Código	FPI-009
Objeto	Guía de elaboración de Informe final de proyecto
Usuario	Director de proyecto de investigación
Autor	Secretaría de Ciencia y Tecnología de la UNLaM
Versión	5
Vigencia	03/9/2019

Departamento de Ingeniería e Investigaciones Tecnológicas

Programa de acreditación:
PROINCE

Código del Proyecto: C 222

Título del proyecto:

Desarrollo de un banco didáctico de ensayo de motores térmicos

Director:
ETEROVIC, Jorge Esteban

Codirector:
RODOFILE, Hugo Guillermo

Integrantes:
PEREZ, Alejandro Sigfrido
FOURCADE, Alejandro
PEREZ ARAUZ, Alejandro Luis

Resolución Rectoral de acreditación: N° 352/19

Fecha de inicio: 01/01/2019

Fecha de finalización: 31/12/2020

A. Desarrollo del proyecto (adjuntar el protocolo)

A.1. Grado de ejecución de los objetivos inicialmente planteados, modificaciones o ampliaciones u obstáculos encontrados para su realización (desarrolle en no más de dos (2) páginas)

Etapa I.1 – Búsqueda bibliográfica proveedores e insumos

Para diseñar una plataforma de medición de un banco de ensayo de motores fue necesario visitar algunas unidades instaladas en otras Universidades para relevar sus principales características funcionales. Desde el punto de vista electrónico, se relevaron las interfaces de conexión más usuales, las variables a medir, el método de prueba y las facilidades auxiliares necesarias. (unidad de enfriamiento, sistema de alimentación de combustible, tratamiento de gases, etc.).

Los valores que entrega el freno deben consolidarse en un punto de control que permita la visualización de los mismos obtenidos en el ensayo. Una vez que se contó con la información operativa mínima se especificó el alcance inicial del proyecto de investigación: tomar las variables fundamentales de un motor en prueba de rendimiento para almacenarlas y graficarlas.

Etapa I.2 – Definición y diseño del banco de ensayo de motores

Para la definición del sistema fue necesario relevar las tareas que el usuario espera. Para ello se entrevistó a profesionales, usuarios y docentes. Se acordó que el banco de ensayo de motores tendría un uso exclusivamente didáctico y que no soportaría pruebas extremas de desempeño. Por estas mismas razones la precisión de las mediciones no necesita ser extremadamente alta, sino que se priorizaría su funcionamiento integral.

Dado que no existía al momento de comenzar el desarrollo de la parte electrónica certeza de la ubicación de la explanada de ensayo, los controles y la computadora (que son las partes principales del sistema), se decidió hacer la conectividad entre módulos lo más flexible posible. Por eso se propuso diseñar una interfaz de toma de datos analógico/digital que transmita la información consolidada a un nodo central vía WIFI/Ethernet.

Teniendo en cuenta que el desarrollo electrónico responde a las normas especificadas por µ-Framework (un entorno conceptual de generación de hardware y software embebido), se siguieron los pasos aconsejados para la obtención de un MVP (mínimo producto viable).

Este MVP incluía el nodo central de proceso, la interfaz CAN, la interfaz a sensores, la pantalla de información de los datos obtenidos y la conectividad internet/WIFI intermodular. Se definió una estrategia de desarrollo modular basado en hardware sustentable (de adquisición comercial habitual) y un criterio concéntrico de la extensión de los alcances. Como se muestra en la Figura 1, para la primera parte de este proyecto se optó por un modelo de desarrollo modular:

Otra decisión de diseño que atañe al sistema de medición y adquisición de datos es la definición de la interfaz al usuario. Se optó por tres vías de comunicación: un display gráfico en el punto de ensayo, un informe consolidado de datos obtenidos en una computadora en el nodo de control y un acceso a través de la Web. En el diseño de las interfaces se priorizó la visualización de datos críticos para la operación y la seguridad en cada lugar.

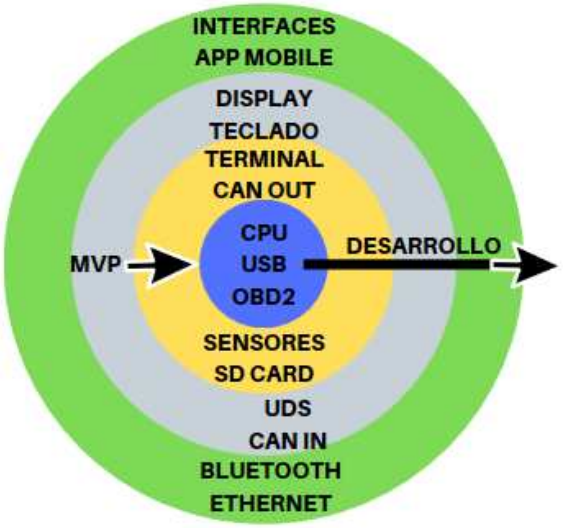

Figura 1. Modelo concéntrico de desarrollo modular

Los datos que se mostrarán en el punto de ensayo resumen el avance de la prueba y detalles de las alertas de seguridad. En el punto de control se mostrarán curvas e información estadística básica y en la interfaz Web los resultados globales y los datos ya procesados.

Como se mencionó, la definición de la electrónica para esta etapa del proyecto prevé posibles cambios en la matriz tecnológica y en los alcances propuestos. Para acompañar estos cambios el sistema es fácilmente expandible, interconectable, posee capacidades de proceso sobredimensionadas y puede implementarse en un entorno de tolerancia a fallos.

Etapa II.3 – Diseño y desarrollo del subsistema electrónico de medición y digitalización

En la primera etapa de relevamiento, se detectaron posibles ventajas operativas y de prestación si se tomaba información tanto analógica como digital y se las procesaba en conjunto complementariamente. Para incorporar la información de los sensores se probaron varios sistemas de conversión AD, obteniendo resultados consistentes con el módulo ADS1115. Se seleccionó este módulo por tener como protocolo de comunicación al I2C, que coincide con la elección del protocolo interno del sistema.

El sistema de toma digital de datos (CAN BUS) se implementó alrededor de un módulo MPC2515 que si bien es SPI está acompañado de una unidad de proceso modular basada en Raspberry Zero que traduce lo obtenido a I2C.

El módulo central de proceso es una placa Raspberry Pi corriendo Raspbian optimizado especialmente para este fin.

Etapa II.4 – Diseño y desarrollo del subsistema informático y de representación gráfica

Se probaron varios display para el punto de ensayo, HDMI touch, Nextion HMI, optando alternativamente entre ellos en diferentes ensayos para probar sus cualidades. El front end en el punto de proceso se realizó en Raspbian con desarrollos en Shell Scripting y en Phyton corriendo en Raspberry PI y Zero.

Dentro del desarrollo de este subsistema y como hardware auxiliar de investigación, fue necesario generar un entorno de pruebas portable de protocolo CAN. Para llevar a cabo esta tarea se incorporaron tableros de automóviles, se desarrollaron CAN sniffers y se programó un control gráfico touch para activar las capacidades del tablero de instrumentos. Este conjunto de desarrollos adicionales facilitó mucho las tareas de conexión con la camioneta Toyota Hilux SW4 del DIIT, que se utilizó para los ensayos de campo.

Cada nodo medidor tiene un pequeño display OLED que visualizará el proceso activo y algunos parámetros de entorno. Este sistema con la conexión OBD2, un nodo remoto de sniffing, el análisis de los datos obtenidos y el control vía un panel touch de un tablero CAN fue presentado en Expo Proyecto 2020.

Etapa II.5 – Diseño, calculo y desarrollo del subsistema de alimentación

Para el diseño, calculo y desarrollo del subsistema de alimentación de combustible se debe considerar que los motores aspirados utilizan una presión de combustible que están en el orden de 0,069 bares (6,89 Kilo Pascales) a 0,2 bares (27,57 Kilo Pascales).

Los sistemas de inyección electrónica de combustible en general trabajan por debajo de la presión máxima que entrega la bomba de combustible, es decir aproximadamente entre 2,8 bar (280 kilo Pascales) y 3,2 bar (320 kilo Pascales). La presión debe comprobarse con un manómetro y regularla.

El sistema se calculó con un medidor volumétrico, válvulas de corte eléctrico, bomba de combustible, regulador de presión, tanque y cañerías de 8 mm del tipo alta presión, estándar, con conectores de acople rápido.

El diagrama del circuito de combustible es el indicado en la Figura 2.

Etapa II.6 – Diseño, calculo y desarrollo del subsistema de escape

El subsistema de gases de escape constará de un caño de escape convencional conectado al múltiple de salida del motor y llevará un silenciador del mismo tipo que el instalado en el vehículo al cual pertenece el motor. El caño de escape se conectará a la tubería de salida de gases calientes de la sala, como se muestra en la Figura 3.

Etapa II.7 – Diseño, calculo y desarrollo del subsistema de refrigeración

El sistema está constituido por: un tanque de 60 litros para el llenado del circuito motor - radiador y mantener constante el mismo, un segundo tanque de 30 litros para la descarga del circuito, permitiendo retirar el motor, un circuito de cañerías y electroválvulas para el llenado, vaciado y circulación del fluido durante el ensayo. En la cañería de descarga del tanque de recupero hay una bomba, la cual se encargada de recircular el fluido al tanque para el llenado.

Descripción del funcionamiento del circuito de refrigeración.

- Llenado del sistema motor radiador: Para el llenado las válvulas V1, V2; V5, V6, y V4 deben estar abiertas, para que el fluido circule, la electroválvula V4 es la que permite la salida del aire, cuando la

cañería de esta válvula alcanza el nivel (vasos comunicantes) del tanque de carga, el circuito queda sin aire, por lo tanto se cierran las válvulas V1, V4, V7 y las V3, V2, V5 y V6 quedaran abiertas. En esta condición el circuito está listo para realizar el ensayo.

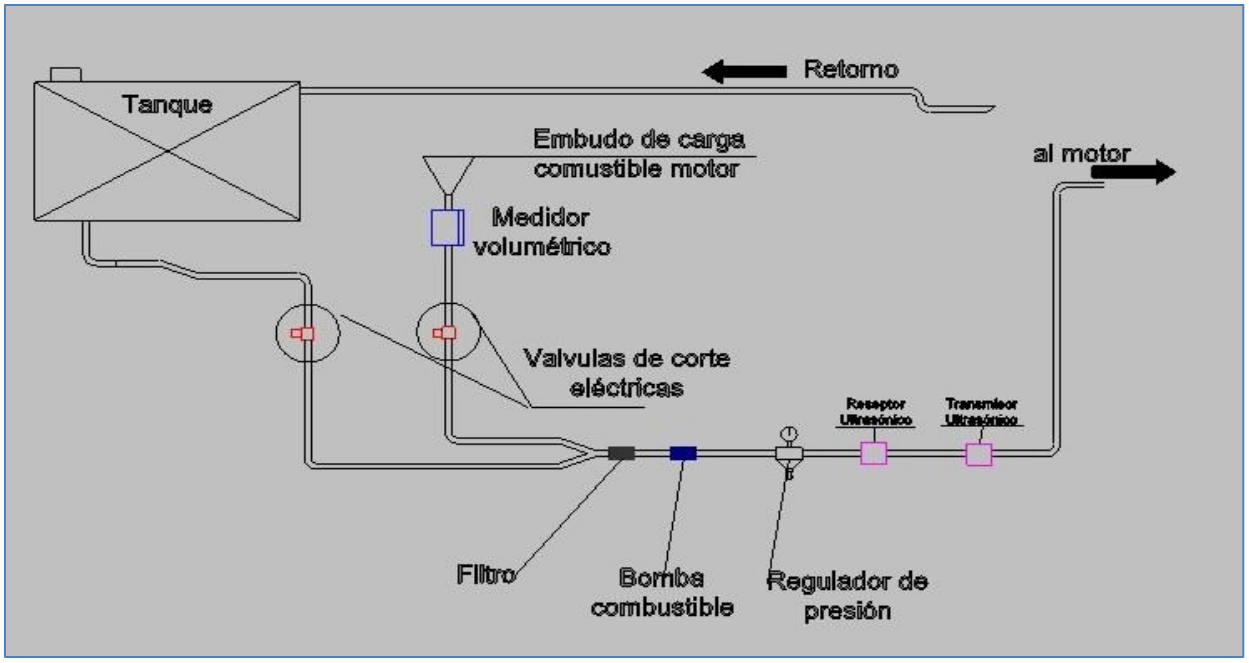

Figura 2. Diagrama del circuito de combustible

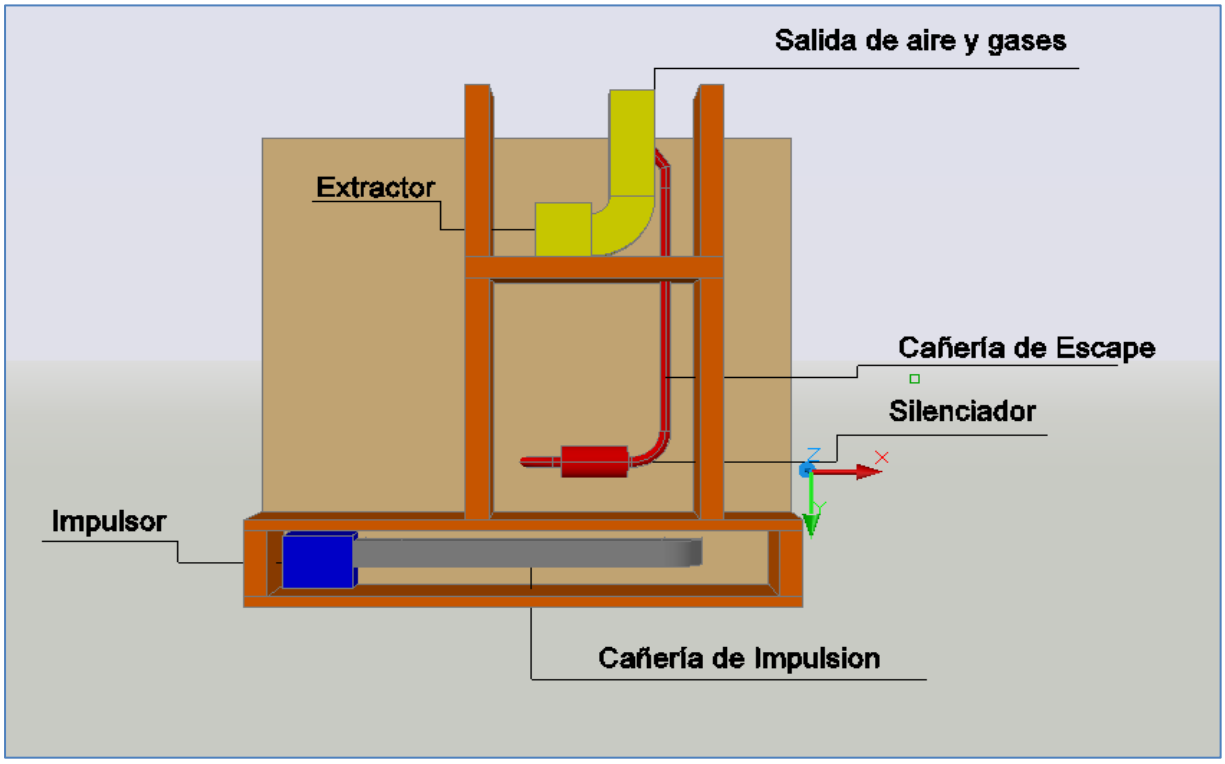

Figura 3. Subsistema de salida de gases de escape

- Vaciado del sistema motor radiador: Para el vaciado del sistema, las válvulas V1, V2 y V3 deben estar cerradas y las válvulas V7, V5, V6 y V4 deben estar abiertas, V4 es la que permite la entrada de aire para que se produzca la descarga, cuando el tanque de descarga alcance un caudal preestablecido, se encenderá la bomba de recupero enviando el caudal al tanque de llenado.

El sistema permite el cambio del subsistema radiador – electroventilador, una vez vaciado el circuito, se cierran todas las válvulas, esto permite la desconexión y retiro del radiador. El soporte universal permite montar cualquier versión de radiador – electroventilador según se requiera.

El esquema general del subsistema de refrigeración se muestra en la Figura 4 y la Figura 5 es una vista del subsistema de refrigeración del motor y del filtro de aire.

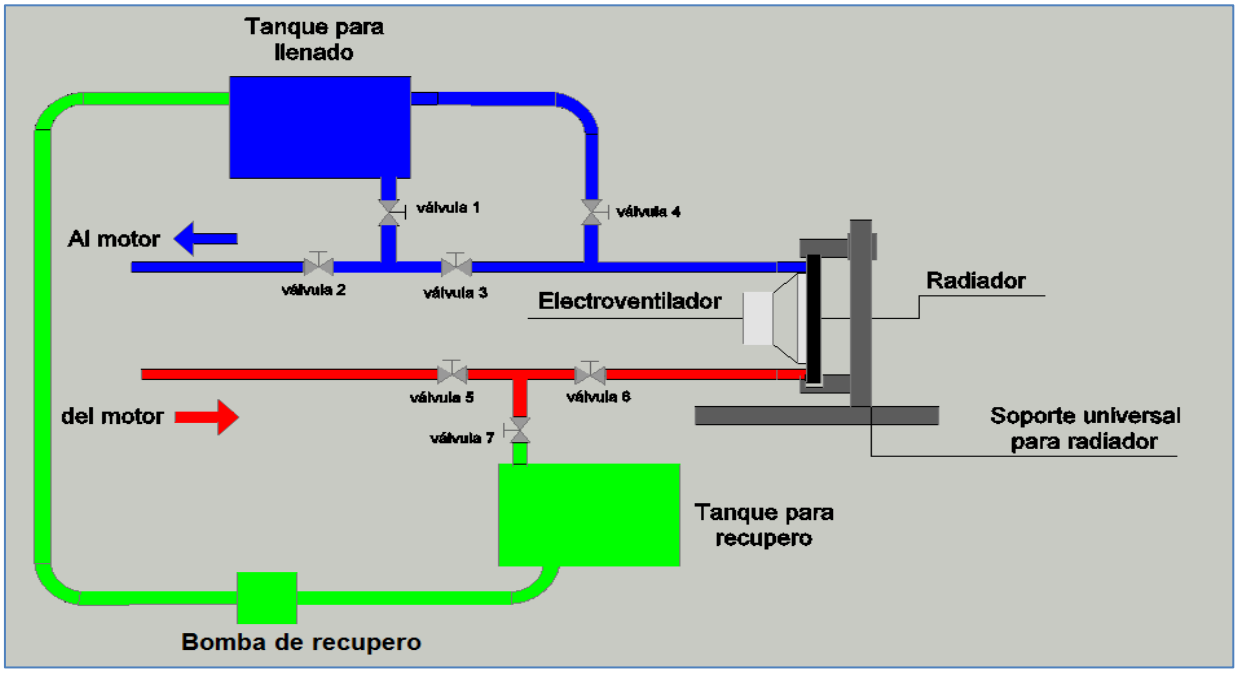

Figura 4. Circuito de Refrigeración

Etapa II.8 – Diseño, calculo y desarrollo de la sala contenedora

El diseño general de la sala contenedora del banco de ensayo de motores contempla una sala de control, la sala de máquinas, un depósito de herramientas y la sala de ensayo de motores térmicos. Este diseño se muestra en la Figura 6.

Etapa II.9 – Diseño, calculo y desarrollo del subsistema de anclaje

Por disponibilidad comercial, costo, resistencia y rigidez, se seleccionó un perfil estructural de acero del tipo U para el armazón principal. Se han previsto puntos de anclaje regulables en posición y en altura para el posicionamiento de diferentes motores según diseño y configuración de soportes originales, utilizando torretas con tornillos de potencia.

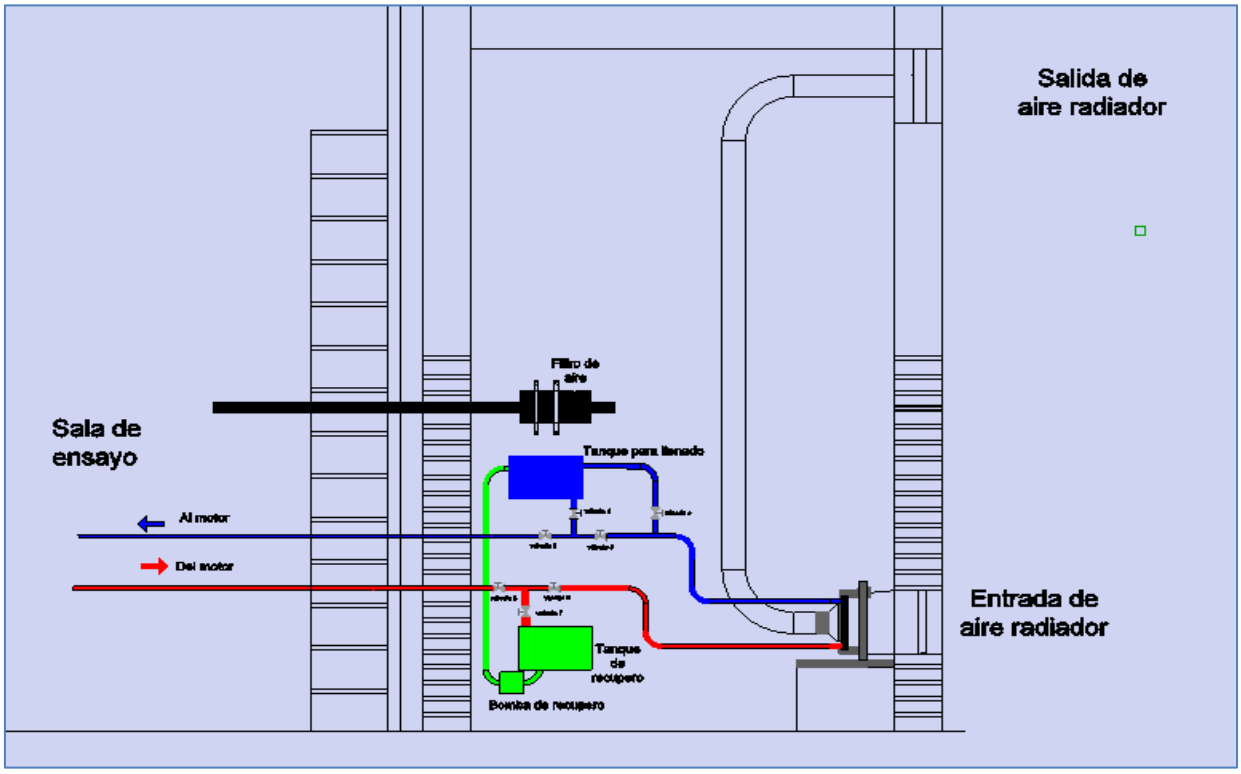

Figura 5. Subsistema de refrigeración del motor y del filtro de aire

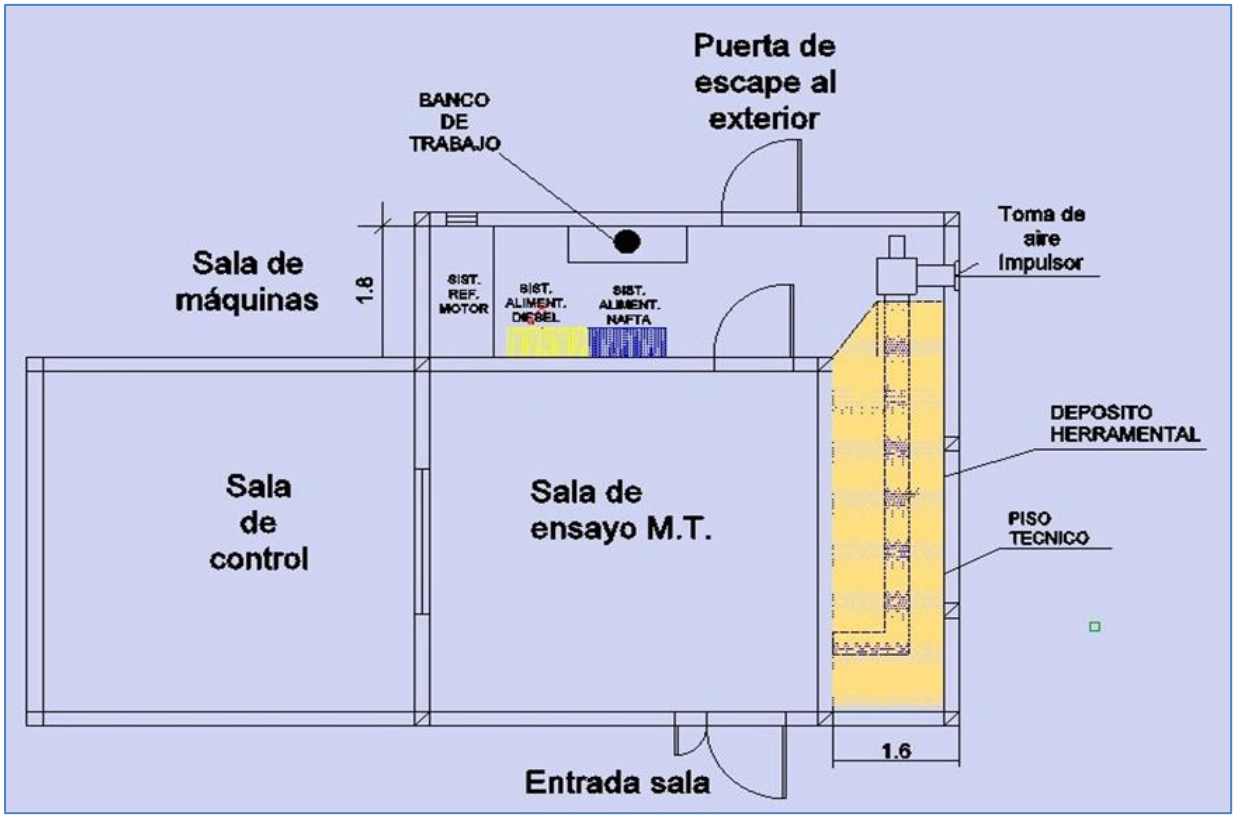

Figura 6. Diseño de la Sala Contenedora

La bancada se ha aislado del suelo mediante elementos de absorción de vibraciones, los cuales se han seleccionado atendiendo a las cargas transmitidas por el motor durante el proceso de arranque. Para el diseño de la bancada se ha tenido en cuenta: la exigencia de resistencia y rigidez tanto a las cargas estáticas generadas por el peso de los componentes, como las cargas dinámicas que se producen al tener el motor y el freno en funcionamiento, la flexibilidad para realizar los montajes de diferentes motores, la facilidad de acceso para la manipulación del banco, la seguridad para el personal que lo manipule.

El diseño constructivo de la estructura del banco puede apreciarse en la Figura 7.

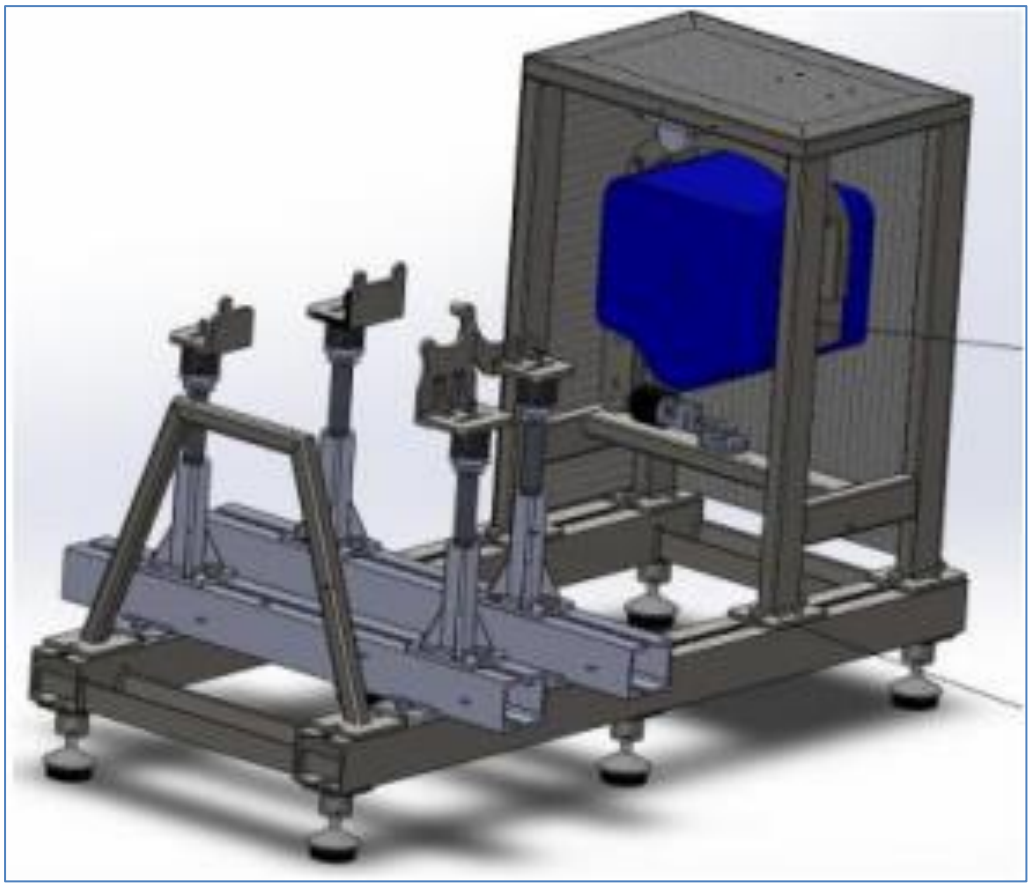

Figura 7. Anclaje de Motores

Para el acople mecánico entre el dinamómetro y los motores a ensayar, se ha previsto dos platos de acople en los extremos para la unión con la volante y el dinamómetro, y una transmisión cardánica, la cual además de acoplar mecánicamente el motor de combustión y el dinamómetro, transmitiendo el par y el régimen de giro, sirve para salvar la posible desalineación existente entre las dos máquinas y, por su flexibilidad torsional, la frecuencia natural de las vibraciones torsionales que aparecen en la unión. Una caja de material desplegable sirve para prevenir al personal de la exposición directa ante situaciones de peligro provocadas por la rotura accidental de la transmisión cardánica.

El detalle del acople mecánico entre el dinamómetro y los motores a ensayar se puede ver en la Figura 8.

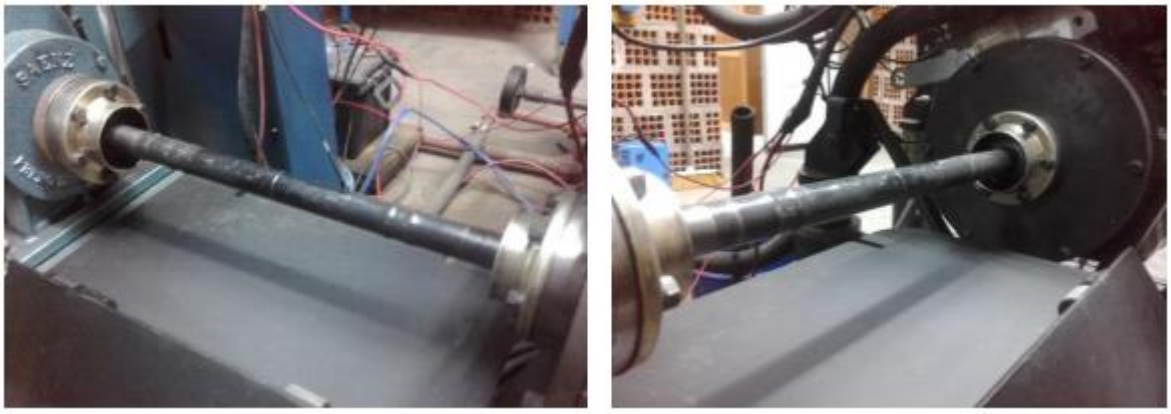

Figura 8. Acople mecánico Dinamómetro-Motor

Etapa II.10 – Diseño, calculo y desarrollo del subsistema de control

Esta Etapa fue estudiada y resuelta en las Etapa II.3 – Diseño y desarrollo del subsistema electrónico de medición y digitalización y Etapa II.4 – Diseño y desarrollo del subsistema informático y de representación gráfica.

En la Figura 9 se muestra cómo se vería ser la sala de control.

Figura 9. Sala de Control

Etapa II.11 – Diseño, calculo y desarrollo de los subsistemas auxiliares y de seguridad

El caudal de aire es un concepto fundamental en todas las instalaciones de impulsión y extracción, dado que, nos indica, la cantidad de aire que se renueva en la sala. El caudal para este estudio es la cantidad de aire que discurre por un lugar en la unidad de tiempo. Si tomamos como ejemplo el caso de salas de máquinas, se tiene la información estadística que la renovación debería estar entre 30 y 60 veces el volumen de la sala por hora.

La temperatura de la sala de ensayos aumentará hasta que haya un equilibrio entre el calor generado y el calor disipado por el sistema de ventilación, que es función de la diferencia de temperaturas entre la entrada y la salida.

Para nuestra sala utilizaremos el sistema mixto de impulsión-extracción. La impulsión provoca la introducción de aire desde el exterior a la sala, y la extracción es para la expulsión del aire caliente al exterior. Considerando este sistema de renovación de aire, existe una depresión provocada por los extractores y una sobre presión provocada por los impulsores.

Etapa II.11 – Diseño, calculo y desarrollo de los subsistemas auxiliares y de seguridad

El caudal de aire es un concepto fundamental en todas las instalaciones de impulsión y extracción, dado que, nos indica, la cantidad de aire que se renueva en la sala. El caudal para este estudio es la cantidad de aire que discurre por un lugar en la unidad de tiempo. Si tomamos como ejemplo el caso de salas de máquinas, se tiene la información estadística que la renovación debería estar entre 30 y 60 veces el volumen de la sala por hora.

La temperatura de la sala de ensayos aumentará hasta que haya un equilibrio entre el calor generado y el calor disipado por el sistema de ventilación, que es función de la diferencia de temperaturas entre la entrada y la salida.

Para nuestra sala utilizaremos el sistema mixto de impulsión-extracción. La impulsión provoca la introducción de aire desde el exterior a la sala, y la extracción es para la expulsión del aire caliente al exterior. Considerando este sistema de renovación de aire, existe una depresión provocada por los extractores y una sobre presión provocada por los impulsores.

Para que la instalación dé un mejor resultado, trabajaremos con presión positiva en la sala, o sea los impulsores deben dar más aire del que desalojan los extractores. Este valor de acuerdo con la experiencia, se estima la relación en un 20% más la impulsión que extracción.

Es muy importante tener en cuenta que tanto el motor de impulsión como de aspiración sean de velocidad variable, ya que esto permite regular el caudal volumétrico que se renueva, para regular la temperatura de la sala.

Este sistema de control debe ser automático, por medio de un sensor de temperatura en la sala y un sistema de termostato. El termostato deberá dar las señales a los actuadores para el aumento o disminución de la velocidad de ambos motores en forma simultánea, manteniendo la diferencia de caudales del 20% constante.

El diseño del subsistema de extracción de aire de la sala es el que se muestra en la Figura 10.

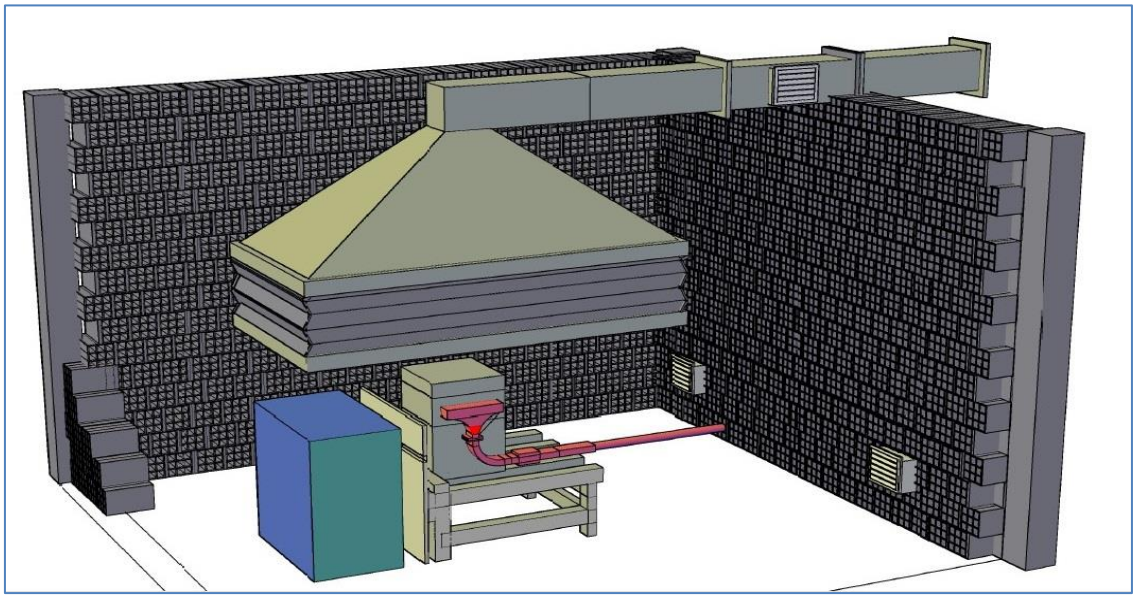

Figura 10. Subsistema de extracción de aire

Etapa III.12 – Armado, Instalación y Prueba
Etapa III.13 – Ajuste final y puesta en funcionamiento
Estas Etapas no se han desarrollado debido a que no se encuentra construido el edificio del Laboratorio de Ingeniería Mecánica.

B. Principales resultados de la investigación

B.1. Publicaciones en revistas (informar cada producción por separado)

No se han realizado publicaciones en revistas.

B.2. Libros

No se han realizado publicaciones en libros.

B.3. Capítulos de libros

No se han realizado publicaciones en capítulo de libros.

B.4. Trabajos presentados a congresos y/o seminarios

Autores	*Alejandro Fourcade; Jorge Eterovic; Alejandro Pérez; Guillermo Rodofile.*
Título	*µ-Framework: Una Visión Actual para el Desarrollo de Sistemas Embebidos*
Año	*2019*
Evento	*CoNaIISI 2019*
Lugar de realización	*San Justo, Argentina*
Fecha de presentación de la ponencia	*14 y 15 de noviembre de 2019*
Entidad que organiza	*CONFEDI; RIISIC; UNLaM; DIIT*
URL de descarga del trabajo (especificar solo si es la descarga del trabajo; formatos pdf, e-pub, etc.)	*https://conaiisi2019.unlam.edu.ar/pdf/2019-CONAIISI-ACTAS-7MA-EDICION.pdf*
Autores	*Alejandro Fourcade; Jorge Eterovic.*
Título	*Análisis de la Seguridad del Protocolo de Comunicaciones CAN*
Año	*2020*
Evento	*XXII Workshop de Investigadores en Ciencias de la Computación – WICC 2020*
Lugar de realización	*El Calafate – Evento virtual*
Fecha de presentación de la ponencia	*07 y 08 de mayo de 2020*
Entidad que organiza	*Red UNCI; Universidad de la Patagonia Austral*
URL de descarga del trabajo	*http://sedici.unlp.edu.ar/handle/10915/103151*
Autores	*Alejandro Fourcade; Jorge Eterovic.*
Título	*Un Análisis de los Aportes de Agile al Desarrollo de Sistemas Embebidos*
Año	*2020*
Evento	*CoNaIISI 2020*
Lugar de realización	*San Francisco, Argentina*
Fecha de presentación de la ponencia	*05 y 06 de Noviembre de 2020*
Entidad que organiza	*CONFEDI; RIISIC; UTN-FRSF*
URL de descarga del trabajo	*http://conaiisi2020.frsfco.utn.edu.ar/*

B.5. Otras publicaciones

No se han realizado otras publicaciones.

C. Otros resultados.

No hay otros resultados.

C.1. Títulos de propiedad intelectual. Indicar: Tipo (marcas, patentes, modelos y diseños, la transferencia tecnológica) de desarrollo o producto, Titular, Fecha de solicitud, Fecha de otorgamiento

No se han registrado Títulos de propiedad intelectual.

C.2. Otros desarrollos no pasibles de ser protegidos por títulos de propiedad intelectual. Indicar: Producto y Descripción.

No se han realizado otros desarrollos.

D. Formación de recursos humanos. Trabajos finales de graduación, tesis de grado y posgrado. Completar un cuadro por cada uno de los trabajos generados en el marco del proyecto.

No aplica.

D.1. Tesis de grado

No se han realizado tesis de grado.

D.2 Trabajo Final de Especialización

No se han realizado trabajos finales de especialización.

D.2. Tesis de posgrado: Maestría

Director (apellido y nombre)	Tesista (apellido y nombre)	Institución	Calificación	Fecha / En curso	Título de la tesis
Eterovic, Jorge	Fourcade, Alejandro	UNLaM	--------	En curso	µ-Framework: Una Visión Actual para el Desarrollo de Sistemas Embebidos

D.3. Tesis de posgrado: Doctorado

No se han realizado tesis de doctorado.

D.4. Trabajos de Posdoctorado

No se han realizado posdoctorados.

E. Otros recursos humanos en formación: estudiantes/ investigadores (grado/posgrado/ posdoctorado)

No se han realizado otras formaciones de RRHH.

F. Vinculación[1]: Indicar conformación de redes, intercambio científico, etc. con otros grupos de investigación; con el ámbito productivo o con entidades públicas. Desarrolle en no más de dos (2) páginas.

No se han realizado vinculaciones.

G. Otra información. Incluir toda otra información que se considere pertinente.

No aplica.

H. Cuerpo de anexos:

- Anexo I: Copia de cada uno de los trabajos mencionados en los puntos B, C y D, y certificaciones cuando corresponda.[2]
- Anexo II:
  - FPI-013: Evaluación de alumnos integrantes. (si corresponde)
  - FPI-014: Comprobante de liquidación y rendición de viáticos. (si corresponde)
  - FPI-015: Rendición de gastos del proyecto de investigación acompañado de las hojas foliadas con los comprobantes de gastos.
  - FPI-035: Formulario de reasignación de fondos en Presupuesto.
- Anexo III: Alta patrimonial de los bienes adquiridos con presupuesto del proyecto (FPI 017)
- Nota justificando baja de integrantes del equipo de investigación.

_____

Jorge Esteban Eterovic
Director del proyecto

San Justo, 19 de febrero de 2021

[1] Entendemos por acciones de "vinculación" aquellas que tienen por objetivo dar respuesta a problemas, generando la creación de productos o servicios innovadores y confeccionados "a medida" de sus contrapartes.
[2] En caso de libros, podrá presentarse una fotocopia de la primera hoja significativa o su equivalente y el índice.

Anexo I

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.

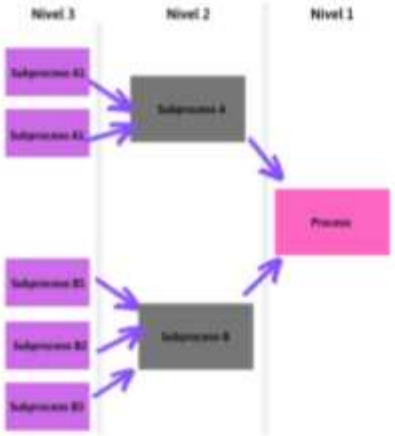

Figura 4. División de procesos

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	Boton de parada

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

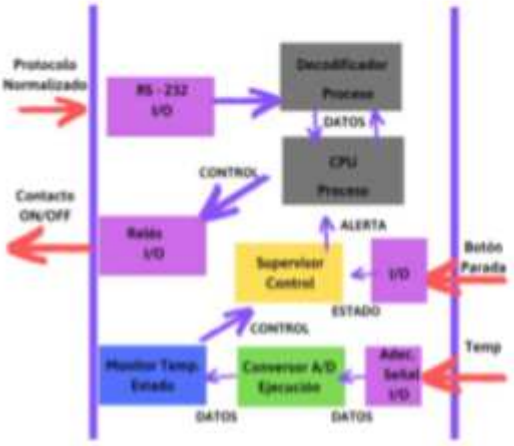

Figura 5. Procesos y mensajes

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.

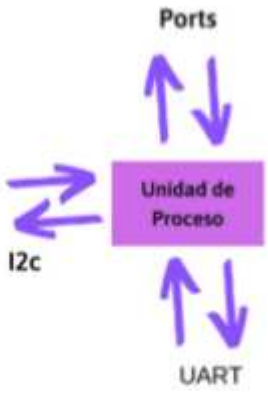

Figura 6. Módulo Central

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).

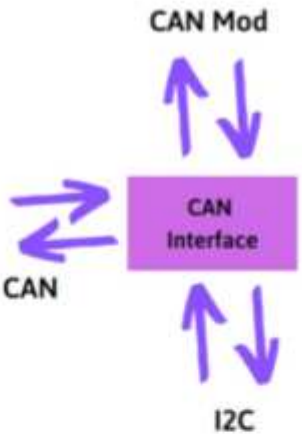

Figura 7. Módulo CAN de borde

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.

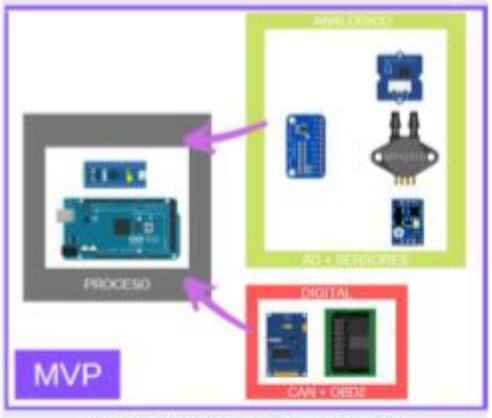

Figura 8. Módulos que forman MVP

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.

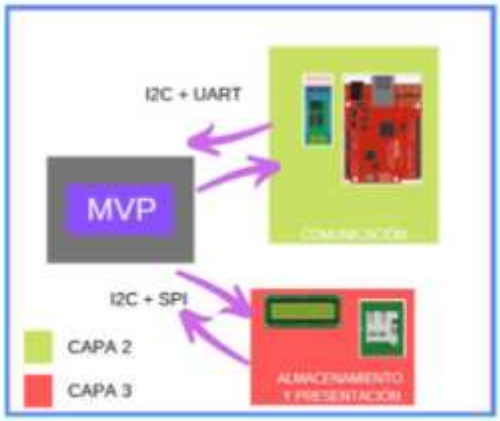

Figura 11. Iteraciones sobre el MVP

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.

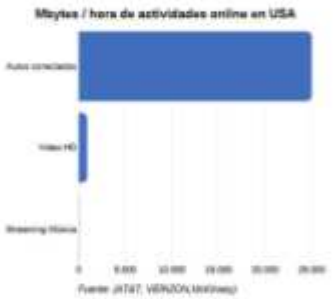

Figura 11. Aporte de datos online

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 /IPL 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

CONAIISI

VII Congreso Nacional de Ingeniería Informática - Sistemas de Información

2019

San Justo, 5 de diciembre de 2019

Se certifica que **Alejandro Fourcade, Jorge Eterovic, Alejandro Pérez, Guillermo Rodofile** han participado como Autores del artículo 349 *"µ-Framework: Una Visión Actual para el Desarrollo de Sistemas Embebidos"*, aceptado en el VII Congreso Nacional de Ingeniería Informática – Sistemas de Información, CONAIISI 2019, realizado los días 14 y 15 de noviembre en la Universidad Nacional de La Matanza.

Ing. Claudio D'Amico
Coord. Gral. CONAIISI

Dr. Carlos Neil
Coordinador RIISIC

Mg. Jorge Eterovic
Decano DIIT

UNLaM confedi DIIT RIISIC

ANÁLISIS DE LA SEGURIDAD DEL PROTOCOLO DE COMUNICACIONES CAN

Ing. Alejandro Fourcade, Mg Jorge Eterovic
Departamento de Ingeniería e Investigaciones Tecnológicas Universidad Nacional de La Matanza
Florencio Varela 1903 (B1754JEC), San Justo, (5411) 4480-8900

afourcade@unlam.edu.ar; eterovic@unlam.edu.ar

RESUMEN

Desde su concepción, el protocolo de comunicaciones CAN (Controller Area Network), trata de dar solución a la transmisión de datos en los exigentes entornos industriales. Creado por Bosch en 1986, ofrece una opción robusta y confiable, por lo cual se ha transformado en el estándar para la industria automotriz [1].

Su arquitectura está pensada para satisfacer eficientemente las necesidades operativas de un vehículo. Pero hoy, el automóvil tal como se lo conocía una década atrás, ya no existe. Sus mecanismos han sido invadidos por dispositivos digitales, sensores, actuadores y tiene hasta 80 ECUs (Engine Control Unit: Unidad de Control de Motor) comunicadas entre sí. A esta complejidad se ha sumado una aún mayor: estar conectado a Internet constantemente.

Como sucedió recientemente con algunos protocolos inalámbricos ante el advenimiento del Internet de las Cosas (IoT), pasar de un dominio cerrado a uno abierto conectado a la nube, propicia la aparición de amenazas, vulnerabilidades y brechas de seguridad.

Este cambio de escenario no estuvo en los planes en el momento de la concepción de la arquitectura, y en el caso de CAN termina transformándose en una bomba de tiempo. La diferencia entre los dos escenarios es que no tiene las mismas consecuencias un ataque informático a una heladera que a un camión de carga circulando por una autopista.

El principal problema es que algunas de las fortalezas del protocolo CAN, se transforman en limitaciones a la hora de adaptarse a entornos abiertos. Este dilema no es nuevo, pero como el riesgo es creciente, es de crucial importancia encontrar soluciones de seguridad.

Palabras Clave: Seguridad de Datos. Protocolo CAN. Seguridad en Automóviles Conectados.

CONTEXTO

Este proyecto de investigación se desarrolla en el marco de un Programa de Incentivos a Docentes Investigadores de la Secretaría de Políticas Universitarias (PROINCE) en el Departamento de Ingeniería e Investigaciones Tecnológicas de la Universidad Nacional de La Matanza.

El proyecto es financiado por el propio Departamento y es del tipo investigación aplicada. El mismo consiste en el desarrollo de un banco didáctico de ensayo de motores térmicos. Los trabajos de campo y relevamientos realizados aportaron información valiosa y sirvieron como base para el presente trabajo.

1. INTRODUCCIÓN

Los automóviles eran hasta hace unos años dispositivos casi exclusivamente electromecánicos, pero en las últimas décadas se ha incorporado masivamente la electrónica digital y ha tomado preponderancia y criticidad en su funcionamiento. También la Tecnología de la Información (IT) tiene un papel fundamental en los diseños automotrices.

El uso de la IT en los vehículos es cada vez mayor y sus objetivos eran brindar conectividad y confort. Si bien se han alcanzado estas metas y se han maximizado las prestaciones, no se han tenido en cuenta las amenazas y vulnerabilidades ante ataques externos que tiene todo dispositivo conectado a una red abierta como Internet.

Es muy importante la gestión de la seguridad en los sistemas de información. Uno de los principales factores a considera cuando se evalúan

los posibles daños que una intrusión maliciosa puede ocasionar, es el tipo y tiempo de respuesta del sistema.

Los automóviles son sistemas que reaccionan en tiempo real y una de las características del protocolo CAN es su respuesta rápida garantizada. Por eso el daño eventual ante un ataque al sistema de control de un automóvil es muy alto, teniendo en cuenta que, por ejemplo, la dirección, los frenos y el acelerador dejaron de tener control mecánico y pasaron a ser comandados electrónicamente.

Las ECU son las que leen los sensores y comandan los actuadores de un vehículo. En las primeras unidades que incluían esta tecnología había exclusivamente una ECU central, pero en los vehículos actuales se pueden encontrar hasta 80 de ellas conectadas a través de buses de diferentes jerarquías. Por ejemplo, el bus CAN destinado al sistema de confort, tiene menor prioridad y velocidad de respuesta que el que controla el airbag.

Este complejo sistema maneja actualmente casi todos los dispositivos de un automóvil, desde su dirección hasta los frenos, pasando por las trabas de las puertas, el tablero de instrumentos, las ópticas y hasta en algunos modelos, las bombas de agua y combustible. En muchos casos no existe una opción mecánica que prevalezca por sobre la electrónica. Si la ECU decide doblar a la derecha, el volante no puede evitar que esto suceda.

El periodista de la revista Forbes Andy Greenberg, pudo constatar que su acompañante podía habilitar o deshabilitar desde su notebook MacBook, los frenos de una camioneta Ford Escape de 1500 kg. de peso [2]. La situación descripta deja ver la gravedad de una intrusión al sistema.

En los años venideros este panorama se hará todavía más complejo, lo que lleva a los investigadores de la seguridad de la información como Eugene Kaspersky, ampliamente conocido por su software antivirus, a agregar al entorno automotriz dentro de los posibles objetivos de un ataque informático [3].

En el futuro tendremos nuevas formas de conectividad, como CAR TO CAR (C2C) o CAR TO INFRASTRUCTURE (C2I), lo que aumentará la integración de los vehículos a las redes de datos, por lo que las amenazas y vulnerabilidades expuestas son solo la punta del iceberg.

Varios trabajos han demostrado la posibilidad de atacar la estructura de datos de un automóvil con los métodos tradicionales (sniffing, spoofing) o generando mensajes falsos. El bus CAN puede accederse desde varios puntos de ingreso, directamente desde el puerto OBD2 (On Board Diagnostics) disponible debajo del tablero, o mediante "wire tapping", o sea hackeando los conductores internos.

La primera opción de seguridad es el cifrado de los datos [4], no es de fácil aplicación por las características del protocolo CAN, como ser: la velocidad de transmisión, la latencia y su estructura multimaestro/multicast.

En el escenario descripto, las soluciones parecerían provenir del análisis estadístico de los mensajes que ocupan el bus. De esta forma es posible observar desviaciones con respecto a datos históricos y alertar así de una intrusión externa maliciosa. En este sentido se han expuesto trabajos sobre la detección de anomalías que corresponden tanto a problemas técnicos como a ataques a la seguridad. Los procedimientos que se utilizan incluyen análisis de frecuencias, análisis multivariado de series temporales [5] y los basados en la entropía de la información [6].

Gracias al Big Data están disponibles las grabaciones de miles de vehículos que alimentan una gran base de datos que describe modelos de comportamiento usuales. Con estos datos podrían establecerse patrones de comportamiento, cuyas desviaciones podrían ser analizadas para determinar su origen.

A partir de 2006, un nuevo protocolo llamado FlexRay [7] entró al mercado, con mayor ancho de banda (dos canales de 10 Mbps) y la posibilidad de cifrado. Si bien su implementación es más cara y todavía no es un estándar, demuestra el interés de las grandes fábricas de automóviles de migrar a un nuevo protocolo más seguro.

En base al panorama expuesto y al estado del arte, el proyecto de investigación propone el análisis de las amenazas y vulnerabilidades a los automóviles y la búsqueda de soluciones de seguridad para el parque automotor existente.

2. LÍNEAS DE INVESTIGACION Y DESARROLLO

Como sucede frecuentemente, en nuestro caso el interés por la detección de los problemas de

seguridad en el bus CAN fue un efecto colateral de los estudios e investigaciones realizadas en pos de obtener información de los diferentes dispositivos electrónicos y mecánicos de un automóvil.

El proyecto que propició el presente trabajo fue el de la construcción de un banco de pruebas de motores. Para fabricar un banco de ensayo normalmente se utilizan sensores y dispositivos auxiliares (frenos, sistemas de enfriamiento), pero en las ocasiones en que el dispositivo bajo ensayo conserva el sistema CAN, este puede ofrecer interesante información complementaria.

Se realizaron ensayos sobre algunos modelos de automóviles para estudiar los diferentes sistemas de información de un vehículo. Se decidió, luego de conocer las capacidades de los sistemas, trabajar con tableros de instrumentos, que generan y reciben tramas CAN y también con vehículos completos. Para esta tarea se utilizaron tableros de Volkswagen Suran, Volkswagen Polo, una camioneta Toyota Hilux y una Volkswagen Suran.

El primer paso fue contactar especialistas en electrónica automotriz para que aportaran sus experiencias y expandieran así los alcances de la investigación. Se realizaron varias reuniones con conclusiones sumamente interesantes:

- Crecimiento del equipamiento electrónico en los automóviles (principalmente los de alta gama).
- La complejidad y diversidad de implementaciones en vehículos.
- Los cambios en el perfil del especialista que debe resolver los nuevos problemas relacionados con el ambiente digital.
- Al participar de pruebas de conexión al bus CAN en automóviles, ayudaron a comprender la facilidad con que se ingresa, la variedad de herramientas informáticas que existe para ello y lo potencialmente peligroso que resultaría un acceso malicioso.

Una vez finalizados los relevamientos de campo se comenzó con la elección del dispositivo a ensayar. Era necesario que reciba, transmita y que además muestre los resultados en forma gráfica. En este punto la decisión era utilizar como dispositivo a una ECU o a un tablero de instrumentos. Se optó por el tablero, por ajustarse mejor a los requisitos expuestos.

Si bien existen implementaciones estándar de la mensajería CAN, la mayoría de los fabricantes utilizan, además, mensajes exclusivos. Por eso para establecer comunicación con el tablero, se utilizaron varios métodos, diferentes velocidades y diferentes combinaciones de mensajes.

Para contar con la flexibilidad necesaria para hacer estas pruebas, era necesario tener acceso directo a la trama CAN, tanto para leer como para generar mensajes. Se desarrolló para esa tarea una herramienta informática llamada "CAN Sniffer" que permite la lectura y grabación de mensajes CAN y la generación de tramas. Está basada en el MCP2515, un chip que hace de interfaz del CAN SPI y la plataforma Raspberry Pi (3 y Zero). Los dispositivos se probaron en entornos de maestro/esclavo y de multimaestro con varias ECUs [8].

Una vez que se contó con la posibilidad de capturar mensajes se determinó el procedimiento para leer los mensajes de control de una herramienta especializada de diagnóstico automotriz especial para Volkswagen. De esa forma se pudo descifrar cómo establecía la comunicación con el tablero de instrumentos. Esto permitió determinar el método más eficiente para tratar de que el tablero reaccione a mensajes externos generado por nuestro dispositivo. Luego de probar varias formas, la más eficaz demostró ser el ataque por fuerza bruta aplicado de forma metódica. Se enviaron mensajes generados secuencialmente para luego analizar la reacción del tablero.

Se verificó también, que al cambiar el tablero por uno externamente idéntico, pero del modelo Polo, fue necesario realizar nuevamente todo el procedimiento de detección de mensajes.

Para la grabación y análisis de las tramas recibidas y generadas se utilizó una herramienta de código abierto llamada Wireshark en su versión para Raspbian (sistema operativo de Raspberry Pi)

En paralelo a las pruebas con el tablero, se realizaron ensayos con una Toyota Hilux y un Volkswagen Suran, para verificar que lo que sucedía en las pruebas de laboratorio, se repitiera en vehículos reales.

3. RESULTADOS OBTENIDOS/ESPERADOS

Luego de realizar los ensayos descriptos, los resultados fueron interesantes tanto desde el punto de vista de la lectura de lo que sucede con algunos dispositivos conectados al bus CAN como en lo que respecta a su reacción ante la generación de mensajes. Como se mencionó, si bien existen varios estándares de protocolo CAN (SAE J1708, SAE J1587 SAE J1939), los fabricantes implementan mensajería propietaria y diferentes variantes. Una de ellas es, por ejemplo, el protocolo UDS (Unified Diagnostic Service), un protocolo de capa 7, adaptación del KPW2000 en CAN. Es por eso que la tarea de tratar de encontrar un dispositivo que se comunique con todos los sistemas es casi imposible y explica a su vez la variedad de herramientas, para diferentes marcas y usos que se ofrecen en el mercado.

Dentro de los resultados obtenidos podrían citarse los obtenidos en el desarrollo de dispositivos para llevar a cabo la investigación. En el proceso hacer ingeniería inversa fue necesario desarrollar hardware y programarlo. Los lenguajes de programación utilizados fueron C++ de diferentes plataformas (Arduino, Raspberry, STM32) y Python sobre Raspbian. Los dispositivos desarrollados fueron:

- CAN Sniffer básico
- CAN Sniffer extendido
- Driver de Tablero
- Nodo de relevamiento remoto

En la Figura 1 se pueden ver el tablero con su driver y el Sniffer. Esto fue presentado funcionando en la feria Exproyecto 2019, en la UNLaM.

En cuanto a los resultados de la investigación propiamente dichos fueron, hasta el momento:

- Ingeniería inversa de implementación CAN (estándar y propietaria).
- Inyección de mensajes a tableros y vehículos que interfieren su funcionamiento maliciosamente.
- Lectura de parámetros del automóvil en tiempo real vía ODB2.
- Lectura de parámetros de un automóvil interviniendo su cableado interno.
- Transmisión inalámbrica al nodo de registro de datos operativos de un vehículo.
- La verificación de que la mayoría de los resultados obtenidos en el banco de pruebas, eran consistentes con vehículos reales.

Figura 1: Tablero y Sniffer

El objetivo principal era obtener datos e interactuar con entornos automotrices, pero una de las conclusiones adicionales fue que la seguridad de un vehículo de calle está en riesgo dada la facilidad para ingresar y atacar su lógica de control.

Otra de las conclusiones fue que la electrónica automotriz ya tenía latente estos riesgos, pero las nuevas tecnologías de vehículos "siempre conectados" le suman gravedad. Hace años que la seguridad en los vehículos está en discusión, pero hasta hace poco era necesario el contacto físico para realizar la intrusión. Hoy eso ha comenzado a cambiar. En el horizonte cercano vemos camiones autónomos sin chofer surcando las rutas, por eso es indispensable hacer más robusta la seguridad de los sistemas vehiculares.

Las metas por cumplir para garantizar la seguridad de los datos en sistemas tecnológicos, confidencialidad, integridad, autenticidad, disponibilidad y no repudio; están lejos de ser alcanzadas actualmente por cualquier dispositivo que basa su comunicación en el protocolo CAN [9].

Si bien la industria está trabajando en mitigar estos riesgos en los nuevos modelos, hay un parque automotor existente que está en riesgo. Esta necesidad abre una línea de investigación posible: el desarrollo de un dispositivo conectable al puerto ODB2 para detectar intrusiones maliciosas por comparación estadística. En caso de camiones o vehículos de flota también sería instalable en un bus específico. Por ejemplo, el que

controla el consumo de combustible, el airbag, la dirección o los frenos.

Atacar un dispositivo puede ser tan sencillo como bombardearlo con mensajes y generar lo que se conoce como denegación de servicio (Denial of Service), dejándolo fuera de servicio en forma temporal o definitiva.

Como ejemplo del resultado obtenido en el proceso de ingeniería inversa del tablero de instrumentos de un VW Vento, se puede apagar o encender el testigo del airbag. Si se envía al identificador 80 (0X050 hexadecimal) el dato 0X000001, si X=0 apaga y si X=1 prende. Si se envía al identificador 1136 (0X470) los datos (X0000000) para X=1 prende la luz de giro a la izquierda, X=2 la derecha, X=3 la baliza. Ese mismo identificador maneja la luz del baúl y la luz de batería.

Los códigos para realizar las funciones descriptas se obtuvieron haciendo un barrido de identificadores y enviando datos al azar. También se determinó que, por ejemplo, en el caso del velocímetro, medidor de RPM y de combustible, es necesario enviar una combinación de varias tramas.

En resumen, los resultados obtenidos evidencian una interesante línea de investigación para desarrollar dispositivos no invasivos que aumenten la seguridad de los vehículos.

4. FORMACIÓN DE RECURSOS HUMANOS

El equipo de trabajo de este proyecto está formado por un ingeniero mecánico, un ingeniero electrónico y un especialista en seguridad. Como se mencionó anteriormente, este trabajo se desarrolló en el marco del proyecto de investigación: "Desarrollo de un banco didáctico de ensayo de motores térmicos".

El desarrollo del proyecto de investigación generó varias líneas de trabajo, de múltiples disciplinas. Dada la complejidad, fue necesaria la colaboración de varios expertos con amplia experiencia en la industria y la investigación académica, tanto en la parte de electrónica automotriz, como de bancos de prueba de motores y de seguridad de datos.

Uno de los miembros del equipo de investigación se encuentra desarrollando su trabajo de tesis de posgrado de la Maestría en Informática de la UNLaM titulada: "µFramework: marco de referencia para desarrollos de sistemas embebidos" y su tutor es el Mg. Jorge Eterovic, integrante del proyecto de investigación [10].

5. BIBLIOGRAFIA

[1] Protocolo de Comunicaciones CAN; http://www.bosch-semiconductors.com/en/ubk_semiconductors/ip_modules_3/produkttabelle_ip_modules/can_literature_1/can_literature.html. 2019.

[2] Greenberg A., Hackers Reveal Nasty New Car Attacks — with me behind the wheel, https://www.forbes.com/sites/leemathews. 2019.

[3] Kapersky E., Viruses coming aboard? - https://securelist.com. 2005.

[4] Weimerskirch A., Wolf M., Wollinger T., State of the art: embedding security in vehicles, *Horst-Gortz-Institute for IT Security, Ruhr-University Bochum, Universitatsstraße, 44780 Bochum, Germany*. 2007.

[5] Theissler A., Anomaly detection in recordings from in-vehicle networks, IT-Designers GmbH, Esslingen, Germany, 2014.

[6] Marchetti M., Stabili D., Guido A., Colajani M. Evaluation on anomaly detection for in-vehicle networks through information-theoretic algorithms, University of Modena and Reggio Emilia, Italy, 2015.

[7] Hartzell, S., Stubel C. Automobile CAN bus network security and vulnerability, University of Washington, Seattle, USA. 2018.

[8] Miller C., Valasek C. Car Hacking: for poories, http://illmatics.com/car_hacking_poories.pdf. 2018.

[9] Hoppe T., Kilz S., Security threats to automotive can networks – practical examples and selected short-term countermeasures, Otto von Guericke University of Magdeburg, Germany. 2010.

[10] Fourcade A., Eterovic J., Pérez A., Rodofile G., µFramework: marco de referencia para desarrollo de sistemas embebidos; CoNaIISI 2019; RIISIC-CONFEDI-UNLaM, San Justo. 2019.

UNPA RedUNCI WICC 2020

Universidad Nacional de la Patagonia Austral
XXII WORKSHOP DE INVESTIGADORES EN CIENCIAS DE LA COMPUTACION 2020

Se certifica que JORGE E. ETEROVIC **(UNLAM)** ha participado en calidad de autor del artículo **ANÁISIS DE LA SEGURIDAD DEL PROTOCOLO DE COMUNICACIONES CAN (12781 - SI)** aceptado en el XXII WORKSHOP DE INVESTIGADORES EN CIENCIAS DE LA COMPUTACIÓN – WICC 2020, organizado por la Universidad Nacional de la Patagonia Austral - Junio 2020.

Lic. Patricia Pesado
Coordinadora
RedUNCI

Ing. Hugo Santos ROJAS
Rector
UNPA

CERTIFICADO Nº 325 /2020 /UNPA

Un Análisis de los Aportes de Agile al Desarrollo de Sistemas Embebidos

Alejandro Fourcade, Jorge Eterovic
Departamento de Ingeniería e Investigaciones Tecnológicas
Universidad Nacional de la Matanza
afourcade@unlam.edu.ar, eterovic@unlam.edu.ar

Resumen

La aplicación de modelos para optimizar resultados de los procesos de desarrollo de software ha diversificado la forma en que se genera código para sistemas informáticos. La búsqueda de caminos más rápidos y flexibles, que mejoren la respuesta ante necesidades dinámicas del cliente, trajo nuevos marcos teóricos de trabajo. Los beneficios generados por el reciente cambio de percepción del proceso de desarrollo, no alcanzaron a todos los campos por igual y en especial no encontraron aplicación en los sistemas embebidos. Argumentos tecnológicos sumados a numerosas variaciones de entorno, expandieron el campo de acción de los sistemas integrados y obligan hoy a repensar la conformación de sus procesos. Se propone, entonces, analizar las ventajas que podría aportar la metodología Ágil al desarrollo de embebidos. Este documento teoriza sobre las causas de esta disociación, sus efectos y propone opciones operativas que compatibilicen ambos mundos, hoy en la práctica, distanciados.

Introducción

El 12 de febrero de 2001 Kent Beck, convocó a un grupo de 17 desarrolladores que se reunieron para debatir sobre el futuro de la generación de software. Ese fue el nacimiento de la metodología Ágil, que formalizó prácticas y unificó criterios para dar respuesta a algo que la experiencia había dejado al descubierto: ciertos tipos de escenarios necesitaban otros modelos más flexibles y efectivos que los que ofrecían los métodos aplicados hasta ese momento. En la década del 90, modelos alternativos de gestión de proyectos comenzaron a rivalizar contra los llamados métodos tradicionales o "pesados". Estas formas alternativas de interpretar los nuevos, complejos y cambiantes requerimientos del mercado tenían características que los diferenciaban de los métodos pesados, que eran cuestionados por tener regulación excesiva, documentación intensiva, ser micro gerenciados y no tener una visión cenital de los proyectos. Los nuevos métodos livianos prometían ser más dinámicos, adaptativos, rápidos y eficientes.

El manifiesto Ágil, da un marco conceptual para que muchas metodologías se manejen dentro de sus mandamientos, de formas diferentes e innovadoras. Estas ideas rectoras, ya famosas, marcan un conjunto de aseveraciones que todo método ágil debiera respetar:

- Individuos e interacciones sobre procesos y herramientas.
- Software funcionando sobre documentación extensiva.
- Colaboración con el cliente sobre negociación contractual.
- Respuesta ante el cambio sobre seguir un plan.

Las metodologías ágiles, combinan visiones no tradicionales que se comenzaron a gestar antes que el 2001, año de publicación del manifiesto. Si bien parecieran conceptos disruptivos para la época, no es otra cosa que la suma y consolidación de algunos métodos livianos de los 90s. Los métodos ágiles incorporan conceptos de RAD (Rapid Application Developement), UP (Unified Process), DSDM (Dynamic Systems Developement Method), Scrum, Crystal Clear y XP (Extreme Programming).

La velocidad con que hoy se desarrolla un producto tecnológico, la constante superación e innovación de su rendimiento y la exigencia de un tiempo de salida al mercado optimizado generó la necesidad de procedimientos de producción de software más ligeros. Por eso la alternativa Ágil ganó adeptos rápidamente, sobre todo en proyectos "express" o que cambian sus alcances dinámicamente. Los proyectos con requisitos iniciales borrosos o inciertos en los que los objetivos a satisfacer pueden ir modificándose a lo largo su desarrollo, encontraron en Agile una solución procedimental que permite evitar el lastre burocrático de los métodos tradicionales.

Los sistemas de desarrollo como cascada, espiral o incremental, han mostrado su eficacia en procesos críticos y complejos como aplicaciones financieras, aeroespaciales, de defensa y también, en sistemas embebidos. Todos ellos tienen factores comunes que los hacen ideales para estos tipos de desarrollos: los procesos marchan según planificaciones realizadas en el comienzo del proyecto y tienen abundante documentación funcional, de control de avance y de seguimiento financiero. La regla de oro a respetar son los plazos de entrega parciales y totales y la interacción con el cliente es idealmente mínima, solamente al inicio y al final del proceso.

Generalmente en los métodos "pesados", la forma de gestionar un proyecto se resume en realizar los

relevamientos para determinar alcances y características de la aplicación, negociar con el cliente cuestiones financieras y de tiempos de entregas, desarrollar la solución a puertas cerradas y presentar el producto. Si existen errores, se corrigen y si fueran necesarios cambios o funcionalidades adicionales se presupuestan y planifican dentro de la etapa de mantenimiento.

En el universo Ágil, se ofrecen opciones más actuales, adaptadas al medio, y proponen posicionar al cliente en el centro de la escena, modificando los alcances y especificaciones según sus necesidades durante el transcurso de proyecto. Según la variante a utilizar, se ejecutan ceremonias con diferentes funciones y procedimientos iterativos de producción. Existen figuras y roles diferentes y las guías de seguimiento y control son diversas, pero mantienen la esencia que es ofrecer un camino ordenado para obtener un producto en tiempo corto, con mayor adaptabilidad durante todo el proceso, mayor eficiencia, con un alto grado de interacción y satisfacción del cliente.

La metodología ágil ha pasado por diferentes fases: adhesión, expectativa, decepción y estabilidad. Las desventajas de su aplicación están constantemente bajo análisis y tomando como referencia el ciclo de hype de Gartner [1] se podría decir que está en la etapa de meseta de productividad. Como todos los cambios significativos, cosecha en su evolución seguidores y detractores, y hoy los métodos ágiles se encuentran maduros para dar certezas en cuanto a sus bondades y el precio que conlleva aparejado obtenerlas.

1.1 Definiciones

Los sistemas embebidos pueden definirse de varias formas y su clasificación se ha diversificado en las últimas décadas, pero en su definición formal, son sistemas desarrollados sobre plataformas que no poseen sistema operativo, se ejecutan sobre hardware dedicado y fueron diseñados para un objetivo determinado. Se caracterizan por tener código con poco o nulo nivel de abstracción sobre el hardware, lo que comúnmente se denomina trabajar sobre "bare metal". Una de las características importantes, por lo menos en lo que respecta a este documento, es que un sistema embebido se compone de hardware y software, no solamente software. No se puede pensar un desarrollo que mantenga al hardware y al software por caminos independientes con la idea de integrarlos en la etapa final. Generalmente, y casi como una regla de oro de sistemas integrados, se desarrolla primero el hardware y luego el software asociado. Dentro de los módulos de software que forman un sistema embebido, se encuentran el firmware y software de aplicación. El firmware es aquel que interacciona con el hardware directamente, para configurarlo y el software de aplicación es lo que relaciona al producto con el usuario. No necesariamente el firmware actúa como middleware entre el hardware y el software de aplicación, en muchos casos ambos se ejecutan en forma paralela.

La profunda relación entre el código y el circuito donde se ejecuta, además de la creciente variedad de dispositivos que pueden seleccionarse a la hora de diseñar, hace que los sistemas embebidos deban tratarse con reglas y cuidados particulares. Hoy, son pocos los equipos de programación que imaginan un proyecto de embebidos con la alteración del orden de desarrollo de hardware y software. Esta premisa genera una secuencia inevitable en las tareas del proyecto, la de generar los prototipos para luego comenzar el desarrollo del código.

Otro punto principal a analizar, es que en el proceso de desarrollo de productos integrados se requiere la interacción de muchos conocimientos diferentes y los grupos de trabajo suelen ser multidisciplinarios y de muy alto grado de especialización. Por eso la articulación y coordinación de estos grupos, suele ser diferente a los de los sistemas de desarrollo de software para plataformas tradicionales. El nivel de abstracción de un programa que funciona sobre un sistema operativo determinado, aporta un mayor grado de independencia sobre el hardware, normalización en el uso de recursos, arbitraje en el acceso y seguridad de operación, entre otras ventajas. Estas características brindan una plataforma operativa segura y estable, virtudes fundamentales para facilitar el desarrollo. Tanto los teléfonos celulares, que escaparon a la definición de sistema embebidos, como las computadoras ofrecen entornos amigables y blindados para las aplicaciones, con un desentendimiento casi total del hardware en que se ejecutan.

El proceso de diseño de sistemas embebidos, es más sensible y crítico en lo que respecta al acceso a los recursos del hardware. Una falla en la programación puede anular el funcionamiento o poner en peligro al usuario del sistema. No existe una capa de abstracción que arbitre y maneje las consecuencias que puedan ocasionar la ejecución de una aplicación con problemas en su código. Esa es solo una de las razones por las cuales el desarrollo del hardware y el software no pueden transitar carriles separados. Ambos se integrarán en un sólo dispositivo y se debe asegurar desde la génesis del producto una operación con la menor cantidad de fallas posibles. Una aplicación en una PC, por ejemplo, puede incorporar fixes, patches o actualizaciones, pero el firmware de un horno microondas no tiene esa posibilidad y debe funcionar desde el primero hasta el último día de su vida útil. Aunque el hardware fuese excelente, un mal código arruinaría el producto, y viceversa.

Una vez definido el diseño del hardware, se mantiene constante, salvo en caso de correcciones. Si se desea incorporar una modificación o agregar una funcionalidad, es posible hacerlo siempre y cuando no modifique el circuito. Esta limitación, dificulta la adaptabilidad y la permeabilidad a los cambios en el transcurso del proyecto.

Si bien los lenguajes de programación utilizados en sistemas embebidos siguen siendo, en la mayoría de los casos, los tradicionales (C, C++, Assembler), seguidos de lejos por Java o Python, la experiencia y el conocimiento que aporta cada proyecto al grupo de trabajo se capitaliza en menor medida, dado que el hardware y las herramientas de desarrollo pueden variar en cada proyecto. Hace algunas

décadas era normal resolver todos los diseños con el mismo microcontrolador y el mismo entorno de desarrollo. Hoy, elegir esa política sería un error, porque limitaría alcances y potencialidades de un diseño, restándole competitividad. Cada nueva especificación debe, como parte del análisis técnico/funcional inicial, estudiar qué plataforma es la indicada para resolver más eficientemente los problemas y las metas de diseño. Podar los requisitos deseables según las limitaciones de una plataforma, por el solo hecho de estar familiarizados con ella, pone en riesgo la efectividad y novedad de un producto al momento de salir al mercado.

El tiempo que involucra el diseño y la construcción del hardware, suele ser uno de los hitos que limita las fechas de entrega en plazos cortos. Aún con el prototipo finalizado y varias unidades disponibles, el trabajo coordinado de diferentes grupos sobre el mismo diseño de hardware es dificultoso. La integración en general es uno de los pasos más riesgosos ya que puede llegar hasta a anular la factibilidad técnica de producción de un diseño que llevó tiempo y recursos. Este es uno de los principales escollos para la incorporación de metodologías más actuales y eficaces al proceso de generación de código embebido.

Aumentar la abstracción, en general no está bien visto por quienes generan software embebido. Incrementar la distancia entre hardware y software, implica una pérdida del control directo del sistema y dependencia de código de terceros. Cuando se utiliza código de bajo nivel, las órdenes son ejecutadas directamente contra el hardware, al aumentar la abstracción, aumenta también la indeterminación. La imposibilidad en muchos sistemas embebidos de actualizar el firmware hace que, por ejemplo, errores de baja ocurrencia, multipliquen su peligrosidad en productos de producción masiva. Dada la criticidad del buen funcionamiento del código, las pruebas en embebidos son exhaustivas, y comprenden trazabilidad del hardware (series de fabricación), pruebas de esfuerzo, repetibilidad, bajo condiciones adversas (alta temperatura, baja tensión, fluctuación de alimentación, ambientes húmedos).

La incertidumbre inevitable que genera acceder a los recursos de hardware a través funciones, debe tenerse en cuenta a la hora de programar módulos críticos. En algunos casos el mismo fabricante del hardware provee las rutinas, pero en otros casos lo hace el proveedor del periférico u otros programadores. Estas piezas de software serán como engranajes de un reloj y deben ser confiables.

Simplificar y facilitar las funciones en muchos casos ayuda a implementar más rápidamente, pero casi siempre, implica delegar parámetros de configuración y por ende perder opciones de optimización. No contar con todo el abanico de posibilidades, conlleva aparejado una limitación de prestaciones.

Por las razones descriptas, la selección y configuración de las plataformas de desarrollo son puntos críticos. Al utilizar diferentes fabricantes y familias de dispositivos, es inevitable el cambio de plataformas de desarrollo.

Existen IDEs (Integrated Developement Environment), por ejemplo, Keil o Eclipse, que son compatibles con varias familias de dispositivos y ofrecen un entorno único de trabajo. Estas herramientas son valiosas ya que permiten programar para varios microcontroladores y unificar la plataforma. En otros casos cada fabricante, ofrece su propia IDE de desarrollo, por ejemplo, Atmel ofrece Atmel Studio, una completa herramienta gratuita para su familia dispositivos.

El caso de STMicroelectronics es una muestra de cómo los fabricantes se adaptan a las nuevas reglas. Para posicionar su línea STM32 como una opción alterativa y superadora al popular Arduino, compró a la firma Atollic[1] e incorporó su herramienta de desarrollo TrueStudio, integrándola a Cube MX, su aplicación de selección de hardware y generación de entorno. Configuró así un IDE que puso a disposición sin cargo. Para captar la atención de los numerosos usuarios de la IDE de Arduino, financió también el desarrollo de bibliotecas para que su STM32 pudiera usarse en ese entorno. Con esa visión poniendo énfasis en la compatibilidad, integración y precios competitivos, STMicroelectronics logra posicionar a su microcontrolador de 32 bits en la lista de los más utilizados para desarrollos no específicos.

El advenimiento de Arduino cambió las reglas del mercado, facilitando el acceso al mundo de los microcontroladores con tres pilares: hardware de bajo costo, herramientas de desarrollo gratuitas y sencillas y amplia disponibilidad de código y documentación de ejemplo. Entre los logros de Arduino está el de permitir que alguien que no tuviese la menor idea de microcontroladores, pudiera programarlos copiando código disponible y lograra hacer funcionar sus desarrollos.

La programación y el diseño con microcontroladores fue durante muchos años una disciplina reservada a especialistas o programadores con profundo conocimiento de electrónica digital. La perspectiva no fue mejorando en los primeros años, sino todo lo contrario, el hardware se hacía cada vez más completo y complejo y las herramientas de desarrollo más caras. La idea de comenzar a diseñar en una plataforma, era una decisión que traía aparejada altos costos, tanto de horas como de dinero. La situación actual es muy diferente, los kits de desarrollo son extremadamente accesibles y se simplificó su programación, a veces a costa

[1] STMicroelectronics adquiere Atollic, en 7 millones de dólares en diciembre de 2017.

de aumentar la abstracción. La reutilización y adecuación de código, es una herramienta poderosa, más aún para el que se inicia. Por ejemplo, en los microcontroladores ARM de STMicroelectronics, se cuenta en la plataforma de desarrollo con un extenso conjunto de rutinas provistas por el fabricante llamadas HAL (Hardware Abstraction Layer). Estas rutinas en conjunto con las aplicaciones de generación de entorno y configuración gráfica han simplificado y fortalecido la codificación, ya que mantiene el código a bajo nivel y sólo normaliza y simplifica el acceso. Este concepto, no es novedoso, es asimilable a las rutinas de BIOS del microprocesador Intel 8088.

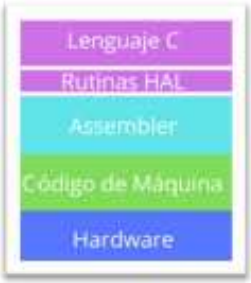

Figura 1: Capas de abstracción

En la **Figura 1** se ve dónde se posicionan las rutinas HAL, en el stack de lenguajes. Aun así, en concordancia con lo que se mencionó anteriormente sobre la mala fama de la abstracción en embebidos, muchos programadores evitan o minimizan el uso de estas rutinas en sus códigos y su utilización sigue siendo materia de discusión.

Las cuestiones descriptas son sólo algunas de las tantas que caracterizan las peculiaridades y diferencian a los sistemas embebidos de otras aplicaciones. Es por ello, que las herramientas que facilitan y optimizan la gestión de proyectos de software de extenso uso en los entornos tradicionales, no encuentran un camino de aplicabilidad en el ámbito de los sistemas integrados.

Lo que se propone en este documento es realizar un resumen de los cambios y evoluciones acaecidas en los últimos años y analizar si estos hechos modificaron el entorno de forma tal que puedan acercar a los sistemas embebidos a las prácticas modernas de gestión de proyectos. Las preguntas que se proponen desde este documento y servirán como guía son:

- ¿Es el ideario del manifiesto ágil aplicable en todo o en parte al escenario embebido?
- ¿Pueden las virtudes ágiles, aportar visibilidad y adaptabilidad?
- ¿Puede mejorar la interacción del grupo de desarrollo entre sí y con el cliente?
- ¿Si no ha sido aplicable hasta ahora, por qué comenzaría a serlo?

Para contestar estas preguntas, es necesario analizar primero los factores de cambio y la conformación de los escenarios de aplicabilidad.

1.2 Razones que ralentizan en cambio

Las causas para que la incorporación de nuevas metodologías de trabajo no se produzca rápidamente en los sistemas embebidos son múltiples. Se enumerarán los principales argumentos, detallando luego los inconvenientes que aportan cada uno.

- No se pueden obtener modelos funcionales en plazos cortos.
- La integración continua no es posible.
- Las pruebas automáticas son complejas de implementar.
- Los equipos son multidisciplinarios.
- Los cambios durante el proceso de desarrollo no son factibles.
- El tiempo de los ciclos de iteración no es compatibles.
- Las historias de usuarios no aplican en este entorno.
- Primero se desarrolla el hardware luego el software.
- Cambiar la forma de trabajo sin asegurar resultados puede resultar un esfuerzo inútil y costoso.

No se pueden obtener modelos funcionales en plazos cortos: Una de las características de los métodos Agiles es la entrega continua, concepto que va asociado al denominado MVP (Minimun Viable Product). Esto permite despachar productos totalmente funcionales en varias instancias del proceso. Los tiempos de desarrollo de hardware, elección de la tecnología, diseño, fabricación e implementación hacen imposible la obtención de modelos funcionales en plazos cortos. A primera vista, pensar en desarrollos embebidos con entrega continua no es posible en las condiciones actuales.

La integración continua no es posible: El concepto de integración continua, cambió la visión sobre las implementaciones y facilita la unificación de los desarrollos. En el caso de embebidos, la integración continua se dificulta ya que la automatización de las pruebas y los la complejidad de los ciclos de corrección, que incluyen tanto al hardware como al software, son incompatibles con su aplicación. La visión de estos procesos, plasmadas en algunas metodologías, como por ejemplo DEVOPS, cuentan con otras condiciones de entorno que hacen favorable su implementación. Las trabas de los largos ciclos de modificación de prototipos o la compleja corrección y depuración en los sistemas embebidos, parecen mostrar a la integración continua como algo imposible.

Las pruebas automáticas son complejos de implementar: la automatización de las pruebas de software ha avanzado notablemente en los últimos años permitiendo un proceso continuo de generación, test y entregas. Sin embargo, la inclusión de hardware dificulta la posibilidad de generar un conjunto de tests automáticos que aseguren el correcto funcionamiento, por lo menos a nivel de permitir la entrega de un producto funcional al cliente. Proponer

instancias de pruebas automáticas a entornos embebidos no es algo que sea usual, dado que los algoritmos de prueba no tienen en cuenta la interacción mecánica con el hardware

Los equipos son multidisciplinarios: a diferencia de los desarrollos tradicionales, en los sistemas embebidos es necesaria la integración de expertos en distintos campos. Especialistas en electricidad, electrónica, mecánica, hidráulica, neumática, mecatrónica, intervienen frecuentemente tanto el en diseño del hardware como el del software. Normalmente la fusión de estas disciplinas está coordinada por un líder de proyecto. Los métodos ágiles aconsejan grupos de trabajos reducidos, de grandes cualidades técnicas y elevada capacitación, con un alto grado de interacción entre sus miembros y sin la estructura jerárquica tradicional.

Los cambios durante el proceso de desarrollo no son factibles: uno de los pilares del método Ágil es poner al cliente en el centro de la discusión y darle las potestades de intervenir en el proceso de desarrollo y de cambiar las especificaciones y alcances dinámicamente. A simple vista estas características no son compatibles con un proceso de desarrollo en el cual intervengan hardware y software como una unidad monolítica. Sería impensable cambiar a voluntad del cliente el diseño del hardware durante el desarrollo de proyecto. Integrar el conocimiento del cliente al conjunto de saberes del grupo de desarrollo, es solamente posible a través de la toma de requisitos correcta, proceso que se realiza antes de comenzar el desarrollo.

El tiempo de los ciclos de iteración no son compatibles: en la metodología Ágil, los ciclos iterativos de entrega o sprints tienen una duración aconsejada de entre 2 y 3 semanas. Al finalizar cada ciclo, o de un número pactado de ellos, se debería contar con un producto funcional. Estos tiempos no son posibles en embebidos dado que el tiempo de fabricación del hardware excede las 2 o 3 semanas extendiéndose a veces a varios meses.

Las historias de usuarios no aplican en este entorno: las historias de usuario son en la metodología Scrum la presentación de requisitos, expresados de forma simple y directa. Se generan a lo largo del proyecto y son administradas por el Product Owner que representa la voz del cliente dentro del proyecto. Es la forma de tomar y actualizar requisitos de los métodos Agiles. En embebidos, como se mencionó, no es posible cambiar los requisitos en la mitad del proyecto, las especificaciones deben ser detalladas al inicio del proceso y mantenerse sin cambios hasta la finalización. Por esa razón, la inclusión del cliente, como componente activo del equipo de trabajo, tampoco es posible.

Primero se desarrolla el hardware luego el software: es imposible pensar en embebidos la posibilidad de desarrollar en paralelo hardware y software o peor aún escribir código para un sistema que todavía no existe. El orden lógico es que una vez que esté desarrollado y construido el software, se desarrolle el firmware o el código de aplicación que comandará ese hardware. De otra forma, los tests y las pruebas funcionales serían imposibles. Gran parte del proceso de desarrollo y depuración se basa en un ciclo de prueba y error en conjunto con el hardware.

Cambiar la forma de trabajo sin asegurar resultados puede resultar un esfuerzo inútil y costoso: al momento de comenzar un cambio deben establecerse metas, objetivos y perspectivas de mejora, para incentivar y motivar a los actores que serán el motor del proceso. Ante la incertidumbre del resultado y las ventajas que se lograrán luego de aplicar un proceso de cambio, parecería no tener sentido arriesgarse a llevarlo a cabo. Hasta hace unos años, no había grandes jugadores que hayan tomado a Ágil como metodología y hayan experimentado mejoras, como para inspirar y motivar al resto del mercado.

1.3 Factores de Cambio

Si bien las razones expuestas dejan claro que las metodologías de desarrollo de ambos mundos deben ser necesariamente distintas, hay factores que comenzaron a acercar diferencias y deben ser tenidos en cuenta. La idea de los métodos Agiles es la de dinamizar procesos, agregar valor, aumentar la eficiencia y dar mayor visibilidad y adaptabilidad que la que ofrecían los marcos de trabajo tradicionales. Pero, aun así, al tratar de aplicar algunas de sus implementaciones como Kanban, Scrum o Extreme Programming a embebidos, "out of the box" o tal como se lo aplica a desarrollos tradicionales, parece no arrojar resultados satisfactorios. El interrogante no es si el proceso de sistemas embebidos puede adaptarse a alguna metodología ágil, sino debería ser, qué factores de esas metodologías pueden aportarle cambios positivos. Adaptar las implementaciones ágiles y consensuar alteraciones en post de flexibilizar y dinamizar los procesos actuales debería ser la llave para incorporarlos gradualmente.

Los hits y búsquedas en internet sobre métodos ágiles han disminuido en el tiempo [1] y se ha circunscripto su uso a las áreas donde aporta valor. Por esa razón no tiene sentido tratar de utilizar a la fuerza métodos ágiles, sólo por el hecho de seguir una tendencia, sino que es indispensable preguntar qué parte de sus reglas y principios suman a la dinámica de los proyectos.

La intervención de factores que modifican las condiciones de entorno, hace necesario replantear el estado actual de los procesos de desarrollo. Estos avances actúan como factores de cambio y cohesión que modifican el análisis de factibilidad de aplicación de nuevos marcos de trabajo a los métodos en uso. Entre estos factores de cambio podemos mencionar los siguientes:

- Desarrollo en base hardware modular.
- Disponibilidad de prototipos rápidos y económicos.
- Posibilidad de definir y generar MVPs totalmente funcionales.
- Simulación de hardware con interacción de desarrolladores.
- Grupos de trabajo con integrantes en centros distribuidos.
- Optimización de tiempos de armado de prototipos.

- Mejores herramientas de automatización de tests.
- Nuevas prácticas de programación (Code refactor, pair programming, nuevos estándares de programación).
- Amplia disponibilidad de kits de desarrollo a bajo precio.
- Visión de los fabricantes para enfocar sus esfuerzos en adaptarse y compatibilizar sus productos a nuevos estándares.

La forma de generar sistemas integrados ha cambiado, desde la concepción y el diseño del hardware hasta la creación e implantación del código. El concepto de salir desde la hoja en blanco hasta el modelo terminado, va quedando en desuso, y los ingenieros de diseño hoy son integradores de partes. Los fabricantes de dispositivos están adoptando una visión modular de sus productos ofreciendo kits de desarrollo económicos que pueden ser utilizados como unidades funcionales que se suman fácilmente formando modelos más complejos. Las grandes marcas, fabrican y alientan la producción de kits que contengan sus productos, por lo que en los últimos años, la posibilidad de acceder a hardware de desarrollo se ha incrementado exponencialmente, multiplicando la oferta de microcontroladores de todo nivel y haciendo diversa la variedad de accesorios y periféricos disponibles.

El desarrollo sobre hardware modular aumenta las probabilidades de éxito en el diseño de prototipos, ya que se parte de subsistemas que funcionan y están extensamente probados y en la mayoría de los casos cuentan con código de aplicación de ejemplo. Esta ventaja, junto con la optimización de tiempos de entrega de las placas de circuito impreso y la opción de entregarlas armadas hace que se pueda contar con prototipos funcionales en plazos cortos.

Por otro lado, la posibilidad de tener modelos complejos y precisos de simulación 7x24 para todos los ingenieros de diseño, donde se pueda interactuar en forma ordenada y eficiente, es comparable a la revolución que generó el concepto de una computadora en cada escritorio, poniendo un prototipo en cada escritorio.

La facilidad que aportan las comunicaciones actuales permite generar reuniones con recursos distribuidos en diferentes partes del mundo, casi sin notar la distancia y con total comodidad. La mejora de la conectividad y las aplicaciones de reuniones virtuales posibilita generar reuniones frecuentes de feedback y reporte como las que implementan las metodologías Agiles.

En resumen, ante una profunda modificación de las reglas de juego en lo que respecta a los recursos que pueden influir en el proceso de desarrollo, se hace indispensable replantear la forma de planificar y generar sistemas embebidos.

1.4 Marco de aplicación

Las nociones que contiene este documento desde el punto de vista operativo, están pensadas para desarrollos de pequeña escala con grupos reducidos de trabajo. Incluye también un escenario frecuente en el ámbito de embebidos, el del programador solitario. Uno de los valores ágiles es la preferencia de "individuos e interacciones sobre procesos y herramientas" y por ejemplo si aplicáramos Scrum no tendría sentido tener un grupo de una sola persona. Ese único componente debería ocupar todos los roles (Product Owner, Scrum Master y equipo de programadores), y debería realizar las "Stand up Meetings" de inicio de jornada parado en soledad. Las ceremonias y rituales de cada implementación de Ágil son de cumplimiento obligatorio y resultan ser un conjunto de reglas rígidas para un marco conceptual destinado a flexibilizar los métodos tradicionales de programación. Kanban en este sentido es algo más adaptable y es más fácilmente asimilable al entorno embebido.

La idea de este documento nace en el ámbito académico. Cuando se enseñan las bases de los sistemas micro controlados en las universidades, es frecuente realizar proyectos de aplicación. Los proyectos individuales suelen resultar de menor complejidad y cuando se hacen grupales, los problemas más frecuentes son la división equitativa de tareas y asegurar la finalización de proyecto.

En un caso testigo, en una cátedra avanzada de ingeniería electrónica, se trató de utilizar una adaptación de prácticas ágiles. Para ecualizar las responsabilidades, se implementó un esquema de distribución de tareas, dividiendo el proyecto en módulos con entrada y salida normalizada, para facilitar la integración. Se aplicaron directivas de Agil y de TDD (Test Driven Developement). En este esquema el docente hace los roles equivalentes de Product Owner y de Scrum Master y según el avance del proyecto, puede agregar, complicar, simplificar o eliminar características funcionales.

El resultado de esta experiencia fue altamente positivo, ya que se pueden realizar proyectos de mayor complejidad, los alumnos aprenden a cumplir objetivos y a respetar formatos y normas de trabajo, que son fundamentales para su futuro profesional. Ante los buenos resultados, se continúa perfeccionando el método, expandiendo su alcance a desarrollos de pequeña escala, de características dinámicas y alcances específicos.

Algunas veces lo que pasa en un aula es el reflejo de lo que sucede en la realidad. En varias ocasiones, cuando se consultó a los alumnos quiénes querían hacer el trabajo grupal y quiénes individual, aproximadamente un 90% votó que prefería hacerlo en forma individual. Esto se debe, en gran parte, a no conocer los beneficios que puede aportar el trabajo en equipo coordinado con una visión específica. Tratar de aplicar métodos ágiles con este panorama es todo un reto y puede no tener los resultados esperados. En la industria sucede lo mismo, tratar de aplicar métodos ágiles en un entorno altamente normalizado y burocratizado, atenúa las ventajas de adaptabilidad y dinámica y resalta las desventajas de falta de control de los procesos y poca documentación. Si bien las ideas ágiles pueden modificar el entorno y hacerlo más responsivo y adaptado a cambios, a veces no logra modificar las reglas o costumbres impuestas.

2) Planteamiento del Problema

En la década del 70, Texas Instruments ponía en el mercado los primeros microcontroladores, la serie TMS1000, procesadores que con los que venía equipando sus calculadoras desde 1972. La posibilidad de contar con un dispositivo con memoria RAM interna, que pudiese ser programable por el usuario (mandando el programa al fabricante, para que este lo programe), abría un panorama amplísimo de aplicación. Este chip fue usado dentro de instrumentos de medición, electrónica automotriz hasta para darle vida al famoso juguete Simon. En 1980 Intel sacó a la venta su microcontrolador 8051, el procesador con la vida más larga en el mercado, que llega aún hasta nuestros días. Existen hoy variantes y diferentes modelos de 8051 y se sigue usando para nuevos productos.

Pensar en un procesador que mantenga su vigencia luego de 40 años, en el entorno de PCs de escritorio o laptops sería impensable. Más aún si analizamos que la velocidad de reloj se ha mantenido hasta nuestros días. Es normal programar un chip en 8 o 16Mhz, la velocidad de los primeros modelos. Una de las causas de que los microcontroladores no tengan el avance arrollador de los microprocesadores es que el objetivo para lo que fueron creados, no ha variado demasiado en estos años. La necesidad de resolver problemas puntuales como manejar una heladera, una cafetera a monedas, un molinete de subte o medir el nivel de combustible puede satisfacerse con creces con la velocidad de los microcontroladores más básicos. La posibilidad de solucionar rápida y eficientemente estos problemas, en plataformas maduras en las que los programadores se hayan familiarizado y hayan consolidado su experiencia es una ventaja importante. La confiabilidad y estabilidad del código en sistemas embebidos es vital para asegurar el éxito del producto.

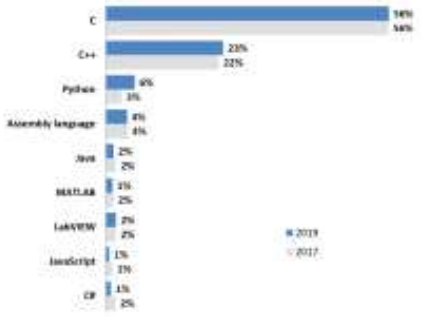

Figura 2: Lenguajes en embebidos.
Recuperado de https://www.cnx-software.com

El fenómeno descripto para la evolución del hardware se aplica también hoy a los lenguajes de programación que se utilizan para programar sistemas embebidos. Los lenguajes de programación más utilizados son C, C++ y Assembler. El lenguaje C data de 1972, C++ de 1979. Las aplicaciones críticas, kernels, sistemas de bajo nivel están programadas en C y C++. El C++, informalmente denominado C con clases, es una evolución a lenguaje orientado a objetos de C y si bien son diferentes comparten gran parte de su vocabulario. Una de las razones principales de esta no evolución de los lenguajes es que los sistemas embebidos necesitan control a nivel de bit y prefieren lenguajes como el C que mantienen cercanía con la arquitectura de hardware.

En resumen, el comportamiento evolutivo conservador de los entornos de programación de los sistemas embebidos (Ver **Figura** 2), es diferente al de los sistemas de PC y más parecido al del entorno mainframe, donde la previsibilidad es un punto clave.

Un cambio clave ha sido la evolución en los requisitos de los productos y su apertura e interconexión. Se ha sumado conectividad, integración y generación de datos, en algunos casos en tiempo real y ha multiplicado las interfaces con otros sistemas y con la nube. Los métodos tradicionales parecen no dar respuesta a algunos campos de aplicación de los sistemas integrados, que como se mencionó aumentaron sus prestaciones reales y potenciales, aumentando su complejidad. Se han realizado varios estudios e investigaciones sobre la aplicación de métodos ágiles al desarrollo de embebidos [10, 5] y las conclusiones son parcialmente coincidentes. Es necesario solucionar algunos conflictos del dominio embebido antes de poder aplicar métodos ágiles eficientemente. Los estudios analizan los resultados de los métodos, que porcentaje del mercado los utiliza, que tipo de práctica aplica y las perspectivas de cambiar de metodología en el corto plazo. Es verdad que los ciclos cortos y de fecha inamovible de SCRUM, no favorece el desarrollo de software y hardware en paralelo, o que permitir los cambios de requisitos dinámicamente de XP (Extreme Programming) puede complicar la interacción y los alcances de la relación software hardware. Pero la idea de esta propuesta es capitalizar la aplicación de los valores ágiles que pueden dar ventajas en el desarrollo embebido, por ejemplo, la posibilidad de coordinar acciones entre programadores con un objetivo común, la revisión de objetivos en plazos cortos, la colaboración del cliente en la especificación del producto y la relación de Agile con el concepto de MVP (Minimun Viable Product).

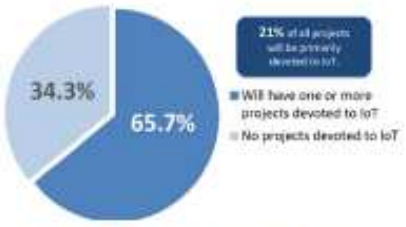

Figura 3: IOT en proyectos futuros
Recuperado de https://www.embedded.com

En la **Figura 3** se puede ver la fuerte influencia del IOT, en los futuros desarrollos. En la misma encuesta [15] se determinó que los temas de mayor desarrollo son visión inteligente, comunicación oral, realidad aumentada y virtual y que el 68% considera utilizar para sus proyectos la tecnología machine learning.

Muchos síntomas dejan prever una evolución de los desarrollos embebidos hacia sistemas más demandantes y competitivos, tanto que pueden poner a prueba la definición de sistemas embebidos como sucedió con la evolución de teléfono celular al smartphone. Para acompañar eficientemente la expansión y diversidad de objetivos, es imprescindible imaginar nuevas metodologías, más plurales y menos estrictas en el apego a normas y rituales.

Existen actualmente muchas empresas comenzando a utilizar variantes adaptadas de los métodos Agiles a embebidos y se organizan mundialmente simposios y conferencias en las que se exponen implementaciones y casos de éxito [12, 7].

2.1 El MVP como concepto y herramienta

La definición de MVP es un proceso iterativo basado en la realimentación continua, obtenida de los llamados "early adopters" o usuarios pioneros. Este término, definido por Frank Robinson[2] en 2001 es un concepto de evolución permanente, y actualmente se refiere a una versión de un producto que permite obtener la mayor cantidad de experiencia con el menor esfuerzo. Eric Ries[3] propone una metodología de trabajo orientada a los emprendedores llamada Lean Startup, que usa el MVP como punto principal de aprendizaje y evolución. Uno de sus principios es Construir-Medir-Aprender y permite adquirir conocimientos sobre cómo mejorar y completar un producto, a través de iteraciones de las cuales se obtiene información.

Al incorporar el desarrollo incremental en los sistemas embebidos, el MVP puede transformarse en una poderosa herramienta para optimizar y materializar un producto en ciclos de transformación.

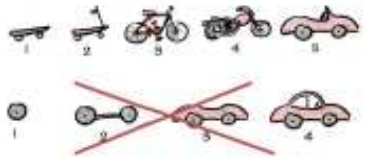

Figura 4: Concepto de MVP
Recuperado de www.medium.com

La **Figura 4**, se usa frecuentemente para explicar la visión del método Ágil sobre la evolución de un producto. Es aquí donde el MVP interviene marcando las metas parciales del desarrollo. La adecuación del concepto de MVP a los estados intermedios de un producto desarrollado en Ágil, complementa la idea de completar etapas con productos funcionales.

Uno de los problemas más frecuentes en el desarrollo de embebidos es la determinación de puntos de control e integración parcial. En general el proceso de diseño sigue el modelo "V" [3], simulación, prototipo y preproducción y la integración se da en las tres etapas. Se trabaja por separado, con mínima interacción y al momento de nuclear los desarrollos en un diseño, surgen problemas, en algunos casos tan graves, que comprometen la factibilidad del producto.

Es necesario proponer puntos intermedios de ajuste con desarrollos lo suficientemente maduros para generar un MVP. La escala de producción, las prestaciones indispensables, la criticidad operativa, el tiempo de introducción al mercado, son factores a tener en cuenta para configurar las pruebas a realizar y estimar qué factores se dejan librados a la realimentación de los usuarios pioneros.

La determinación de puntos de control que exijan productos o subproductos operativos, aclara las metas y genera un esquema de desarrollo concéntrico que sale desde un producto con características básicas y llega a uno que funciona cumpliendo todos los alcances propuestos. Esta forma de acercarse a la generación de sistemas embebidos permite la introducción del cliente en el proceso como principal fuente de realimentación.

2.2 Realimentación en proceso de diseño

Los cambios en un proceso de diseño en el cual intervenga también el hardware, no son en general aceptados o bienvenidos. Pensar en modificar los alcances o prestaciones en este escenario, no permite hacer pie o cimentar una base donde construir un diseño. Esta limitación impide pensar en la intervención sobre las especificaciones durante el proceso de creación.

Con el concepto de MVP como herramienta, puede implementarse un cronograma de intervención del cliente en el desarrollo, que aporte seguridad y realimentación sobre la dirección en que se está yendo. Para ello deben establecerse fronteras que delimiten las áreas dinámicas y las áreas estáticas; es decir, qué se puede cambiar y qué no y en qué medida.

Tomando como ejemplo el desarrollo de una placa de control automotriz, se analizará si en el proceso pueden agregarse o modificarse prestaciones que complementen o amplíen su funcionamiento. Por ejemplo, agregar un conjunto de mensajes de control que comanden un módulo nuevo de apertura de puertas que se sume al bus I2C existente, no debería resultar problemático. Pero si en cambio, se quisiera disminuir la latencia de respuesta a ciertos eventos o aumentar la cantidad de puertos de control, estas modificaciones estructurales podrían no ser satisfechas por el hardware seleccionado.

Por las razones expuestas el proceso de toma de requisitos y la distinción y selección de los requerimientos funcionales y no funcionales es esencial. Identificar la zona dinámica y la zona segura de un proyecto es lo que define y

[2] Conocido también como desarrollo sincrónico, aplicado a B2B (Business to Business).

[3] MVP aplicado startups rápidos para nuevas empresas.

especifica la intervención de la realimentación del usuario o conjunto de usuarios expertos.

3. Propuestas de Solución

De acuerdo con lo expresado, el acercamiento correcto a cómo capitalizar algunas características de los métodos ágiles por parte de desarrollos de embebidos es diversificar la visión y las herramientas. La estrategia no es tratar de adaptar los procesos de desarrollo en uso a otros más ágiles, sino tomar características ágiles y sumarlas a los procesos existentes. Varios análisis de las posibilidades de usar Ágil en embebidos, describen como morigerar algunos conflictos y como algunos factores hacen imposible su implementación[4]. Mientras otros, tomando prácticas ágiles y adaptándolas establecen que puede favorecer ampliamente las operaciones[5].

Si se hiciese la pregunta sobre si se puede utilizar, por ejemplo, SCRUM en embebidos, la respuesta sería u rotundo no. Sin embargo, si estudiando los factores de cambio, se concluye que aportan principios que permiten incorporar prácticas ágiles a los procesos, puede resultar altamente favorable.

Se proponen a continuación lineamientos y conceptos para compatibilizar e incorporar algunas ventajas de los métodos ágiles a los procesos usuales de desarrollo de sistemas embebidos.

3.1 Toma de requisitos

El proceso de toma de requerimientos tradicional da como resultado un conjunto de objetivos y alcances que el proyecto deberá cumplir. Normalmente los requisitos se dividen en funcionales, los que definen el comportamiento del software, y no funcionales, los que definen características de funcionamiento. Estos últimos también llamados atributos de calidad se refieren a cómo deberá el sistema ejecutar su función (tiempo, disponibilidad, estabilidad, tiempo de entrega, cantidad de usuarios, etc.). Entonces, la clasificación de los requisitos, en términos simplificados, definen qué es lo que debe hacer el sistema (funcionales) y cómo debe hacerlo (no funcionales). En el entorno de sistemas integrados ambos impactan en la selección de hardware y la planificación del código. La división de los requisitos, facilita la visión de las zonas verde y roja de cambios, por eso es fundamental realizarla con las precauciones debidas.

La posibilidad de acordar con el cliente ciertos hitos que no sufrirán cambios en el transcurso del proyecto y algunas zonas que pueden sufrir alteraciones controladas. Esto como se mencionó, genera un mapa de la dinámica del proyecto que regulará la forma de integrar al cliente al proceso de desarrollo. Esta meta es imprescindible para un correcto dimensionamiento del hardware y una correcta especificación del código.

Es necesario que el resultado de la toma de requisitos sea claro y conciso, poniendo énfasis en el correcto relevo de los requisitos que definirán el hardware, teniendo en cuenta a los no funcionales. Otro resultado deseable del relevamiento inicial es el análisis de las funciones complejas junto con el cliente, para evaluar la factibilidad de dividirlas en procesos más simples. Esta operación propiciará la mejor comprensión de las secuencias que forman cada tarea, la sucesión de acciones que definen cada operación y facilitará la visión modular del proyecto. Por último, en esta etapa debe establecerse un canal de comunicación con el cliente para informarlo de los avances.

Si bien excede el alcance de este documento, la incorporación del concepto de LEL (Léxico Extendido del Lenguaje) [4] a las prácticas de toma de requisitos, colabora notablemente en la correcta interpretación de las metas y alcances.

3.2 Análisis y Proceso

Una vez obtenidos y seleccionados los requisitos, para intervenir con prácticas ágiles en el proceso de desarrollo es necesario dividir modular y jerárquicamente los bloques que conformarán el mapa.

Como primer paso deben identificarse las operaciones que sucederán, con el fin de definir unidades de proceso que tendrán las siguientes características:

- **Función**: Qué tarea realiza: *transmisión de datos, medición de temperatura, presentación de información.*
- **Entradas**: Cantidad y tecnología de las entradas: *(4 entradas SPI, 2 entradas I2C, 5 puertos digitales).*
- **Salidas**: Cantidad de salidas y tipo de tecnología de comunicaciones: *(1 salida RS232).*
- **Resultado**: Cuál es la función que realiza *(reporte, protección, alerta, control).*
- **Información**: Qué datos adicionales a la función principal aporta (*cantidad de conversiones, registro de tensión de entrada, tiempo online*)
- **Maestro**: Si maneja algún módulo (*display TFT*)
- **Esclavo**: Si depende de algún módulo (*Controlador central*)
- **Relación**: Si es interno o de borde, es decir si se relaciona sólo con módulos internos o si tiene salidas al exterior del diseño. (*interno / de borde*)
- **Proceso**: Describe el nivel de inteligencia del módulo, por ejemplo, si es sólo un sensor o si procesa información. (*Procesa / No procesa*)
- **Tiempo de atención**: Describe si necesita atención periódica. (*módulo watchdog de control).*
- **Recursos que utiliza**: Qué recursos necesita exclusivamente del sistema (*4 puertos digitales*).

[4] Ver Timo Punkka, "Embedded Agile". Allí se explica como adaptar algunos factores y la imposibilidad de hacerlo con otros.

[5] Andrea Tomasini and Bent Myllerup en el ciclo de conferencias "Embedded meets Agile" en Munich, en el 2014.

• **Recursos que comparte:** Qué recursos utiliza no exclusivamente. (*DMA, Timer Output Compare)*

Estas características pueden aumentar o disminuir según la complejidad del proyecto, pero en este ejemplo se plantea el concepto y el criterio de clasificación. Supongamos el caso de un control de RPM de un automóvil

• Función: Controlar y reportar las RPM del vehículo
• Entradas: Cantidad: 1 / protocolo CAN
• Salidas: Cantidad: 2, 1 salida CAN, 1 salida I2c
• Resultado: Reporte de desvío de límites inferior y superior de RPM.
• Información: Tiempo de actividad/inactividad
• Maestro: Display OLED
• Esclavo: Procesador Central
• Relación: Interno
• Proceso: Si
• Tiempo Atención: No
• Recursos Ex: No
• Recursos CP: Bus CAN, BUS I2C

Junto con estas características es preciso definir también los buses que interconectarán los módulos. Para ello es necesario dividirlos según el objetivo que cumplen y qué nodos comunican. Por ejemplo, que protocolos serán internos al diseño y comunicarán módulos y cuales conectarán el sistema con el exterior. Suponiendo el caso de un sistema de control de procesos industriales, el protocolo interno podrá ser SPI y los externos Ethernet y RS485. La definición de protocolos clarifica la interconexión entre módulos y especifica las necesidades de diseño. No es lo mismo, por ejemplo, el diseño de una conexión CAN interna entre módulos que una entrada o salida CAN del producto. La definición de los protocolos y su división en internos o externos simplifica la declaración de entrada de los módulos que integrarán el sistema.

En caso de ser necesario el diseño de una trama de comunicaciones propietaria, deberá documentarse, especificar si es interna o externa y qué módulos la utilizarán. Esta información debe especificarse también en la información de cada módulo, pero sea dónde fuere debe estar documentada ya que define la forma en que se comunica cada módulo con sus pares o con el exterior.

Protocolos

Inter Modulares
BUS1 = SPI
BUS2 = I2C
BUS3 = UART (*)

Externo
BUS4 = MIDI
BUS2 = USB

Figura 5: Definición de protocolos

A modo de ejemplo, la definición de buses de un controlador MIDI (Musical Instrument Digital Interface) se puede apreciar en la **Figura 5**. Tiene display I2C, conversores para potenciómetros encoders SPI y comunicación entre dos microcontroladores a través de la UART con trama propietaria. Se conecta vía USB a una computadora y también tiene salida MIDI nativa.

3.3 División modular

La división modular y su definición tanto operativa como tecnológica permite seccionar tareas en diferentes grupos, tanto jerárquicos como operativos y generar tantos subproyectos como módulos se hayan definido. Este concepto, asimilable al de MVP para un cliente interno, propone generar conjuntos totalmente funcionales que concentren tareas específicas. La visión de un módulo funcional como una caja estanca, con entradas y salidas definidas, interacción controlada y una función transferencia conocida, permite:

• Simular su funcionamiento más fácilmente.
• Medir sus parámetros internos y de intercambio.
• Especificar su desempeño (consumo, error de medición, calor que disipa) para luego inferir los del diseño completo.
• Utilizar código más sencillo para pruebas.
• Ensayar módulos con las mismas especificaciones de entorno, pero con otras tecnologías.
• Prever escalabilidad en sus funciones y características.

Por ejemplo, el caso de un medidor que informa constantemente la corriente que circula por el devanado de un motor. Su entrada es un sensor de corriente y su salida un interfaz RS232 que informa la corriente, muestra parámetros de configuración y tiene indicadores de alama y estado. Si se conocen las convenciones y reglas de conexión e intercambio, cuál es el parámetro que excita al módulo (corriente) y cuál es su salida (información digital), puede verse sin problemas como un pequeño proyecto, con los requisitos que impone un cliente interno que es el mismo diseño. Generar el entorno adecuado para verificar su funcionamiento es relativamente sencillo, una fuente variable de tensión y un analizador de protocolo o una PC con entrada RS232. Este módulo puede ser desarrollado totalmente aparte del resto del diseño e integrarse luego sin problemas. A su vez los diseños monolíticos e indivisibles pueden sumarse en sub conjuntos más complejos, que pueden verse como piezas de diseño y ser tratados de la misma forma que los módulos que los componen.

Siguiendo con el ejemplo del medidor de corriente, pueden sumarse al conjunto de módulos para formar un subconjunto de medición de motores, medidores de tensión, torque, RPM, vibración y sus parámetros ser consolidados por un pequeño microcontrolador que informe a una unidad central. En otras palabras, la organización de un automóvil moderno.

La visión de un proyecto complejo como la sumatoria de MVPs ayuda a poder paralelizar los desarrollos y mientras los módulos estén correctamente definidos

(entrada, salida, recursos, tiempos de atención, etc.), podrán integrarse sin problemas.

La interacción con los módulos de desarrollo provistos por los fabricantes es una herramienta de diseño poderosa que permite incorporar subconjuntos que vayan sumando habilidades hasta llegar a completar las funciones del proyecto. Es decir, utilizar módulos provistos por fabricantes o desarrolladores externos como nodos operativos es una estrategia de gran efectividad. Este acercamiento a la integración modular, tiene numerosas ventajas:

- Funcionamiento probado.
- Herramientas de desarrollo actualizadas.
- Soporte técnico.
- Fácilmente intercambiables.
- Código de ejemplo y bibliotecas
- Foros y comunidades de usuarios.
- En general, precios competitivos.

Este tipo de dispositivos, a los que nos referiremos como *hardware sustentable*, han sido el motor de las plataformas de desarrollo más famosas como Arduino, ESP32, STM32, Raspberry Pi, etc. Por ejemplo, si se necesitara para un diseño un display TFT táctil, no es necesario consultar como se hacía anteriormente, a una hoja de datos del display, otra del controlador de display, otra del conversor de paralelo a I2C, otra del controlador táctil y diseñar el circuito que los interconectaba (con todos los riesgos y problemas que eso significaba). Actualmente se ofrecen cientos de conjuntos armados con displays de esas características, a precio competitivo (ya que se fabrican de a miles), con bibliotecas que permiten manejarlos fácilmente y con garantía de correcto funcionamiento.

Figura 6: Desarrollo MVP con hardware COT
Recuperado de Agile in embedded software: What´s wrong with it?, 42 Agile Conference 2014

Este acercamiento a otra forma de diseño favorece y posibilita el cambio de especificaciones y alcances durante el transcurso del proyecto. Por eso se mencionó la importancia de los requisitos no funcionales al momento de dimensionar el hardware, ya que permite prever expansiones e incorporación de requisitos adicionales en zona segura.

Se muestra en la **Figura 6** el proceso de diseño de hardware basado en componentes provistos por los fabricantes como kits de desarrollo. En este caso es un ejemplo de diseño modular presentado por la empresa Ericsson en una presentación sobre Agile aplicado a embebidos. El paso final del desarrollo es el diseño del prototipo de producción que soportará el proyecto, pero el camino se realizó con hardware sustentable de proveedores no específicos. Esta metodología de desarrollo permitió trabajar paralelamente en la generación tanto de firmware como de código de aplicación. Es decir, se invierte el orden tradicional y la implementación del hardware final puede sufrir modificaciones intermedias.

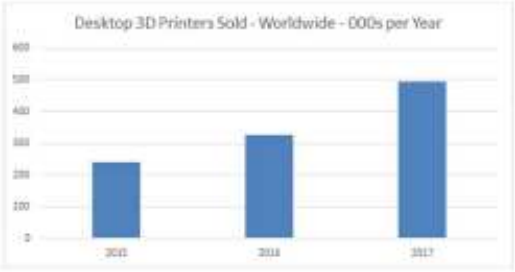

Figura 7: Ventas de impresoras 3d (2015-2017)
Recuperado de https://3dprintingindustry.com

Otros casos de éxito que pueden citarse como ejemplos de esta visión del proceso de generación de embebidos son las pequeñas impresoras 3D, en su gran mayoría hechas con hardware sustentable y código abierto gratuito. Se puede ver (**Figura 7**) el aumento de ventas de las impresoras 3D, en los últimos años. Es el resultado de haber simplificado el diseño, bajar su costo y facilitar su adquisición. Solamente Prusa Research del emprendedor Joseph Prusa lleva vendidas hasta 2019, 130000 unidades.

Por otro lado, Julián Fernandez, a través de financiación colectiva (crowfounding), logró construir uno de los satélites más pequeños del mundo (5cm) y ponerlo en órbita para brindar entre otras funciones, internet gratuita. La plataforma utilizada para el desarrollo fue PocketQube, un kit para armar orientado a aplicaciones aeroespaciales, que permite hacer un satélite por menos de U$S6000. En una nota periodística[6], Julián Fernandez se refiere al hardware sustentable con el término *COTS hardware*. COTS significa

[6] Ver "The Open-Source Revolution in Outer Space" en www.medium.com

Commercial Off The Shelf o componente tomado del estante y describe productos que no necesitan un desarrollo específico y se comercializan masivamente. La inclusión de productos COTS [16] en las políticas de compra de insumos en instituciones gubernamentales estadounidenses, les permitió bajar significativamente costos y reducir los tiempos de adquisición.

3.3 Pruebas

La imposibilidad de automatizar tests y verificaciones, atenta contra la entrega continua. La idea de tener productos funcionales luego de ciclos de iteración es posible solo si existe una fase de prueba ágil y eficiente.

Como se mencionó, la confección de prototipos con división modular y con la ayuda de hardware sustentable, resuelve un eslabón de la cadena de la entrega continua. La pregunta es cómo generar ciclos de pruebas que garanticen resultados y sean eficaces.

Generar escenarios que pongan a prueba a los prototipos y encuentren sus fallas, no es sencillo, pero debe tenerse en cuenta que las facilidades de diseño que se han detallado también aplican a los dispositivos de prueba. Estos deben incluirse como elementos prioritarios para el desarrollo, a la misma altura de la plataforma de desarrollo o de los kits de hardware. Y más allá de las aplicaciones que organizan versiones de código, ambientes o entornos de validación (sandboxes) para desarrollar y probar la programación, en embebidos debe incluirse el hardware en el planeamiento de las pruebas.

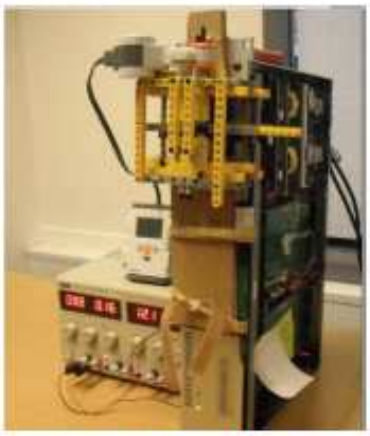

Figura 8: Máquina de test de hardware
Recuperado de Agile in embedded software:
What's wrong with it?, 42 Agile Conference 2014

En la **Figura 8** se ve un dispositivo diseñado por una empresa que produce teléfonos celulares [7] para simular la presión de teclas, construido en el propio laboratorio con elementos no específicos. Los sistemas que prueban hardware, integrados con software de medición e interacción con el sistema de producción, son solo una parte de las posibilidades disponibles a la hora de probar y simular un sistema embebido. La visión que prevé la confección de dispositivos de pruebas como parte del desarrollo, permite diversificar y especializar las herramientas disponibles.

Por otro lado, el avance del software de simulación, permite contar con opciones poderosas a la hora de diseñar, verificar e interactuar. Por ejemplo, el software Simics, especializado en modelar hardware embebido, permite pensar en que cada ingeniero de diseño tenga un prototipo virtual en su escritorio y que los cambios que se vayan realizando impacten actualizando un modelo global. La compartición de un modelo sobre el cual trabajar grupalmente en forma controlada y sincronizada, reduce las limitaciones impuestas por la distribución de recursos en diferentes lugares del mundo.

En resumen, la modificación del ciclo de generación de prototipos y la optimización de los ciclos de prueba, permite evaluar la posibilidad de alterar la secuencia de las tareas, paralelizar procesos y posibilita la generación de ciclos de entrega más cortos similares a los de las metodologías ágiles.

3.4 Dinámica de trabajo

Si la generación de hardware y software puede ocurrir en paralelo, es factible aplicar más fácilmente estrategias de generación de código más actuales como programación de a pares o el desarrollo orientado a pruebas (test driven developement). La generación de módulos funcionales y su división jerárquica, en un marco de delimitaciones de zonas segura y dinámica, permite aceptar historias de usuario durante el proceso de desarrollo y allana el camino que conduce a la integración continua.

La naturaleza heterogénea de los grupos de trabajos que necesita el desarrollo de los sistemas embebidos, puede ahora trabajar en conjunto más fácilmente a través de las reuniones virtuales y la unificación de los modelos de simulación. Muchas empresas tienen grupos de desarrollo propios o contratados distribuidos en diferentes sedes y si bien es necesario coordinarlos y ordenar sus actividades, la interacción es posible. La dinámica de los métodos ágiles es difícil de alcanzar virtualmente, pero la división de tareas y la división del proyecto en pequeños subproyectos colabora para que la interacción sea necesaria fundamentalmente en la etapa de integración.

La incorporación de kits de desarrollo como unidad de diseño facilita la estandarización de código. El concepto de hardware sustentable o COTS hardware facilita la unificación de criterios de codificación, ya que se cuenta con bibliotecas y notas de aplicación que colaboran con el Code Refactoring, otras de las tendencias que ha ganado adeptos rápidamente. Tal como se generan pequeñas unidades funcionales de hardware pueden generarse unidades de código reutilizables asociadas con cada módulo o conjuntos de módulos de hardware. Esta visión facilita el reemplazo, incorporación y eliminación de componentes en un proyecto.

La generación de documentación en embebidos es fundamental para su depuración y mantenimiento. Las condiciones de operación descriptas facilitan el registro de avances, por lo menos en las etapas de prototipo hasta avanzado el diseño. De todas formas, es imprescindible establecer normas y bases para asegurar el correcto y completo registro tanto de hardware como de software.

3.4 Determinación de zona segura y zona dinámica

La correcta obtención y clasificación de requisitos, permite confeccionar un conjunto de ideas tecnológicas, que delimitan distintas áreas de un diseño, principalmente según su función específica. Una vez confeccionado el mapa tecnológico del diseño, se podrán dividir zonas que podrán sufrir cambios sin alterar el núcleo de desarrollo y zonas que permanecerán inmóviles. Un sistema puede diseñarse con algunas previsiones para expansiones futuras, sin que eso signifique aumentar el costo del hardware. El diseño del PCB, la localización de los componentes, la distribución de los elementos en el chasis, los tipos y cantidad de conectores, la asignación de espacios internos son partes del diseño que pueden realizarse con visión modular y con previsiones de escalabilidad.

Es verdad que esta visión es más lógica, por ejemplo, en un concentrador de red o en una computadora automotriz, que en un teléfono celular, pero aun así, mientras no afecte otros factores o metas de diseño, la visión que aporta, la generación de diseños con posibilidad de ser escalables es valiosa.

Si por ejemplo se estuviera diseñando un controlador para una unidad de control numérico, podrían preverse algunas de las siguientes potencialidades:

- la posibilidad de contar con un display externo además del básico incorporado.
- la conexión en red con pares para configurar ta reas más complejas.
- el agregado del controlador de un eje extra.
- la posibilidad de slot de expansión para un interfaz de comunicación de diferentes normas industriales.
- la conexión a red ethernet con configuración automática de actualizaciones de firmware.
- Set de accesorios para montar en diferentes medidas de rack.
- Conexión de doble fuente para aumentar la redundancia a fallos.

Todas estas expansiones, agregan valor al producto, lo hacen más competitivo y escalable y muy probablemente no necesiten cambiar el núcleo de proceso, ni la cantidad de entradas o salidas, o el tamaño de la fuente de alimentación. Es decir, se pueden prever opciones que no modifiquen sustancialmente el costo del diseño. En muchos casos se sub utilizan las capacidades de los microcontroladores y periféricos, dejando ociosa una parte de la potencia del hardware que podría agregar valor.

Es entonces importante, determinar que parte del diseño es permeable a cambios y expansiones y que parte no lo es. Esta operación también se realiza en el código, a la hora de pensar los módulos operativos y la forma en que se comunican.

Figura 9: Mapa de zonas

Cuando se analizó la toma de requisitos, se resaltó que la toma de requisitos no funcionales es importante a la hora de pensar en el diseño. En el ejemplo citado del controlador industrial, si se solicitara el envío de mensajes a un teléfono celular en caso de falla, podría implementarse sin demasiado problemas. Pero sin embargo si se pidiera que la velocidad de reacción a impulsos fuese un orden de magnitud menor a la que el controlador detalla en su especificación, podría ser un inconveniente que impacta directamente en el tipo de CPU seleccionada al comenzar el diseño o al tipo de conversor analógico digital elegido. Claramente, como se ve en la **Figura 9**, la primera solicitud se encontraría en área verde y la segunda en área roja.

Una vez realizada esta división, se puede pactar con el cliente ciertos límites para activar la aplicación dinámica de cambios. La determinación y documentación de qué alcances pueden modificarse en el transcurso del proyecto y cuáles son las potencialidades de expansión, formarán un conjunto de reglas de juego que regirán la relación y el flujo de los pedidos, solicitudes que pueden tener la forma de las historias de usuario.

Con estas reglas de juego, los que tienen la última palabra son los ingenieros de diseño, que son quienes analizan el impacto de las modificaciones solicitadas, pero si aun así ejercita el feedback y la inmersión del cliente en el proceso de diseño.

4. Conclusiones y futuras líneas de investigación

El progreso y la evolución de las condiciones de entorno hace necesario la evaluación periódica de los métodos en busca de nuevas variantes que ayuden a mejorar procesos. Existen ambientes más conservadores que otros en lo que a adoptar nuevas tendencias se refiere. Está a la vista que la gestión de proyectos o generación de códigos de sistemas embebidos no ha incorporado cambios profundos, por lo menos hasta hace unos años, cuando la evolución de los

requisitos y las necesidades tecnológicas ha sido tan notable que obligó a replantearse los procesos. El hecho de que los mismos lenguajes de programación se utilicen desde hace 40 años y las plataformas o IDEs de desarrollo tampoco se hayan modificado profundamente es una circunstancia, pero no necesariamente un problema. Dentro de la inmovilidad del C, que no tiene la ventaja de la portabilidad que pueden ofrecer JAVA o Python, hay ventajas que lo transforman en un standard de diferentes plataformas y sistemas de desarrollo. En embebidos C tiene la virtud de permitir una rápida adaptación y migración de código a la hora de cambiar de hardware.

Los métodos tradicionales dan respuesta a la secuencialidad y a la necesidad de documentación de los desarrollos embebidos, pero no a la dinámica y efectividad que se necesita hoy. Los métodos ágiles, no se adaptan al entorno embebido porque la mecánica de desarrollo atenta contra sus pilares conceptuales. La visión de tomar lo que sirve y adaptarlo para mejorar los procesos, puede ser el camino para permitir el aporte ágil a embebidos. Se propone que la clave sea limitar la zona de cambios, adoptar herramientas y métodos que viabilicen y diversifiquen procesos alternativos que permitan el acercamiento a algunos conceptos ágiles. También se propone no poner énfasis en implementar alguna metodología ágil completa en las prácticas de desarrollo sino, tener como núcleo conceptual la cuestión de si aporta o incrementa valor al proceso. Incorporar rituales o ciclos innecesarios, sólo por el hecho de que regulan la interacción o fomentan la comunicación, puede no ser lo ideal en un entorno embebido. Si bien esos conceptos son valiosos e indispensables para la optimización de los procesos modernos, se pueden prever formas alternativas de aplicación más adaptadas a los ambientes reales de embebidos, que son heterogéneos, reducidos, con recursos de gran experiencia y alta capacitación.

Las líneas de investigación que se seguirán son la posibilidad de aplicar puntualmente programación de a pares en entornos embebidos, la reusabilidad de conjunto hardware/código, la escalabilidad en una familia de procesadores con independencia de código base, la inclusión de Python para GUI (Interfaz Gráfica de Usuario) en integración transversal y la factibilidad de aplicar Agile a programación gráfica en PyQt y Tkinter.

5. Consideraciones finales

La inducción de embebidos a características ágiles descripta en el presente documento, se está utilizando actualmente en algunos proyectos en el ámbito académico universitario. Se han aplicado también en desarrollos de pequeña escala de sistemas de control y dispositivos sencillos.

El tema que abarca este documento es parte de una tesis de maestría sobre la generación de un nuevo marco conceptual orientado a sistemas embebidos, que se encuentra en proceso de compleción.

Referencias

[1] A. Janes, G. Succi, "The Dark Side of Agile Software Developement", Free University of Bolzano/Bozen, Bolzano, Italia.

[2] J. Greening, "Agile Embedded Software Development", Embedded Systems Conference (ESC), San José, California, Estados Unidos, 2013.

[3] B. Broekman, E. Notenboom, "Testing Embedded Software", Addison & Wesley, Pearson Limited Ed., Holanda, 2003.

[4] G. Kaplan, G. Hadad, J. Doorn, J. Sampaio do Prado Leite, "Inspección del Léxico Extendido del Lenguaje", Workshop de Ingeniería de Requerimientos (WER2000), Río de Janeiro, Brasil, 2000

[5] T. Punkka, "Embedded Agile", Embedded System Conference (ESC 2010), Boston, Estados Unidos, 2010.

[6] V. Lenarduzzi, D. Taibi, "MVP Explained: A Systematic Mapping Study on the Definitions of Minimal Viable Product", Free University of Bolzano/Bozen, Bolzano, Italia, 2011.

[7] A. Tomasini, B. Myllerup, "Agile Embedded Software, What's wrong with it?", Embedded Mets Agile Conference, Berlin, Alemania, 2014.

[8] F. Deprettere, J. Teich, S. Vassiliadis "Embedded Processors Design Challenges", Springer Ed. 2002

[9] K. Fowler, "What Every Engineer Should Know About Developing Real Time Embedded Products", CRC Press, Boca Raton, Florida, Estados Unidos 2008.

[10] M. Kaisti, V. Rentala, T. Mujunen, ,S. Hyrynsalmi, K. Konnola, ,T. Makila, T. Lehtonen, "Agile Methods for Embedded Systems Development – A Literature Review and a Mapping Study", Eurasip Journal on Embedded Systems 2013, Marruecos, 2013.

[11] M. Rowe, "IEEE Survey of Programming Languages", EE Times, Recuperado: www.eetimes.com/ieee-survey-ranks-programming-languages/#, 2018.

[12] Bent Myllerup "Scrum Embedded Systems: An Experience Report of tc Electronics", Orlando Scrum Gathering, Florida, Estados Unidos, 2019

[13] J. Greening, "Test Driven Develpement in Embedded C", Recuperado de: www.wingman-sh.com, Pragmatic Bookshelf, 2014.

[14] K. Beck, "Extreme Programming Explained", Addison & Wesley, Pearson Education Ed., 2000.

[15] K. Beck, "Embedded 2019 Market Study" , Recupendo de: https://www.embedded.com/2019-embedded-markets-study-reflects-emerging-technologies-continued-c-c-dominance/, 2020.

[15] "The Advantages of COTS Hardware" , Recuperado de: https://freeandfair.us/articles/the-advantages-of-cots/ , 2020.

50 UTN ANIVERSARIO FACULTAD REGIONAL SAN FRANCISCO - CÓRDOBA - ARGENTINA

VIRTUAL 2020 CONAIISI 8° CONGRESO NACIONAL INGENIERÍA INFORMÁTICA / SISTEMAS DE INFORMACIÓN 05 06 NOV.

CERTIFICADO

DE ASISTENCIA

Por cuanto, **Eterovic Jorge** *D.N.I. 11.917.135 ha participado del 8° Congreso Nacional de Ingeniería Informática / Sistemas de Información (CoNaIISI 2020) organizado por la Red de Carreras de Ingeniería Informática / Sistemas de Información (RIISIC) perteneciente al CONFEDI, realizado de forma Virtual por la Universidad Tecnológica Nacional Facultad Regional San Francisco, los días 05 y 06 de Noviembre de 2020, se otorga el presente certificado.*

Firma Digital

Firma Digital

RIISIC confedi