diseño esquemático) necesario para atender los requisitos planteados para el sistema (ver herramientas obligatorias).

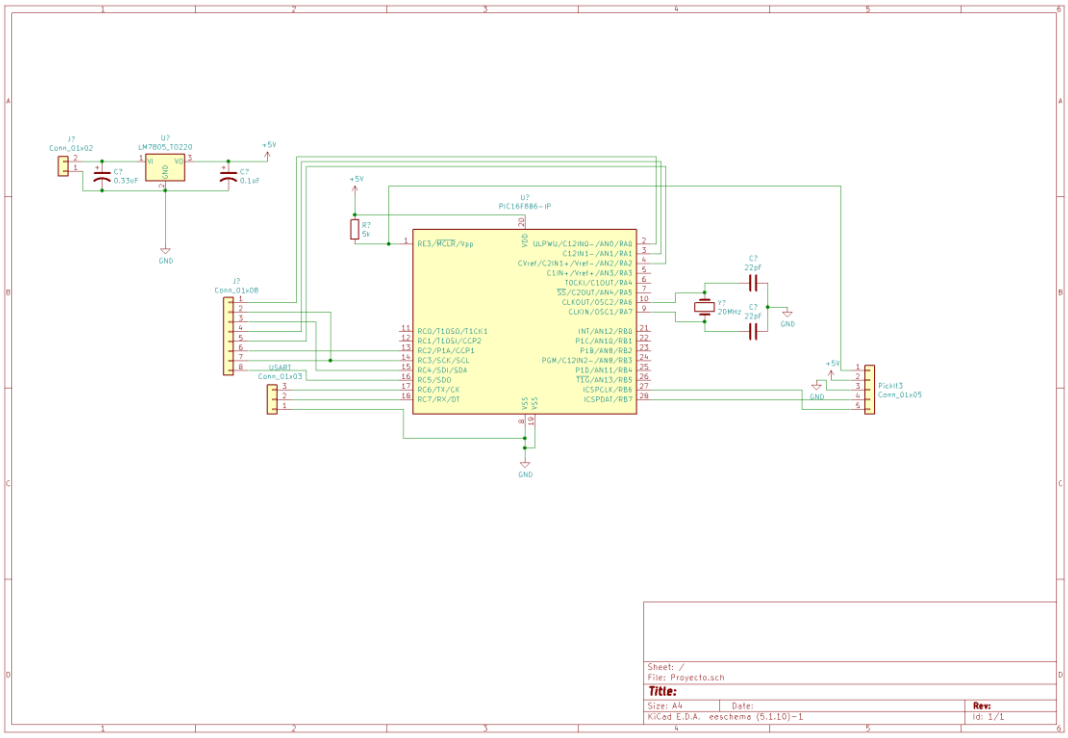
Según las indicaciones de la documentación del Microchip PIC 16F886, el circuito necesita de alimentación y señal de reloj para funcionar. Según el manual, entre Vpp y Vdd debe haber una resistencia mayor de 1kΩ pero menor de 10kΩ, y opcionalmente un condensador que conecte Vpp y GND. Nosotros hemos elegido aplicar 5kΩ y no emplear condensador. Además ambas entradas Vss deben estar conectadas, en nuestro caso a tierra. Queremos que la señal de reloj sea de 20MHz, así que a CLKOUT y CLKIN debemos conectar un cristal de dicha frecuencia, que

según las indicaciones del manual debe ir acompañado de dos condensadores de 22 pF conectados cada uno a un lado del cristal, y entre sí a tierra.

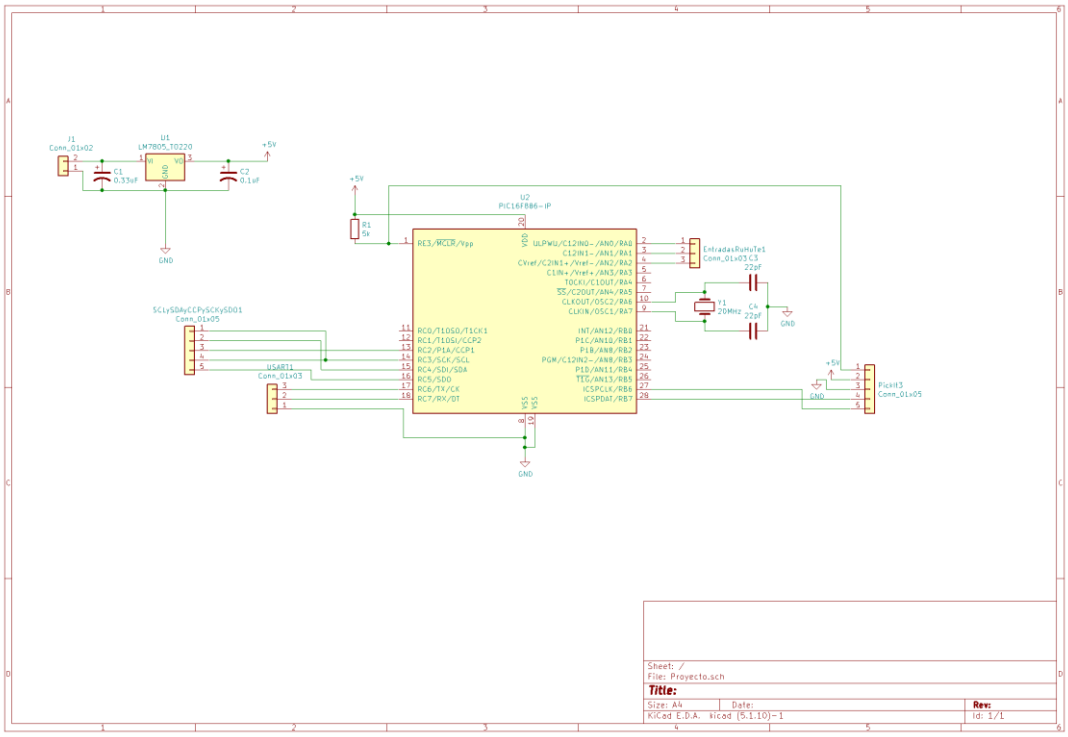
Además hemos tenido en cuenta el regulador lineal A7800 para una salida fija y eliminación de ruido, por el cuál debíamos asignar al regulador dos condensadores polares en las patas Vi y Vo de 0.33 y 0.1 µF, respectivamente.

Ya con las partes más obvias para el proyecto, necesitamos de una conexión a nuestro ordenador para programar la placa (con PICKIT3) y las entradas y salidas pedidas en el enunciado. Así, hemos conectado el PICKIT3 según las especificaciones (PIN1 al MCLR, PIN2 a Vdd, PIN3 a GND, PIN4 a RB7 y PIN5 a RB6, dejando PGM sin implementar); creado un Conector 01x03 para la USART (PIN3 es TX, PIN2 ES RX y PIN1 es GND) y colocado un conector 01x08 en el que los pines 1, 4 y 5 son de entradas analógicas de ruido, humedad relativa y temperatura y conectan con los pines analógicos AN0, AN1 y AN2, los pines 2 y 3 con SCL y SDA de I2C y el PIN6 para el motor con PWM al módulo CCP1. Desafortunadamente para nosotros, en nuestro chip el bus I2C y el bus SPI están conectados en mismos pines, por lo que como alternativa más sencilla deberemos simular uno de ellos (específicamente el SDI sobre el I2C), lo que supone que el pin 7 se conecta a SCK/SDL y será muy probable que aunque el pin 8 se conecte al SDO del SPI, luego acabe sin usarse.

Los sensores de CO2 y luminosidad se comunicarán por I2C

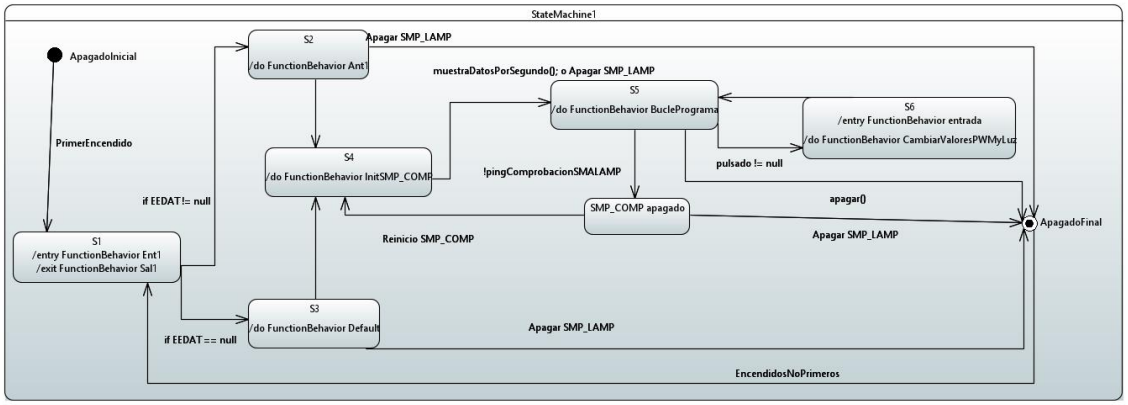


Sin embargo, al final por limitaciones a nivel físico (específicamente asegurarse de que todos los cables pudieran pasar sin cruzarse) realizamos unos cambios al diseño lógico, en el que el Connector de 8 lo dividimos en un Connector de 3 (para las entradas de Ruido, Humedad y Temperatura, pines 1,2, 3) y otro de 5 (1 y 2 para SCL y SDA, 3 para CCP1 y 4 y 5 para SCK y SDO)



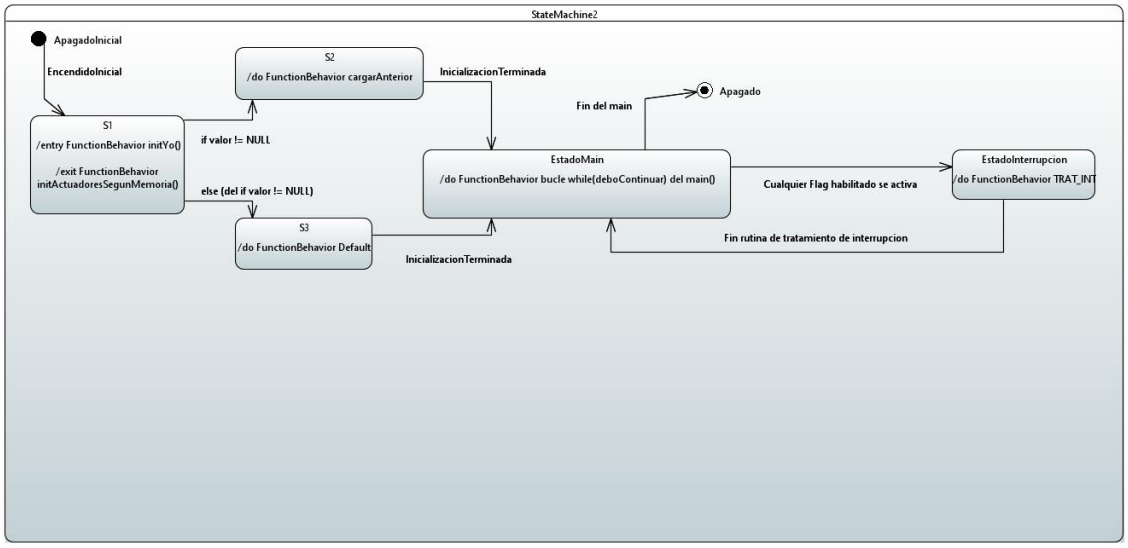
3. Diseño software detallado. Por medio de diagramas de estado, diagramas de secuencia u otros recursos que se consideren apropiados, debe plantearse el diseño detallado del componente software del sistema.

A partir del diseño SW de alto nivel hemos elaborado un diagrama de estados en el que partiendo de un estado de apagado inicial se inicia la placa SMA_LAMP y los periféricos, y al acceder a la memoria o bien se pone la configuración anterior, o si no existe la configuración por defecto de luz blanca y motor apagado. Una vez hecho eso se arranca el SMA_COMP, que iniciará sus ventanas, y se quedará en un bucle infinito que comprobará que se puede conectar al SMA_LAMP. Si no puede se cierra y deberá reiniciarse para volver a intentarlo. Si se conecta con éxito procederá a mostrar los datos de los sensores en un periodo de tiempo preestablecido (1 o 5 segundos) y dará la opción de aceptar por entrada de terminal o botón el permitir modificar la luz y la velocidad del ventilador. Una vez la configuración se ha realizado el SMA_LAMP procederá a escribir los datos en la memoria para un posible acceso futuro tras apagar.



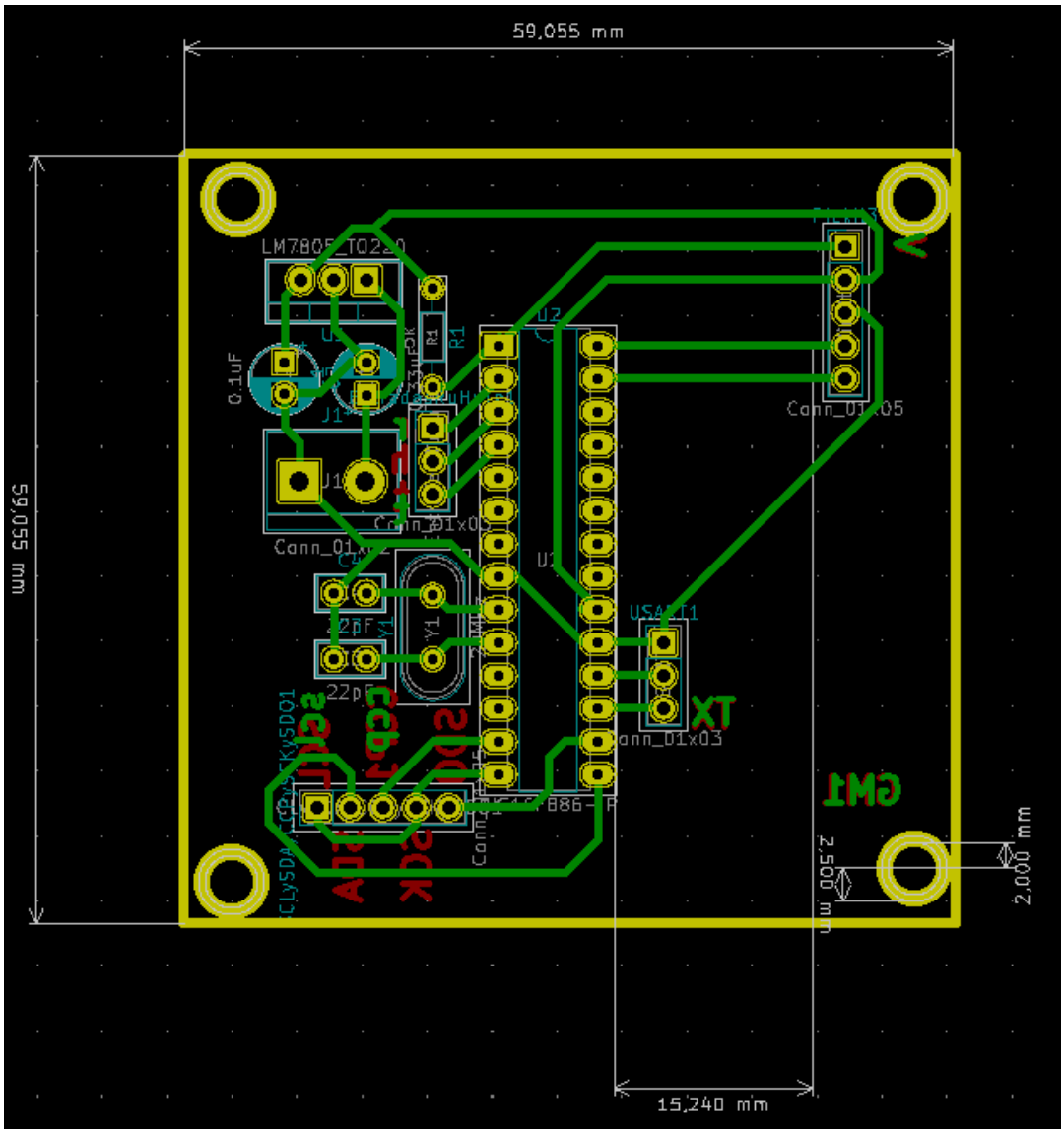
Más tarde tras recibir información adicional realizamos un segundo diagrama de Estados que indicaba de forma básica en qué momento el SMA_LAMP se encontraba en el programa main o en la rutina de tratamiento de interrupción. Este diagrama se centraba más en la lámpara y nos permitió definir con mucho más detalle como finalmente funcionaría la lámpara: el main iniciaría el sistema, comprobaría la memoria y según los datos obtenidos, actualizará los valores de la tira de LEDes por el bus SPI y del ventilador por señal PWM con el CCP1 y la nueva configuración se guardará en la memoria; tras hacer esto se quedaría en un bucle infinito en el que se irían haciendo algunas de las funciones pedidas, como actualizar el valor que luego se enviaría a SMA_COMP. La rutina de tratamiento de interrupción se encarga de un gran número de funciones importantes, regulación de los Timers 0 (cuenta procesos generales del main, cada segundo y cada 5 s), 1 (regulación del tiempo sample and hold del ADC, monitoriza el nivel de ruido cada 10 ms) y 2 (para generación de la señal PWM para nuestro ventilador), los flags de escritura completa en memoria, el de I2C para los sensores de luminosidad y CO2 y por último flags de la USART, tanto de TR como RX, para poder ejecutar diferentes opciones según una cadena enviada o recibida:

- "a" sería devolver los valores de sensores al SMA_COMP: TXREG = 'A' + valoresSensores(); valoresSensores devolvería una cadena de caracteres indicando el estado de las variables asociadas a los sensores indicados, según unos valores de variables globales manejados por funciones que informarían de si los valores son basura o no.
- "b" es hacer ping: TXREG = 'B'; (nos bastaría con esto para que el SMA_COMP supiera que ahí estamos).
- "c" es cambiar el PWM, setPWM(valor); donde valor sería el segundo char recibido de RCREG.
- "d" es cambiar LED, setLED(red, green, blue, luminosidad); siendo los parámetros el segundo, tercero, cuarto y quinto char recibido de RCREG
- "e" sería apagar el SMA_LAMP, se podría hacer con una señal de reset específica, la que mejor nos convenga a bajo nivel, y una variable int deboContinuar (inicialmente a 1) que se pusiera a 0 y así rompiera el bucle del programa principal.



4. Diseño físico hardware. A partir de la utilización de alguna herramienta de diseño, debe obtenerse el diseño físico hardware, respetando para ello el diseño lógico hardware previamente planteado (ver herramientas obligatorias).

Acá se eligió nombrar algunos de los pines como TX, el pin 1 de pickit3, los pines de ruido (r) y temperatura (t), el SCL y el CCCP1 (ccp) pero no el resto por falta de espacio. También se decidió conectar el LM7805 con el pin 3 del pickit3 en vez de con el de 5V del microchip porque a pesar de que la ruta era más corta hubiera supuesto pasar entre varios pines del microchip y eso acarrearía problemas a la hora de atacar la placa.



5. En el apartado de diseño deben plantearse las siguientes trazas:

a) Traza de componentes del diseño software de alto nivel contra requisitos.
b) Traza de diseño físico (HW) contra diseño lógico (HW) haciendo las siguientes comprobaciones.

    1) Todos los componentes del diseño lógico aparecen en el diseño físico.
    2) Todas las conexiones establecidas en el diseño lógico aparecen en el diseño físico.

Al comprobar el diseño físico no se ha visto que falten conexiones ni componentes del diseño lógico.

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

c) Comprobaciones añadidas del diseño físico (HW).
1) No hay soldaduras con imposibilidad de ser realizadas.
COMPROBADO.

2) El espacio reservado para el conector de programación permite insertar la sonda sin chocar con ningún componente (Pickit3).
COMPROBADO.

3) Las líneas de conexión entre componentes tienen el ancho adecuado y separación suficiente para el proceso de fabricación a utilizar.
4) Todos los componentes están correctamente alimentados.
COMPROBADO.

5) Se han introducido marcas de serigrafiado adecuadas para identificar correctamente el circuito y su interfaz.
COMPROBADO.

6) Se incluyen las marcas para el mecanizado de sujeción. Por defecto taladros de métrica 4 (deja espacio adicional).
COMPROBADO. AUNQUE EN UNA NOS SALIMOS AL REALIZAR EL AGUJERO, NINGUNA PISTA PELIGRÓ.

7) No hay pistas que hagan contacto con las partes metálicas del mecanizado de sujeción.
COMPROBADO.