Proyecto Parcial 1:Sistema de Control para Semáforos Inteligentes

Emanuel Araya Esquivel
Fabián Ramírez Arrieta
Jose David Sánchez Schnitzler
Jorge Wajib Zaglul Chinchilla

***Resumen*—En el presente documento se denotan los procesos llevados a cabo para el diseño de un código en ensamblador que permite crear un sistema de control pensado para semáforos inteligentes que regulan el tránsito en una intersección de dos calles.Para efectuar el diseño del sistema de control deseado, se hará uso del lenguaje ensamblador y se utilizará el compilador Keil que emplea el IDE Microvision versión 5.34.**

I. Introducción

El desarrollo de los microcontroladores en la industria electrónica ha hecho que estos dispositivos estén vinculados cada vez más en ámbitos cotidianos del ser humano. Ejemplo de ello es la implementación de cámaras y sensores en los sistemas de control vial, tales como semáforos inteligentes, cámaras que permiten determinar la velocidad con la que viaja un vehículo e incluso fotografiar cuando un auto ha cometido una infracción como saltarse un alto.

El objetivo del proyecto es diseñar un Sistema de Control basado en sensores que detecten la presencia de autos y personas que deban cruzar una intersección de dos carreteras, con la finalidad de que el sistema empleado para controlar los semáforos permita reducir los atascos viales y que ni los autos deban esperar por largos periodos para poder cruzar la intersección.

II. Desarrollo

Previo a iniciar la creación de código en lenguaje ensamblador se efectuaron varios diagramas que hacen alusión a los diferentes casos que pueden ocurrir en la intersección. De esta forma se tendrá un panorama que abarque la mayor cantidad de posibilidades y así evitar que el Sistema de Control pueda fallar ante un caso muy particular.

El diseño de tales diagramas se efectuó partiendo del hecho que la intersección posee dos carriles en sentidos opuestos, cuatro semáforos que regulan el tránsito y además considerando un señalamiento que permite a los conductores girar a la derecha con el semáforo en rojo cuando la maniobra así lo permita y ocho semáforos peatonales. Dichos turnos se muestran en la sub sección referente al diseño de la solución del problema.

II-A. Criterios para la Elección de la Arquitectura Empleada

Para este caso en específico se optó por trabajar con arquitectura ARM M3, la razón por la cual se escogió esta arquitectura radica en el hecho que está pensada para ser utilizada en sistemas que involucren el uso de sensores [1]. Tal como es el caso para el sistema de control el cual está diseñado para que funcione por medio de sensores capaces de detectar la presencia de autos en los carriles de las intersecciones. Estos sensores se encargarán de avisar al Sistema de Control cuando sea necesario cambiar las luces de los semáforos para dar paso a los carriles más congestionados. De igual manera los semáforos peatonales reciben una señal cuando hay personas esperando para cruzar la calle.

Otro motivo por el cual se implementó esta arquitectura es debido su bajo costo en consumo de potencia, lo cual es beneficioso para un sistema que debe estar operando en todo momento. El procesador del ARM M3 ofrece un consumo de energía de 4,5 mW . Sumado a esto, se encuentra el hecho de que es una de las arquitecturas que con núcleo más reducido lo cual beneficia para que tamaño de Sistema de control sea pequeño.[2].

El procesador Cortex-M3 ha sido diseñado para ser rápido y fácil de programar, sin que sea necesario que los usuarios deban tener un conocimiento profundo de la arquitectura para crear aplicaciones simples como es el caso de este proyecto.[3].

II-B. Diseño de la solución al problema

Se tomará como vía principal la calle en dirección Norte-Sur y como segunda en prioridad será la avenida de Este-Oeste. En las siguientes figuras se presentan los diagramas.

Los números (0,1,2,3) representan los carriles de la intersección, los puntos marcados en rojo indican que dicho carril tiene su semáforo en rojo, por otra parte el carril con una marca verde, indica que el semáforo está en verde.

Las flechas verdes indican las trayectorias que podrían seguir los vehículos cuando el semáforo del carril respectivo está en verde.

Las flechas naranja indican las trayectorias que pueden realizar los vehículos que estén en carriles con semáforo en rojo y pueden realizar un giro a la derecha, esto será viable siempre y cuando las trayectorias de las flechas verdes y naranjas no se crucen.

Las secciones marcadas en color morado representan los pasos peatonales, aquellos pasos peatonales resaltados en color turqueza indican que el semáforo peatonal está en verde, en caso contrario, indica que el semáforo está en rojo.

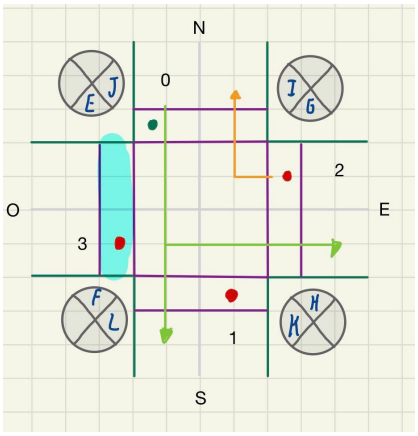

Fig. 1: Primer turno: autos que vayan en sentido Norte Sur o Norte Este

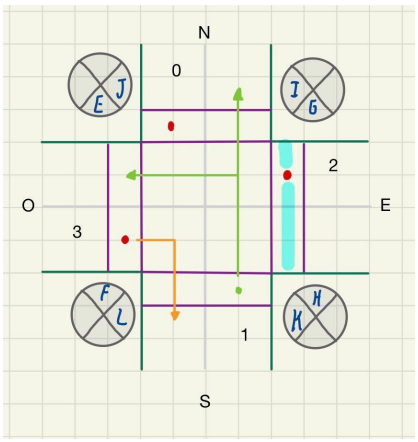

Fig. 2: Segundo turno:autos que vayan en sentido Sur-Norte o Sur-Oeste

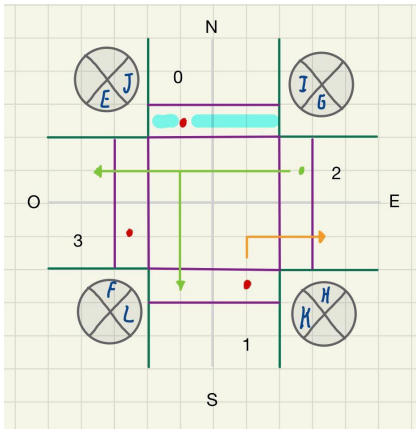

Fig. 3: Tercer turno:autos que vayan en sentido Este-Oeste o Este-Sur

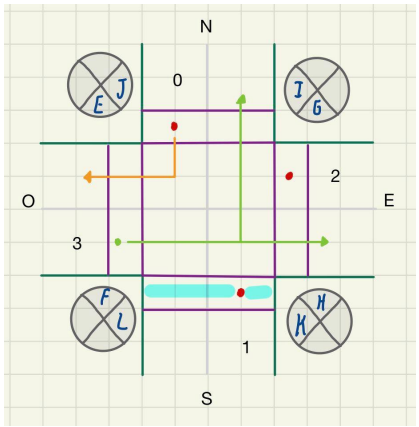

Fig. 4: Cuarto turno: autos que vayan en sentido Oeste-Sur u Oeste-Norte

Los círculos en los cuatro lados de la intersección en las figuras de arriba representan las posiciones donde se ubicarán los semáforos peatonales.

Una vez establecidas las diferentes trayectorias que pueden tomar los vehículos de acuerdo con el color que toma cada uno de los 4 semáforos, se pueden establecer los casos extremos a los que puede enfrentarse el Sistema de Control. Los casos extremos que pudieron encontrarse son:

1) Peatones en todos los cruces y vehículos en todos los carriles.
2) Vehículos en todos los carriles y ningún peatón en los cruces.
3) Ningún vehículo en los carriles y peatones en todos los cruces.
4) Ningún vehículo en los carriles y ningún peatón en los cruces.

Una vez establecidos los diseños preliminares sobre como funcionará el sistema que controla los semáforos y definido el diagrama de estados que describe el comportamiento de los semáforos. Se procede a elegir la arquitectura más óptima para llevar a cabo el proyecto.

II-C. Máquinas de estado

El diagrama de estados, mostrado en la figura 5, se desarrolló a un nivel general de forma que quede plasmado los turnos planteados anteriormente. Nuestro sistema de semáforos los turnos se mueven de manera secuencial respetando un orden ascendente y cuando se llega al turno 4, se vuelve a iniciar el ciclo.

Siguiendo la lógica de los turnos explicados anteriormente, el turno uno está asociado al estado cero, el dos al estado uno y así sucesivamente. Sin embargo, hay un estado extra, el estado cinco, el cual corresponde al manejador de casos extremos. Dentro de este estado el sistema funciona con la misma lógica normal solamente que modifica la espera entre estados para cumplir con el requisito de diseño de los tiempos de espera. En el momento que se notifique que el estado de caso extremo se acabe, el sistema volverá al estado cero y volverá a su funcionamiento normal.

Además del diagrama de estados, se considera de gran importancia elaborar un diagrama de flujo que muestre el comportamiento esperado del sistema el cual se muestra a continuación en la figura 6.

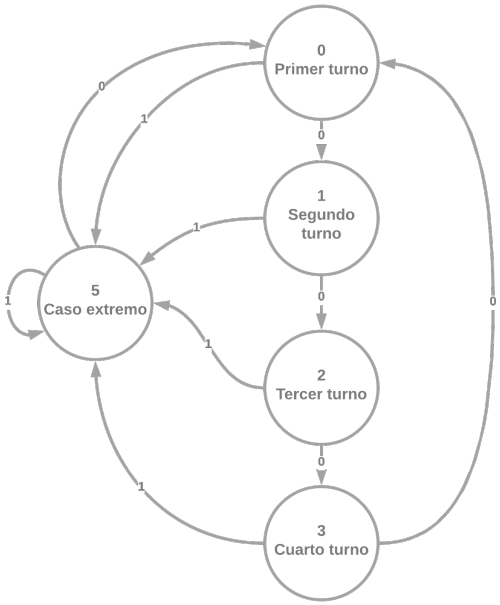

Fig. 5: Sistema de semáforos representado en máquina de estados

II-D. Simbología en el código de ARM en Keil

Para inicializar el código de ensamblador se utilizan posiciones de memoria fijas para guardar el estado de las luces de cada semáforo además deben contener un valor inicial para su debido funcionamiento. Se les asigna nombres significativos por lo que es intuitivo. Dichas posiciones de memoria vienen definidas en la tabla I.

Tabla I: Direcciones de memoria asignadas para el estado de los semáforos.

Nombre significativo	Dirección de memoria	Valor inicial
SemaNorteSurVerde	0x20000000	1
SemaNorteSurAmarillo	0x20000004	0
SemaNorteSurRojo	0x20000008	0
SemaEsteOesteVerde	0x2000000C	0
SemaEsteOesteAmarillo	0x20000010	0
SemaEsteOesteRojo	0x20000014	1
SemaPeatonalNorteSurVerde	0x20000018	1
SemaPeatonalNorteSurRojo	0x2000001C	0
SemaPeatonalEsteOesteVerde	0x20000020	0
SemaPeatonalEsteOesteRojo	0x20000024	1

Seguidamente se definen el uso general de los registros que se utilizaron en el desarrollo del código. Dichos registros se presentan en la tabla II.

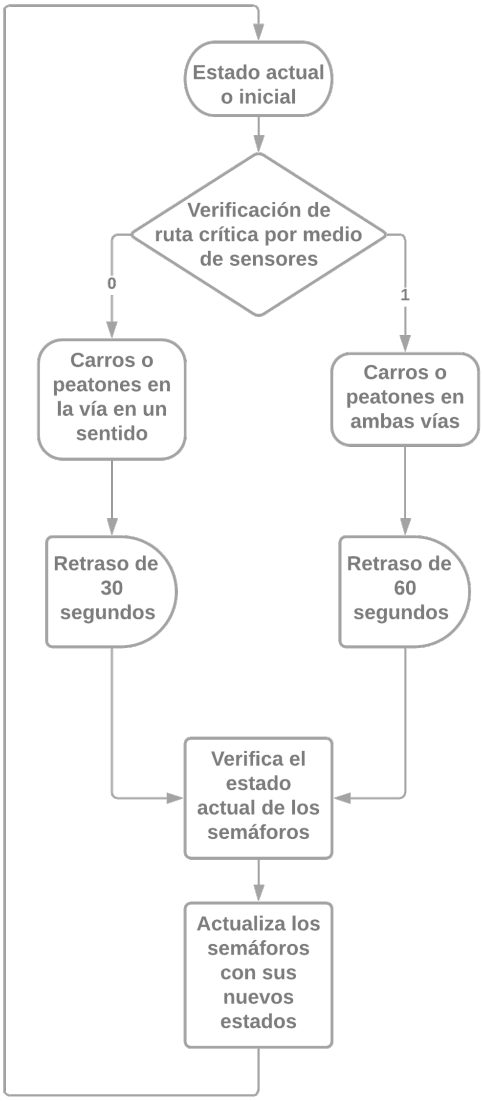

Fig. 6: Diagrama de flujo del sistema

Tabla II: Registros a utilizar, el papel que juega en el código además de sus valores iniciales

Registro	Papel	Valor inicial
R0	Contador	0
R1	Valor de memoria	0
R2	Puntero de memoria	0
R3	Bandera de carro Norte a Sur	0
R4	Bandera de carro Este a Oeste	1
R5	Bandera de peaton Norte a Sur	0
R6	Bandera de peaton Este a Oeste	1
R7	Bandera de ruta crítica	0
R8	Bandera de peaton o carro de Norte a Sur	0
R9	Bandera de peaton o carro de Este a Oeste	0

II-E. Complicaciones en cuanto a código de ARM en Keil

En esta subsección se muestran las complicaciones presentes en el desarrollo del código de ARM en Keil. Para empezar cabe destacar que la función de Branch Link Equal la que funciona asociada a una comparación no funciona correctamente

en el ambiente de trabajo de Keil por lo que hacer un salto por medio de un Link Register (LX), Branch y comparación se tuvo que resolver haciendo un bloque aparte únicamente para saltara nuevamente donde apuntaba el registro Link Register con la instrucción BX asociado a LX.

III. Resultados

En este apartado se presenta la herramienta de Debugger en el ambiente de trabajo de Keil para ARM. En dicha herramienta se pueden ver los valores específicos tanto de los registros como de las direcciones de memoria. Para empezar se muestra la reserva de direcciones de memoria para los estados de las luces de los semáforos, esto se muestra en la figura 7.

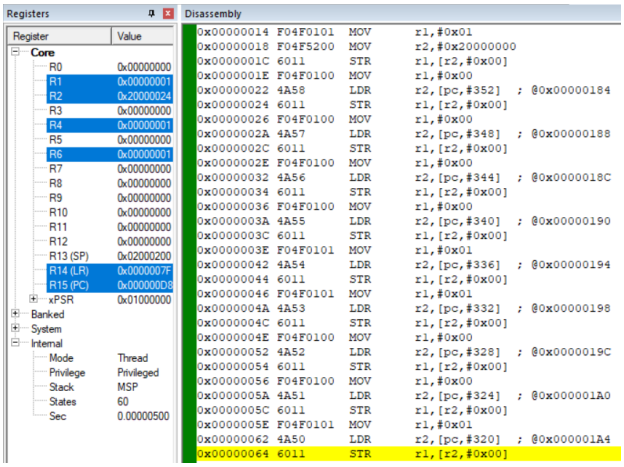

Fig. 7: Reserva de las direcciones de memoria para los estados de los semáforos

Seguidamente se muestran los valores iniciales de los registros que se van a utilizar, esto se muestra en la figura 8. En este caso de R0 a R7, tomando como referencia la tabla II.

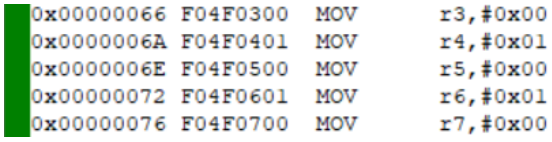

Fig. 8: Inicializando los registros con sus valores de caso inicial

Luego de inicializar los registros con sus valores iniciales y además de reservar las direcciones de memoria para los estados de los semáforos se procede a crear una lógica básica por medio de operadores booleanos usando los registro R8 y R9 para alterar R7 que es la bandera que se va a activar en caso de que se detecte un caso crítico. Dicha ejecución de lógica se muestra en la figura 9 donde se alteran las banderas de estado de las vías y seguidamente se inicializa la verificación de ruta crítica para saber qué contador usar, esto se observa en la figura 10.

El siguiente paso es ejecutar los contadores dependiendo del valor de R7 el cual va a ser la bandera de estado crítico, en el caso que R7 tenga un 0 significa que no hay un caso crítico por lo que va a usar el contador de 30 segundos. En el

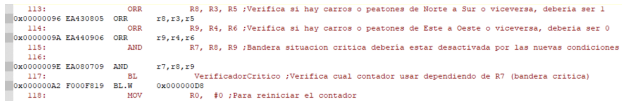

Fig. 9: Lógica para saber el estado de las vías

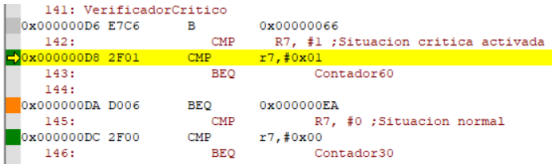

Fig. 10: Verifica si el caso es crítico para saber qué contador usar

caso que R7 tenga el valor 1, se va a usar el contador de un minuto. Dichos contadores se prueban de igual manera por lo que solamente se hizo la prueba del contador de 30 segundos como se muestra en la figura 11.

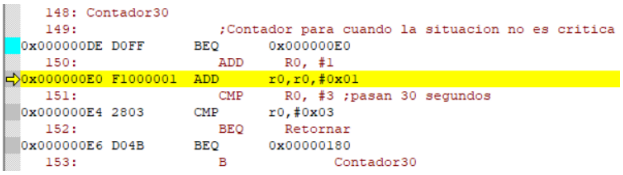

Fig. 11: Prueba de contadores

Luego de que se completa el proceso de los contadores, se procede a cambiar los semáforos, esto sucede primero con un bloque de código que compara el estado actual para saber el estado futuro a cambiar. Una vez que se sabe el estado futuro se dirige a su respectivo bloque de código para alternar las luces de los semáforos. Dicha verificación se prueba en la figura 12.

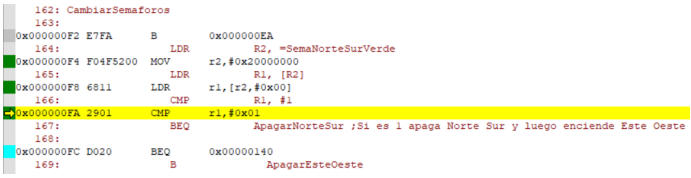

Fig. 12: Prueba del bloque de código para cambiar semáforos

Una vez verificado el estado actual se compara el valor para saber a cuál bloque de código ejecutar y de esta manera simular el encendido y apagado de los estados de las luces de los semáforos. Esto se aprecia en las figuras 13 y 14 respectivamente.

Una vez que se termina el código vuelve a empezar verificando los nuevos estados actuales de las vías en los registros R3, R4, R5 y R6 como se había visto en casos anteriores por medio de un bloque de código asociado al Link Register, dicho bloque se le nombra Retornar y se prueba en la figura 15.

IV. Conclusiones

Luego de realizar este sistema de semáforos se puede resaltar una serie de puntos importantes a concluir. Primeramente es

```
   207: ApagarNorteSur
   208:
0x0000013E E01F      B        0x00000180
   209:                MOV          R1, #0
0x00000140 F04F0100  MOV      r1,#0x00
   210:                STR          R1, [R2] ;Apaga el semaforo SemaNorteSurVerde para carros
0x00000144 6011      STR      r1,[r2,#0x00]
   211:                LDR          R2, =SemaPeatonalNorteSurVerde
0x00000146 4A14      LDR      r2,[pc,#80]  ; @0x00000198
   212:                STR          R1,  [R2] ;Apaga el semaforo SemaNorteSurVerde para peatones
   213:
0x00000148 6011      STR      r1,[r2,#0x00]
   214:                MOV          R1, #1
0x0000014A F04F0101  MOV      r1,#0x01
   215:                LDR          R2, =SemaNorteSurAmarillo ;Eciende SemaNorteSurAmarillo para carros
0x0000014E 4A0D      LDR      r2,[pc,#52]  ; @0x00000184
   216:                STR          R1, [R2]
0x00000150 6011      STR      r1,[r2,#0x00]
   217:                MOV          R1, #0
0x00000152 F04F0100  MOV      r1,#0x00
   218:                STR          R1, [R2] ;Apaga SemaNorteSurAmarillo para carros
   219:
0x00000156 6011      STR      r1,[r2,#0x00]
   220:                MOV          R1, #1 ;
0x00000158 F04F0101  MOV      r1,#0x01
   221:                LDR          R2, =SemaNorteSurRojo
0x0000015C 4A0A      LDR      r2,[pc,#40]  ; @0x00000188
   222:                STR          R1, [R2] ;Eciende SemaNorteSurRojo para carros
0x0000015E 6011      STR      r1,[r2,#0x00]
   223:                LDR          R2, =SemaPeatonalNorteSurRojo
0x00000160 4A0E      LDR      r2,[pc,#56]  ; @0x0000019C
   224:                STR          R1, [R2] ;Eciende SemaNorteSurRojo para peatones
```

Fig. 13: Prueba de semáforos con simulación de poner en rojo los semáforos de Norte a Sur y viceversa

```
   228: EncenderEsteOeste
   229:
0x00000164 E7FF      B        0x00000166
   230:                MOV          R1, #0
0x00000166 F04F0100  MOV      r1,#0x00
   231:                LDR          R2, =SemaEsteOesteRojo
0x0000016A 4A0A      LDR      r2,[pc,#40]  ; @0x00000194
   232:                STR          R1, [R2] ;Apaga SemaEsteOesteRojo para carros
0x0000016C 6011      STR      r1,[r2,#0x00]
   233:                LDR          R2,  =SemaPeatonalEsteOesteRojo
0x0000016E 4A0D      LDR      r2,[pc,#52]  ; @0x000001A4
   234:                STR          R1, [R2] ;Apaga SemaEsteOesteRojo para peatones
   235:
0x00000170 6011      STR      r1,[r2,#0x00]
   236:                MOV          R1, #1
0x00000172 F04F0101  MOV      r1,#0x01
   237:                LDR          R2, =SemaEsteOesteVerde
0x00000176 4A05      LDR      r2,[pc,#20]  ; @0x0000018C
   238:                STR          R1, [R2] ;Enciende SemaEsteOesteVerde para carros
0x00000178 6011      STR      r1,[r2,#0x00]
   239:                LDR          R2,  =SemaPeatonalEsteOesteVerde
0x0000017A 4A09      LDR      r2,[pc,#36]  ; @0x000001A0
   240:                STR          R1, [R2] ;Enciende SemaEsteOesteVerde para peatones
   241:
0x0000017C 6011      STR      r1,[r2,#0x00]
   242:                B            Retornar
```

Fig. 14: Prueba de semáforos con simulación de poner en verde los semáforos de Este a Oeste y viceversa

esencial recalcar como la implementación de la ingeniería aplicada en soluciones de la vida cotidiana es imprescindible en la sociedad moderna.

Por otro lado, como equipo de trabajo se aprendió a valorar de gran manera la etapa previa de diseño para tener claro el panorama del problema planteado y de esta forma poder formular una solución integral, en la cual se tomen las mejores decisiones tanto de comportamiento, como es el caso de los turnos, así como en el equipo a utilizar más eficiente, como en el caso del procesador seleccionado.

Además, se concluye que es de vital importancia el orden a la hora de diseñar estas aplicaciones para poder analizar todos los posibles casos, para de esta forma ser tomados en cuenta y prevenir cualquier tipo de accidente que la falta de alguno de estos pueda causar. Es por esto que el uso de diagramas fue de suma importancia a la hora de desarrollar el proyecto, ya que permitió visualizar fácilmente todos los posibles casos en las vías de tránsito.

```
   244: Retornar
0x0000017E E7FF      B        0x00000180
   245:                BX           LR
```

Fig. 15: Retornar al Main para seguir con el código en este último bloque

V. Cronograma

Tabla III: Colaboradores

Nombre	Escuela	Horas
Emanuel Araya Esquivel	Electrónica	24 Horas
Fabián Ramírez Arrieta	Computadores	24 Horas
Jose David Sánchez Schnitzler	Computadores	26 Horas
Wajib Zaglul Chinchilla	Computadores	24 Horas

Tabla IV: Cronograma

Actividad	Responsables	Tiempo
Diseño de Ingeniería	Todos	8 Horas
Diseño de Software	Todos	4 Horas
Evaluación de Prototipo	Jose Sánchez Schnitzler	2 Horas
Creación de parámetros de Prueba	Jose Sánchez Schnitzler	1 Hora
Reuniones y preparación de documentación	Todos	3 Horas
Pruebas de funcionamiento	Jose Sánchez Schnitzler	0.5 Horas
Elaboración del documento formato IEEE	Todos	9 horas

VI. Referencias Bibliográficas

[1] AMR, ARM Cortex M3 Datasheet.Disponible en: https://developer.arm.com/documentation/#sort=relevancy

[2] ARM, An Introduction to the ARM Cortex M3 Processor. ARM KEIL. 2006. Disponible en: https://class.ece.uw.edu/474/peckol/doc/StellarisDocumentation/IntroToCo M3.pdf

[3] Elahi. A. Sistemas Computacionales. Connecticut: Universidad del Estado de Connecticut.