µ-Framework: Una Visión Actual para el Desarrollo de Sistemas Embebidos

Ing. Alejandro Fourcade Mg. Jorge Eterovic Ing. Alejandro Pérez Ing. Guillermo Rodofile
Departamento de Ingeniería - Universidad de La Matanza
afourcade@unlam.edu.ar eterovic@unlam.edu.ar aperez@unlam.edu.ar
hurodofile@unlam.edu.ar

Resumen

Los paradigmas de diseño de sistemas embebidos han sufrido en los últimos tiempos cambios fundamentales. Los requisitos básicos, los alcances y límites de un sistema integrado se han expandido y a la vez, se han impuesto funcionalidades que hoy resultan casi obligatorias y están fuera del radar de los sistemas de desarrollo tradicionales.

Las nuevas tecnologías, plantean nuevas visiones de los problemas tradicionales y los sistemas embebidos se han mantenido mayormente al margen de los métodos actuales de desarrollo de software.

Este trabajo propone un marco referencial llamado µ-Framework para el desarrollo de sistemas integrados, teniendo en cuenta los nuevos límites operativos y funcionales que los desarrollos modernos requieren.

1. Introducción

El concepto de IoT (Internet of Things) y sus adaptaciones a diferentes escenarios IIoT (Industrial IoT) y EIoT (Enterprise IoT) cambian los enfoques sobre las prestaciones operativas que deben ofrecer hoy las soluciones tecnológicas. Al incorporar conectividad, adquisición masiva de datos, interoperabilidad, control y supervisión remota, tableros de control en la web, entre otras características, se impone una nueva forma de toma de requisitos, análisis de alcances e implementación.

Otro factor fundamental en la evolución del diseño de sistemas embebidos es que el crecimiento explosivo de la oferta plataformas de desarrollo, ha abaratado sustancialmente los costos y facilitado el ingreso a la programación de microcontroladores.

Las metodologías de desarrollo de software tradicionales (Agile, Cascada), tienen un enfoque centrado en el software, con un alto nivel de abstracción que aleja al código del hardware que lo ejecutará. En sistemas embebidos existe una fuerte dependencia entre software y hardware, aun si se trabaja en lenguajes de alto nivel. Por razones de rendimiento, memoria, consumo y proceso, el firmware tiene parámetros de diseño, totalmente diferentes que el software que se ejecutará en un sistema ya existente, previsible y compatible.

Timo Punkka en su trabajo Embedded Agile[1] concluye que para implementar el método Agile en sistemas embebidos es necesario adaptar varias funciones. Entre ellas, desarrollos incrementales de lapsos cortos, ciclos de inspección y adaptación continuos y una cultura del aprendizaje constante. Estas modificaciones permiten introducir, aunque sea parcialmente, al hardware dentro del ciclo de desarrollo.

Además de los factores expuestos que distancian a los sistemas integrados de los métodos de desarrollo de software para plataformas tradicionales, se propone desde µ-Framework el concepto de la utilización de hardware sustentable. El universo IoT ofrece hoy infinidad de desarrollos intermedios de hardware con extenso soporte de bibliotecas y código. Estos módulos de hardware de uso múltiple acercan el objetivo de cumplir los lineamientos IoT en cuanto a bajo consumo, transferencia de datos, comunicaciones y eficiencia de código. Los fabricantes ofrecen soporte y aportan notas de aplicación con ejemplos de uso para tratar de popularizar sus productos. En el mercado se encuentran además una serie de sensores, transmisores, interfaces, displays compatibles y preparados para estas plataformas, lo que facilita en extremo el armado del hardware, más que nada en la etapa de prototipo.

Este concepto de generar diseños con módulos creados para operaciones no específicas no es nuevo. El reglamento llamado FAR (Federal Adquisition Regulations)[2] utilizado para adquirir materiales para entes gubernamentales en los Estados Unidos de América, prevé la adquisición de elementos genéricos de fabricación comercial para usos no críticos del ámbito militar. Esta figura se denomina COTS (Commercial Off The Shelf) y permite la compra de elementos NDI (Insumos No Desarrollables) que se comercializan regularmente a menor costo que los fabricados específicamente para uso militar. Entre las ventajas que aporta esta mecánica de adquisición están la rapidez, la actualización tecnológica constante, además del factor económico. Otro punto positivo por considerar es el mantenimiento más sencillo y menos especializado, tanto de software como de hardware.

Un factor que ha tomado preponderancia en los últimos años en los diseños integrados, es el de la seguridad. Es imprescindible tener en cuenta que, al conectar un sistema a

la nube, a internet o a una red (tanto LAN (Local Area Network) como WAN (Wide Area Network)), se lo expone a riesgos potenciales y crecientes de seguridad, aunque el dispositivo sea una simple cafetera. Es por esta razón que están comenzando a ofrecerse plataformas de desarrollo IoT con seguridad certificada, tanto software como hardware, por ejemplo, Microsoft a través de Azure IoT Edge[1].

Teniendo en cuenta los factores citados, µ-Framework propone un *road map* a seguir para obtener modelos embebidos funcionales compatibles con los nuevos lineamientos tecnológicos. Este marco conceptual incluye varias disciplinas como se muestra en la Figura 1.

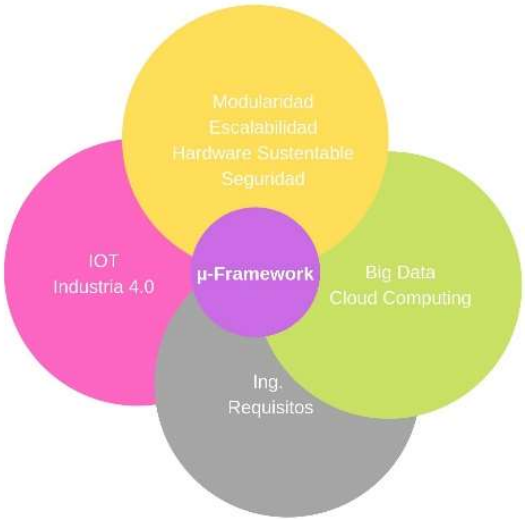

Figura 1. Conceptos y disciplinas

2. Elementos de trabajo y metodología

Como se ha mencionado µ-Framework integra diferentes disciplinas que impactan desde lo formal hasta lo técnico en el proceso de diseño.

Algunos de los lineamientos se han tomado de Agile, por ejemplo, el de mejora continua y de MVP (Mínimo Producto Viable). MVP resulta interesante para ordenar los pasos del desarrollo, mostrar efectividad y aceitar la cooperación de los clientes visionarios o "earlyvangelists"[2] en el proceso de diseño. Estos usuarios claves, convencidos de las bondades del producto, son normalmente más tolerantes y predispuestos a dar la realimentación de los resultados de sus experiencias. Si bien los principios mencionados son más aplicables a sistemas informáticos, se aplican también a los sistemas embebidos fortificar y actualizar la metodología de desarrollo. La Figura 2 muestra las entradas y salidas finales de µ-Framework y los preceptos que toma como guías principales.

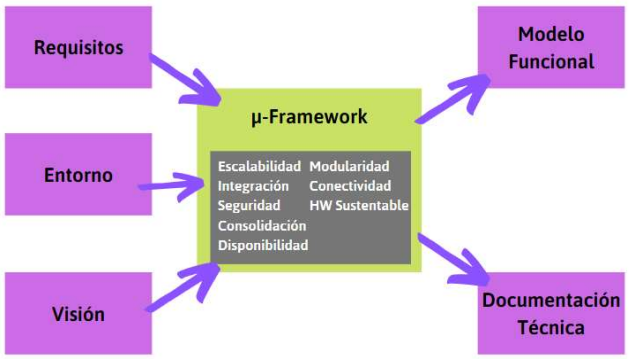

Figura 2. Entradas y Salidas

Las reglas de diseño de los sistemas integrados deben incluir a las nuevas tecnologías en los relevamientos, como puntos deseables y a veces indispensables.

Otro factor que se ha modificado es que actualmente la generación de un producto personalizado se ha vuelto menos frecuente. Entre las razones de esta disminución se encuentran las numerosas variantes de desarrollos y accesorios que se ofrecen en el mercado, cubriendo muchas de las necesidades que anteriormente generaban un desarrollo a medida. Dentro de las ventajas que otorga la estandarización del hardware y software de automatización, está la de no dejar en manos de proveedores únicos la operación de procesos críticos. Hace años era normal que el hardware que manejaba un proceso de fabricación hubiera sido diseñado especialmente y si el proveedor decidía no dar más servicio, comprometía la continuidad de la línea. En cambio, hoy en general, los procesos críticos están controlados por módulos comerciales no específicos, asegurando soporte y confiabilidad a través del tiempo.

Esta expansión en la oferta de opciones técnicas y la estandarización de software y hardware en pos de la operación y mantenimiento más seguros han influido en la forma de pensar un sistema. Los desarrollos actuales deben prever la integración, tanto horizontal como vertical (con sistemas pares o sistemas maestros).

Se han citado, a modo de ejemplo, dos factores que impactan en la forma de planear un desarrollo. El entendimiento de que un sistema actual no debe ser una solución aislada es fundamental. Es necesario llevar al mínimo las debilidades y apoyarse en las fortalezas para generar un producto preparado y permeable a nuevos requerimientos y principalmente a los nuevos requisitos de comunicación.

Un factor central para que el producto generado tenga las capacidades citadas, es que desde el instante cero del desarrollo, se prevean vías de expansión, conexión e integración. Una estrategia para cumplir ese objetivo es dividir la topología de diseño en bloques funcionales con entradas y salidas estandarizadas.

Como se verá a continuación la ingeniería de requerimientos cumple un papel protagónico en generar los

[1] https://azure.microsoft.com/es-es/services/iot-edge/

[2] Según Steve Blank en su libro "The Four Steps to Epiphany", usado como sinónimo de earlyadopter o cliente faro.

lineamientos y alcances, tanto tecnológicos como operativos, que definirán los bloques y su interacción.

Para explicar este marco conceptual, se propone recorrerlo a través de un caso de estudio: un banco de ensayos de motores.

2.1 Pasos para el desarrollo

Se propone a continuación una guía para el diseño de sistemas embebidos, aplicándolos luego a un caso de estudio.

Paso 1: LEL y escenarios

El primer paso es familiarizarse con el léxico y la terminología del entorno del proyecto. Tal como lo propone la Ingeniería de Requisitos tradicional el primer paso es generar el Léxico Extendido del Lenguaje (LEL)[3] e identificar y describir los escenarios.

El LEL es una especie de diccionario que contiene los términos utilizados en la jerga técnica y coloquial del lugar donde se realizará el relevamiento. El aporte del LEL fundamental en la incorporación del lenguaje natural que ayudará a reconocer el problema y facilitar así su modelización. Es en suma un tamiz del universo discursivo y nos ayudará a que la elicitación de requerimientos sea precisa.

Los escenarios en cambio definen entornos, protagonistas y roles. El contexto en que se define el problema, cuales son los actores que interactúan y los recursos con que se cuenta.

Aplicación al caso de estudio

En el caso de uso seleccionado, el banco de ensayo de motores, va a ser esencial saber interpretar términos como: *torque, revoluciones, curva de rendimiento, freno hidráulico, celda de carga, presión de aceite, intercambiador de calor, banco de rodillos, banquear, patas del motor, ensayar un motor, etc.* Existe una infinidad de palabras utilizadas para describir mediciones, elementos y operaciones aplicables a un banco de ensayos de un motor.

Los escenarios en que se desenvolverá el proyecto, contienen dos tipos de actores: ingenieros mecánicos, encargados de la parte de estructura e implementación mecánica, e ingenieros electrónicos que cumplimentarán los requisitos de adquisición y administración de datos, supervisión, control y conectividad. No está definido claramente el alcance en cuanto a lo técnico/funcional, en primera instancia debe interactuar con un motor o un vehículo y no prevé la posibilidad de cambiarlo rápidamente, como en un banco de motores comercial.

En resumen, en cuanto al LEL, existe una gran cantidad de léxico especializado que es necesario interpretar precisamente para que la operación de elicitación sea válida. No solo para comprender las necesidades sino para evaluar el costo beneficio de desarrollos adicionales que se incorporarán al MVP.

En lo que respecta a los escenarios, para encontrar usuarios claves o especializados se ha recurrido a otras universidades y empresas con instalaciones equivalentes. Se ha consultado con especialistas en electrónica automotriz para conocer los protocolos de intercambio de información, sus formatos y versiones y con fabricantes comerciales de sistemas de ensayo mecánico, para evaluar el tipo de estructura necesaria (banco o rodillos), las características de intercambio térmico para una instalación fija, el tipo de freno y su dimensionamiento.

Como no se cuenta con usuarios claves, se optó por formar referentes dentro del equipo que puedan guiar el desarrollo basándose en el estado del arte, evaluando opciones de mínima y máxima en cuanto a las prestaciones y dimensionando el alcance de la primera etapa del proyecto.

Un resumen gráfico de lo referido se puede ver en la Figura 3.

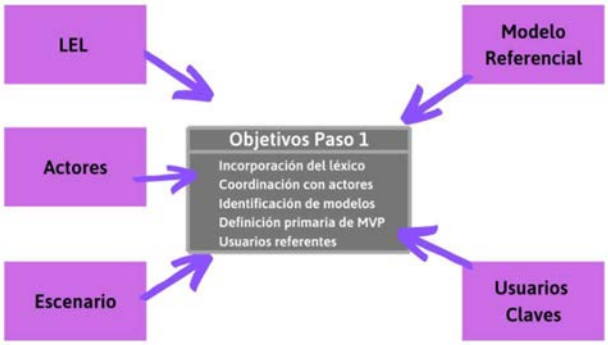

Figura 3: Objetivo del paso 1

Paso 2: Requisitos: tipos y niveles

El objetivo del paso 2 es obtener un diagrama de entradas y salidas y una lista de requisitos funcionales y no funcionales.

Los requisitos que definen un proyecto de implementación, son los que especifican las acciones que deberá ejecutar el sistema. La clasificación de estos requisitos, tal como indica la ingeniería de requisitos clásica, los divide en funcionales o no funcionales. Los funcionales tienen que ver directamente con las aptitudes del sistema, mientras que los no funcionales detectan necesidades colaterales, imprescindibles para el correcto desarrollo de las acciones descriptas en los funcionales.

Como los sistemas embebidos la interacción entre software y hardware es tan cercana es necesario definir diferentes niveles de abstracción. La separación en diferentes capas según las necesidades operativas, facilita la detección de los procesos a definir. Si se tomara como ejemplo un cajero automático, las acciones que espera que ejecute el cliente que lo opera, tienen que ver estrictamente con lo práctico (por ejemplo, hacer un depósito). Para completar esta acción, que se definirá como de Nivel 1, deben suceder otras, que se definirán como de Nivel 2: operaciones financieras (acreditar el depósito), mecánicas (leer los billetes), de seguridad (evitar fraude). Estas últimas, ya invisibles al usuario final, contienen también acciones de niveles superiores cada vez más específicas.

La correcta estratificación y jerarquización de los subprocesos que conforman el proceso final es fundamental

para los desarrollos embebidos. Si bien el postulado del manifiesto Agile pondera la capacidad de cambiar los requisitos en cualquier momento del proceso, en los sistemas integrados se hace complicado cumplir esa premisa. Por eso es indispensable que la división de tareas sea lo más detallada posible y se visualice el proceso como una cadena de subprocesos de diferente complejidad entrelazados y contenidos unos en otros.

Uno de los objetivos principales de la división de subprocesos en niveles (Ver Figura 4) es también prever y asegurar la integración y escalabilidad. En el ejemplo anterior del cajero automático, si se cambiara la lectora de tarjeta magnética por una que acepta también tarjetas con chip, solo debería cambiarse el módulo que maneja el hardware específico, si el entorno se encuentra normalizado (trama de datos / control / estado, líneas de comunicación, flags, alarmas, etc.). Seguir estos criterios de diseño puede ser la diferencia entre un costoso proyecto de upgrade y una actualización remota automática.

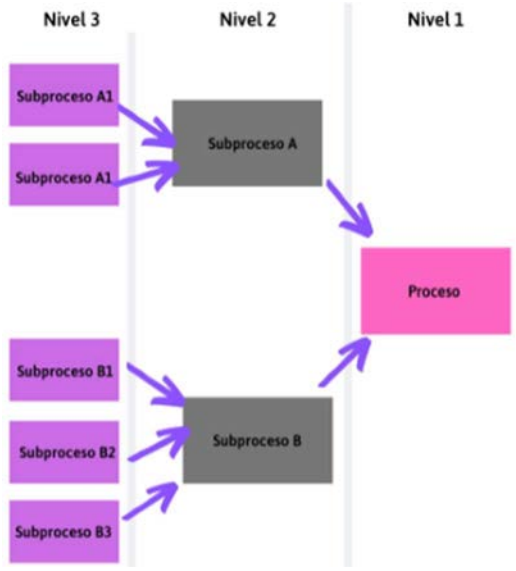

Figura 4. División de procesos

Aplicación al caso de estudio

Es necesario en primer lugar definir los requisitos funcionales que debe tener el sistema para analizar su factibilidad. Es fundamental para evitar frustraciones y no incluir premisas imposibles desde un comienzo. Las habilidades mínimas con que debe contar el producto para ser viable definirán el MVP y los mojones (milestones) serán los objetivos parciales a cumplir posteriormente. Las iteraciones (en Agile llamados Sprints) irán generando ciclos de entrega que corregirán el MVP e irán sumándole funcionalidades.

Aplicándolo a nuestro caso de estudio los requisitos funcionales básicos del sistema son:

- El sistema deberá tomar los datos de un motor a explosión de un automotor, almacenarlos y procesarlos.
- Los datos se obtendrán durante el ensayo, algunos se mostrarán en tiempo real en un monitor y otros se procesarán luego.
- Los datos principales son RPM, TORQUE, TEMPERATURA DEL BLOCK, CONSUMO DE COMBUSTIBLE, TEMPERATURA DE GASES DE SALIDA.
- El sistema deberá adquirir datos de varias fuentes posibles (Sensores del motor, sensores externos, tramas digitales (OBD2, CAN)).
- El sistema deberá contar con un botón de parada de emergencia.
- El sistema deberá operarse en forma automática, semiautomática o manual.

Los requisitos no funcionales serían:

- El error en muestras analógicas debe ser menor al 2%.
- El lapso mínimo entre muestras deberá ser de 1 ms.
- El sistema, en lo que a adquisición de información se refiere, se calificará como no crítico, por lo que no es necesario prever tolerancia a fallos.
- Se deberá prever almacenamiento en disco externo, tarjetas de memoria, pen drives y repositorios web.

Dada esta toma de requisitos inicial, se analiza con los medios con que se cuenta actualmente para planificar la dirección del desarrollo. Si bien no se cuenta todavía con un motor, se cuenta con un vehículo al cual se le pueden realizar mediciones y pruebas no invasivas o destructivas. Para eso, deben tomarse datos de su trama digital (puertos OBD2 y CAN) y pueden agregarse sensores.

En base a la información con que se cuenta, se puede comenzar a delinear un MVP conectado a la trama digital que almacene información para posterior análisis y muestre en tiempo real RPM, TORQUE y CONSUMO. Para terminar de definir el MVP es necesario determinar las interfaces.

Paso 3: Definición de interfaces

Más allá del resultado de los requisitos funcionales la especificación de las interconexiones internas y externas del sistema en entornos embebidos merece un tratamiento especial. El camino para definir las necesidades de interconexión de un sistema se puede dividir en:

- *Interfaces operativas*
- *Interfaces de expansión*
- *Interfaces a redes*

Interfaces operativas

Las interfaces operativas son las que tienen que ver con las necesidades actuales del sistema. Las conexiones a otros sistemas o repositorios existentes. Estos sistemas se clasifican a su vez en:

- *Sistemas pares*
- *Sistemas maestros*
- *Sistemas esclavos*

Los sistemas pares son aquellos iguales o de características similares, que comparten funcionalidades del mismo nivel. Por ejemplo, en el caso de una inyectora de plástico, podría ser la relación con otra inyectora similar para

compartir un solo repositorio de datos, o un mismo sistema de alimentación de materia prima.

Los sistemas maestros son los que administran y manejan nodos de producción, volviendo al caso de las inyectoras de plástico un tablero de control que consolide la información de varias inyectoras, informando de sus funcionamientos en forma centralizada.

Los sistemas esclavos son sistemas cuya operación depende del sistema maestro. En el ejemplo utilizado, sería el control de una tolva que puede abastecer a la inyectora de materia prima.

Interfaces de expansión

Las interfaces de expansión son aquellos que teniendo en cuenta la posibilidad de que el sistema necesite sumar funcionalidades actuales o futuras, deja abierta la posibilidad de conexión a través de una vía normalizada. Volviendo al ejemplo de la inyectora, podría ser un puerto paralelo para generar reportes de fin de jornada en impresoras de matriz de punto, o un puerto HDMI para conectar un monitor extra.

Interfaces a redes

Los interfaces a redes son los que proporcionan salidas a redes de datos (PAN, LAN o WAN). Pueden ser redes Ethernet, Bluetooth, Zigbee, RF y son configurables. En general se conectan a un nodo que oficia de repositorio de datos o a la web a repositorios en la nube. Estas opciones también incluyen Big Data, Cloud Computing o Almacenamiento en la nube.

La información en estos casos tiene mayor volumen que en el caso de las interfaces anteriores. El mayor ancho de banda y las posibilidades de almacenamiento en la nube se convierten en herramientas poderosas para el procesamiento estadístico o inteligencia de negocio (BI).

Un concepto fundamental a la hora de especificar las vías de comunicación que tendrá el sistema con el exterior es que la integración, y la conectividad permitan generar información de producción o control disponible en la web. Estas potencialidades tanto operativa como comercialmente son importantes puntos a favor de un diseño.

Paso 4: Definición de entradas, salidas, mensajes y protocolos

El mapa mental de nuestro desarrollo, en este punto debería comenzar a materializarse. Los escenarios, actores y requisitos deberían comenzar a definir los alcances del MVP. Pero para asegurar la capacidad del proyecto de ir creciendo tanto horizontal como verticalmente es necesario definir las formas y lenguajes de comunicación. Se debería pensar el proyecto como un conjunto de módulos o tomando el concepto electrónico, cuadripolos conectados entre sí estratégicamente para cumplir una función. Para ayudar a generar el mapa de módulos de un proyecto es necesario dividirlos en diferentes categorías según su ubicación y función.

Según su ubicación:

- Módulos de borde
- Módulos internos

Y según su función:

- Módulos de control
- Módulos de proceso
- Módulos de operación
- Módulos de estado
- Módulos de entrada/salida

Cada uno de ellos cuenta con características propias y dividirlos es una forma metódica de comenzar a definir sus funciones dentro del proyecto.

Para caracterizar un módulo según su ubicación se deberá preguntar: ¿Qué contacto tiene el módulo con el exterior? / ¿Entrega o toma datos/señales/comandos? La importancia de estas preguntas simples es determinar qué módulos son las puertas de entrada y salida de información.

A su vez los protocolos de comunicación se clasifican en internos y externos. Los protocolos internos comunican los módulos entre sí, pero no salen al exterior. Los externos son los que determinan las opciones de conectividad del diseño.

Esta clasificación diferencia no solo la función y puntos de conexión de los protocolos sino sus características eléctricas, operativas y la compatibilidad. Ejemplos de protocolos internos son: SPI, I2C, Serie, PWM, Paralelo. Estos protocolos conforman las redes interiores del proyecto y en muchos casos forman los buses de comunicación que unen los módulos internos. La capa física de estos protocolos se adapta a las características del bus interno del sistema (5V o 3,3 V, por ejemplo).

Los protocolos externos, no unen módulos entre sí, sino que son los que le dan las opciones de conectividad al proyecto. Algunos ejemplos de normas/protocolos externos son: Modbus, Ethernet, USB, Zigbee, CAN, Profibus, Bluetooth, WIFI.

Los protocolos externos poseen normas eléctricas y mecánicas que definen su implementación física por ejemplo MODBUS puede implementarse en RS-232, RS485, RS422 o Ethernet (IEEE803.3) sobre 10BaseT o 10Base2.

La necesidad de categorizar los protocolos radica en comprender las necesidades de conectividad internas y externas del proyecto. Un módulo diseñado para el ambiente industrial podría prescindir de comunicación Bluetooth, pero no de comunicación USB o el RS232. El tipo de protocolo de conexión de cada módulo, también los clasifica ya que aquellos que tengan en su entrada y salida protocolos interiores serán módulos internos, mientras que, si tienen protocolos externos en su entrada o salida, serán módulos de borde.

Los mensajes enviados entre los módulos internos son de cuatro tipos diferentes:

- Mensajes de Estado
- Mensajes de Control
- Mensajes de Datos
- Mensajes de Alerta

Los mensajes de estado son los que informan sobre el funcionamiento de los módulos internos. Por ejemplo, entran en esta categoría los mensajes de Ocupado / Disponible que sería la respuesta de un módulo al recibir un mensaje que pida información sobre su disponibilidad.

Los mensajes de control son los que disparan acciones dentro del sistema e indican la necesidad de poner algún proceso en marcha. Por ejemplo, un mensaje para comenzar

la recolección de datos. Los mensajes de alerta son mensajes de control de alta prioridad, que deberían transmitirse en los canales de mayor velocidad.

Los mensajes de datos son aquellos que ingresan, envían o intercomunican información. Estos mensajes pueden conectar el proyecto con el exterior o ser entre módulos internos. Aquí se puede obtener otra clasificación complementaria que resulta útil fundamentalmente en topología complejas de diseño, que es la de mensajes de I/O o mensajes inter modulares. Diferenciarlos colabora con un correcto diseño del formato de la trama. Mientras que los mensajes de I/O deben tener en general un formato normalizado ya que interactuarán con actores externos al diseño, los inter modulares pueden tener una trama propietaria.

Un ejemplo claro de esta clasificación, aplicable al caso de uso seleccionado, se da en los automóviles actuales que tienen varios buses CAN con tramas propietarias que comunican sus numerosos módulos internos y mensajes normalizados para sus salidas ODB2.

La definición de módulos según su función es fundamental a la hora de aislar y asignar tareas. Cuando se define el MVP, debe considerarse cuales son los caminos para escalar el proyecto y llevarlo a su alcance final. Para ejecutar esa tarea correctamente es necesario segmentar las operaciones, asignarlas a diferentes módulos y determinar correctamente los protocolos de entrada y salida, para que, de esa forma, permita el desarrollo por capas sin necesidad de rediseñar constantemente las anteriores.

Para continuar con la clasificación de los módulos, los de control se definen como aquellos que no actúan sobre el sistema, sino que lo supervisan y generan mensajes que provocarán acciones en los módulos de proceso y estado.

Los módulos de proceso son los que realizan análisis y ejecutan procedimientos que informan a otros módulos de operación sobre acciones a realizar, por ejemplo, un módulo que cada 10 ms. ordena actualizar un dato en un display.

Los módulos de operación son los que realizan operaciones específicas, reciben órdenes para ejecutar tareas. Por ejemplo, el módulo que escribe en el display del ejemplo anterior.

Los módulos de estado son aquellos que disponen de la información de cómo se encuentran diferentes dispositivos y variables interiores y exteriores. En algunos diseños más pequeños se fusionan con los módulos de control. Por ejemplo, los módulos de estado consultan la temperatura, nivel de tensión, consumo y consolidan los datos para determinar el nivel operativo de la unidad.

Los módulos de entrada/salida son los que comunican el diseño con el exterior intercambiando datos y conectando líneas de control y señalización.

En un diseño, del punto de vista de la codificación los módulos pueden ser una subrutina, un conjunto de subrutinas y funciones, una biblioteca, o un programa independiente. Lo importante es tener definidos los alcances de cada módulo y que cumpla con las reglas impuestas de formatos de entrada y salida.

En la Tabla 1 se dan algunos ejemplos de módulos, según lo descripto anteriormente.

Tabla 1. Tabla de módulos

Tipo	Función	Ejemplo
Control	Supervisión	Watchdog
Proceso	Decisión	Cruce por cero
Operación	Ejecución	Conversor A/D
Estado	Información	Estado de bomba
I/O	Intercambio	Botón de parada

En la Figura 5 se grafican los modelos de un conjunto de módulos que interactúan en un diseño.

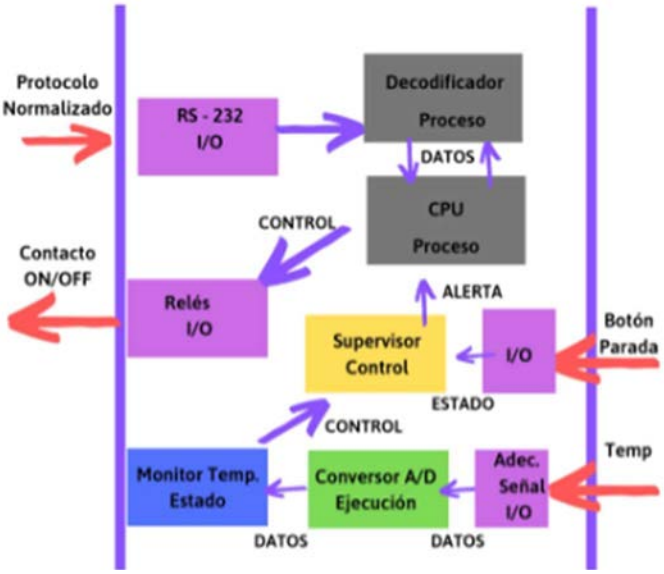

Figura 5. Procesos y mensajes

Aplicación al caso de estudio

Para el banco de prueba de motores, es necesario especificar primero la CPU, una unidad central de proceso que tomará datos aportados por diferentes módulos y los mostrará y exportará en batch o tiempo real. Eso implica la definición de una primera capa de proceso que tendrá como protocolo de entrada y salida I2C, un protocolo multimaestro y multiesclavo, que resulta ideal como transporte para buses internos. Si bien en general los desarrollos embebidos están más cerca de la arquitectura Harvard (que Von Neumann) porque el almacenamiento del programa es interno, la clasificación que se ha propuesto se basa específicamente en lo funcional. Entonces el primer módulo será de proceso, interno con entradas y salidas I2C. Como se ve en la Figura 6, se toman también como entradas y salidas los ports digitales y el interfaz serie asincrónico para conectar fundamentalmente módulos de borde. En caso de ser necesario puede habilitarse también el SPI sin ninguna penalidad, salvo la de dejar los pines libres, en previsión de futuras expansiones.

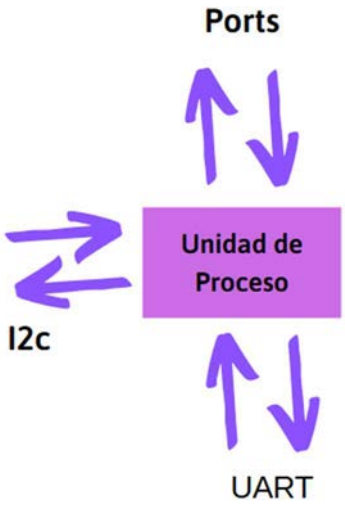

Figura 6. Módulo Central

Serán necesarios módulos de borde para recibir tramas CAN y decodificarla. En caso de que la CPU tenga entradas y salidas CAN (microcontroladores de la familia STM32, por ejemplo) sólo decodificarán las tramas de nivel superior y harán la adecuación de niveles eléctricos. En caso de no contar con ellas también codificarían a I2C. Se tomarán en cuenta las dos posibilidades (Ver Figura 7).

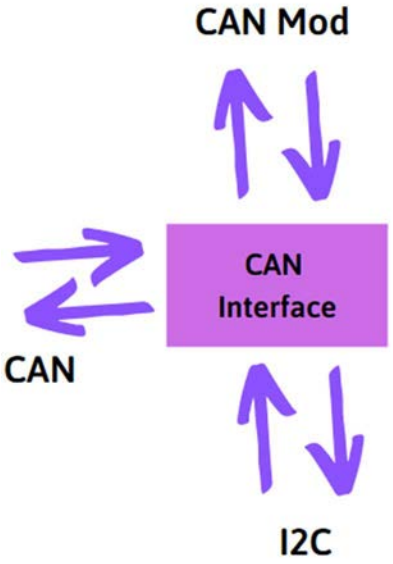

Figura 7. Módulo CAN de borde

Con los mismos criterios utilizados anteriormente se tendrán que generar los módulos de:

- Adecuación de señales
- Ethernet
- USB
- HDMI
- Display interno
- Tarjeta SD
- Control de parada emergencia
- Supervisor

Como no se tiene la certeza de cuáles serán las fuentes de datos a procesar, las definiciones de entrada y salida deberán ser amplias, dado que el mismo desarrollo se deberá poder usar en ambos escenarios.

Paso 5: Determinación del MVP

En este punto, es necesario tener algunos conceptos fundamentales en cuenta. Es diferente determinar el software mínimo que el hardware mínimo. Definir el MVP no es necesariamente definir los alcances del producto final, es solo una parte de ese proceso, un eslabón en la cadena. Para que esto pueda cumplirse es fundamental mantener y propiciar los conceptos de modularidad e integración, pero principalmente cumplir con las definiciones de estandarización de entradas, salidas y protocolos de comunicación, tal como se detalló en el apartado anterior.

Antes de comenzar a definir el MVP debemos imaginar el proyecto como una cebolla o una muñeca rusa, con capas concéntricas relacionadas entre sí. Cada capa deberá estar preparada para conectarse con la capa anterior y recibir las conexiones de la capa siguiente. Esto implica que la correcta definición de entradas y salidas y de las capacidades y velocidades de proceso en este paso son vitales.

La utilidad del MVP en Agile es entregar al cliente algo mínimamente operativo que irá mejorándose y completándose en los springs siguientes. En sistemas embebidos, si bien esa ventaja está presente, el MVP ayuda a resolver el problema del orden del desarrollo. **El objetivo es discernir lo fundamental de lo accesorio, sin perder de vista el desarrollo completo**. Como se mencionó anteriormente las modificaciones en sistemas embebidos no son tan sencillas por estar el software y el hardware estrechamente ligados, entonces al ir tomando decisiones es necesario ver la foto completa.

Una vez definidos los módulos, se evalúa cuales forman la primera capa de desarrollo, los más cercanos a la funcionalidad básica.

En general esa capa la ocupan las unidades funcionales básicas como la CPU y hardware necesario para el desarrollo, por ejemplo, los puertos USB. En la segunda capa se encuentran los periféricos y módulos de estado. En la tercera los que manejan las entradas y las salidas. En la cuarta los de presentación gráficas e interfaces.

Esta división, si bien subjetiva, ayuda a orientar la forma de poblar el modelo, pero muchas veces el orden lógico se ve modificado por razones presupuestarias, o de logística. Piezas que no llegan a tiempo, detenidas en la aduana o que por su precio todavía no han podido adquirirse. La gran foto del proyecto, también debe tener en cuenta esas vicisitudes.

Más allá del orden el MVP debe ser una parte del diseño que abarque los requisitos básicos y permita demostrar la efectividad del desarrollo, para generar confianza aportando resultados intermedios y delinear con más precisión las siguientes etapas del proyecto.

Aplicación al caso de estudio

El núcleo del desarrollo del banco de motores será la plataforma de desarrollo el ATMEGA 2560 junto con uno o más microcontroladores F103C8T6. Los productos comerciales que ofrecen estas alternativas son Arduino Mega y STM32 “Blue Pill”. Ambos cuentan con conexión USB para facilitar su programación y seguimiento de código. Lo principal en este caso es lograr tomar datos de manera consistente de un automóvil, ya sea de su bus digital de información, o a través de sensores existentes o complementarios.

Para terminar de definir el sistema mínimo, es necesaria la interfaz de datos. Para ello se utilizarán dos tipos de módulos: el conversor AD de 16 bits ADS1115 y la interfaz CAN (A1050 + MCP2515), que se encargarán de tomar información del bus CAN y de los sensores. En ambos casos son módulos de expansión disponibles en el mercado.

Los buses a utilizar son I2C en el caso de los conversores AD y SPI en el caso de la interfaz CAN.

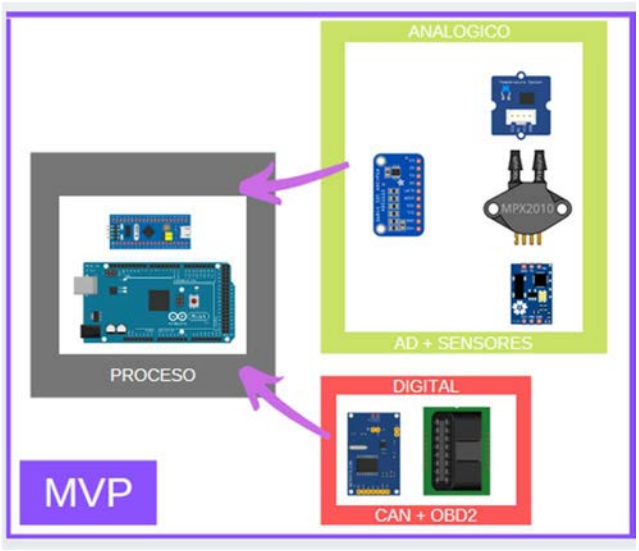

Figura 8. Módulos que forman MVP

Los módulos mostrados en la figura 8, constituyen la unidad funcional mínima entregable, pero además los módulos básicos de desarrollo. Una vez que el sistema es capaz de activar la unidad de proceso y tomar información analógica y digital, es necesario analizar, almacenar y consolidar la información.

Figura 9. Pruebas de mensajes CAN

En este momento (octubre 2018) se encuentra finalizada la capa de proceso y toma de datos. Para ello se desarrollaron varios submódulos que permitieron completar esta etapa, por ejemplo, un analizador de tráfico CAN y un generador de mensajes para manejar un tablero de instrumentos de control digital como se ve en la figura 9. Se utilizó un tablero de instrumentos porque genera y recibe mensajes CAN de varios tipos, es transportable y económico.

Cabe aclarar que para finalizar la primera etapa fue necesario visitar especialistas en talleres de electrónica automotriz, y realizar una trabajosa investigación ya que las implementaciones del protocolo CAN de buses internos son propietarias de cada modelo o marca.

Los detalles técnicos (circuitos, códigos, tramas, etc.) se incluirán en la tesis sobre µ-Framework que está en elaboración para presentar como trabajo final de la Maestría en Informática y se obvian en este trabajo por ser muy extensos.

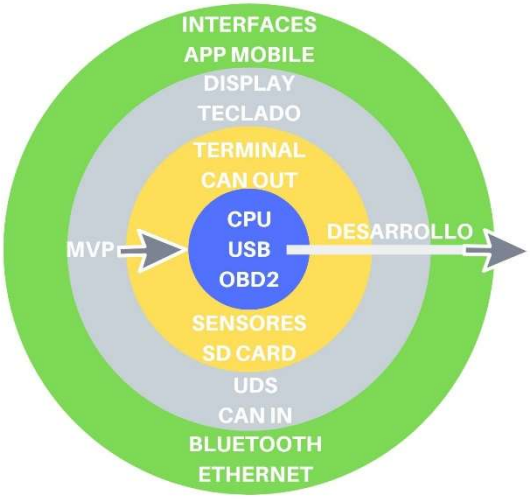

Figura 10. Capas de desarrollo

Paso 6: Iteraciones de desarrollo

Una vez alcanzado el MVP, se procede con el desarrollo de las capas siguientes considerando siempre el orden propuesto de antemano, respondiendo a una estrategia de implementación (figura 10). Aplicado al caso de estudio, la etapa siguiente a la generación del MVP incluye el almacenamiento, la comunicación de información operativa básica y la expansión de las fuentes de adquisición de datos.

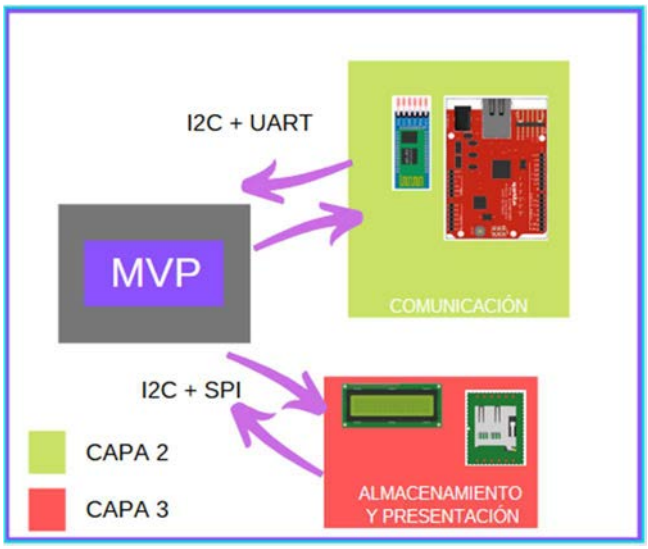

Figura 11. Iteraciones sobre el MVP

Con los sucesivos sprints se van agregando capas, en este caso las de comunicación y presentación. Los pasos que seguirían para completar el modelo, serían las capas de interfaces de administración y análisis de información (almacenamiento Web, exportación a Excel, etc.).

Es de observar que dentro de la metodología no se propuso ningún método de seguimiento de tareas en particular (PERT, GANTT), ni se refirió al control del orden de los subprocesos. Estas acciones son complementarias al marco de desarrollo y su uso queda librado a la experiencia del usuario. Tal como existen hoy herramientas para realizar GANTT con los términos de Agile, podrían usarse sin interferir en los pasos propuestos.

2.2 Hardware Sustentable

Al comienzo de este trabajo se especificaron de las ventajas que podía aportar el software sustentable. Tal como sucedía con el citado criterio COTS, evaluar la viabilidad de utilizar subconjuntos de hardware de fácil adquisición a la hora de diseñar; tomarse el tiempo para hacer un análisis de los diferentes caminos que están disponibles para lograr un desarrollo estable y con buenas perspectivas de soporte en el futuro, suele dar generosos frutos. El concepto de usar hardware selecto para generar productos difíciles de mantener y comprender ha caído en desuso y cada vez la apertura tecnológica es mayor. Actualmente la tendencia para pequeños desarrollos es utilizar el hardware más estable, confiable y económico.

En el excelente libro escrito en 1975 por Thomas Blackeslee [4] sobre diseño digital, se aborda no solo la importancia de la subdivisión de tareas en módulos funcionales, sino el arte de discernir los verdaderos retos del diseño. En su capítulo sobre división modular, inserta el concepto de "black box", bloques de proceso de los cuales sólo importa su función y comienza a establecer el concepto de hardware sustentable, refiriéndose a la correcta elección de los componentes.

Cuando en µ-Framework se refiere a hardware sustentable, significa utilizar subconjuntos que cumplan funciones del diseño, que sean de fácil adquisición, de amplio uso en el mercado, con rendimiento ampliamente probado. En los últimos años, en medio de lo que se dio llamar la "*democratización del conocimiento*"[3] en el campo de los microcontroladores, comenzaron a producirse innumerables piezas de hardware, listas para usar con sus bibliotecas asociadas y múltiples ejemplos de uso. Entre estas piezas hay sistemas de desarrollo (desde los más básicos a los más avanzados) e infinidad de periféricos y accesorios que se conectan fácilmente a estos sistemas.

Se pone así, a disposición del diseñador de baja escala, hardware testeado, con terminales para cablear y soldar. Esta característica facilita el armado y evita la manipulación de componentes pequeños y frágiles, difíciles de utilizar en prototipos sin el instrumental adecuado.

Se ofrecen, además, shields (placas suplementarias) de conexión, conectores a medida de las placas de desarrollo, pequeñas placas madre y placas Piggyback (que se insertan sobre las placas principales) que agregan capacidades y aceleran el desarrollo notablemente.

Las razones antedichas propician el uso de hardware sustentable por costo, rendimiento y disponibilidad. Además, inevitablemente, el hardware sustentable genera software sustentable, resultando código más fácil de entender y mantener.

Utilizar subconjuntos armados, es sumamente útil en la etapa de prototipo o en las producciones a baja escala. Pero el criterio de elección de componentes se extiende aún cuando la escala sea mayor y usar subconjuntos deje ya de ser práctico.

Si bien este fenómeno comenzó con Arduino, hoy muchos fabricantes tomaron el mismo criterio al ver los resultados obtenidos por abrir las puertas de sus productos. Se ha intensificado el soporte y el acceso a documentación, teniendo como objetivo no sólo a los diseñadores expertos. Se pueden conseguir hoy en el mercado, microcontroladores de alta performance con plataformas de desarrollo sencillas y en algunos casos, compatibles con la básica IDE de Arduino.

[3] David Cuartielles sobre su proyecto Arduino (Diario El País, España, 28 septiembre de 2006)

En el caso de uso, se utilizará la línea STM32F103, el software de generación de entorno STM32CubeMX[4] y Atollic TrueSTUDIO[5] e IDE Arduino como plataformas de desarrollo para lenguaje C, Assembler y Phyton. El microcontrolador STM32 es uno de los casos de integración de un dispositivo potente con una plataforma de desarrollo sencilla y gratuita.

En caso de ser necesario un módulo DSP para proceso de señales de medición se desarrollará con Audio Weaver[6] otra aplicación que facilita la programación DSP en la familia de microcontroladores ST.

De esta forma, queda claro como diferentes módulos se pueden desarrollar con múltiples herramientas.

2.3 Influencia de los cambios tecnológicos en la toma de requisitos

En el momento de establecer los alcances del proyecto, el entorno tecnológico aporta nuevas sendas para encaminar el desarrollo.

Una pregunta clave en el desarrollo de un proyecto es: ¿Es este un proyecto que resuelve una necesidad puntual o podría transformarse en un producto?

Si un diseño resuelve un problema puntual, por ejemplo, el control de la línea de laminación de una metalúrgica, es probable que el desarrollo fuera cerrado y propietario. O por lo menos, es lo que sucedía en las últimas décadas del siglo pasado. La tendencia era diseñar un gran circuito basado en una familia o la combinación de varias familias lógicas (CMOS, TTL, etc.) con el único objetivo de satisfacer las necesidades operativas propuestas.

Hoy, el objetivo de generar un diseño, debería cambiarse por el de generar un producto. A los conceptos tratados en párrafos anteriores, se suman otros, que facilitan la expansión, integración y conectividad. Expansión, es sumar capacidades a un producto y a esta altura del documento, se puede entender fácilmente que un proyecto nunca debería darse por terminado, sino que, al alcanzar la fase de puesta en producción, debería ser su destino natural seguir evolucionando.

Figura 11. Aporte de datos online

Plantear algunos interrogantes durante el proceso de diseño, ayuda a completar los alcances, por ejemplo:

- ¿Será necesario en algún momento de la vida tecnológica del producto la conectividad con nodos similares?
- ¿La información generada por esos nodos podría consolidarse? ¿Ayudaría eso a una mejor toma de decisiones de gestión?
- ¿En caso de centralizar la información en un nodo único, el flujo de datos será horizontal, vertical o ambos?
- ¿El almacenamiento de información histórica de procesos con fines estadísticos agrega valor?
- ¿El monitoreo remoto y central de las actividades de varios sistemas similares, es deseable?
- ¿La integración a sistemas mayores o menores sería una característica diferencial?
-

Todas las respuestas a esas preguntas llevan al mismo universo: IoT, BI, Big Data, Web Services. Los sistemas más autónomos que se puedan imaginar, se encuentran hoy conectados y generan datos constantemente. El la Figura 11 se puede ver la comparación entre la información que suben a la web los sistemas de los autos conectados, versus otras actividades online. La posibilidad de monetarizar el acceso a estos datos es un negocio incipiente, ya que sería de gran utilidad para aseguradoras y fabricantes de automóviles para determinar costumbres de manejo y otros parámetros de la conducción.

Demás está decir que el volumen de datos que aportan los automóviles conectados, que crecen en número día a día, traerá problemas de almacenamiento y proceso a corto plazo. Por esa razón ha avanzado el desarrollo de la tecnología llamada Edge Computing o Fog Computing que procesa datos antes de subirlos a la nube con el objetivo de minimizar así, su volumen.

2.4 Seguridad

La multiplicación del volumen de datos ha también multiplicado la importancia de mantenerlos seguros. El tópico es tan extenso que excede los alcances de este documento, pero aun así es obligatoria su mención. El ingreso de la seguridad en la estructura de los microcontroladores está impactando en los diseños desde sus raíces para poder compatibilizarlos con IoT. En los próximos años, su influencia en la programación y diseño será capital.

3. Conclusiones y trabajos futuros

El objetivo del presente trabajo es analizar sucintamente las causas y efectos de los cambios en el tablero tecnológico y como eso repercute en los procesos de diseño de sistemas embebidos. Este análisis, desde lo secuencial y

[4] https://www.st.com/en/development-tools
[5] https://atollic.com/truestudio/
[6] https://dspconcepts.com/st

metodológico es frecuente en el desarrollo de software, pero no lo es tanto en los sistemas integrados.

µ-Framework es en sí, una propuesta metodológica que toma procesos y conceptos de diseño clásicos y los adapta a las nuevas reglas. Si bien, la metodología propuesta, tal como su dominio de aplicación, sufrirán cambios constantes; se trata de aportar desde la formación del sentido común, ya que no es la intención proponer un método férreo de diseño, sino una forma de pensar los desarrollos.

La ventaja que otorga la rápida adaptación a los escenarios tecnológicos es un rasgo diferencial, rector del desarrollo sustentable y de las buenas reglas del diseñador. Es, en resumen, el gran objetivo de este documento; hacer notar que los cambios tecnológicos no solo influyen en las plataformas, sino en los criterios con que se aborda el diseño.

Por cuestiones de extensión, el presente trabajo se concentra más en aclarar el marco conceptual en que se basa µ-Framework que en ahondar en detalles técnicos generados por el caso de estudio. Existe una tesis en redacción que incluirá tanto los resultados metodológicos como técnicos con mayor detalle.

El abordaje a este tema recién comienza y su aplicación proveerá mejoras y extenderá el alcance. Los objetivos próximos son generar una documentación concisa que guíe los pasos del desarrollo y un método claro para evaluar resultados. Estos puntos ayudarán a la aplicación y evaluación de resultados, principalmente en el entorno académico.

Agradecimientos

Al Ing. Fernando Szklanny. Al Lic. Carlos Maidana, Ing. Hugo Tantignone del Departamento de Ingeniería de UNLAM, por su constante apoyo y guía.

Al Ing. Jorge Casanova, por su inagotable memoria y amistad.

Al Sr. Eugenio Paredes Rojas y el Sr. Pablo Lombardi por compartir desinteresadamente su extensa experiencia de campo.

Referencias

[1] Punkka, T., Embedded Agile, Embedded System Conference 2010, ESC 241, Boston, USA, 2010

[2] Department of Defense USA, FAR, Part12, Cap. Adquisition of Commercial Items, USA, 2018.

[3] Kaplan G., Hadad G., Doorn J., Sampaio do Prado Leite J. Inspección del Léxico Extendido del Lenguaje, III Workshop de Engenharia de Requisitos (WER), 2000.

[4] Blakeslee, T, Digital Design with Standard MSI and LSI, John Wiley and sons, New York, Cap. 2, 1975

[5] Embedded, Yes, you can develop embedded software using Agile methodology, https://www.embedded.com/, 2016{{[{{

[6] Holzman G., The Power of Ten – Rules for Developing Safety Critical Code, IEEE, Nasa /JPL Laboratory for Reliable Software, 2018

[7] University of Washington, Embedded Systems Design and Development, https://class.ece.uw.edu/, Cap 12.0 – Design Cycle

[8] Manifesto for Agile Development Software, Twelve Principles of Agile Manifesto, https://agilemanifesto.org

[9] Microsoft, Creating Security Controls for IoT at Microsoft, Microsoft IT Showcase, USA, 2017

[10] Steve Blank, Perfect by Subtraction – The Minimum Feature, https://steveblank.com, 2010