

4. Especificación *mikro_STR* con Extensión *UML*

Como segundo acercamiento se introduce la EGS *mikro_STR* con extensión *UML*. Se sigue aplicando el proceso iterativo (Fig. 1), ahora introduciendo el desarrollo de software con extensiones de tiempo real del *UML* (Herrera, 2011; Douglass, 2014).

La idea en esta nueva etapa es que el desarrollador, pueda tener acceso sin complicaciones metodológicas, pudiendo adaptar los conceptos impartidos en *mikro_STR*, al diseñar una variedad de sistemas de tiempo real, a partir de pequeños sistemas de microcontroladores de bajos requerimientos, extendiendo a grandes sistemas en red de varios SE. Así podrá prepararse para experiencias complejas como la utilización de sistemas operativos de tiempo real como MaRTE u otro, tomando ventaja de los métodos modernos de ingeniería basados en modelos y normas de la industria correspondientes para superar estas limitaciones (Selic y Gerad, 2013; Douglass, 2014).

4.1 Notación Gráfica

Es importante representar las características del sistema apoyados en diagramas *UML* con las extensiones para sistemas de tiempo real (Herrera, 2011; Douglass, 2014). El *UML* fue desarrollado en respuesta a la necesidad de un lenguaje de modelado de objetos estandarizados. Puede ser adaptado para diseñar una variedad de sistemas de tiempo real, a partir de pequeños sistemas de microcontroladores de 8 bits para grandes sistemas en red de varios procesadores. *UML* incluye características para la funcionalidad de modelado, objetos, estados, patrones de diseño y características de extensibilidad (Douglass, 2014; Wolf, 2012; Goma, 2011; Douglass, 2004).

Si bien *UML* no ha sido desarrollado específicamente para modelar sistemas de tiempo real, algunos de los conceptos tradicionales de la orientación a objetos, como las clases y los paquetes se han mejorado desde la tradicional *UML*. Dentro de este contexto, tres estructuras principales se puede citar para ayudar en el modelado de tiempo real: unidades funcionales, puertos y conectores (Douglass, 2014).

- Las unidades funcionales o clases activas que representan módulos de la arquitectura, cuyos únicos puntos de interacción con otros módulos son por los puertos.
- Los puertos representan puntos de interacción de una clase, o interfaces que especifican las operaciones y las señales ofrecidas por una clase, llamados protocolos, lo que permite la comunicación con el ambiente externo.
- Los conectores especifican un enlace que permite la comunicación entre dos o más entidades, siendo este último, por ejemplo, ya sea una clase, o un objeto, o un actor en un caso de uso.

UML propone que el modelado se realiza a través de varios diagramas que muestran visualmente características del software de modelado. Como se muestra en la Figura 12, en total hay ocho diagramas divididos en dos subconjuntos llamados diagramas estructurales y de comportamiento, cada uno con su propia función, que en conjunto permiten la comprensión del software del sistema como un todo.

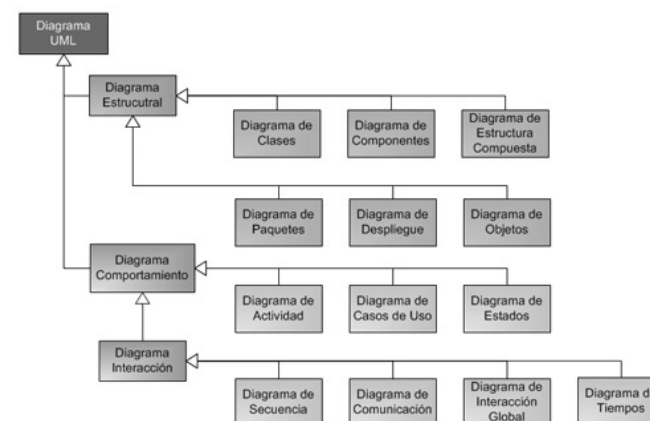


Figura 12. Diagramas UML (OMG, 2005)

A continuación se explican los diagramas representativos que se usan en este trabajo producto de una análisis crítico realizado (Douglass, 2014; Wolf, 2012; Douglass, 2004)

4.1.1 Diagrama de Caso de Uso

Un Caso de Uso especifica una manera de usar un sistema sin revelar la estructura interna del mismo. Los Casos de Uso han sido adoptados casi universalmente para capturar los requerimientos de los sistemas de software; sin embargo, los Casos de Uso son más que una herramienta de especificación ya que tienen una gran influencia sobre todas las fases del proceso de desarrollo tales como el diseño, la implementación y las pruebas del sistema.

Respecto a los elementos del Diagrama de Casos de Uso, presentados en la tabla 4, siendo sus definiciones las siguientes:

Actores: alguien o algo externo al sistema que interactúa con él desempeñando un rol.

Casos de uso: interacción entre actores y el sistema que produce un resultado observable de valor para un actor.



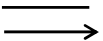
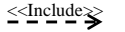
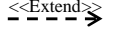

La relación posible entre un actor y un caso de uso es la Asociación.

Se pueden establecer tres relaciones entre casos de uso:

- **Inclusión**: es una relación mediante la cual se re-usa un Caso de Uso encapsulado en distintos contextos a través de su invocación desde otros Casos de Uso.
- **Extensión**: es una relación que amplía la funcionalidad de un Caso de Uso mediante la extensión de sus secuencias de acciones.

- **Generalización:** es una relación que amplía la funcionalidad de un Caso de Uso o refina su funcionalidad original mediante el agregado de nuevas operaciones y/o atributos y/o secuencias de acciones.

Tabla 4. Elementos del Diagrama de Casos de Uso.

Componentes	Elementos	Notación
Diagrama de Casos de Uso	Casos de Uso	
	Actores	
	Asociaciones	
	Relaciones	
	Inclusión	
	Extensión	
	Generalización	

4.1.2 Diagrama de Componentes

Se utilizan para modelar la vista estática de un sistema. Muestra la organización y las dependencias entre un conjunto de componentes. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente se realizan por partes. Cada diagrama describe un apartado del sistema. Con este tipo de diagramas se representan entidades reales, esto es, componentes de software. Un componente de software puede ser una tabla, un archivo de datos, un ejecutable, documentos, un *applet* de Java, etc.

Respecto a los elementos del Diagrama de Componente, presentados en la tabla 5, siendo sus definiciones las siguientes:

Componente: Elemento de funcionalidad del sistema reutilizable. Un componente proporciona y utiliza el comportamiento a través de las interfaces y puede hacer uso de otros componentes.

Puerto de interfaz proporcionada: Representa un grupo de mensajes o llamadas que un componente implementa y que otros componentes o sistemas externos pueden utilizar. Un puerto es una propiedad de un componente que tiene una interfaz como tipo.




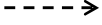
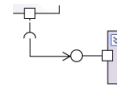
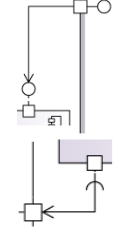
Puerto de Interfaz necesaria: Representa un grupo de mensajes o llamadas que el componente envía a otros componentes o sistemas externos. El componente está diseñado para que se acople a los componentes que proporcionan al menos estas operaciones. El puerto tiene una interfaz como tipo.

Dependencia: Se puede utilizar para indicar que una interfaz necesaria de un componente se puede satisfacer mediante una interfaz proporcionada de otro. Las dependencias también se pueden utilizar con más frecuencia entre los elementos del modelo para mostrar que el diseño de uno depende del diseño del otro.

Ensamblado de elementos: Conexión entre los puertos de la interfaz necesaria de un elemento y los puertos de la interfaz proporcionada de otro. La implementación de un ensamblado de elementos puede variar de un componente a otro. Los elementos conectados deben tener el mismo componente primario.

Delegación: Vincula un puerto a una interfaz de uno de los elementos del componente. Indica que los mensajes enviados al componente se administran en el elemento o que los mensajes enviados desde el elemento se envían fuera del componente primario.

Tabla 5. Elementos del Diagrama de Componentes.

Componentes	Elementos	Notación
Diagrama de Componentes	Componente	
	Puerto de interfaz proporcionada	
	Puerto de interfaz necesaria	
	Dependencia	
	Ensamblado de elementos	
	Delegación	

4.1.3 Diagrama de Secuencia

Los diagramas de secuencia modelan el flujo de la lógica dentro del sistema de forma visual, permitiendo documentarla y validarla. Pueden usarse tanto en análisis como en diseño, proporcionando una buena base para identificar el comportamiento del sistema. Típicamente se usan para modelar los escenarios de uso del sistema, describiendo de qué formas puede usarse. La secuencia puede expresar tanto un caso de uso completo como quizá un caso concreto del mismo contemplando algunas alternativas.

Respecto a los elementos del Diagrama de Secuencia, presentados en la tabla 6, siendo sus definiciones las siguientes:

Línea de vida: Una línea vertical que representa la secuencia de eventos que se producen en una unidad funcional durante una interacción, mientras el tiempo avanza. Esta unidad funcional puede ser una instancia de una clase, componente o actor.

Unidad coherente de funcionalidad: *Classifier* y por lo tanto posee una colección de operaciones (con sus correspondientes métodos) describiendo su comportamiento. O un Actor, externo al sistema que está desarrollando.

Mensaje sincrónico: representa la invocación de una operación en la cual el objeto invocante se queda bloqueado esperando la terminación de la misma.

Mensaje asincrónico: representa la invocación no bloqueante, cuando el objeto que invoca la operación continúa de inmediato su hilo de ejecución, sin esperar respuesta ni que la operación sea ejecutada por el objeto invocado.

Incidencia de ejecución: Un rectángulo sombreado vertical que aparece en la línea de la vida de una unidad funcional y representa el período durante el que la unidad funcional está ejecutando una operación. La ejecución comienza donde la unidad funcional recibe un mensaje. Si el mensaje inicial es un mensaje sincrónico, la ejecución finalizará con una flecha de «devolución» al remitente.

Mensaje de devolución de llamada: representa el retorno de un mensaje sincrónico.

Automensaje: Un mensaje de una unidad funcional a sí mismo. La aparición de ejecución resultante aparece encima de la ejecución de envío.

Crear mensajes: Un mensaje que crea una unidad funcional. Si una unidad funcional recibe un mensaje de creación, este debe ser el primer mensaje que recibe.

Mensaje encontrado: Un mensaje asincrónico de una unidad funcional desconocida o no especificada.

Mensaje perdido: Un mensaje asincrónico a una unidad funcional desconocida o no especificada.

Uso de interacción: Agrega una secuencia de mensajes que se definen en otro diagrama.

Tabla 6. Elementos del Diagrama de Secuencia.

Componentes	Elementos	Notación
Diagrama de Secuencia	Línea de vida	
	Unidad coherente de funcionalidad	
	Mensaje sincrónico	
	Mensaje asincrónico	
	Incidencia de ejecución	
	Mensaje de devolución de llamada	
	Automensaje	
	Crear mensajes	
	Mensaje encontrado	
	Mensaje perdido	
	Uso de interacción	

4.1.4 Diagrama de Actividades

En un diagrama de actividades se muestra un proceso de negocio o un proceso de software como un flujo de trabajo a través de una serie de acciones. Estas acciones las pueden llevar a cabo personas, componentes de software o equipos.

Respecto a los elementos del Diagrama de Actividades, presentados en la tabla 7, siendo sus definiciones las siguientes:

Actividad: Una actividad es la especificación de una secuencia parametrizada de comportamiento. Una actividad muestra un rectángulo con las puntas redondeadas adjuntando todas las acciones, flujos de control y otros elementos que constituyen la actividad.

Acción: Una acción es un paso de un proceso o actividad que tiene la semántica *run to completion* (Se inicia para ser terminado).

Flujo de Control: Muestra el flujo de control de una acción a otra. Su notación es una línea con una punta de flecha.

Nodo inicial: Un nodo inicial o de comienzo se describe por un gran punto negro.

Nodo final de actividad: denota el final de todos los flujos finales dentro de la actividad.

Nodo final de flujo: Marca el punto final de un flujo, dejando en ejecución el resto de flujos.

Nodo de decisión: as decisiones representan las alternativas de flujo de control en un diagrama que se llevan a cabo en función de una condición

Condición de flujo o Guarda: Las condiciones de guarda asociadas a cada rama de salida determinan la opción de flujo de control que se sigue.

Nodo de combinación: dos flujos de control al pasar por el nodo de unión produce un solo flujo de salida.

Comentario: Proporciona información adicional sobre los elementos a los que está vinculado.

Subactividad: Actividad que se describe más en detalle en un diagrama de actividades aparte.

Nodo de división o bifurcación: pasa cualquier flujo de control directamente a través de esta. Si dos o más flujos de entrada se reciben por un símbolo de combinación, la acción a la que el flujo de salida apunta se ejecuta dos o más veces.

Nodo de unión: Sincroniza dos flujos de entrada y produce un solo flujo de salida. El flujo de salida desde una unión no se puede ejecutar hasta que todos los flujos se hayan recibido.

Acción de señal de envío: Representa la acción de enviar una señal.

Acción de aceptación de evento: Representa la acción de aceptar una señal.

Alfonsi A., Alfonsi A.R. y Yáñez R. Estructura Global de Software para Aplicaciones de Control Empotrado con Restricciones Temporales en Microcontroladores con Recursos Limitados. Informe Final Proyecto de Investigación CI-UDO. Universidad de Oriente, Barcelona, 2015.

Aceptar evento temporal: Tipo particular de acción 'acceptar' en la que la señal es una señal de tiempo.

Región que se puede interrumpir: Representa un grupo de actividades que se pueden interrumpir.

Excepción: Representa la ocurrencia de una excepción.

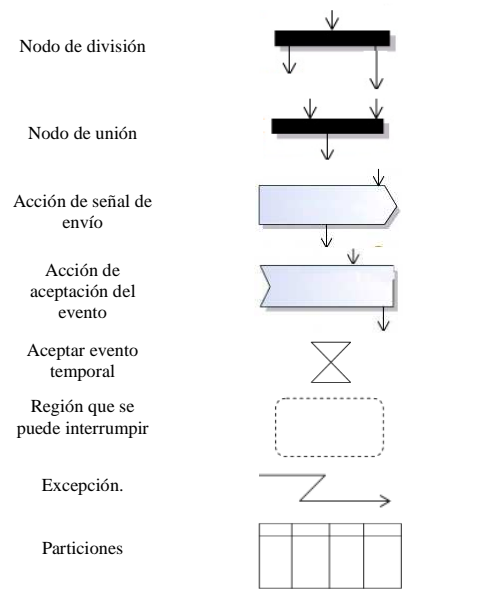
Particiones: Se pueden hacer particiones en un diagrama de actividades para identificar las acciones que tienen alguna característica en común. Por ejemplo que se llevan a cabo por un mismo actor.

Tabla 7. Elementos del Diagrama de Actividades

Componentes	Elementos	Notación
Diagrama de Actividades	Acción	
	Flujo de Control	
	Nodo inicial	
	Nodo final de actividad	
	Nodo final de flujo	
	Nodo de decisión	
	Condición de flujo	[Condición de flujo]
	Nodo de combinación	
	Comentario	
	Comportamiento de llamada	

Alfonsi A., Alfonsi A.R. y Yáñez R. Estructura Global de Software para Aplicaciones de Control Empotrado con Restricciones Temporales en Microcontroladores con Recursos Limitados. Informe Final Proyecto de Investigación CI-UDO. Universidad de Oriente, Barcelona, 2015.

Continuación Tabla 7



4.1.5 Diagrama de Estados

Los Diagramas de Estados muestran una Máquina de Estado. Son útiles para modelar la vida de un objeto. Un diagrama de estados muestra el flujo de control entre estados (en qué estados posibles puede estar “cierto algo” y como se producen los cambios entre dichos estados).

Respecto a los elementos del Diagrama de Estados, presentados en la tabla 8, siendo sus definiciones las siguientes:

Estado: Un estado identifica una condición o una situación en la vida de un objeto durante la cual satisface alguna condición, ejecuta alguna actividad o espera que suceda algún evento. Un objeto permanece en un estado durante un tiempo finito (no instantáneo).

Un estado se representa gráficamente por medio de un rectángulo con los bordes redondeados y con una o tres divisiones internas. El de un compartimiento aloja el nombre del estado. El de tres compartimientos aloja el nombre del estado, el valor característico de los atributos del objeto en ese estado y las acciones que se realizan en ese estado, respectivamente.

Respecto al valor característico de los atributos del objeto en ese estado aquí pueden ser listados y asignados. Los atributos son aquellos de la clase visualizados por el diagrama de estado.

El tercer compartimento es el compartimento de las transiciones internas, donde se listan las actividades o las acciones internas ejecutadas en respuesta a los eventos recibidos mientras el objeto está en un estado, sin cambiar de estado. La sintaxis formal dentro de este compartimento es:

nombre-evento ('lista-argumentos') '['guard-condition']' '/' expresión-acción

Las siguientes acciones especiales tienen el mismo formato, pero representan palabras reservadas que no se pueden utilizar para nombres de eventos:

entry '/' expresión-acción

exit '/' expresión-acción

Las acciones *entry* y *exit* no tienen argumentos, pues están implícitos en ellas. Cuando se entra al estado o se sale del estado, se ejecuta la acción atómica especificada en expresión-acción.

La siguiente palabra clave representa la llamada de una actividad:

do '/' expresión-acción

La transición especial *do* sirve para especificar una actividad que se ejecuta mientras se está en un estado, por ejemplo, enviando un mensaje, esperando o calculando. Dicha actividad es la que aparece en expresión-acción.

Por último, las transiciones internas, que son esencialmente interrupciones, se especifican teniendo en cuenta la sintaxis formal. Por ejemplo:

help / display help

La transición interna *help* representa un evento que no causa ningún cambio de estado, pues sólo muestra, en cualquier momento, una ayuda al usuario que viene expresada en *display help*.

Eventos: Un evento es una ocurrencia que puede causar la transición de un estado a otro de un objeto. Esta ocurrencia puede ser una:

- Condición que toma el valor de verdadero (normalmente descrita como una expresión booleana). Es un Evento Cambio.
- Recepción de una señal explícita de un objeto a otro. Es un Evento Señal.
- Recepción de una llamada a una operación. Es un Evento Llamada.
- Paso de cierto período de tiempo, después de entrar al estado actual, o de cierta hora y fecha concretas. Es un Evento de Tiempo.

Acción: Una acción es una operación atómica, que no se puede interrumpir por un evento y que se ejecuta hasta su finalización. Una acción puede ser:

- Una llamada a una operación (al objeto al cual pertenece el diagrama de estado o también a otro objeto visible),

- La creación o la destrucción de otro objeto,
- El envío de una señal a un objeto.

Un punto de comienzo, el estado inicial, que se dibuja mediante un círculo sólido relleno, y un (o varios) punto de finalización, el estado final, que se dibuja por medio de un círculo conteniendo otro más pequeño y relleno.

Una transición simple es una relación entre dos estados que indica que un objeto en el primer estado puede entrar al segundo estado y ejecutar ciertas operaciones, cuando un evento ocurre y si ciertas condiciones son satisfechas.

Una transición simple se representa gráficamente como una línea continua dirigida desde el estado origen (*source*) hasta el estado destino (*target*). Puede venir acompañada por un texto con el siguiente formato:

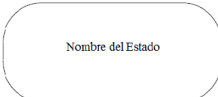
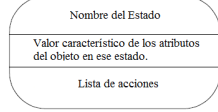
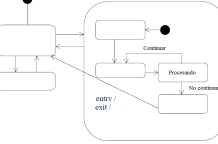
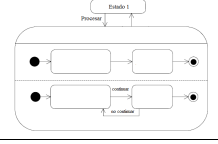
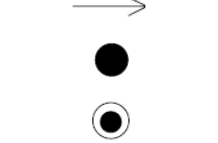
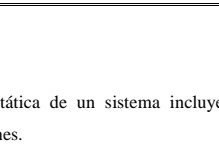



nombre-evento ‘(‘lista-argumentos‘)’ ‘[‘guard-condition‘]’ ‘/’ expresión-acción ‘^’ cláusula-envío

Donde “nombre-evento” y “lista-argumentos” describen el evento que da lugar a la transición y forman lo que se denomina *event-signature*, y *guard-condition* es una condición (expresión booleana) adicional al evento y necesaria para que la transición ocurra. Si la *guard-condition* se combina con una *event-signature*, entonces para que la transición se dispare tienen que suceder dos cosas: debe ocurrir el evento y la condición booleana debe ser verdadera. El campo “expresión-acción” es una expresión procedimental que se ejecuta cuando se dispara la transición. Es posible tener una o varias expresión-acción en una transición de estado, las cuales se delimitan con el carácter “/”. Y, “cláusula-envío” es una acción adicional que se ejecuta con el cambio de estado, por ejemplo, el envío de eventos a otros paquetes o clases.

Este tipo especial de acción tiene una sintaxis explícita para enviar un mensaje durante la transición entre dos estados. La sintaxis consiste de una expresión de destino y de un nombre de evento, separados por un punto.

Estados Avanzados: Un estado simple es aquel que no tiene estructura. Un estado que tiene subestados, es decir, estados anidados, se denomina estado compuesto. Un estado compuesto puede contener bien subestados secuenciales (disjuntos) o bien subestados concurrentes (ortogonales). En estos últimos se encuentran estados anidados en dicho estado pero separados por una línea discontinua.

Tabla 8. Elementos del Diagrama de Estados

Componentes	Elementos	Notación
Diagrama de Estados	Un compartimiento	
	Estado	
	Tres compartimientos	
	Subestados Secuenciales	
	Estado Avanzado	
	Subestados Concurrentes	
	Eventos	
	Punto de comienzo o estado inicial	
	Punto de finalización, el estado final	
	Elementos	

4.1.6 Diagrama de Clases

Los diagramas de clases sirven para representar la estructura estática de un sistema incluyendo una colección de elementos de modelización estáticos, tales como clases y relaciones.

Respecto a los elementos del Diagrama de Estados, presentados en la tabla 9, siendo sus definiciones las siguientes:

Clase: Una clase es la definición de objetos que comparten ciertas características estructurales y de comportamiento.

- Atributo: describe genéricamente una propiedad de los objetos de una clase (generalmente, describe hechos estáticos o estructurales).

- Operación: describe genéricamente un servicio que puede ser requerido a cualquier objeto de una clase para que muestre un comportamiento.

Relaciones: Se pueden establecer cuatro tipos de relaciones entre clases. A continuación se describen cada una de ellas.

- Asociación: describe una relación genérica entre objetos de clases. Contiene el nombre de la asociación. La multiplicidad que describe el número mínimo y máximo de enlaces posibles (<min..max>, 0..*, 1..*, 1..1, *, n..m, 1).
- Generalización: es una asociación entre una clase y otra más general de modo que la primera describe una subfamilia de objetos de esta última.
- Agregación: es una asociación que describe una relación entre un todo y sus partes de modo que las partes pueden existir por sí mismas.
- Composición: es una asociación que describe una relación entre un todo y sus partes de modo que las existencias de las partes se perciben como totalmente dependientes del todo.

Tabla 9. Elementos del Diagrama de Clase

Componentes	Elementos	Notación
Diagrama de Clase	Un campo	Nombre
	Dos campos	Nombre
		Atributos
	Clase	Nombre
		Atributos
		Operaciones
Relaciones	Asociación	<Nombre>
	Generalización	—>
	Agregación	—◇
	Composición	—◆

4.2 Estrategia de Mapeo para utilizar Modelos UML con SECTRA

Puesto que *UML* no tiene soporte directo para lenguajes de sistemas basados en microcontroladores de bajos recursos y el lenguaje C, se presenta una estrategia de mapeo que define las reglas para utilizar modelos *UML* con

sistemas embebidos basados en estos lenguajes. Los tipos de diagramas primarios definidos en esta estrategia se detallan en la Tabla 10.

Tabla 10. Representación desde los diagramas UML a C u otro lenguaje de microcontroladores de bajos recursos.

Diagramas básicos UML	Diagramas basados en C UML	Descripción
Diagrama Caso de Uso	Diagrama Caso de Uso	Representan los casos de uso del sistema con respecto a los actores. Crean un archivo .c y uno .h correspondiente a cada caso de uso.
Diagrama de Componentes	Diagrama de Construcción	Muestra el conjunto de artefactos construidos desde el archivo fuente, tales como ejecutables y librerías.
Diagrama de Clases	Diagramas de Petición o Llamadas	Muestra las llamadas o peticiones y sus secuencias entre un conjunto de funciones.
Diagramas de Secuencia	Diagramas de Mensajes	Representan secuencias de llamadas y envíos de eventos entre un conjunto de archivos, incluyendo valores por paso de parámetros.
Diagramas de Estado	Statechart	Muestran la máquina de estado para un archivo y cuando sus funciones son incluidas, así como también, las acciones que son ejecutadas como eventos son recibidos.
Diagrama de Actividades	Diagrama de Flujo (Flowchart)	Detalla el flujo de control para una función o unidad funcional (tal como el caso de uso).

Cada caso de uso de los diagramas de casos de uso puede describir dos archivos de C (uno.c y otro archivo de cabecera.h), cada caso de uso define un comportamiento dando un resultado observable. Los diagramas de secuencia identifican la secuencia de llamadas a las funciones. En el caso de que un objeto del sistema puede tener diferentes estados, tal como se especifica en la Máquina Estado correspondiente, que puede ser descrito utilizando un *statechart UML*. El Diagrama de Actividad en *UML* puede representar el diagrama de flujo para un uso función. Disparadores de eventos en esos diagramas se implementan mediante la activación de los temporizadores, las entradas del usuario, las interrupciones, etc. (Wang, 2009).

Se resalta el proceso *tarea* con el propósito de tener funcionalidades con características tanto lógicas como temporales dentro de la EGS. Posee todas las propiedades funcionales de un componente, con la diferencia de poseer también, atributos y características propias del concepto de tareas de tiempo real, donde se encuentran el tiempo de cómputo (*Ci*), el plazo de finalización (*Di*), el período (*Ti*) y el tiempo de liberación (*ri*).

4.3 Definición de los requisitos del sistema

La primera etapa tiene que ver con la identificación general de las características en las que se debe desarrollar el sistema, estas son: sistema hardware, entorno de operación, tareas a realizar, consideraciones especiales, arquitectura general. Las pautas que se sugieren para la definición de requisitos se resumen a

continuación y pueden ser ampliadas o modificadas de acuerdo a las características requeridas por cada usuario; no obstante, se presentan las que cumplen con la mayoría de necesidades que se requieren a este respecto.

Identifique las necesidades de diseño del SE para llevar a cabo sus funciones y bajo qué limitaciones operará. Tiene como propósito visualizar y describir el sistema, mediante un levantamiento de los diferentes equipos y dispositivos, tanto de hardware, software y de instrumentación. El resultado de este paso será un conjunto de requisitos del sistema que describen lo que hace el sistema y puede también dividir los requisitos entre el hardware y el software. Las restricciones en un sistema embebido de tiempo real suelen incluir tiempo, capacidad de memoria y procesador de rendimiento.

Actores

Se entiende por actores, aquellos elementos Software/Hardware que interactúan con el sistema o personas, en este punto, se trata de definir con más nivel de detalle, los actores del paso anterior e identificar otros no contemplados.

A continuación se presentan los actores elementales con las cuales el sistema propuesto tendrá interacción:

- Dispositivos de Arranque compuesto por el Pulsador de inicio o dispositivo que emite una señal de control que indica el inicio de operaciones y, el Pulsador de reinicio, dispositivo que emite una señal de control externa que indica al sistema que se debe reiniciar desde el comienzo la ejecución del programa.
- Sensores: tienen la finalidad de suministrar al sistema la magnitud de la variable de la planta o proceso del lazo de control asociado; esta es proporcionada en forma continua en el tiempo.
- Actuadores: elementos finales de control que modifican las variables del proceso del lazo de control asociado, según la respuesta que le proporcione el sistema.

Las entidades externas que se nombran son opcionales dentro de la estructura general, ya que dependerán de la aplicación específica que se vaya a desarrollar y del propósito de la misma:

- Punto de consigna (i): es la entidad correspondiente de proporcionar al sistema el valor deseado en estado estacionario de las variables a controlar en los diferentes lazos de control. Este dato es suministrado por el usuario. A pesar de suministrar un dato indispensable para correcto funcionamiento del sistema, la existencia o no de esta entidad externa dependerá de las características del proceso, ya que si éste no requiere de constantes modificaciones en el punto de consigna durante su funcionamiento, puede reservarse un espacio en la memoria del sistema para almacenar estos datos, y reducir la complejidad del sistema.
- Reloj del sistema: encargado de llevar el histórico del tiempo de ejecución del sistema, y es proporcionado de forma continua en el tiempo. Su presencia dentro de la estructura global va a depender de la política de planificación que se decida implementar, como por ejemplo aquellas donde las prioridades de las tareas varían en forma dinámica.
- Temporizadores: generan interrupciones periódicas al sistema cuando se cumple el tiempo para el cual fueron programados. Sirven como apoyo para el cumplimiento de los requerimientos temporales, y la

presencia de éstos dentro del sistema también va a depender de la política de planificación seleccionada y las características temporales de las tareas.

Desarrollo de los casos de uso.

Con los requisitos del sistema como punto de partida, se desarrollan los casos de uso que cubran estos requisitos. Los casos de uso ilustran las comunicaciones entre un SECTRA y los actores externos. Los casos de uso para sistemas de tiempo real también deben definir plazos y requisitos de sincronización. Resulta en un diagrama general, el cual ofrece una primera visión que especifica el sistema a desarrollar.

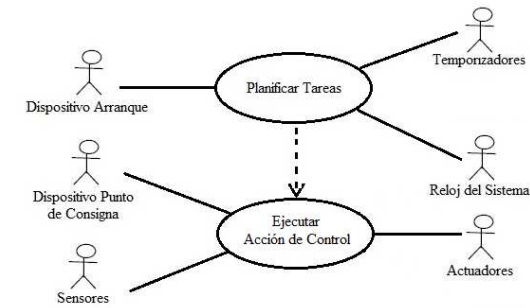


Figura 12. Modelo Casos de Uso mikro_STR con extensión UML.

Diagrama de componentes

Con el fin de tener un nivel de organización del modelo, se agruparon las clases en los paquetes: Medición, Aspectos de calidad y Dominio de Aplicaciones.

Diagrama de clases

Definir estructura del objeto. Una vez que los casos de uso se han definido, la estructura del objeto puede ser definido. La estructura del objeto de un sistema embebido en tiempo real incluye la definición de las clases de objetos y los datos de cada clase de objeto que contendrá. Los diagramas de clases y diagramas de objetos pueden ser utilizados para modelar la estructura del objeto.

Diagramas de Estado, Diagrama de Secuencia

Definir el comportamiento del objeto. Los objetos en el diseño de su estructura de objetos tendrán comportamientos que corresponden a la funcionalidad necesaria para los casos de uso. Comportamiento de los objetos puede ser modelado a través de **diagrama de estado y diagramas de secuencia**.