Robot Competición 2022

**Tabla de contenido**

[Enunciado de trabajo 3](#_Toc117205662)

[Mini Plan del Proyecto 1](#_Toc117205663)

[Identificación de Riesgos 1](#_Toc117205664)

[Metodología 3](#_Toc117205665)

[Estrategia de control de versiones 4](#_Toc117205666)

[Especificación de requisitos 11](#_Toc117205667)

[Diseño de Software 14](#_Toc117205668)

[Registros de trazabilidad 18](#_Toc117205669)

[Casos y Procedimientos de prueba del software 21](#_Toc117205670)

[Registro de aceptación 22](#_Toc117205671)

# Enunciado de trabajo

**1. Descripción del producto**

* 1. **Propósito**

Equipos de estudiantes ensamblarán un kit de robot, desarrollarán el software para controlarlo y participarán en una carrera, que consiste en tres tramos con características diferentes.

**1.2 Requisitos generales**

La pista de carreras consiste en tres tramos, cada uno de los tramos se deberá de completar por medio de un algoritmo diferente. Para completar cada tramo se cuenta con un tiempo máximo de 5 min. Se sumará el tiempo que llevó completar cada tramo, el equipo con menor tiempo será el equipo ganador. Los tramos de la pista son:

* Seguidor de Líneas
* Evasión de obstáculos
* Activación del robot por ruido

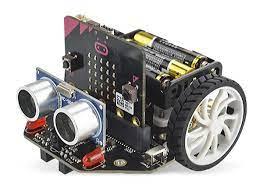


Figura 1. Robot

1. **El estudiante ensamblará el robot.**

* Instrucciones para ensamblar los equipos
  + Tiempo estimado de ensamblaje: 10 minutos
* Componentes del Kit de Robots

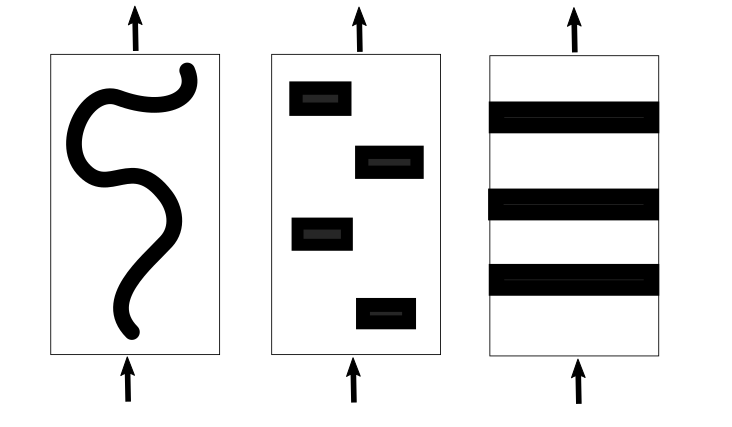
1. Tarjeta micro:bit (microcontrolador y sensores)
2. Kit maqueen for micro:bit (base, llantas, motores, sensores,battery holder)
3. 3 Pilas AAA recargables
4. **Descripción del alcance de lo que está incluido y lo que no está incluido.**

* El robot se mueva hacia adelante
* El robot se mueva hacia atrás
* El robot haga el alto total
* El robot pueda girar a la derecha
* El robot pueda girar a la izquierda
* Pruebas de evasión de obstáculos
* Prueba de medición de ruido

1. **Objetivos del proyecto**

Desarrollar el software de un robot para competir en la carrera usando la guía de gestión e ingeniería del perfil básico del ISO/IEC 29110 para competir con el robot en la carrera.

La figura 2 muestra la pista de carreras, que consiste en tres tramos, cada uno de los tramos se deberá de completar por medio de un algoritmo diferente. Para completar cada tramo se cuenta con un tiempo máximo de 5 min. Se sumará el tiempo que llevó completar cada tramo, el equipo con menor tiempo será el equipo ganador. Los tramos de la pista son:



(a) (b) (c)

Figura 2. Pista de carrera para la competición del robot (a) seguidor de líneas; (b) evasión de obstáculos y (c) activación del robot por ruido

1. **Entregables**

La configuración de software del Robot de acuerdo a como se describe en la guía de gestión e ingeniería del perfil básico del ISO/IEC 29110 ilustrado en la Figura 3.

1. Mini plan del proyecto (2 páginas)
2. Documento de especificación (2-3 páginas)
3. Documento de identificación de riesgos (1-2 páginas)
4. Documento de diseño (2-3 páginas)
5. Documento de casos y procedimientos de prueba (3-5 páginas)
6. Documentos de software (ej. código) (5-10 páginas)
   1. El software del robot se programará en lenguaje C
   2. Una guía de programación debe ser usada
7. Matriz de trazabilidad (1-2 páginas)
   1. Requisitos-Diseño-Código-Pruebas
8. Presentación (2 diapositivas)
9. Participación en la carrera de robots usando el software desarrollado

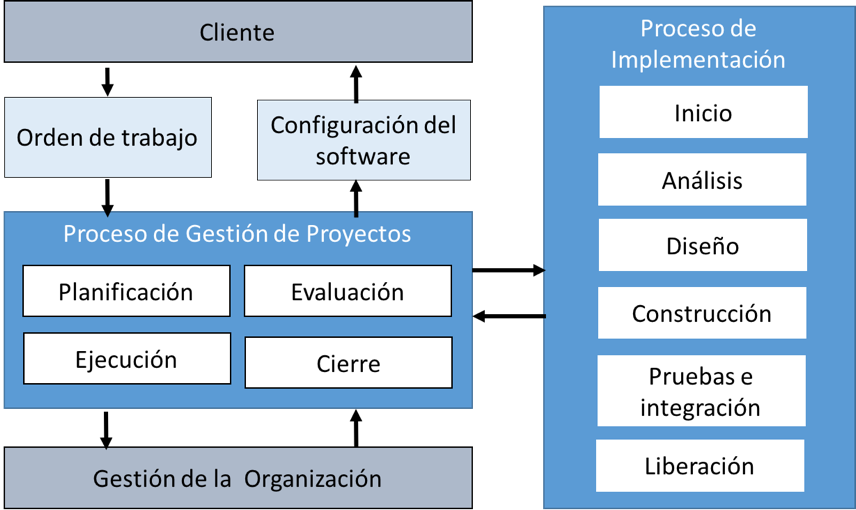
****

Figura 3. Visión general de procesos y actividades del perfil básico del ISO 29110

1. **Reglas y penalizaciones**

Los equipos seguirán un conjunto de reglas para asegurar que la competencia del robot se realice de manera segura por todos los participantes. Si en algún momento la operación del robot es considerada por un jurado como insegura, el robot será detenido. El robot será inspeccionado antes de que éste pueda participar en otra carrera.

El jurado tendrá la última palabra y será la autoridad que proporcione los puntos durante la competencia de robots. Sus decisiones son definitivas e inapelables.

1. **Reglas**
2. Acceso a la pista de la carrera
   * A ningún equipo le está permitido probar la carrera antes de la competencia.
3. Tiempo entre carrera
   * Los equipos tienen un mínimo de 5 minutos para su participación entre dos carreras consecutivas.
4. Destrucción, daño, golpes, etc.
   * No se permiten estrategias y/o acciones enfocadas en la destrucción, daño, vuelco, golpe o enredo entre Robots.
5. Extensión del robot fuera del perímetro de la pista de carreras
   * Si alguna parte del robot hace contacto con algo fuera del perímetro de la pista de carreras.
6. Gestión del equipo
   * Cada equipo incluirá hasta dos Pilotos designados.
7. Forzar a un oponente a romper una regla
   * Las acciones de un equipo o su robot no deben ser causa para que otro equipo rompa una regla.
8. Tomar el robot de otro equipo
   * No se tiene permitido que los equipos toquen el robot de otro equipo una vez iniciada la carrera.
9. Bloquear un robot de otro equipo
   * Los robots no deben ser bloqueados por otros robots
10. **Penalizaciones**

* Es la consecuencia por la violación de una regla identificada por un árbitro.
* Cada penalización equivale a quitar 10 puntos al equipo.

**Desarrollado por**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Fecha** (dd-mm-aaaa): \_\_\_\_\_\_\_\_\_\_

**Aprobado por**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Fecha** (dd-mm-aaaa): \_\_\_\_\_\_\_\_\_\_

# Mini Plan del Proyecto

Los elementos de un mini plan de proyecto son:

* Duración estimada para ejecutar cada tarea y salida
* Composición del equipo de trabajo asignando a los roles y sus responsabilidades

El esfuerzo real y estimado son estimados y medidos con una precisión de +/- 15 minutos

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Rol** | **Nombre** | **Nombre de la tarea** | **Salida** | **Esfuerzo estimado (hora)** | **Esfuerzo real (hora)** |
| **Analista** | LUIS ALBERTO LORENZO PÉREZ | * Identificar el problema. * Levantamiento de requerimientos. * Identificar materiales a usar. * Definir el tipo de metodología a implementar. * Definir cronograma de actividades. | * Plan de proyecto. | 4hrs |  |
| **Diseño** | SAMUEL PERALTA ARECHIGA | * Elaborar el diseño del sistema. * Diseñar la arquitectura del software. | * Diseño del sistema. | 4hrs |  |
| **Programador** | SAMUEL PERALTA ARECHIGA | * Desarrollar el software. * Identificar errores. * Realizar pruebas y correcciones. | * Programación y mantenimiento. | 4hrs |  |
| **Líder Técnico** | LUIS ALBERTO LORENZO PÉREZ | * Verificar que el programa ejecute las instrucciones que se marcan en el diseño. * Comunicar al cliente el progreso del proyecto. * Gestionar los materiales necesarios para que el equipo cumpla con el objetivo del proyecto. * Apoyar y aportar ideas al presentar algún problema. | * Gestión del funcionamiento del proyecto. | 5hrs |  |

**Desarrollado por**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Fecha** (dd-mm-aaaa): \_\_\_\_\_\_\_\_\_\_

**Aprobado por**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Fecha** (dd-mm-aaaa): \_\_\_\_\_\_\_\_\_\_

# Identificación de Riesgos

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID Riesgo** | **Descripción del riesgo** | **P** | **I** | **E** | **Mitigación del riesgo** |
| 1 | No revisar el proyecto con el equipo de trabajo en el taller por falta de tiempo. | B | A | M | Haciendo uso de las herramientas de comunicación en línea como lo es el teams. |
| 2 | Mala organización en el desarrollo del software. | M | A | A | Realizando un cronograma de las horas que se invertirán para cada proceso del software. |
| 3 | Desconocimiento de la tecnología base del proyecto | A | A | A | Tomar un tutorial básico del uso de la herramienta. |
| 4 | Falta de apoyo y compromiso. | A | B | M | Se compensa con los miembros que estén activos en el proyecto. |
| 5 | Falta de pruebas en el producto físico. | A | A | A | Con ayuda de simuladores podemos realizar más pruebas del software. |
| 6 | Riesgos externos, fenómeno natural (Huracán en aproximación a las costas). | A | M | A | Estar atentos a la información que brinda el sistema de protección civil y adaptarse al cambio para las nuevas estimaciones de fechas de entrega. |

**Desarrollado por**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Fecha** (dd-mm-aaaa): \_20/10/2022\_\_\_\_\_

**Aprobado por**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Fecha** (dd-mm-aaaa): \_\_\_\_\_\_\_\_\_\_

Tabla de Clasificación

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Impacto** | | |
| **Probabilidad** | **Bajo** | **Medio** | **Alto** |
| **Bajo** | Bajo | Bajo | Medio |
| **Medio** | Bajo | Medio | Alto |
| **Alto** | Medio | Alto | Alto |

# Metodología

**Análisis**: en esta etapa vamos a comprender todo lo que está relacionado con el proyecto, se debe entender de manera detallada los requisitos que se deben de cumplir. Como lo es, el diseñar el código para que robot microbit sea capaz de recorrer tres pistas.

1. Seguir la línea de la pista de color blanco, el robot debe ser capaz de moverse hacia delante, hacia atrás, girar a la derecha y girar a la izquierda.
2. El robot debe ser capaz de esquivar obstáculos con ayuda de un sensor ultrasónico.
3. El robot debe ser capaz de llegar a un punto a otro con ayuda de un sensor de ruido.

Es importante tener en cuenta los requerimientos que debe cumplir el código para que sea eficiente.

**Diseño**: en esta parte nos sirve para formular soluciones específicas de acuerdo con el problema que queremos resolver. En este caso vamos a diseñar un código que le indique al microbit que avance y si deja de detectar una línea o si se encuentra un objeto que gire o retroceda.

Implementación: para este punto ya tenemos el software desarrollado para el robot, entonces hacemos las pruebas necesarias para saber si el producto obtenido cumple o no con lo deseado. En este punto tenemos nos encontramos en la busque de errores para poder corregir o reducir esos errores.

**Verificación**: entramos en las pruebas de aceptación del producto, donde nos podemos apoyar de una bitácora para hacer un checklist respecto a los puntos que si se cumplieron con respecto al producto final obtenido. Se acepta el desempeño del robot en cada pista de prueba.

**Mantenimiento**: en esta etapa final de la metodología se puede aprovechar en un futuro para la realización de mejoras en futuras competencias de microbit, esto va ligado con el control de versiones que serán de ayuda para poder analizar la versión en la que se encuentra y realizar las mejoras correspondientes o estudiar sus posibles deficiencias de esa versión.

# Estrategia de control de versiones

Repositorio

Para poder manejar la estrategia de control de versiones optamos por utilizar un repositorio independiente para la documentación generada y otro para el código ejecutable, apoyandonos de la herramienta GitHub el cual es un sistema de control de versiones distribuido gratuito y de código abierto diseñado para manejar todo, desde proyectos pequeños hasta proyectos muy grandes, con rapidez y eficiencia.

Al ingresar en GitHub tenemos el proyecto dividido en dos repositorios, la documentación y la codificación.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Como varios integrantes del equipo están involucrados se crearon ramas en el cual se trabajaron el analista y el programador.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Dentro del repositorio de codificación tenemos las pruebas que se han realizado en este transcurso de tiempo con sus diferentes versiones dependiendo de las pruebas realizadas

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

De igual manera contamos con el repositorio de documentos en el cual anexamos las versiones de documentos que se han realizado en el transcurso del proyecto.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. **Elementos requeridos para entrega del producto (hardware, software, documentación)**

Verificar el cumplimiento del proyecto de acuerdo a las instrucciones de entrega establecidas en el plan del proyecto, proveyendo apoyo para su aceptación y obteniendo el Registro de Aceptación firmado.

|  |  |
| --- | --- |
|  | |
| **Objetivos** | Elementos requeridos para entregar el producto o servicio a conformidad del cliente |
| **Razón: Fundamental:** | Cerrar y entregar formalmente el proyecto certificando la conformidad del cliente. |
| **Roles:** | Gestor de proyecto |
| Cliente. |
| **Artefactos:** | Instrucciones de entrega. |
| Registro de aceptación. |
| Reporte Cierre Proyecto. |
| Robot |
| **Pasos:** | Ejecutar las actividades de cierre formal de proyecto. |
| Entrega y aceptación del proyecto. |
| Evaluación post implementación y registro de lecciones aprendidas. |
| Versión de la documentación generada. |
| **Descripción de Pasos:** | **Paso1: Ejecutar las actividades de cierre formal de proyecto.**  Actualizar, consolidar y revisar la documentación final del proyecto. Validar que todos los errores reportados han sido cerrados satisfactoriamente y que se hayan cumplido las Instrucciones de entrega definidas en la fase de planeación. Dejar documentado cualquier pendiente o mejora sugerida. Garantizar que las pruebas de aceptación se han ejecutado completamente y que los errores reportados han sido cerrados y el proyecto cumple con los criterios de aceptación establecidos.  **Paso2: Entrega y aceptación del proyecto.**  En este paso se hace la entrega formal del proyecto al Cliente, es importante que el cliente acepte la entrega satisfactoria del proyecto directamente o a través de quienes lo representen. Finalmente se deberá informar a todos los interesados el cierre formal del proyecto y se diligenciará el artefacto Registro de aceptación avalando la entrega del producto o servicio por parte del cliente.  **Paso3: Evaluación post implementación y registro de lecciones aprendidas.**  Este paso busca recopilar las experiencias del proyecto, consolidar y evaluar los resultados obtenidos y validar las lecciones aprendidas para ajustar y mejorar el proceso, y para revisión de los históricos de futuros proyectos. Finalmente se recopila la experiencia del proyecto a través del artefacto Reporte de Cierre del Proyecto.  **Paso 4: Versión de la documentación generada.**  Todos y cada uno de los productos o formatos usados en la actividad de cierre, deben ser versionados en la herramienta GitHub. |

1. **Requisitos de entrega**
2. **Orden secuencia de las tareas a ser realizadas para la entrega**

Tareas

Planeamiento y Preparación de Repositorio.

|  |  |
| --- | --- |
|  | |
| **Objetivos** | Desarrollar una estrategia de control de versiones e instalar un ambiente para ejecutar el control de versiones. |
| **Razón: Fundamental:** | Debido a que cada proyecto tiene diferentes características, se tienen que adaptar/aplicar las prácticas de control de versiones que encajen con los requerimientos del proyecto y aseguren que todos los participantes tengan conocimiento de cómo el control de versiones será implementado. El repositorio es usado para almacenar y controlar la configuración de software (código fuente y artefactos asociados) de un sistema. |
| **Roles:** | Gestor de proyecto |
| Líder Técnico |
| **Artefactos:** | Estrategia de control de versiones |
| Repositorio del proyecto |
| **Pasos:** | 1. Crear la estrategia de control de versiones |
| 2. Crear un repositorio |
| **Descripción de Pasos:** | **Paso 1. Crear la estrategia de control de versiones:**  El gestor de proyecto crea la estrategia de control de versiones. El control de versiones es documentado en el plan de proyecto  **Paso 2. Crear un repositorio:**  El líder técnico crea un repositorio para el proyecto y lo configura.   * Crear nuevo repositorio. * Crear espacios o carpetas (por ejemplo, espacio de trabajo y espacio controlado). * Determinar accesos al repositorio. * Proveer mecanismos para el almacenamiento, descarga, elementos cambiantes, respaldo y recuperación. * Entrenar a los miembros del equipo y stakeholders acerca de: estrategia de control de versiones, repositorio, procedimientos y herramientas. * Realizar regularmente un respaldo del repositorio. * Verificar que el respaldo se realizó satisfactoriamente. |

1. **Liberaciones aplicables identificadas**
2. **Identificar todos los componentes del software con información de versiones**

Establecer el repositorio del proyecto usando la estrategia de control de versiones.

|  |  |
| --- | --- |
|  | |
| **Objetivos** | Implementar la estrategia para controlar las versiones de los documentos para la dirección del proyecto. |
| **Razón: Fundamental:** | Gestionar de forma controlada, segura y oportuna la información del proyecto. |
| **Roles:** | Gestor de proyecto |
| Líder Técnico |
| **Artefactos:** | Documentación del plan de proyecto. |
| Codificación del proyecto. |
| **Pasos:** | Se crea la estructura de espacios. |
| Se cargan los artefactos a cada carpeta. |
| **Descripción de Pasos:** | **Paso 1. Se crea la estructura de espacios.**  El gestor de proyecto o líder técnico crea, en la herramienta de GitHub, carpetas para el almacenamiento del proyecto.  **Paso 2. Se cargan los contenidos a cada espacio.**  Se cargan los artefactos a cada carpeta, dependiendo de la estructura definida para el control de los documentos de cada actividad. |

1. **Identificar todos los procedimientos de respaldo y recuperación**

Realizar backup del repositorio del proyecto de acuerdo a la estrategia de control de versiones

|  |  |
| --- | --- |
|  | |
| **Objetivos** | Versionar los documentos del proyecto y notificaciones de aprobaciones, en la herramienta de GitHub. |
| **Razón: Fundamental:** | Gestionar los cambios de la información del proyecto de forma clara, segura y controlada, permitiendo revisar los cambios que ha tenido la información y la justificación de cada cambio, en el repositorio de GitHub del proyecto. |
| **Roles:** | Gestor de proyecto |
| Líder Técnico |
| **Artefactos:** | Documentación del plan de proyecto. |
| Codificación del proyecto. |
| **Pasos:** | Listar documentos modificados. |
| Listar documentos nuevos. |
| Listar documentos aprobados, identificando la persona que aprueba y quien remite para aprobación. |
| Actualizar documentación en el repositorio. |
| Codificación del proyecto. |
| **Descripción de Pasos:** | **Paso 1. Listar documentos modificados.**  El gestor de proyecto o líder técnico identifica los documentos del plan del proyecto modificados.  **Paso 2. Listar documentos nuevos.**  El gestor de proyecto o líder técnico identifica los documentos nuevos del plan del proyecto Paso  **Paso 3. Listar documentos aprobados, identificando la persona que aprueba y quien remite para aprobación.**  El gestor de proyecto o líder técnico identifica los documentos, a los cuales se les ha generado una acción para aprobación.  **Paso 4. Actualizar documentación en el repositorio.**  El gestor de proyecto o líder técnico versiona los documentos modificados, nuevos y registra las evidencias de las aprobaciones sobre ciertos documentos.  **Paso 5. Codificación del sistema.**  El gestor de proyecto o líder técnico versiona el código que se esta elaborando para la construcción del proyecto. |

Realizar la recuperación del repositorio del proyecto usando el backup del repositorio del proyecto, si es necesario.

|  |  |
| --- | --- |
|  | |
| **Objetivos** | Verificar que el repositorio de GitHub opera correctamente |
| **Razón: Fundamental:** | Realizar pruebas de que la documentación está siendo gestionada correctamente por la herramienta de GitHub. |
| **Roles:** | Gestor de proyecto |
| Líder Técnico |
| **Artefactos:** | Verificación de resultados. |
| **Pasos:** | Verificar históricos de cambio, de un contenido. |
| Descargar las dos últimas versiones. |
| Verificar que los contenidos tengan los cambios respectivos. |
| Registrar verificación de resultado. |
| **Descripción de Pasos:** | **Paso 1. Seleccionar artefacto versionado en el repositorio.**  El gestor de proyecto o líder técnico selecciona un artefacto del plan del proyecto y verifica que tenga más de dos versiones generadas.  **Paso 2. Recuperar y evaluar las dos últimas versiones de un artefacto.**  El gestor de proyecto o líder técnico recupera las dos versiones del documento y verifica el contenido del artefacto para evaluar el cambio respectivo. |

**Desarrollado por**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Fecha** (dd-mm-aaaa): \_\_\_\_\_\_\_\_\_\_

**Aprobado por**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Fecha** (dd-mm-aaaa): \_\_\_\_\_\_\_\_\_\_

# Especificación de requisitos

**1. Introducción**

El equipo deberá construir un software capaz de poder controlar un robot para tres tramos de una carrera. Cada tramo de la carrera se debe completar en un tiempo máximo de 5 minutos, tras finalizar cada tramo, el tiempo acumulado se sumará y el equipo con menor tiempo de ejecución de carrera ganará.

Cabe destacar que cada tramo de la pista consta de una tarea diferente a desarrollar para el robot y entonces el robot debe ser capaz de:

* El robot se mueva hacia adelante
* El robot se mueva hacia atrás
* El robot haga el alto total
* El robot pueda girar a la derecha
* El robot pueda girar a la izquierda
* El robot pueda seguir una línea
* Pruebas de evasión de obstáculos
* Prueba de medición de ruido

1. **Descripción de Requisitos**
   1. Funcionalidad

REQ-1 Capacidad de trasladarse.

REQ-2 Capacidad de trasladarse en reversa.

REQ-3 Capacidad de hacer alto total.

REQ-4 Capacidad de girar 90 grados en dirección a la derecha.

REQ-5 Capacidad de girar 90 grados en dirección a la izquierda.

REQ-6 Capacidad de identificar entorno de desplazamiento.

REQ-7 Capacidad de evasión de obstáculos.

REQ-8 Capacidad de medición acústica del entorno.

REQ-9 Capacidad identificar un obstáculo potencial que intervengan con la

trayectoria del dispositivo.

REQ-10 Capacidad de cuantificar la distancia entre el dispositivo y el obstáculo.

REQ-11 Capacidad de evadir obstáculos.

* 1. Interfaces de usuario, interfaces externas
* Se codificará utilizando la herramienta MakeCode de Microsoft
* El robot será codificado de manera secuencial.
* Se utilizará un kit de robot Maqueen que incluye: Tarjeta microbit   
  (microcontrolador y sensores), Kit Maqueen for micro;bit: (base, llantas, motores, sensores battery holder) y 3 pilas AAA recargables.
* La interfaz externa puede ser a través de dos tipos de conexión una es vía inalámbrica (bluetooth) y la otra es la vía alámbrica por medio de un cable tipo micro b, para el desarrollo del proyecto se utilizará la segunda opción.
  1. Confiabilidad, eficiencia, mantenibilidad y portabilidad

• El robot debe contar con una documentación actualizada que permita realizar operaciones de mantenimiento con el menos esfuerzo posible.

• El robot no debe sobrepasar los 5 minutos para desplazarse en los 3 escenarios.

• Se debe contar con una copia de seguridad del código a cargar para prevenir riesgos de funcionamiento.

• El robot no debe sobrepasar los 5 minutos para desplazarse en los 3 escenarios.

* 1. Limitaciones / restricciones del diseño y construcción

Se debe entregar un robot con las funcionalidades mencionadas en el punto 2.1, por lo cual no se podrán agregar más funcionalidades.

* 1. Interoperabilidad

Con el fin de intercambiar la información y poder hacerla llegar de la forma más práctica, fácil y eficientemente la herramienta Makecode nos ofrece la posibilidad de compartir mediante un link nuestro proyecto creado.

* 1. Reusabilidad

Es importante mencionar que el código del proyecto se debe respaldar y almacenar en dispositivos de fácil transporte para su rápido y practico uso.

* 1. Legales y de regulación

• Utilizar la norma ISO/IEC 29110 para el desarrollo del software.

• No se permite la venta del código del robot ni sus componentes.

• Respetar el uso de reglas y penalizaciones.

**Reglas**

1. Acceso a la pista de la carrera
   * A ningún equipo le está permitido probar la carrera antes de la competencia.
2. Tiempo entre carrera
   * Los equipos tienen un mínimo de 5 minutos para su participación entre dos carreras consecutivas.
3. Destrucción, daño, golpes, etc.
   * No se permiten estrategias y/o acciones enfocadas en la destrucción, daño, vuelco, golpe o enredo entre Robots.
4. Extensión del robot fuera del perímetro de la pista de carreras
   * Si alguna parte del robot hace contacto con algo fuera del perímetro de la pista de carreras.
5. Gestión del equipo
   * Cada equipo incluirá hasta dos Pilotos designados.
6. Forzar a un oponente a romper una regla
   * Las acciones de un equipo o su robot no deben ser causa para que otro equipo rompa una regla.
7. Tomar el robot de otro equipo
   * No se tiene permitido que los equipos toquen el robot de otro equipo una vez iniciada la carrera.
8. Bloquear un robot de otro equipo
   * Los robots no deben ser bloqueados por otros robots
9. **Penalizaciones**

* Es la consecuencia por la violación de una regla identificada por un árbitro.
* Cada penalización equivale a quitar 10 puntos al equipo.

**Estado (**verificado, validado y en línea base)**:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Desarrollado por**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Fecha** (dd-mm-aaaa): \_\_\_\_\_\_\_\_\_\_

**Aprobado por**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Fecha** (dd-mm-aaaa): \_\_\_\_\_\_\_\_\_

NOTA 1: Cada requisito se identifica, es único y verificable o puede ser evaluado.

NOTE 2: Si una sección del formulario no se utiliza para un proyecto específico. El autor debe indicarlo con a siguiente declaración: NO APLICA

# Diseño de Software

**1. Introducción**

Para el desarrollo de software de este proyecto se debe tener en cuenta que diseño del robot debe cumplir las especificaciones que se mencionaron en los requerimientos funcionales del punto 2.1.

La placa del micro bit que se codifica en makecode es capaz de recibir las instrucciones y ejecutarlas al instante una vez que estas se descargan y se cargan a la placa.

**2. Información gráfica y textual de la estructura del software.**

1. **Diseño de alto nivel de la arquitectura del software:**

**Componentes de Hardware**:

* Placa micro: bit
* 2 sensores de infrarrojo.
* Motores variables por PWM.
* Un sensor ultrasónico.
* 2 ruedas
* Set de pilas

**Componentes de Software:**

* Plataforma de código abierto MakeCode.

**Relación de Componentes:**

* A través de la plataforma MakeCode y con la extensión macqueen, por medio de programación por bloques se especifican las instrucciones para encender los motores variables, así como la dirección que deben tomar con respecto a los valores obtenidos por medio de los 2 sensores infrarrojos.
* Será capaz de desplazarse siguiendo una línea de color blanca ya que, al no tener una continuidad de luz en los infrarrojos, no hay una transmisión de señal.
* Nota: El color de la línea dependerá del tipo de infrarrojos que se utilicen ya que puede ser línea negra o línea blanca.

1. **Diseño detallado del desarrollo de software** – Incluye detalles de los componentes de software para facilitar su construcción y prueba dentro del entorno de programación;

* Proporciona un diseño detallado (puede ser presentado como prototipo, diagrama de flujo, diagrama de relaciones, seudocódigo, etc.)
* Proporciona formatos de entrada y salida de datos
* Proporciona especificación de las necesidades para almacenamiento de datos
* Establece los convenios de nomenclatura de datos requerida
* Define los formatos requeridos para la estructura de datos
* Define los campos de datos y el propósito de cada elemento de datos requerido
* Proporciona las especificaciones de la estructura del programa

**Programa Seguidor de líneas:**

**Descripción:** El diseño secuencial del seguidor de líneas consiste que, durante la instrucción “Por siempre” ambos sensoresavancen mientras tengan un piso blanco(sensor derecho e izquierdo detecten valores en 1) en caso de encontrar una superficie que no se la pista (se recupere un valor 0) el motor contrario a la orientación del sensor girara, esto con el fin mantener el seguimiento de la línea.

**Estructura:**

**Texto

Descripción generada automáticamente con confianza baja**

**Programa Evasor de Obstáculos.**

**Descripción:**

Dentro de la instrucción “Por siempre”, ambos motores comenzaran a avanzar a una velocidad de 40, es aquí donde el ciclo while comenzara, al momento de detectar un obstáculo a una distancia de 10cm, se mostrará un indicador en la pantalla led con una “X”, estando aquí iniciara una reversa con un retardo de 1000s para posteriormente girar a la derecha, avanzar y girar a la izquierda para alinearse nuevamente y continuar el trayecto.

**Estructura:**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Estado (**verificado y en línea base)**:**  Terminado.

**Desarrollado por**: Esdras Eduardo García Espinosa

**fecha** (20-10-2022)

**Aprobado por**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Fecha** (20-10-2022)

# Registros de trazabilidad

Documentar la relación entre los requisitos incluidos en la especificación de requisitos, los elementos del diseño del software, los componentes del software, los casos de prueba y los procedimientos de prueba.

Este documento puede incluir:

* Identificar requisitos de la especificación de requisitos para la trazabilidad
* Proporcionar un mapeo hacia delante y hacia atrás de los requisitos hacia los elementos del diseño de software, los componentes de software, los casis de prueba y los procedimientos de prueba.

Los estatus aplicables son: verificados, en línea base y actualizado.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Registro de trazabilidad (robot maqueen)** | | | | | | | | | |
| **Número de Identificación del requisito (ID)** | **Texto de requisito** | **Fuente de Requisito** | **ID arquitectura del componente** | **ID código del Módulo/Unidad** | **ID de prueba del módulo/ unidad** | **Resultados de prueba del módulo/ unidad** | **ID de pruebas de integración** | **Resultados de prueba de integración del módulo / Unidad** | **ID pruebas del sistema** | |
| REQ-1 | Capacidad de trasladarse. | Documento plan de trabajo sección Requisitos. | Arquitectura secuencial | MOD-1 | PMD-1 |  | PI-1 |  | PS-1 | |
| REQ-2 | Capacidad de trasladarse en reversa. | Documento plan de trabajo sección Requisitos. | Arquitectura secuencial | MOD-1 | PMD-2 |  | PI-2 |  | PS-2 | |
| REQ-3 | Capacidad de hacer alto total. | Documento plan de trabajo sección Requisitos. | Arquitectura secuencial | MOD-1 | PMD-3 |  | PI-3 |  | PS-3 | |
| 4REQ-4 | Capacidad de girar 90 grados en dirección a la derecha. | Documento plan de trabajo sección Requisitos. | Arquitectura secuencial | MOD-1 | PMD-4 |  | PI-4 |  | PS-4 | |
| REQ-5 | Capacidad de girar 90 grados en dirección a la izquierda. | Documento plan de trabajo sección Requisitos. | Arquitectura secuencial | MOD-1 | PMD-5 |  | PI-5 |  | PS-5 | |
| REQ-6 | Capacidad de identificar entorno de desplazamiento. | Documento plan de trabajo sección Requisitos. | Arquitectura secuencial | MOD-2 | PMD-6 |  | PI-6 |  | PS-6 | |
| REQ-7 | Capacidad de evasión de obstáculos. | Documento plan de trabajo sección Requisitos. | Arquitectura secuencial | MOD-2 | PMD-7 |  | PI-7 |  | PS-7 | |
| REQ-8 | Capacidad de medición acústica del entorno. | Documento plan de trabajo sección Requisitos. | Arquitectura secuencial | MOD-2 | PMD-8 |  | PI-8 |  | PS-8 | |
| REQ-9 | Capacidad identificar un obstáculo potencial que intervengan con la trayectoria del dispositivo. | Documento plan de trabajo sección Requisitos. | Arquitectura secuencial | MOD-3 | PMD-9 |  | PI-9 |  | PS-9 | |
| REQ-10 | Capacidad de cuantificar la distancia entre el dispositivo y el obstáculo. | Documento plan de trabajo sección Requisitos. | Arquitectura secuencial | MOD-3 | PMD-10 |  | PI-10 |  | PS-10 | |
| REQ-11 | Capacidad de evadir obstáculos. | Documento plan de trabajo sección Requisitos. | Arquitectura secuencial | MOD-3 | PMD-11 |  | PI-11 |  | PS-11 | |

# Casos y Procedimientos de prueba del software

**1. Introducción**

(Descripción general del software y su uso dentro del alcance del negocio del cliente)

**2. Elementos necesarios para probar el código**

1. **Los casos de prueba incluyen:**

* Identificar los casos de prueba
* Elementos probados
* Especificaciones de entrada
* Especificaciones de salida
* Necesidades del entorno
* Requisitos de proceso especiales
* Dependencias de interfaz

1. **Los procedimientos de prueba incluyen:**

* Identificar: casos de prueba, descripción de pruebas y fecha de finalización de la prueba
* Identificar los problemas de implementación potenciales
* Identificar la persona que finalizará el procedimiento de prueba
* Identificar prerrequisitos
* Identificar los pasos de procedimiento incluyendo el número de pasos, la acción requerida por el tester y los resultados esperados.

**Estado (**verificado y en línea base)**:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Desarrollado por**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Fecha** (dd-mm-aaaa): \_\_\_\_\_\_\_\_\_\_

**Aprobado por**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Fecha** (dd-mm-aaaa): \_\_\_\_\_\_\_\_\_\_

# Registro de aceptación

1. **Registrar la recepción de los entregables**
2. **Identificar los elementos entregados**
3. **Registrar la verificación de los criterios de verificación del cliente definidos**
4. **Identificar los problemas abiertos (si aplica)**
5. **Firma de la recepción del cliente**

**Aceptación realizada por: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Fecha (dd-mm-aaaa): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Aceptación aprobada por: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Fecha (dd-mm-aaaa): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\**