**UNIVERSIDAD TECNOLÓGICA DE QUERÉTARO**

**CESEQ**



**Diplomado en Software Embebido**

Proyecto Integrador

**Software Development Plan #CESEQ001**

***Scrum Master***: Preciado, Francisco

**Desarrolladores**. Mendizábal, Rebeca

Velázquez, Aldo

**Fecha**: 20191031

# *Log*

|  |  |  |  |
| --- | --- | --- | --- |
| **Versión** | **Fecha** | **Descripción** | **Revisor** |
| 0.0.1 | 17/08/2019 | Planeación inicial | Mendizábal, Rebeca |
| 0.0.2 | 30/08/2019 | Actualización de las secciones: alcance del proyecto, entregables, metodología de desarrollo y entregables | Preciado, Francisco |
| 0.0.3 | 06/09/2019 | Actualización de las secciones: Estrategia de solución de problemas y Diseño | Mendizábal, Rebeca |
| 0.1.0 | 13/09/2019 | Actualización de las secciones: Liberación, Pruebas y Diseño | Velázquez, Aldo |
| 0.1.1 | 04/10/19 | Actualización de la sección de control. | Mendizábal, Rebeca |
| 0.1.2 | 25/10/19 | Actualización de las secciones de: Pruebas, Diseño | Preciado, Francisco |
| 1.0.0 | 30/10/19 | Ajustes de formato finales | Velázquez, Aldo |

# Índice

Tabla de Contenidos

[*Log*](#_heading=h.gjdgxs)***2***

[Índice](#_heading=h.30j0zll) **3**

[Alcance del proyecto](#_heading=h.1fob9te) **5**

[Identificación del proyecto](#_heading=h.pc9xkevs482s) 5

[Descripción general del proyecto](#_heading=h.vu0qtj735dt) 5

[Entregables con el cliente](#_heading=h.2et92p0) **5**

[Compromisos de entrega de los módulos de software](#_heading=h.kgam937dnz7q) 6

[Metodologías de desarrollo](#_heading=h.c8kv39o2haot) 6

[Metodología de desarrollo](#_heading=h.tyjcwt) **7**

[Estimación](#_heading=h.3dy6vkm) **9**

[Infraestructura y herramientas](#_heading=h.vux6gigm77f1) 9

[Recursos humanos](#_heading=h.eb8n0xt3o3r0) 10

[Listado de riesgos](#_heading=h.7qskl85ob0g5) 10

[Recursos críticos](#_heading=h.bbx5ho2u4zb) 10

[Desglose de actividades](#_heading=h.gh7biv8a3qow) 11

[Planeación](#_heading=h.1t3h5sf) **12**

[Roles y responsabilidades](#_heading=h.j9aazpaurkps) 12

[Trazabilidad de entregables](#_heading=h.71uyp67ldoaf) 13

[Estrategia de Solución de Problemas](#_heading=h.4d34og8) **13**

[Diseño](#_heading=h.2s8eyo1) **13**

[Diagrama de bloques de software](#_heading=h.xa22w5ygm4at) 13

[Diagrama de flujo](#_heading=h.3sisia4ydocd) 13

[Diagrama de control](#_heading=h.3kd6zd9bcb7o) 14

[Estándares](#_heading=h.17dp8vu) 15

[*Naming conventions*](#_heading=h.3rdcrjn) *15*

[**Construcción del software**](#_heading=h.rzx4hynwtmtn) **16**

[**Metodología de revisión de software**](#_heading=h.reygmeubo7j4) **18**

[**Pruebas de software**](#_heading=h.v368108qvp91) **18**

[Pruebas de caja negra](#_heading=h.ecbhj12qg4bl) 19

[Pruebas de caja blanca](#_heading=h.35nkun2) 19

[Liberación](#_heading=h.44sinio) **19**

[Plan de entregas de liberaciones al cliente](#_heading=h.hhwwg297iz17) 20

[Estrategia de pruebas de integración](#_heading=h.2jxsxqh) 21

[Pruebas de validación en campo](#_heading=h.z337ya) 21

[Procedimientos de control](#_heading=h.or25rzct1j06) 21

[**Referencias**](#_heading=h.45hxeg5vpwn6) **22**

# Alcance del proyecto

Se controlará la velocidad de un motor de corriente directa mediante la aplicación de una señal cuadrada variable en su ancho de pulso, cuya frecuencia de trabajo deberá ser constante en un rango de 100 Hz a 1 KHz.

El valor de referencia de entrada será variable y podrá ser establecida por el usuario, a través de un potenciómetro. Se implementará un sistema de control Proporcional Integral Derivativo (PID) para establecer una retroalimentación al sistema. Finalmente, mediante el uso de un sensor de efecto hall acoplado al rotor del motor, se medirá la velocidad del mismo, el cual proveerá una serie de pulsos cada que se complete una vuelta entera. De esta manera, a mayor velocidad del motor, mayor será el número de pulsos leídos y mientras menor sea la velocidad, menor será el número de pulsos.

La definición y análisis de los requisitos se podrá encontrar en el documento: ***Proyecto\_Integrador/ESTRUCTURA DEL PROYECTO/1) Requisitos/SWRA\_20190817.xlsx***

## Identificación del proyecto

Controlar la velocidad de un motor de corriente directa usando como valor de referencia la retroalimentación que proporcione el usuario mediante un potenciómetro. Una interfaz gráfica deberá mostrar la velocidad del motor y SetPoint (ambos en RPM’s); así como el porcentaje de trabajo de la señal cuadrada.

## Descripción general del proyecto

Durante el desarrollo del proyecto, en conjunto con la etapa funcional, se generarán entregables relacionados con la planeación, requisitos, juntas, diagramas de diseño, software implementado, pruebas y documentos de calidad, en donde se reflejen las distintas etapas para poder cumplir con el alcance establecido.

# Entregables con el cliente

A continuación, se hará una breve descripción de los elementos a entregar al término del proyecto:

* **Código (.hex)**. Se entregará el código completo con la trazabilidad y documentación correspondiente, de manera que cualquier persona pueda interpretar y entender lo que se implemente en cada sección. De esta manera también será más sencillo el mantenimiento del mismo.
* **Documentación**. Se entregará una serie de documentos organizados en carpetas definiendo el proceso completo que se llevó a cabo para la elaboración del proyecto integrador desde su inicio hasta el final. De esta manera, quedará como evidencia y respaldo en caso de que se deseara replicar o hacer algún cambio al sistema. La documentación se dividirá en las siguientes categorías: requisitos, planeación, diseño, verificación y documentos de calidad, agregando como documento general el Plan de desarrollo de Software.

Requisitos. En esta carpeta se incluirá toda la información relacionada a la creación, diseño, análisis y aprobación de requisitos. Ésta se capturará en un archivo de Excel, donde se tendrá inicialmente una captura de los requisitos del cliente, posteriormente se hará un análisis validando que los requisitos son válidos y que cumplen con todas las características necesarias.

Después se hará una revisión y aprobación de los mismos donde se establecerá qué competencia debe probar cada uno y se hará una vinculación tanto con las tareas de desarrollo de software como con las pruebas necesarias para verificar su correcto funcionamiento.

Planeación. Se incluirán todos los archivos que describen: los roles y principales responsabilidades de los miembros del equipo, siguiendo la metodología SCRUM; el calendario con las tareas asignadas a cada miembro del equipo, obtenidas de las estimaciones previas; el flujo de trabajo utilizado para el control de tareas; documento utilizado para el Análisis de Modos de Fallas y Efectos (FMEA); y las actividades contempladas durante toda la etapa del proyecto (documentos necesarios, actualizaciones, ejecución de prueba de caja blanca/negra, ejecución de prueba de integración y reuniones contempladas).

Diseño. Carpeta utilizada para archivar todos los diagramas de modelado estático y dinámico del proyecto, que corresponden a: diagramas de bloques, diagramas de flujo, diagramas “*call tree*”, diagramas de máquina de estados, diagramas de secuencia y otros dependiendo del paradigma de programación. Además, se incluirá el diagrama de control donde se definen: entradas, salidas, ruido y su retroalimentación. Por otro lado, se incluirá el documento relacionado al estándar utilizado para el proyecto y la herramienta para evaluarlo, así como el documento donde se define la convención de nombres.

Verificación. La documentación necesaria para la realización de la etapa de pruebas se incluirá en esta carpeta. Los archivos son relacionados a las pruebas que se realizarán de caja negra, caja blanca, integración, validación y CCR (“*Cyclomatic Complexity Redundance index*”). Además, se incluirá el proceso de la prueba de rendimiento y medición de memoria Flash y RAM.

Documentos de calidad. Las auditorías de calidad se incluirán en esta carpeta.

## Compromisos de entrega de los módulos de software

La planeación de las entregas a cliente están establecidas en el siguiente documento: ***Proyecto\_Integrador/ESTRUCTURA DEL PROYECTO/5)Documentos de calidad/Plan de entregas a cliente.docx.*** En él se establecen los módulos que fueron entregados por cada release, el cual se hacía cada 2 sprints, como marca el documento ya mencionado.

## Metodologías de desarrollo

En este tipo de proyectos enfocados al desarrollo de software, es necesario elegir una metodología que sirva de guía y que proporcione herramientas para tener un mejor control del proceso, así como entregables de calidad.

Las metodologías Agile son un conjunto de procedimientos aplicados en la creación de software que basan su desarrollo en un ciclo iterativo. Las ventajas principales de estas metodologías son la mejora en la interacción con el cliente y un desarrollo incremental del software, permitiendo entregas cortas pero funcionales del producto y evaluaciones que permitan la integración de cambios si es necesario.

Hay tres principales tipos de metodologías Agile:

* **SCRUM**: se ocupa de la administración de tareas en un entorno de desarrollo basado en equipos. Es el método Agile más popular. Esto se debe a que es fácil de implementar y resuelve muchos de los problemas de administración comunes en los equipos de desarrollo de software.
* **XP (Extreme Programming)**: es una metodología ágil y radical que se enfoca en el proceso de ingeniería de software y se ocupa de las fases de análisis, desarrollo y prueba. Ofrece enfoques novedosos que mejoran sustancialmente la calidad del producto final.
* **DSDM**: es el método de desarrollo ágil original. Se basa en todos los principios ágiles populares. Este método es el más utilizado en el Reino Unido.

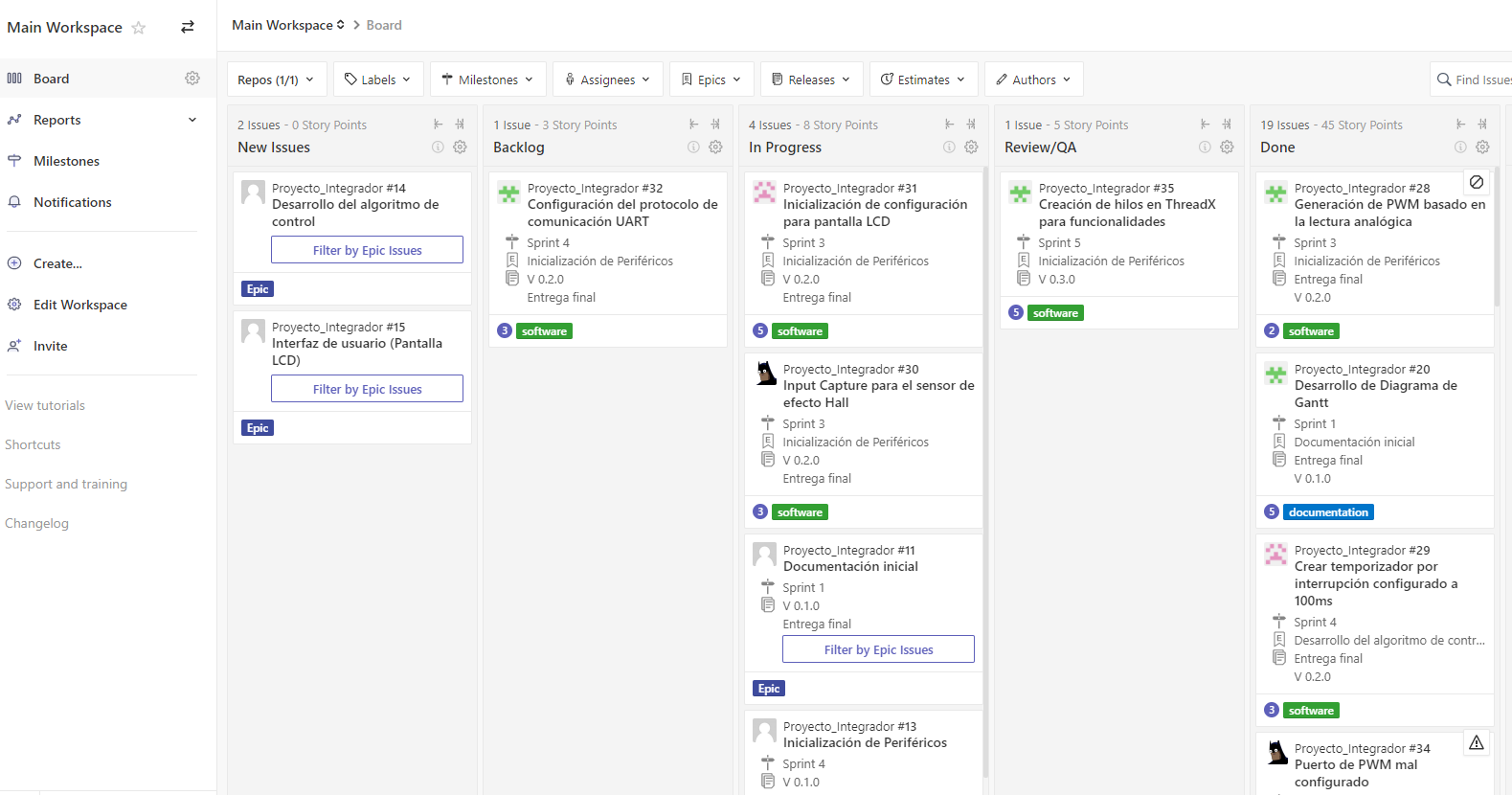
# Metodología de desarrollo

La metodología que se siguió para el desarrollo del presente proyecto es la metodología ***SCRUM***. Para ello se designó a un *Product Owner* y a un *Scrum Master* cuyos nombres se presentan en la siguiente tabla.

|  |  |  |
| --- | --- | --- |
| **Rol** | **Nombre** | **Responsabilidades** |
| ***Product Owner (PO)*** | TBD | Maximizar el valor de los productos creados por el equipo de *scrum.* |
| ***Scrum Master (SM)*** | Preciado, Francisco | Asegurarse de que el equipo sigue los valores y principios de la metodología ágil, siguiendo los procesos y prácticas acordadas por el equipo. Además, mantiene el código actualizado, haciendo las veces de integrador y remueve obstáculos del equipo para mantenerlo concentrado. |
| **Scrum development team member 1** | Mendizábal, Rebeca | Desarrolla apegándose a los requisitos y a los preceptos de la metodología ágil *scrum.* |
| **Scrum development team member 2** | Velázquez, Aldo | Desarrolla apegándose a los requisitos y a los preceptos de la metodología ágil *scrum.* |
| **Scrum development team member 3** | Preciado, Francisco | Desarrolla apegándose a los requisitos y a los preceptos de la metodología ágil *scrum.* |

**Figura 5.1.** Roles del equipo

Debido a la naturaleza de la metodología elegida, se utilizó ZenHub como herramienta para la administración del proyecto debido a su fácil integración con GitHub, el *software* de control de versiones requerido para este proyecto.



**Figura 5.2**. Scrum board en ZenHub

En caso de requerir acceso al *scrum board* localizado en ZenHub, es necesario notificar al equipo para proporcionar los permisos necesarios. Si ya se cuenta con estos permisos, la liga de acceso es la siguiente: <https://app.zenhub.com/workspaces/main-workspace-5d1f89bea509932a55607d90/board>

La longitud de *sprint* que se juzgó pertinente por el tiempo disponible para la implementación del proyecto fue de una semana natural. Empezando los lunes y terminando los domingos. El calendario con los sprints se presenta en la figura 5.3.

Por cuestiones de tiempo, no se llevarán a cabo las *Daily Scrum Meetings****.*** El estatus que normalmente se actualizaría en dichas juntas se controló con un documento interno del equipo en el que se podía consultar el estatus de cada miembro del equipo. Dicho documento fue actualizado por los integrantes del equipo cada que su estatus cambiaba.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Lunes** | **Martes** | **Miércoles** | **Jueves** | **Viernes** | **Sábado** | **Domingo** |
|  |  |  |  |  | **Agosto 17** | **18** |
| **19** | **20** | **21** | **22** | **23** | **24**  *Big Room Planning* | **25** |
| **26** *Sprint* 1 | **27** | **28** | **29** *Review Meeting* | **30** | **31** *Planning & Retrospective Meeting* | **Septiembre 1** |
| **2** *Sprint* 2 | **3** | **4** | **5** *Review Meeting* | **6** | **7** *Planning & Retrospective Meeting* | **8** |
| **9** *Sprint* 3 | **10** | **11** | **12** *Review Meeting* | **13** | **14** *Planning & Retrospective Meeting* | **15** |
| ***16*** *Sprint* 4 | **17** | **18** | **19** *Review Meeting* | **20** | **21** *Planning & Retrospective Meeting* | **22** |
| ***23*** *Sprint* 5 | **24** | **25** | **26** *Review Meeting* | **27** | **28** *Planning & Retrospective Meeting* | **29** |
| ***30*** *Sprint* 6 | **Octubre 1** | **2** | **3** *Review Meeting* | **4** | **5** *Planning & Retrospective Meeting* | **6** |
| ***7*** *Sprint* 7 | **8** | **9** | **10** *Review Meeting* | **11** | **12** *Planning & Retrospective Meeting* | **13** |
| ***14*** *Sprint* 8 | **15** | **16** | **17** *Review Meeting* | **18** | **19** *Planning & Retrospective Meeting* | **20** |
| ***21*** | **22** | **23** | **24** | **25** | **26** |  |

**Figura 5.3.** Calendario de juntas.

# Estimación

## Infraestructura y herramientas

Para esta estimación, se consideró que se trabajará con los insumos del laboratorio únicamente todos los viernes del 23 de agosto de 2019 al 25 de octubre de 2019. Por lo tanto, su disponibilidad medida en horas (h) es la siguiente:

|  |  |
| --- | --- |
| *Microcontrolador, planta, osciloscopio, multímetro, generador de funciones, y fuente de alimentación* | 40 h |

**Figura 6.1**. Disponibilidad del *hardware* de laboratorio.

Esto, asumiendo que las herramientas estén completamente funcionales durante ese tiempo. Esta estimación se considerará en el AMEF en la sección 7 del presente documento.

## Recursos humanos

La estimación de los recursos humanos se realizó bajo el acuerdo que cada miembro del equipo trabajará 8 horas por semana durante el mismo periodo antes descrito en esta misma sección. Dentro del acuerdo se consideró que el *scrum master* deberá invertir 2 horas más por semana para cubrir con sus responsabilidades.

|  |  |
| --- | --- |
| Tiempo de desarrollo | 220 h |
| Tiempo actividades propias del *scrum master* | 40 h |

**Figura 6.2.** Disponibilidad de los recursos humanos

Esto, asumiendo que el equipo no tenga nada que le impida cumplir con este acuerdo. Esta estimación será incluida en el AMEF como parte de los entregables del proyecto en la siguiente dirección: ***Proyecto\_Integrador/ESTRUCTURA DEL PROYECTO/2) Planeación/AMEF.xlsx***

### Listado de riesgos

El cálculo del número prioritario de riesgo se realizará en la sección 8 del presente documento.

Las principales actividades de riesgo consideradas son:

* Hardware dañado o faltante.
* Falta de disponibilidad temporal o permanente de los integrantes del equipo debido a asuntos personales o laborales.
* Vacaciones de la UTEQ.
* Nuevo hardware y/o microcontrolador.
* Falta de conocimiento del lenguaje de programación, SW IDE o sistema operativo.
* Error en la conexión del hardware.
* Cambios a la planeación inicial.
* Mala comunicación entre los integrantes.
* Falta de conocimiento de la metodología SCRUM.
* Errores de comunicación del microcontrolador.
* Mala estimación de recursos a las tareas.
* Error al acceder a la documentación o repositorio del proyecto.

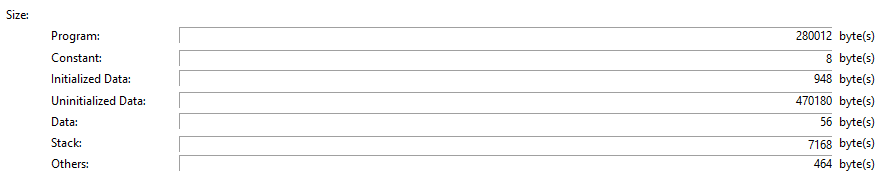
## Recursos críticos

Los componentes que se usarán para la etapa de control están conformados principalmente por el **Starter Kit SK-S7G2** de Renesas Synergy, el cual utiliza un microcontrolador de tipo ARM - MCU (modelo: **R7FS7G27H3A01CFC**). Este microcontrolador está diseñado para aplicaciones de alto cómputo, intensivo rendimiento y control en red que requieren redundancia de memoria completa, el grupo MCU S7G2 tiene de 3 a 4 MB de memoria flash y 640 KB de SRAM. Con una variedad funciones, incluye un controlador de gráficos LCD, un motor de dibujo 2D, una unidad de detección táctil capacitiva, un controlador analógico avanzado, Ethernet MAC con IEEE 1588 PTP, USB 2.0 de alta velocidad y velocidad completa, SDHI, Quad SPI y características de seguridad y protección, para manejar una amplia gama de complejidad. Algunas de sus características son:

|  |  |
| --- | --- |
| **Fabricante:** | Renesas Electronics |
| **Categoría de producto:** | Microcontroladores ARM - MCU |
| **Serie:** | S7G2 |
| **Núcleo:** | ARM Cortex M4 |
| **Ancho de bus de datos:** | 32 bit |
| **Frecuencia de reloj máxima:** | 240 MHz |
| **Tamaño de memoria del programa:** | 4 MB |
| **Tamaño de RAM de datos:** | 640 kB |
| **Resolución del conversor de señal analógica a digital (ADC):** | 12 bit |
| **Número de entradas / salidas:** | 126 I/O |
| **Voltaje de alimentación operativo:** | 2.7 V to 3.6 V |
| **Temperatura de trabajo mínima:** | - 40 C |
| **Temperatura de trabajo máxima:** | + 105 C |
| **Tipo de memoria de programa:** | Flash |
| **Tipo de Ram de datos:** | SRAM |
| **Tipo de interfaz:** | CAN, Ethernet, I2C, IIC, SCI, SPI, SSI, SDHI, USB |
| **Resolución del conversor de señal digital a analógica (DAC):** | 1 x 12 bit |
| **Sensibles a la humedad:** | Si |
| **Cantidad de canales del conversor de señal analógica a digital (ADC):** | 21 Channel |
| **Serie de procesadores:** | S7 |
| **Temporizadores de vigilancia:** | Watchdog Timer |

**Figura 6.3.** Especificaciones del microcontrolador R7FS7G27H3A01CFC

El manual de usuario con más especificaciones de este conjunto de componentes se puede encontrar en la siguiente ruta: **Proyecto\_Integrador/ESTRUCTURA DEL PROYECTO/Documentación de Referencia/RenesasSynergy\_SK-S7G2\_Manual.pdf**

**  
Figura 6.4.** Uso de memoria

## Desglose de actividades

**Lectura y análisis inicial de la descripción del proyecto**

* Planeación para la documentación inicial

**Documentación inicial**

* Estimación de recursos
* Desglose de actividades
* Análisis de dependencias
* Desarrollo de Diagrama de Gantt
  + Establecer milestones
* Detección de riesgos
* Desarrollo AMEF
* Análisis de requisitos de sistema

**Configuración de herramientas**

* Instalación del ambiente de desarrollo (e2studio)
* Creación del repositorio de Git
* Ambiente de edición de documentos

**Inicialización de periféricos**

* Lectura de entrada analógica (ADC)
* Configuración del *PWM*
* Input Capture para el sensor de efecto Hall
* Configuración de protocolo SPI (para pantalla LCD)
* Configuración del protocolo de comunicación UART

**Desarrollo del algoritmo de control**

* Establecer tipo de controlador a utilizar
* Base de tiempo por medio de interrupción o temporizador
* Creación del modelo de control
* Simulaciones del sistema
* Cálculo de la velocidad
* Implementación del algoritmo de control en C

**Interfaz de usuario**

* Formato de la información para el usuario
* Configuración de las características del display
* Despliegue de los datos del sistema
* Configuración de indicador de diagnósticos

**Verificación**:

* Evaluación del estándar de programación
* Pruebas de caja blanca
  + Documentación de pruebas
  + Ejecución de pruebas
* Pruebas de caja negra
  + Documentación de pruebas
  + Ejecución de pruebas
* Cálculo de la complejidad ciclomática
* Pruebas de validación
* Medición de *throughput*, *RAM* y *Flash*

# Planeación

## Roles y responsabilidades

Las definiciones de terminado de cada una de las tareas se encuentran en ZenHub, dado que es el lugar donde los desarrolladores tiene mejor acceso.

En caso de requerir acceso al *scrum board* localizado en ZenHub, es necesario notificar al equipo para proporcionar los permisos necesarios. Si ya se cuenta con estos permisos, la liga de acceso es la siguiente: <https://app.zenhub.com/workspaces/main-workspace-5d1f89bea509932a55607d90/board>

Los roles y responsabilidades de los miembros del equipo se pueden encontrar en la sección 5 del presente documento, en la tabla 5.1.

Para el calendario de actividades del proyecto se generó un diagrama de gantt, el cual se puede encontrar en la siguiente ruta: **Proyecto\_Integrador/ESTRUCTURA DEL PROYECTO/2) Planeación/Gantt\_Diplomado\_complete.pdf**

## Trazabilidad de entregables

La trazabilidad ayuda a tener un seguimiento de los objetos de manera rápida y eficaz. Se encuentra en todos nuestros entregables, siendo estos, requisitos, tareas de software y pruebas. Actualmente está de manera bidireccional, es decir, que en cualquiera de los entregables se puede encontrar una indicación a qué objeto de las otras dos competencias apunta. Los entregables de sistemas se encuentran en la siguiente ubicación: **Proyecto\_Integrador/ESTRUCTURA DEL PROYECTO/1) Requisitos/Análisis de requisitos.xls,** la de *software* se encuentra en las tareas de GitHub y en el código, y la de pruebas en la siguiente ubicación: **Proyecto\_Integrador/ESTRUCTURA DEL PROYECTO/4) Verificación/PruebasCajaNegra.doc** y **PruebasCajaBlanca.doc**

# Estrategia de Solución de Problemas

Partiendo de la sección 11 del documento, donde se establecen los recursos y estimaciones, se desarrolló desde el inicio del proyecto y durante toda su realización, un documento de Análisis de Modo y Efecto de Falla (AMEF) ubicado en la siguiente ruta: **Proyecto\_Integrador/ESTRUCTURA DEL PROYECTO/2) Planeación/AMEF.xls**. En él se establecen todos los posibles modos de falla y riesgos que se pudieran dar a lo largo de la ejecución del proyecto.

# Diseño

En esta sección se encuentran todos los diagramas que generalizan la funcionalidad del código implementado. Éstos se desarrollaron conforme se fue implementando el código y se fueron actualizando de manera periódica. Los diagramas mencionados a continuación son la versión final de los mismos.

## Diagrama de bloques de software

El algoritmo del sistema fue desarrollado como un bosquejo general del funcionamiento global del sistema, se puede encontrar en la siguiente ubicación: **Proyecto\_Integrador/ESTRUCTURA DEL PROYECTO/3) Diseño/Algoritmo y diagrama.docx**

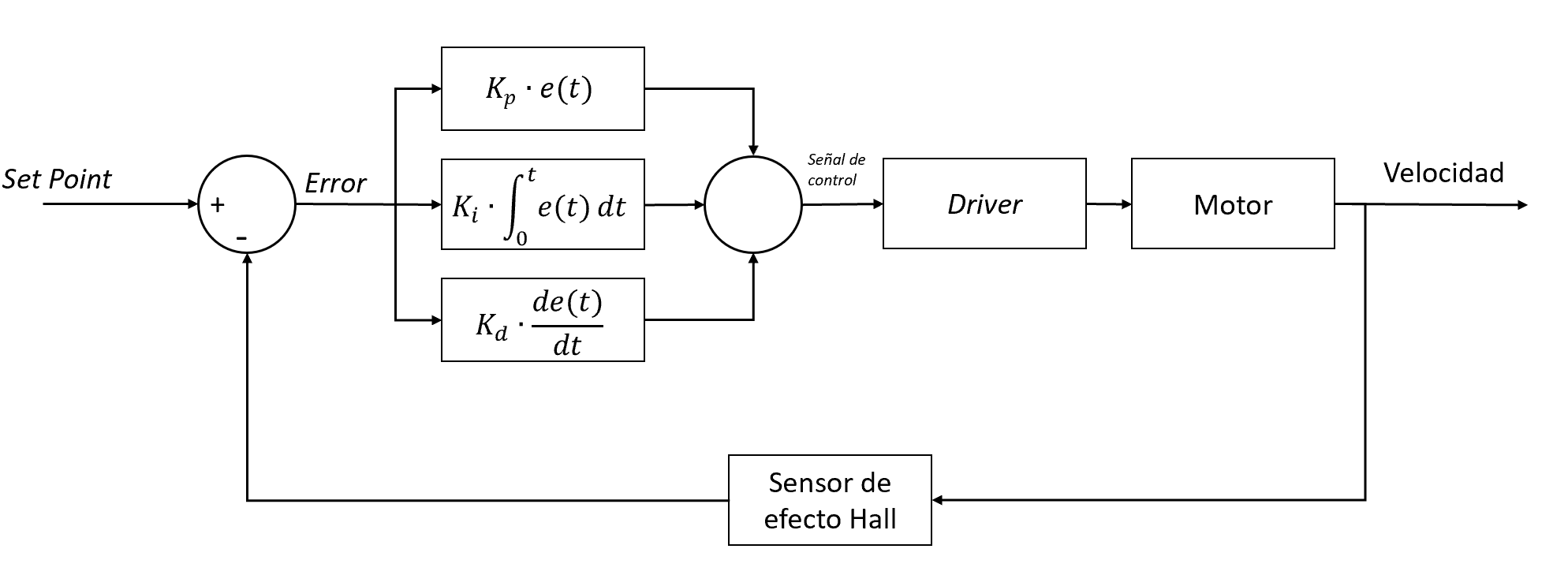
## Diagrama de flujo

El diagrama de flujo del código completo muestra cómo interactúan todos los elementos del *software* y se encuentra en la siguiente ubicación: **Proyecto\_Integrador/ESTRUCTURA DEL PROYECTO/3) Diseño/Diagrama de flujo.pdf**

* 1. Diagrama de estados

Dado que se estableció que el sistema tuviera un control de tiempo de tipo *RTO preemptive,* no se cuenta con un diagrama de estados como tal.

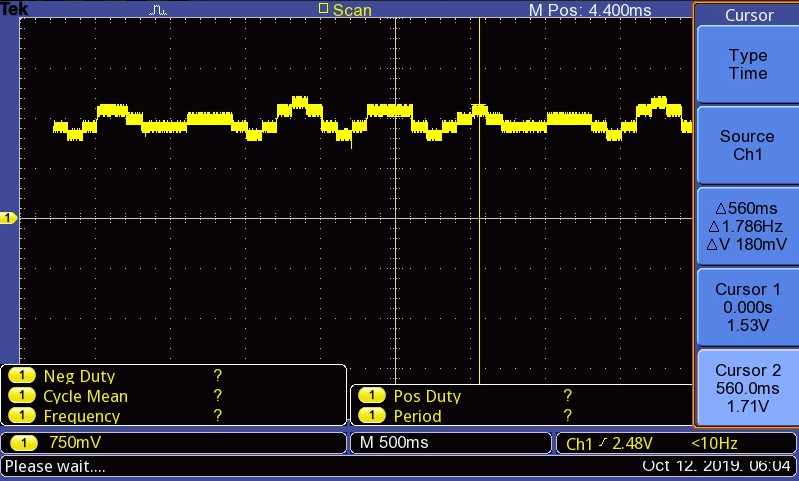
## Diagrama de control

El tipo de controlador elegido fue el Proporcional Integral Derivativo por su facilidad de ser implementado en C en un microcontrolador. En la siguiente figura se puede observar el diagrama de bloques del sistema controlado en lazo cerrado. El ruido del sistema se despreció para simplificar el diseño y la implementación del controlador por lo que no se encuentra en el diagrama de bloques. 

**Figura 9.1** Diagrama de control

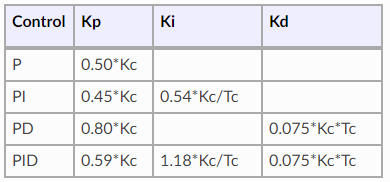
Para la sintonización de las constantes del controlador PID, se utilizó el método de Ziegler-Nichols en lazo cerrado. Para esto, se establecieron ganancias integral y derivativa de cero y se fue aumentando la ganancia proporcional hasta que la respuesta del sistema comenzara a oscilar. Una vez teniendo este comportamiento, se obtuvo la ganancia proporcional establecida como *Kc* (ganancia crítica).

Se midió con el osciloscopio la frecuencia de oscilación del sistema y se denominó *Tc* (tiempo crítico), el cual obtuvo un valor de 0.560 segundos, como se muestra en la imagen a continuación.



**Figura 9.2**. Oscilación crítica del sistema

Partiendo de estos dos valores, se obtuvieron las ganancias proporcional, integral y derivativa con base en la siguiente tabla:



**Figura 9.3** Sintonización de ganancias de un controlador

Sin embargo, al ingresar los valores al controlador, el sistema oscilaba ligeramente, por lo que se optó por reducir la ganancia integral hasta obtener un valor de estado estacionario muy cercano a cero. Las ganancias finales quedaron de la siguiente manera:

## Estándares

El estándar utilizado para el desarrollo de código C en el presente proyecto fue C99 (ISO/IEC 9899:1999). Este estándar introdujo punteros restringidos, matrices de longitud variable, miembros de matriz flexibles, soporte de números complejos, matemática de tipo genérico, int largo largo, identificadores extendidos, constantes hexadecimales de punto flotante, literales compuestos, inicializadores designados, una sola línea, // comentarios , tipo entero extendido, la capacidad de mezclar declaraciones y código, macros variables, nuevas funciones matemáticas, funciones en línea, tipos booleanos, \_Pragma y pragmas estándar, y VA\_COPY.

La herramienta utilizada para evaluar que el código desarrollado sigue el estándar elegido fue Cppcheck. Cppcheck es un analizador estático para código C y C ++. Es de código abierto, gratuito, multiplataforma y fácil de usar.

## *Naming conventions*

Como primer paso, y para seguir la estructura del código generado por el ambiente de desarrollo de *e2 Studio* y para evitar algún problema con los acentos, se definió que el lenguaje a utilizar fuera el inglés para todos los tipos de elementos en el código.

En la tabla a continuación se establecen los *naming conventions* que se definieron para identificar los diferentes elementos en el código.

|  |  |
| --- | --- |
| **Tipo de elemento de código** | ***Naming Convention*** |
| Variable local | Las variables locales **deben** de empezar con minúsculas y guiones bajos. |
| Variable global | Las variables globales **deben** de empezar con mayúscula y después minúsculas con guiones bajos. |
| Función local | Las funciones locales **deben** de empezar con minúsculas y guiones bajos. |
| Función global | Las funciones globales **deben** de empezar con mayúscula y después minúsculas con guiones bajos. |
| *Macros* | Las *macros* **deben** ir todas en mayúsculas. |
| Enumeraciones | Los elementos en las enumeraciones **deben** de ir en mayúsculas y guiones bajos. |
| Estructuras | Los elementos en la estructura **deben** de seguir el estilo de *camelCase*. |
| Archivos | Los nombres de las carpetas **deben** de ir todos en minúsculas y en lugar de espacios tener guiones bajos. |
| Carpetas | Los nombres de las carpetas **deben** de empezar con minúsculas y en lugar de espacios tener guiones bajos. |

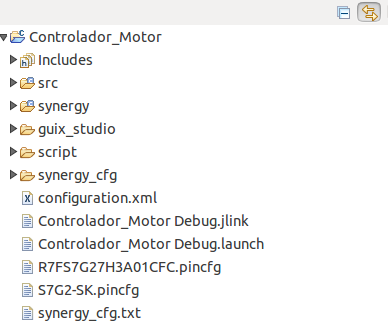
**Figura 9.4** *Naming convention* de los diferentes elementos del código

# Construcción del software

En esta sección se describen los lineamientos que se definieron durante el desarrollo del proyecto para poder seguir en conjunto una misma estructura en la parte de codificación del software.

* 1. Estructura del proyecto

Los directorios definidos en la estructura del proyecto parten del siguiente esquema:

  
**Figura 10.1.** Estructura de los directorios del proyecto

La definición para el uso de cada una de estas carpetas es la siguiente:

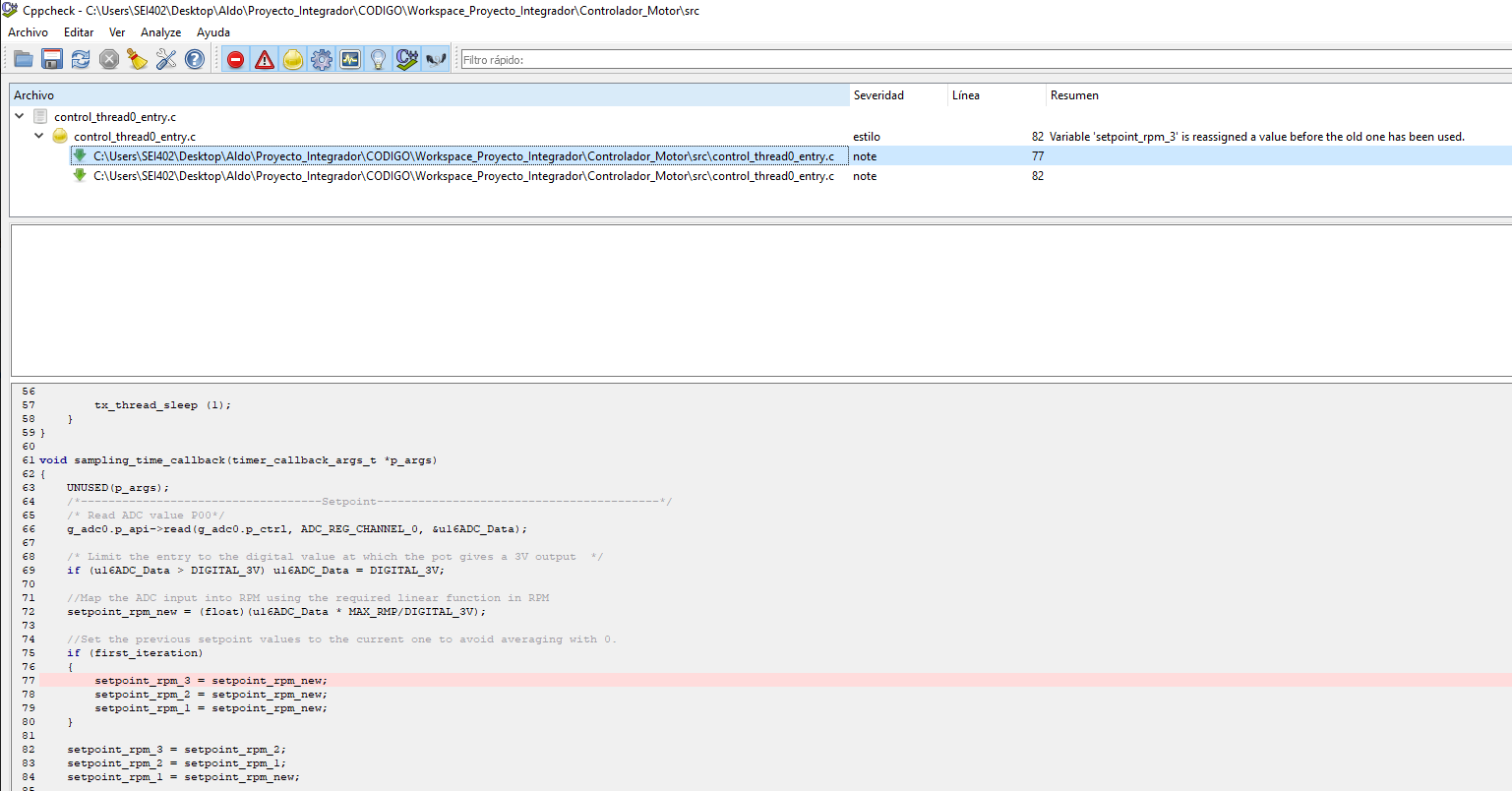
* **src**, se encuentran los archivos más relacionados a la propia aplicación, en este caso el control para un motor de DC.
* **synergy**, las librerías necesarias para utilizar la tarjeta de renesas SK-S7G2 se encuentran en este directorio. No es necesario modificar ningún archivo de esta carpeta manualmente.
* **guix\_studio**, el proyecto y los componentes necesarios para el software de GUIX Studio se colocan en este directorio. Son los archivos que no necesita la tarjeta como tal, pero son los necesarios para poder generar dichos archivos.
* **script**, scripts de enlazador para configurar regiones de memoria. No es necesario modificar ningún archivo de esta carpeta manualmente.
* **synergy\_cfg**, los archivos de configuración para la tarjeta de renesas SK-S7G2 se encuentran en este directorio. No es necesario modificar ningún archivo de esta carpeta manualmente.

En este caso, la carpeta que más usaremos para para codificar nuestras implementaciones y lógica del sistema se encontrarán en la carpeta *src*, y su estructura es la siguiente:

* **gui**, se encuentran los archivos necesarios para la configuración de la pantalla, como los colores predeterminados, pixeles, recursos y datos de inicialización.
* **hardware**, necesario para algunas comunicaciones entre componentes. Aquí se encuentra la definición de la función de configuración del panel LCD SK-S7 a través de la interfaz SPI.
* **synergy\_gen**, aquí se encuentran algunos de los archivos generados por el IDE para el funcionamiento de los threads y datos comunes entre ellos. En este directorio se encuentra ***main.c*** (archivo fuente principal).

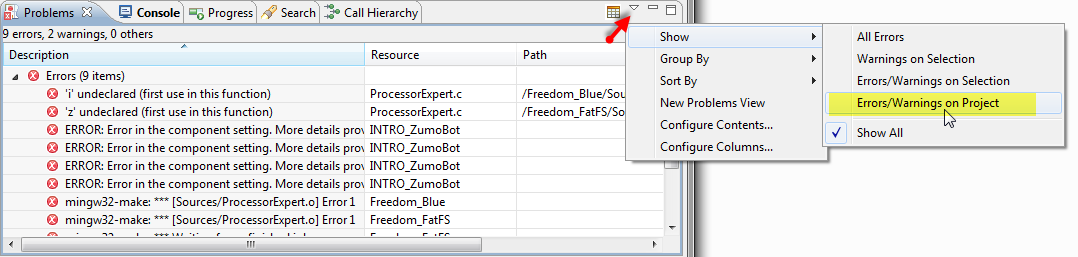
# Metodología de revisión de software

Se optó por tener revisiones de *software* antes de cada liberación, de manera que se asegurara que el código liberado a pruebas cumpliera con ciertos estándares (C99). Para la revisión estática del código se utilizó la herramienta de *Cppcheck,* la cual es capaz de detectar errores que el compilador no puede identificar. Básicamente se enfoca en proveer de un análisis de código para detectar *bugs* y comportamientos indefinidos o construcción peligrosa de código que pueda ocasionar problemas más adelante.



**Figura 11.1.** Captura de pantalla de *Cppcheck*

Además de haber utilizado esta herramienta, se buscaba eliminar todos los errores y *warnings* que indicaba la compilación del código en el ambiente de desarrollo (*IDE*).



**Figura 11.2.** Captura de errores identificados por el compilador

# Pruebas de software

Las pruebas de software contribuyen a que la implementación o producto final se comporte de manera correcta de acuerdo a los requisitos del cliente. Es importante ejecutar este tipo de actividades durante el desarrollo del producto, ya que ayudan a la calidad del mismo y al equipo de desarrollo a reducir costos de errores en el mercado final.

Existen varios tipos de pruebas, dentro de los cuales se seleccionaron las más importantes para el desarrollo del proyecto. Mostradas a continuación:

## Pruebas de caja negra

Este tipo de pruebas siguen la metodología de que la estructura o diseño internos no se prueba o no se conoce por la persona que la ejecuta. Esto es para evitar que la opinión del *tester* o el resultado final de la prueba se vean afectados por el juicio del desarrollador.

Los casos de prueba de este proyecto se hicieron basados en los requisitos finales. Se corrieron cada vez que se terminó un proceso de liberación de software (cada dos sprints) y se documentó lo correspondiente a las mismas. La evidencia se puede encontrar en la siguiente ruta: **Proyecto\_Integrador/ESTRUCTURA DEL PROYECTO/4) Verificación/PruebasCajaNegra.xls.**

## Pruebas de caja blanca

Este tipo de pruebas siguen la metodología de que la estructura y diseño internos se prueban y se conocen por la persona que la ejecuta. De esta manera el código se hace más robusto y evita que existan errores en el campo.

Los casos de prueba de este proyecto se hicieron basados en los requisitos finales. Se corrieron cada vez que se terminó un proceso de liberación de software (cada dos sprints) y se documentó lo correspondiente a las mismas. La evidencia se puede encontrar en la siguiente ruta: **Proyecto\_Integrador/ESTRUCTURA DEL PROYECTO/4) Verificación/PruebasCajaBlanca.xls.**

* 1. Análisis de comportamiento aplicado

Estas pruebas se incluyeron como pruebas de validación o pruebas de aceptación de usuario, ya que sirven en gran medida como evidencia al cliente, y para entender más detalladamente el funcionamiento general del motor.

Las pruebas realizadas se pueden dividir en tres grandes grupos: pruebas de linealidad, pruebas de arranque y paro y pruebas de comportamiento ante perturbaciones. Los resultados se encuentran en la siguiente ubicación: **Proyecto\_Integrador/ESTRUCTURA DEL PROYECTO/4) Verificación/Resultados/Validación.docx**

# Liberación

Se estableció que las liberaciones a pruebas se realizarán cada dos semanas, al igual que al cliente, sin embargo, éstas tendrán un desfase de 1 semana para poder ejecutar las pruebas y corregir los errores encontrados en éstas, de manera que se haga una liberación con una mayor calidad al cliente. También se estableció un acuerdo para nombrar a las versiones, la cual establece a continuación. Un diagrama más detallado sobre las fechas de liberación se encuentra en la siguiente ruta: **Proyecto\_Integrador/ESTRUCTURA DEL PROYECTO/2) Planeación/Liberaciones.pptx**

15/09/2019 - v0.1.0 - HW001

29/09/2019 - v0.2.0 - HW001

13/10/2019 - v0.3.0 - HW001

25/10/2019 - v1.0.0 - HW001

## Plan de entregas de liberaciones al cliente

En el calendario a continuación, se establecen las fechas de liberaciones tanto a pruebas como al cliente.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Lunes** | **Martes** | **Miércoles** | **Jueves** | **Viernes** | **Sábado** | **Domingo** |
|  |  |  |  |  | **Agosto 17** | **18** |
| **19** | **20** | **21** | **22** | **23** | **24** | **25** |
| **26** *Sprint* 1 | **27** | **28** | **29** | **30** | **31** | **Septiembre 1** |
| **2** *Sprint* 2 | **3** | **4** | **5** | **6** | **7** | **8**  Liberación de versión 0.1.0 (pruebas) |
| **9** *Sprint* 3 | **10** | **11** | **12** | **13** | **14** | **15**  Liberación de versión 0.1.0 (cliente) |
| ***16*** *Sprint* 4 | **17** | **18** | **19** | **20** | **21** | **22**  Liberación de versión 0.2.0 (pruebas) |
| ***23*** *Sprint* 5 | **24** | **25** | **26** | **27** | **28** | **29**  Liberación de versión 0.2.0 (cliente) |
| ***30*** *Sprint* 6 | **Octubre 1** | **2** | **3** | **4** | **5** | **6**  Liberación de versión 0.3.0 (pruebas) |
| ***7*** *Sprint* 7 | **8** | **9** | **10** | **11** | **12** | **13**  Liberación de versión 0.3.0 (cliente) |
| ***14*** *Sprint* 8 | **15** | **16** | **17** | **18** | **19** | **20**  Liberación de versión 1.0.0 (pruebas) |
| ***21*** | **22** | **23** | **24** | **25**  Liberación de versión 1.0.0 (cliente) | **26** |  |

**Figura 13.1**. Calendario de liberaciones de *software*.

## Estrategia de pruebas de integración

Las pruebas de integración se pueden definir como el paso inmediato anterior a que se haga cualquier liberación, ya sea a pruebas o al cliente. Éstas sirven para asegurar que la funcionalidad básica del producto no ha sido afectada por los cambios y nuevas implementaciones hechas al código.

La documentación correspondiente se puede encontrar en la siguiente ruta: **Proyecto\_Integrador/ESTRUCTURA DEL PROYECTO/4) Verificación/PruebasIntegracion.docx**

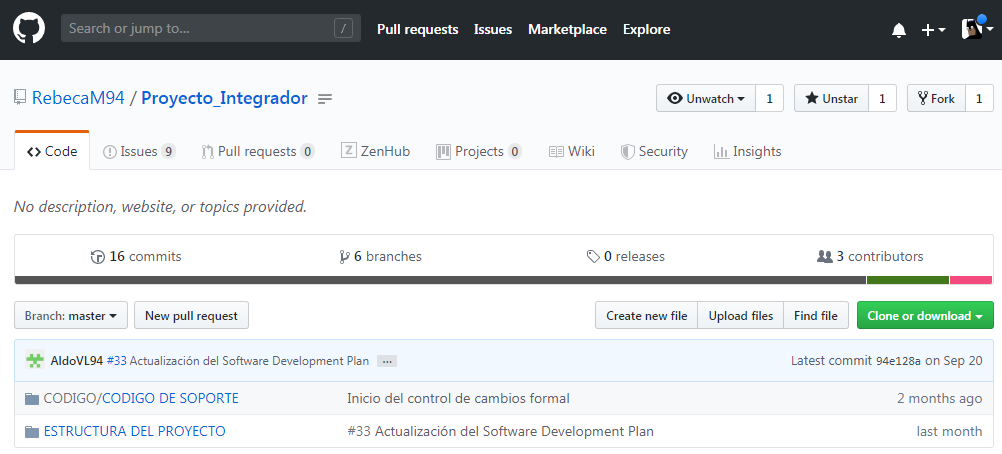
## Pruebas de validación en campo

Las pruebas de validación son aquellas que se ejecutan considerando el escenario o ambiente final del producto. Generalmente son realizadas por un ingeniero de sistemas basándose tanto en los requisitos del cliente como en las acciones más frecuentes de los usuarios finales, por eso son llamadas pruebas de aceptación de usuario (UAT por sus siglas en inglés).

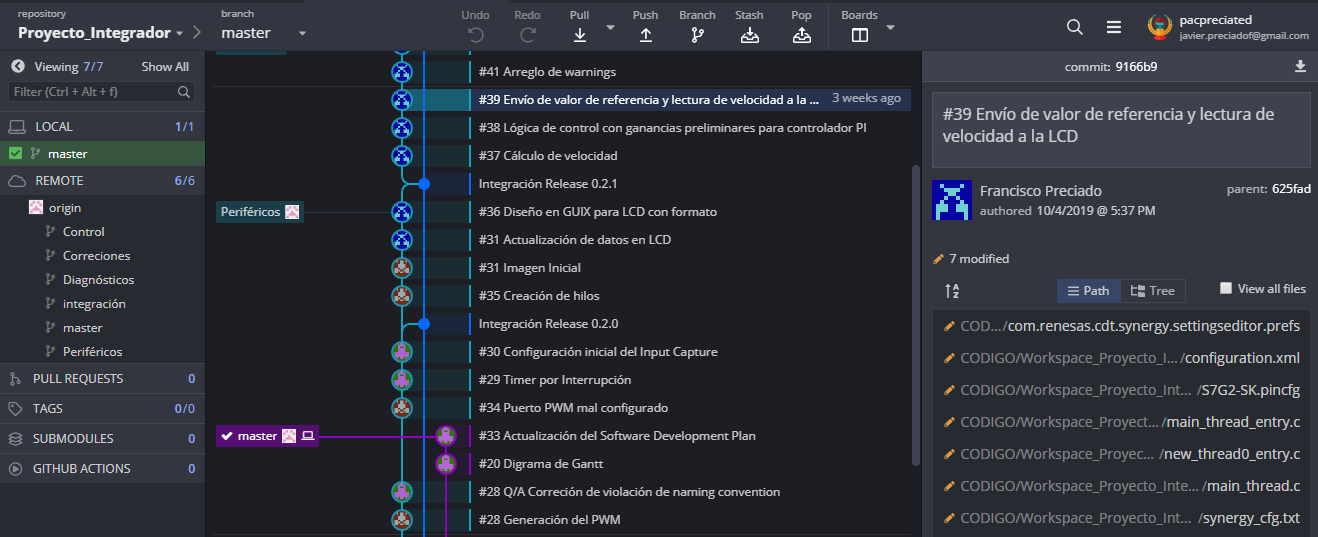
La documentación correspondiente se puede encontrar en la siguiente ruta: **Proyecto\_Integrador/ESTRUCTURA DEL PROYECTO/4) Verificación/Resultados/Liberación 1.0.0/PruebasValidacion.docx**

## Procedimientos de control

Como herramienta de control se decidió usar GitHub, el *software* de control de versiones para realizar integraciones de nuestros cambios en el código.

  
**Figura 13.2**. Repositorio del proyecto en GitHub

Además, se utilizó como herramienta gráfica de git el software *GitKraken* de la empresa Axosoft.

  
**Figura 13.3**. Repositorio del proyecto en herramienta gráfica GitKraken.

En caso de requerir acceso al repositorio localizado en GitHub, es necesario notificar al equipo para proporcionar los permisos necesarios. Si ya se cuenta con estos permisos, la liga de acceso es la siguiente: <https://github.com/RebecaM94/Proyecto_Integrador>

El procedimiento que se siguió para poder controlar nuestros cambios fue sencillo debido a la complejidad y duración de este proyecto. Se decidió crear un branch principal (master) y uno más disponible para las distintas integraciones programadas. Asimismo, se decidió crear distintos branches dependiendo de una funcionalidad mayor. Con esto, los branches contemplados en nuestro repositorio fueron:

* **master,** *branch* principal en el cual se tiene toda la documentación.
* **integración,** *branch* utilizado para realizar el *release* de cada *sprint*.
* **Periféricos,** para la inicialización y configuración de todos los periféricos necesarios en el sistema.
* **Control,** utilizado para la implementación de todos las características necesarias para la etapa de control.
* **Correcciones,** en este branch se suben los cambios relacionados a pequeñas correcciones a problemas identificados antes o después de las pruebas.
* **Diagnósticos,** para la implementación de todo lo relacionado a detección de fallas y sus respectivos indicadores en la pantalla.

# Referencias

Método de Ziegler-Nichols - Control Automático - Picuino. (2019). Recuperado el 28 de Octubre de 2019, de https://www.picuino.com/es/arduprog/control-ziegler-nichols.html

C99 - C (2015). WikiChip. Recuperado el 28 de Octubre de 2019, de https://en.wikichip.org/wiki/c/c99