



UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería  
Informática**

**PrimeBot  
Documentación Técnica**



Presentado por Mario Alonso Pulgar  
en Universidad de Burgos — 10 de junio  
de 2024

Tutor/es: Jesús Enrique Sierra García / César  
Repsa Pérez



---

# Índice general

---

<b>Índice general</b>	i
<b>Índice de figuras</b>	ii
<b>Índice de tablas</b>	iii
<b>Apéndice A Plan de Proyecto Software</b>	1
A.1. Introducción . . . . .	1
A.2. Planificación temporal . . . . .	2
A.3. Estudio de viabilidad . . . . .	6
<b>Apéndice B Especificación de Requisitos</b>	11
B.1. Introducción . . . . .	11
B.2. Objetivos generales . . . . .	12
B.3. Catálogo de requisitos . . . . .	12
B.4. Diagramas UML . . . . .	14
B.5. Especificación de requisitos . . . . .	17
<b>Apéndice C Especificación de diseño</b>	23
C.1. Introducción . . . . .	23
C.2. Diseño de datos . . . . .	23
C.3. Diseño procedimental . . . . .	24
C.4. Diseño arquitectónico . . . . .	26
<b>Apéndice D Documentación técnica de programación</b>	29
D.1. Introducción . . . . .	29
D.2. Estructura de directorios . . . . .	29

D.3. Manual del programador . . . . .	30
<b>Apéndice E Documentación de usuario</b>	<b>37</b>
E.1. Introducción . . . . .	37
E.2. Requisitos de usuarios . . . . .	37
E.3. Instalación . . . . .	37
E.4. Manual del usuario . . . . .	37
<b>Apéndice F Anexo de sostenibilización curricular</b>	<b>43</b>
F.1. Introducción . . . . .	43
F.2. Elección de materiales . . . . .	43
F.3. Optimización energética . . . . .	44
F.4. Reciclaje y Reutilización . . . . .	44
F.5. Gestión de Residuos . . . . .	44
F.6. Conclusión . . . . .	44
<b>Bibliografía</b>	<b>45</b>

---

# Índice de figuras

---

B.1. Diagrama de Flujo . . . . .	14
B.2. Diagrama de secuencia programa siguelineas. . . . .	15
B.3. Diagrama de secuencia programa de resolución de cuadrícula. .	16
B.4. Diagrama de secuencia programa de resolución de laberinto. .	16
D.1. Instalación de Arduino IDE . . . . .	30
D.2. Instalación de Visual Studio Code . . . . .	31
D.3. Instalación de Git . . . . .	32
D.4. Obtener repositorio . . . . .	32
D.5. Definicion de los pines de la placa en el Main . . . . .	33
D.6. Lugar donde añadir nuevas funcionalidades . . . . .	34
D.7. Seleccion de placa en Arduino IDE . . . . .	35
E.1. Posición Modo Calibración . . . . .	38
E.2. Posición Prueba Siguelíneas . . . . .	39
E.3. Posición Prueba Cuadrícula . . . . .	41
E.4. Posición Prueba Laberinto . . . . .	41

---

# Índice de tablas

---

A.1. Resumen horas dedicadas a cada apartado de PrimeBot . . . . .	5
A.2. Tabla de los costes de personal . . . . .	6
A.3. Tabla de los costes desarrollo web . . . . .	6
A.4. Tabla de los costes de hardware . . . . .	7
A.5. Tabla de costes varios . . . . .	7
A.6. Tabla de costes totales . . . . .	7
A.7. Tabla de ingresos potenciales . . . . .	8
B.1. CU-1 Calibración de Sensores. . . . .	18
B.2. CU-2 Seguimiento de Líneas. . . . .	19
B.3. CU-3 Navegación en Cuadrícula. . . . .	20
B.4. CU-4 Resolución de Laberintos . . . . .	21
B.5. CU-5 Ajuste de Parámetros PID. . . . .	22

## *Apéndice A*

---

# **Plan de Proyecto Software**

---

## **A.1. Introducción**

La planificación de un proyecto es un punto básico para estudiar su viabilidad. En esta fase se estima el trabajo, tiempo y dinero que supondrá llevar a cabo el proyecto. Hay que definir minuciosamente las fases y las partes del proyecto para obtener unos datos precisos.

La fase de planificación se puede dividir en:

- **Planificación temporal**
- **Estudio de viabilidad**

En la planificación temporal se elabora un calendario donde se estima el tiempo para la realización de cada una de las partes del proyecto. Se establecerá una fecha de comienzo y una de finalización estimada.

La segunda parte se centra en la viabilidad del proyecto teniendo en cuenta dos aspectos:

- **Viabilidad económica:** se estiman los costes y beneficios que puede suponer la realización del proyecto.
- **Viabilidad legal:** Se analizan las leyes y licencias que pueden afectar al desarrollo del proyecto.

## A.2. Planificación temporal

Al comenzar la planificación del proyecto se decidió utilizar la metodología ágil SCRUM para la gestión del proyecto. [6] No se ha seguido esta metodología al completo ya que el equipo estaba formado solo por mí y no por un grupo de personas, aunque en líneas generales sí que se ha seguido la filosofía.

- Estrategia de desarrollo incremental a través de sprints y revisiones.
- Duración media de los sprints de 2 semanas.
- Al final de cada sprint se hacía la subida de la parte de código correspondiente.
- Para cada sprint había una lista de tareas a realizar.
- Se realizaba una estimación del tiempo que iba a llevar a cabo cada tarea.

Para realizar el seguimiento de los sprints y las tareas del proyecto se ha utilizado la plataforma online YouTrack [2], que permite en un panel canvas organizar las tareas, los sprints, la duración de cada sprint e incluso realizar un Diagrama de Gantt con todas las tareas a realizar en el proyecto.

### Sprint 0 (08/01/2024 - 14/01/2024)

Para llevar a cabo este sprint se realizó una reunión inicial con el tutor Jesús Enrique Sierra García para comenzar con el proyecto.

En esta reunión se trataron los diferentes objetivos para el proyecto. Se trataron temas como planificación que se iba a llevar a cabo, metodología ágil a utilizar, características del proyecto y establecer las diferentes herramientas que se iban a utilizar.

Se estimaron 10 horas de trabajo y se invirtieron finalmente 7,5 horas completando todas las tareas.

### Sprint 1 (16/01/2024 - 16/02/2024)

Los objetivos de este sprint eran: configurar y encargar la fabricación de la PCB, seleccionar los componentes que iba a utilizar PrimeBot y seleccionar las pruebas que se van a realizar con esta versión de PrimeBot.

Se estimaron 100 horas de trabajo y se invirtieron finalmente 90 horas completando todas las tareas.

## Sprint 2 (17/02/2024 - 17/03/2024)

El principal objetivo de este sprint era realizar todo el montaje y todas las pruebas necesarias para asegurar que el funcionamiento de las conexiones y componentes de PrimeBot era el correcto.

Las tareas en las que se dividió este sprint fueron:

- Montar los componentes de PrimeBot.
- Realizar el Script de Prueba de motores.
- Realizar el Script de Prueba de encoders magnéticos.
- Realizar el Script de Prueba del selector de posición.
- Realizar el Script de Prueba de conectividad bluetooth.
- Realizar el Script de Prueba de los sensores de Distancia.
- Realizar el Script de Prueba de los sensores de línea.

Se estimaron 100 horas de trabajo y se invirtieron finalmente 80 horas completando todas las tareas

## Sprint 3 (18/03/2024 - 08/04/2024)

El objetivo de este sprint era realizar el desarrollo del programa que iba a llevar PrimeBot para la prueba de siguelíneas.

Las tareas en las que se dividió este sprint fueron:

- Implementación de un algoritmo PID [3]sencillo.
- Implementar la conectividad Bluetooth y los parámetros configurables dentro del PID.
- Implementar un algoritmo PID con más parámetros que el inicial.
- Realizar la optimización del algoritmo PID para obtener el máximo rendimiento posible.

Se estimaron 80 horas de trabajo y se invirtieron 90 completando todas las tareas

## Sprint 4 (09/04/2024 - 02/05/2024)

El objetivo de este sprint era realizar el desarrollo del programa que iba a llevar PrimeBot para la prueba de la resolución de cuadrícula.

Las tareas en las que se dividió este sprint fueron:

- Conectividad Bluetooth para definir estación de salida y de llegada.
- Implementar la representación gráfica de la cuadrícula.
- Implementar el algoritmo que calcula los movimientos desde el origen al destino.
- Implementar la detección de cruces.
- Implementar los giros que se realizan en la cuadrícula.
- Implementar la gestión de puntos bloqueados y caminos alternativos.

Este sprint fue el más problemático y el que más tiempo extra que no estaba planeado obligó a invertir. Se estimaron 100 horas de trabajo y se invirtieron 120 horas quedando por cumplir la gestión de puntos bloqueados y caminos alternativos para completar en el siguiente sprint.

### **Sprint 5 (03/05/2024 - 03/06/2024)**

El objetivo de este sprint era realizar el desarrollo del programa que iba a llevar PrimeBot para la prueba del laberinto.

- Conectividad Bluetooth para iniciar y parar el robot.
- Implementar la detección de paredes con los sensores de distancia.
- Implementar la detección de huecos con los sensores de distancia.
- Implementar los giros que se realizan dentro del recorrido.
- Implementar la detección del final del laberinto
- Implementar la impresión del recorrido por serial vía Bluetooth.

Se estimaron 100 horas de trabajo y se invirtieron 70 horas terminando todas las tareas, pero fue necesario invertir otras 50 horas para completar la tarea faltante del sprint anterior.

### **Sprint 6 (16/01/2024 - 03/06/2024)**

El objetivo de este sprint era realizar el desarrollo de la página web que iba a acompañar al proyecto.

- Compra del dominio.
- Creación de HomePage.
- Diseño de Landing Page PrimeBot.
- Implementar Landing Page PrimeBot.
- Incluir contenido relacionado con PrimeBot.
- Incluir vídeos multimedia de PrimeBot funcionando.

Se estimaron 90 horas de trabajo y se invirtieron 70 horas completando todas las tareas.

### Sprint 7 (03/06/2024 - 09/06/2024)

El objetivo de este sprint era terminar la memoria, los anexos y todo el material relativo a la entrega del proyecto.

- Terminar la memoria en LaTex.
- Terminar los anexos en LaTex.
- Realizar los dos videos necesarios para la entrega.
- Preparar material para el día de la defensa.

Se estimaron 40 horas de trabajo y se invirtieron 40 horas completando todas las tareas.

## Resumen

Finalmente en el proyecto de PrimeBot se han invertido aproximadamente las mismas horas de trabajo que se había estimado en un inicio pero no con la misma distribución en los sprints, siendo necesarias utilizar horas de un sprint en el sprint posterior.

En la siguiente tabla se pueden ver las horas que se han invertido en cada parte del desarrollo de PrimeBot:

Planificación del proyecto	27,5 horas
Documentación	70 horas
Características	400 horas
Desarrollo web	70 horas
Diseño de piezas	50 horas
Optimización de programas	100 horas
Total	717,5 horas

Tabla A.1: Resumen horas dedicadas a cada apartado de PrimeBot

### A.3. Estudio de viabilidad

#### Viabilidad económica

En este apartado se analizarán los costes y los posibles beneficios que se pueden obtener con este proyecto realizándolo en un entorno empresarial real.

#### Costes

Los costes del proyecto se pueden desglosar en varias categorías, vamos a detallar cada una de ellas:

##### Costes de Personal

El proyecto ha sido llevado a cabo por un equipo de desarrollo web y por un desarrollador empleado a tiempo completo durante cinco meses.

Del desarrollador se considera el siguiente salario:

Concepto	Coste
Salario Neto	1281€
IRPF	1507,15€
Seguridad Social	642,85€
Salario Bruto	2150€

Tabla A.2: Tabla de los costes de personal

El total por tanto durante 5 meses en salario bruto será de 10.750€ El coste del desarrollo web se define como un presupuesto cerrado que tiene un coste total de 500€ IVA Incluido y se divide en diseño y desarrollo:

Concepto	Coste
Diseño Web	200€
Desarrollo	300€

Tabla A.3: Tabla de los costes desarrollo web

#### Costes de Hardware

En este apartado se revisan todos los costes en dispositivos y componentes de Primebot que han sido necesarios para el desarrollo del proyecto.

Por tanto el total de la construcción de PrimeBot a nivel de hardware sería de 141,64 €

Concepto	Coste
Fabricación PCB	29 €
Arduino Nano [1]	19 €
QTR8A [5]	12.04 €
OPT3101 [4]	45.38 €
Bluetooth	4.54 €
Driver de Motores	5.99€
Batería	4.50 €
Motores N20	10.90 €
Encoders Magnéticos	10.29 €

Tabla A.4: Tabla de los costes de hardware

### Costes Varios

En este apartado se revisan el resto de costes del proyecto.

Concepto	Coste
Hosting	6€ / mes
Dominio	14,90€ / año

Tabla A.5: Tabla de costes varios

En el total de los costes hay que incluir que el hosting de momento se ha mantenido durante 5 meses, además del dominio haría un total de 44,90€.

### Costes Totales

A continuación se adjunta una tabla con los costes totales de cada sección de PrimeBot en caso de llevarse a cabo en un entorno industrial.

Concepto	Coste
Costes de Personal	10.750€
Costes desarrollo web	500€
Costes mantenimiento web	44,90€
Costes hardware	141,64€
Total	11.436,64€

Tabla A.6: Tabla de costes totales

### Ingresos

PrimeBot es un proyecto sólido y con un gran rendimiento que se podría emplear en numerosas ediciones del ASTI Robotics Challenge para pelear

por los primeros premios, además también se podría comercializar en kit para que los equipos que quieran participar adquieran el kit a nivel de hardware y desarrollen sus propios programas.

Se considerará que en 5 años, al menos en uno se consiga el primer premio del ASTI Robotics Challenge y que un kit de Hardware se venderá al doble del precio de coste de los materiales.

Concepto	Cantidad
Ganador ASTI Robotics Challenge	3.000€
Venta kit educativo	299€

Tabla A.7: Tabla de ingresos potenciales

## Viabilidad legal

En esta sección se discutirán los temas relacionados con las licencias que puedan estar relacionadas con cualquiera de los ámbitos de este proyecto.

### Software

Primero hay que elegir cuál sería la licencia más conveniente para el proyecto de PrimeBot. Hay que elegir que derechos queremos proporcionar a los usuarios y cuáles no.

En mi caso quiero que PrimeBot sea un proyecto de software libre y código abierto por lo que seleccionaré la licencia más permisible posible con los usuarios.

Dentro de las licencias, la más permisible es la licencia MIT.

MIT Permite a los usuarios realizar uso, copia, modificación, fusión, publicación, sublicencia y venta de copias del software.

MIT Es compatible con muchas otras licencias de software libre y de código abierto.

### Documentacion

Para la documentación he seleccionado utilizar una licencia Creative Commons ya que están enfocadas a licenciar este tipo de contenidos, en concreto he seleccionado Creative Commons Attribution 4.0 International (CC-BY-4.0) que establece lo siguiente:

**Imágenes y vídeos**

En la documentación del proyecto no se han utilizado imágenes de terceros por lo que tanto las imágenes como los vídeos son propias del proyecto y cuentan con la misma licencia que la documentación (CC-BY-4.0)



## *Apéndice B*

---

# Especificación de Requisitos

---

## B.1. Introducción

En esta sección se recoge la especificación de requisitos que define el comportamiento del sistema desarrollado. Esta sección tiene como objetivo documentar correctamente la forma en la que se ha desarrollado PrimeBot.

Se han seguido las recomendaciones del estándar IEEE 830-1998 que manifiesta que una buena especificación de requisitos de software debe ser:

- **Completa:** todos los requerimientos deben estar reflejados en ella y todas las referencias deben estar definidas.
- **Consistente:** debe ser coherente con los propios requerimientos y también con otros documentos de especificación.
- **Inequívoca:** la redacción debe ser clara de modo que no se pueda mal interpretar.
- **Correcta:** el software debe cumplir con los requisitos de la especificación.
- **Trazable:** se refiere a la posibilidad de verificar la historia, ubicación o aplicación de un ítem a través de su identificación almacenada y documentada.
- **Priorizable:** los requisitos deben poder organizarse jerárquicamente según su relevancia para el negocio y clasificándolos en esenciales, condicionales y opcionales.
- **Modificable:** aunque todo requerimiento es modificable, se refiere a que debe ser fácilmente modificable.
- **Verificable:** debe existir un método finito sin costo para poder probarlo.

## B.2. Objetivos generales

El proyecto persigue los siguientes objetivos generales:

- Desarrollar una algoritmo PID eficiente y de alto rendimiento para seguir línea negra.
- Implementar un algoritmo para la resolución de la prueba de cuadrícula.
- Implementar un algoritmo eficiente para la resolución de laberintos.
- Establecer protocolos de comunicación Bluetooth a tiempo real para aumentar el rendimiento de PrimeBot.
- Permitir ejecutar varios programas a la vez sin necesidad de un PC.

## B.3. Catálogo de requisitos

A continuación, se enumeran los requisitos específicos derivados de los objetivos generales del proyecto.

### Requisitos funcionales

- **RF-1 Seguimiento de línea:** PrimeBot debe ser capaz de seguir líneas negras sobre un fondo blanco utilizando los sensores QTR8A.
  - **RF-1.1 Detección de Líneas:** Los sensores QTR8A detectan el contraste entre la línea negra y el fondo blanco.
  - **RF-1.2 Ajuste de Dirección:** El algoritmo PID ajusta la dirección del robot basándose en la información de los sensores.
  - **RF-1.3 Calibración inicial:** antes de comenzar las pruebas, PrimeBot debe calibrar sus sensores QTR8A para seguir correctamente la línea negra.
- **RF-2 Navegacion en cuadricula :** PrimeBot debe moverse de una estación de origen a una estación de destino en una cuadrícula predefinida.
  - **RF-2.1 Detección de Cruces:** El robot detecta los cruces en la cuadrícula utilizando los sensores QTR8A.
  - **RF-2.2 Cálculo de Rutas:** El robot calcula la ruta más eficiente desde la estación de origen a la de destino.
  - **RF-2.3 Evitar puntos bloqueados:** PrimeBot debe ser capaz de calcular rutas alternativas ante puntos de la cuadrícula bloqueados.

- **RF-3 Resolución de laberintos:** PrimeBot debe navegar y resolver laberintos utilizando algoritmos de búsqueda y sensores OPT3101.
  - **RF-3.1 Detección de Paredes:** Los sensores OPT3101 detectan la presencia de paredes en el laberinto.
  - **RF-3.2 Detección de huecos:** PrimeBot debe ser capaz de detectar los huecos que hay en el laberinto para navegar sobre él.

## Requisitos no funcionales

- **RNF-1 Rendimiento:** PrimeBot debe operar con un tiempo de respuesta mínimo y eficiente.
- **RNF-2 Confiabilidad:** PrimeBot debe ser confiable y operar sin fallos durante largos periodos.
- **RNF-3 Usabilidad:** PrimeBot debe ser fácil de usar e interactuar.
  - **RNF-3.1 Interfaz de Usuario:** El robot debe ser controlable mediante una interfaz de usuario sencilla y los ajustes vía Bluetooth deben ser intuitivos.
- **RNF-4 Mantenibilidad:** El código y hardware de PrimeBot deben ser fáciles de mantener y actualizar.
  - **RNF-4.1 Actualización de Software y Hardware:** Las actualizaciones deben poder realizarse sin necesidad de una reestructuración completa.
- **RNF-5 Portabilidad:** PrimeBot debe ser portátil y fácil de transportar.
  - **RNF-5.1 Peso y Dimensiones:** El robot debe ser ligero y compacto, con un peso inferior a 1 kg y dimensiones manejables.
- **RNF-6 Escalabilidad:** PrimeBot debe permitir futuras expansiones y mejoras.
  - **RNF-6.1 Adición de Nuevos Componentes:** La arquitectura debe permitir la adición de nuevos sensores y módulos sin necesidad de un rediseño significativo.
- **RNF-7 Seguridad:** PrimeBot debe operar de manera segura tanto para los usuarios como para el entorno.

## B.4. Diagramas UML

### Diagrama de flujo

A continuación se muestra un diagrama de flujo de la selección de programa de PrimeBot (Figura B.1). Esta selección se realiza con el selector DIP y el pulsador que hay en la parte trasera del PCB de PrimeBot

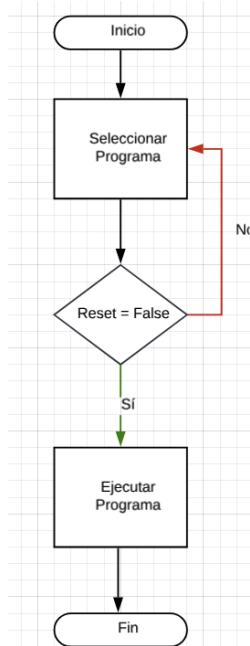


Figura B.1: Diagrama de Flujo.

## Diagramas de secuencias

A continuación se colocan los diagramas de secuencias de las diferentes pruebas que realiza PrimeBot.

### Prueba de siguelíneas

La prueba de siguelíneas se realiza de forma ininterrumpida hasta que se vuelve a pulsar el botón con otro programa seleccionado (Figura B.2).

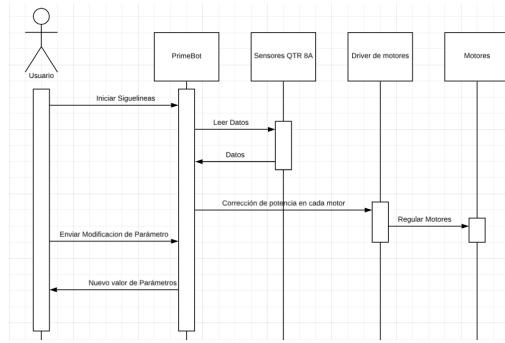


Figura B.2: Diagrama de secuencia programa siguelíneas.

### Prueba de cuadrícula

La prueba de la cuadrícula es la más compleja y se realiza a partir de unos argumentos que el usuario aporta a PrimeBot.

En el diagrama de secuencias se puede ver la ejecución completa del programa (Figura B.3).

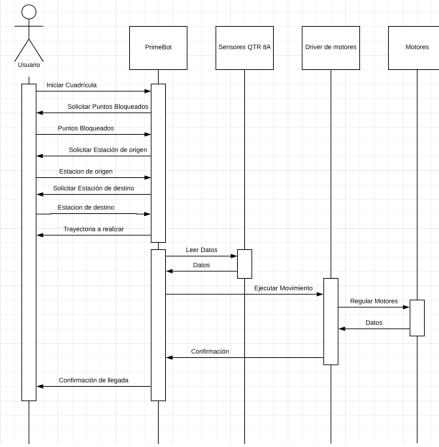


Figura B.3: Diagrama de secuencia programa de resolución de cuadrágula.

### Prueba de laberinto

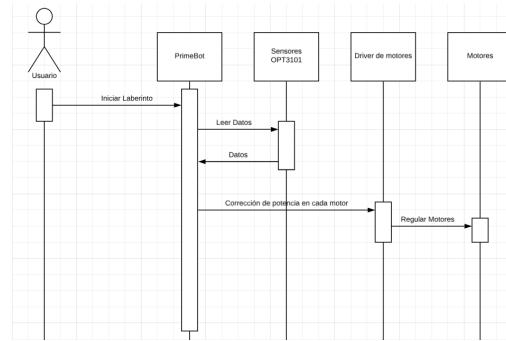


Figura B.4: Diagrama de secuencia programa de resolución de laberinto.

## **B.5. Especificación de requisitos**

En esta sección se desarrollará cada caso de uso del sistema.

<b>CU-1</b>	<b>Calibración de Sensores</b>
<b>Versión</b>	1.0
<b>Autor</b>	Mario Alonso Pulgar
<b>Requisitos asociados</b>	RF-1.3
<b>Descripción</b>	PrimeBot calibra sus sensores QTR para una mejor precisión en el seguimiento de líneas.
<b>Precondición</b>	PrimeBot está encendido y los sensores QTR están conectados.
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1. El usuario selecciona la opción de calibración en el dip-switch.</li> <li>2. PrimeBot enciende los LEDs de calibración.</li> <li>3. PrimeBot calibra los sensores durante 400 iteraciones.</li> <li>4. PrimeBot apaga los LEDs de calibración.</li> </ol>
<b>Postcondición</b>	Los sensores QTR están calibrados y listos para su uso.
<b>Excepciones</b>	Si la calibración falla, PrimeBot avisa al usuario y detiene la operación.
<b>Importancia</b>	Alta

Tabla B.1: CU-1 Calibración de Sensores.

<b>CU-2</b>	<b>Seguimiento de linea</b>
<b>Versión</b>	1.0
<b>Autor</b>	Mario Alonso Pulgar
<b>Requisitos asociados</b>	RF-1, RF-1.1, RF-1.2
<b>Descripción</b>	PrimeBot sigue una línea negra sobre un fondo blanco utilizando los sensores QTR8A.
<b>Precondición</b>	PrimeBot está encendido, los sensores QTR8A están calibrados y está seleccionado el programa 0001
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1. El usuario coloca PrimeBot al inicio de una línea negra.</li> <li>2. PrimeBot detecta la línea utilizando los sensores QTR8A.</li> <li>3. PrimeBot ajusta su dirección mediante el algoritmo PID.</li> <li>4. PrimeBot sigue la línea hasta el final del recorrido.</li> </ol>
<b>Postcondición</b>	PrimeBot sigue la línea negra sin salirse del camino.
<b>Excepciones</b>	Si PrimeBot pierde la línea, recalibra los sensores y vuelve a buscar la línea.
<b>Importancia</b>	Alta

Tabla B.2: CU-2 Seguimiento de Líneas.

<b>CU-3</b>	<b>Navegación en Cuadrícula</b>
<b>Versión</b>	1.0
<b>Autor</b>	Mario Alonso Pulgar
<b>Requisitos asociados</b>	RF-2, RF-2.1, RF-2.2, RF-2.3
<b>Descripción</b>	PrimeBot se mueve de una estación de origen a una estación de destino en una cuadrícula predefinida.
<b>Precondición</b>	PrimeBot está encendido, los sensores QTR8A están calibrados y está seleccionado el programa 0010
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1. El usuario indica las estaciones de origen y destino a través de Bluetooth.</li> <li>2. PrimeBot detecta los cruces en la cuadrícula utilizando los sensores QTR8A.</li> <li>3. PrimeBot calcula la ruta más eficiente.</li> <li>4. PrimeBot se mueve a través de la cuadrícula hasta llegar a la estación de destino.</li> </ol>
<b>Postcondición</b>	PrimeBot llega a la estación de destino.
<b>Excepciones</b>	Si PrimeBot pierde la línea, recalibra los sensores y vuelve a buscar la línea.
<b>Importancia</b>	Alta

Tabla B.3: CU-3 Navegación en Cuadrícula.

<b>CU-4</b>	<b>Resolución de Laberintos</b>
<b>Versión</b>	1.0
<b>Autor</b>	Mario Alonso Pulgar
<b>Requisitos asociados</b>	RF-3, RF-3.1, RF-3.2
<b>Descripción</b>	PrimeBot navega y resuelve laberintos utilizando algoritmos de búsqueda y sensores OPT3101.
<b>Precondición</b>	PrimeBot está encendido, los sensores OPT3101 están calibrados y está seleccionado el programa 0011
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1. El usuario coloca PrimeBot en la entrada del laberinto.</li> <li>2. PrimeBot utiliza los sensores OPT3101 para detectar paredes.</li> <li>3. PrimeBot emplea un algoritmo de búsqueda para planificar su ruta.</li> <li>4. PrimeBot navega a través del laberinto.</li> <li>5. PrimeBot llega a la salida del laberinto.</li> </ol>
<b>Postcondición</b>	PrimeBot encuentra la salida del laberinto.
<b>Excepciones</b>	Si PrimeBot encuentra un obstáculo inesperado, recalcula la ruta y continúa navegando.
<b>Importancia</b>	Alta

Tabla B.4: CU-4 Resolución de Laberintos

<b>CU-5</b>	<b>Ajuste de Parámetros PID</b>
<b>Versión</b>	1.0
<b>Autor</b>	Mario Alonso Pulgar
<b>Requisitos asociados</b>	RNF-3
<b>Descripción</b>	El usuario ajusta los parámetros del controlador PID para optimizar el rendimiento de PrimeBot.
<b>Precondición</b>	PrimeBot está encendido y conectado al sistema de control.
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1. El usuario selecciona el parámetro PID a ajustar (<math>K_p</math>, <math>K_i</math>, <math>K_d</math>, <math>K_v</math>).</li> <li>2. El usuario incrementa o decrementa el valor del parámetro seleccionado.</li> <li>3. PrimeBot ajusta su comportamiento en función del nuevo valor del parámetro PID.</li> </ol>
<b>Postcondición</b>	Los parámetros PID están ajustados para un rendimiento óptimo.
<b>Excepciones</b>	Si el ajuste no es satisfactorio, el usuario puede revertir los cambios.
<b>Importancia</b>	Media

Tabla B.5: CU-5 Ajuste de Parámetros PID.

## *Apéndice C*

---

# Especificación de diseño

---

## C.1. Introducción

El PrimeBot es un robot autónomo diseñado para participar en el ASTI Robotics Challenge, una competición de robótica que fomenta la innovación y la educación en tecnología. Este documento proporciona una descripción detallada del diseño del PrimeBot, abarcando desde la configuración del hardware hasta los algoritmos de control y navegación implementados. El objetivo es ofrecer una comprensión completa de la arquitectura, el flujo de datos y los procedimientos utilizados para asegurar el correcto funcionamiento del robot en diversas pruebas de la competición.

## C.2. Diseño de datos

El diseño de datos del PrimeBot define las estructuras de datos y las variables necesarias para su operación eficiente. Estas incluyen:

### **Variables de Sensores:**

- **sensorValues:** Almacena los valores de los sensores QTR utilizados para el seguimiento de líneas.
- **amplitudes:** Guarda las amplitudes medidas por los sensores de distancia OPT3101.
- **distances:** Guarda las distancias medidas por los sensores OPT3101 en milímetros.

### **Constantes de Control PID:**

- float Kp = 1.2: Constante proporcional del controlador PID.
- float Ki = 0.02: Constante integral del controlador PID.
- float Kd = 0.2: Constante derivativa del controlador PID.
- float Kv = 2.60: Constante de velocidad, optimiza el rendimiento del robot en tramos rectos.

### **Estado del Robot:**

- int posX, posY: Coordenadas actuales del robot en la cuadrícula.
- int destinoX, destinoY: Coordenadas de destino del robot.
- int dirección: Dirección actual del robot (0 = x+, 1 = y-, 2 = x-, 3 = y+).

### **Configuraciones de Hardware:**

- Pines para el control de los motores, encoders, sensores y LEDs.
- Definiciones para la configuración del módulo Bluetooth y la interfaz serial.

### **Datos de Navegación:**

- Arrays para las coordenadas de estaciones de origen y destino.
- Estructuras de datos para los nodos utilizados en el algoritmo A.

## **C.3. Diseño procedimental**

El diseño procedimental del PrimeBot se organiza en funciones que controlan los diversos aspectos del robot, desde la configuración inicial hasta la ejecución de algoritmos complejos. Estas funciones se dividen en varias categorías:

### Configuración y Control de Motores:

- void motorSetup(): Configura los pines de los motores para control de dirección y velocidad.
- void setLeftMotorPWM(int pwm): Establece la velocidad y dirección del motor izquierdo.
- void setRightMotorPWM(int pwm): Establece la velocidad y dirección del motor derecho.
- void setMotorPWM(int left, int right): Establece la velocidad de ambos motores para movimiento coordinado.

### Calibración y Lectura de Sensores:

- void calibrarSensores(): Calibra los sensores QTR para obtener lecturas precisas de la línea.
- void seguirLinea(): Implementa el algoritmo PID para el seguimiento de línea, ajustando la velocidad de los motores en tiempo real.
- bool cruceDetectado(): Detecta la presencia de cruces en la línea mediante la lectura de los sensores.

### Navegación y Planificación de Rutas:

- void calcularTrayectoria(): Utiliza el algoritmo A para calcular la trayectoria óptima desde el origen hasta el destino en la cuadrícula.
- void ejecutarTrayectoria(): Ejecuta la ruta calculada, siguiendo las instrucciones de movimiento (recto, derecha, izquierda).

### Interacción con el Usuario:

- void pedirBloqueos(): Sigue al usuario que introduzca los puntos bloqueados en la cuadrícula.
- void pedirEstaciones(): Sigue al usuario las coordenadas de las estaciones de origen y destino al usuario.
- void mostrarRuta(): Muestra la ruta calculada en la cuadrícula a través de la interfaz serial.

## Control Principal y Ajustes:

- void controlador(): Permite ajustar los parámetros PID y otros configurables en tiempo real mediante comandos Bluetooth.
- void primeBotAction(int function): Ejecuta diferentes acciones del robot según la posición del DIP switch.
- void runRobot(): Controla el flujo principal del robot, determinando la acción a tomar basado en el valor del DIP switch.
- void setup(): Configura el robot al inicio, incluyendo la inicialización de la comunicación y la configuración de hardware.
- void loop(): Bucle principal que mantiene el robot en funcionamiento, verificando continuamente el estado del DIP switch para iniciar las acciones correspondientes.

## C.4. Diseño arquitectónico

El diseño arquitectónico del PrimeBot está estructurado en varias capas que aseguran la modularidad y la fácil integración de los componentes. Estas capas incluyen:

### Capa de Hardware:

- Microcontrolador: El Arduino Nano actúa como el cerebro del sistema, controlando todos los componentes y ejecutando los algoritmos.
- Sensores: Sensores QTR8A para el seguimiento de líneas y sensores OPT3101 para la detección de obstáculos.
- Motores y Encoders: Motores N20 controlados por un driver TB6612FNG, con encoders magnéticos para retroalimentación de posición y velocidad.
- Conectividad Bluetooth: Módulo Bluetooth para la comunicación en tiempo real, permitiendo ajustes y monitoreo sin necesidad de conexiones físicas.

**Capa de Control:**

- Control PID: Algoritmo que ajusta la velocidad de los motores basado en las lecturas de los sensores para mantener el robot en la línea.
- Algoritmo A: Algoritmo de planificación de rutas para navegar en la cuadrícula, optimizando el camino desde el origen hasta el destino.

**Capa de Comunicación:**

- Bluetooth: Proporciona una interfaz para ajustes en tiempo real y monitoreo del estado del robot.
- Serial: Utiliza la interfaz serial para depuración y control manual, permitiendo la visualización de datos y la recepción de comandos.

**Capa de Aplicación:**

- Interfaz de Usuario: Permite la configuración y control del robot mediante comandos enviados a través de Bluetooth y la interfaz serial.
- Ejecución de Pruebas: Implementa funcionalidades específicas para las pruebas del ASTI Robotics Challenge, como seguimiento de líneas, navegación en cuadrículas y resolución de laberintos.

La arquitectura modular del PrimeBot facilita la integración y actualización de componentes, permitiendo adaptaciones rápidas a las diferentes pruebas y escenarios de la competición.



## *Apéndice D*

---

# **Documentación técnica de programación**

---

## **D.1. Introducción**

En este anexo se describe toda la documentación técnica de programación, incluyendo la instalación del entorno de desarrollo, la estructura de archivos que hay en el repositorio de github y la integración de nuevas funcionalidades al robot.

## **D.2. Estructura de directorios**

El repositorio del proyecto se distribuye de la siguiente manera:

- Sprint 1: Esta carpeta contiene los códigos pertenecientes a las pruebas de funcionamiento de los componentes.
- Sprint 2: Esta carpeta contiene los códigos correspondientes a la prueba siguelíneas.
- Sprint 3: Esta carpeta contiene los códigos correspondientes a la prueba de la cuadrícula.
- Sprint 4: Esta carpeta contiene los códigos correspondientes con la prueba del laberinto.
- Sprint 5: Esta carpeta contiene el archivo main y los archivos relacionados con el desarrollo de la web.
- Docs: Esta carpeta contiene los archivos relacionados a la documentación como memoria y anexos.

## D.3. Manual del programador

El siguiente manual tiene como objetivo servir de referencia a futuros programadores que quieran incorporar nuevas funcionalidades a PrimeBot siguiendo la estructura ya existente. En este manual se explica como montar el entorno de desarrollo, obtener el código fuente del proyecto, compilarlo y ejecutarlo.

### Entorno de desarrollo

Para trabajar con el proyecto se necesita tener instalados los siguientes programas y dependencias:

- Arduino IDE.
- Visual Studio Code
- Git.

A continuación se indica como instalar y configurar correctamente cada uno de ellos.

#### Arduino IDE

El principal lenguaje de programación utilizado en el proyecto de PrimeBot es Arduino (una variación de c++) y para llevar a cabo este desarrollo disponer del IDE oficial actualizado es obligatorio.

Se puede obtener para cualquier sistema directamente en la web oficial de arduino en el siguiente enlace: [Arduino IDE](#)



Figura D.1: Instalación de Arduino IDE

### Visual Studio Code

Visual Studio Code es el IDE empleado para el desarrollo y la implementación de la página web de PrimeBot.

Se puede obtener desde el siguiente enlace: [Visual Studio Code](#)

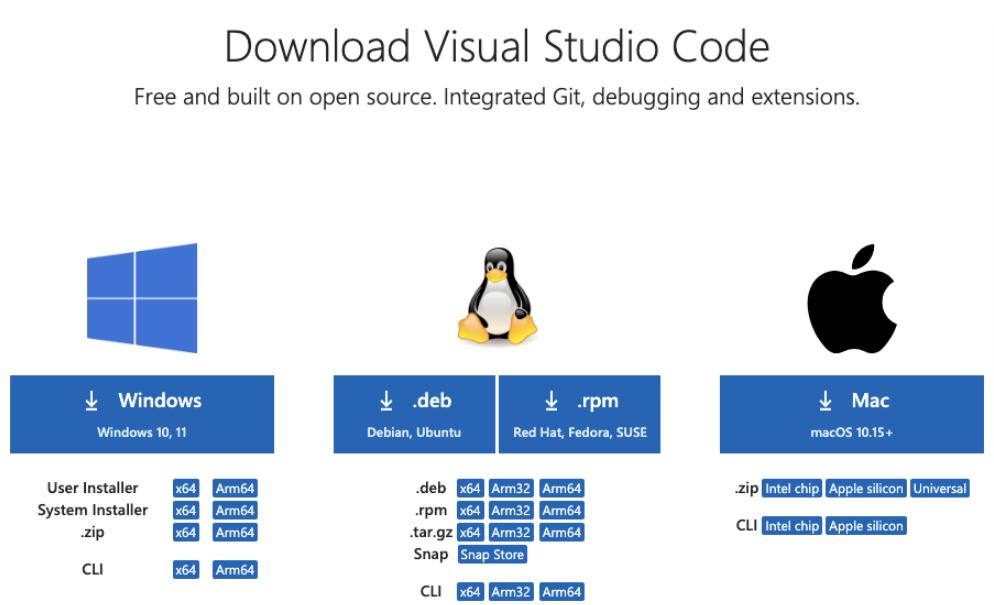


Figura D.2: Instalación de Visual Studio Code

### git

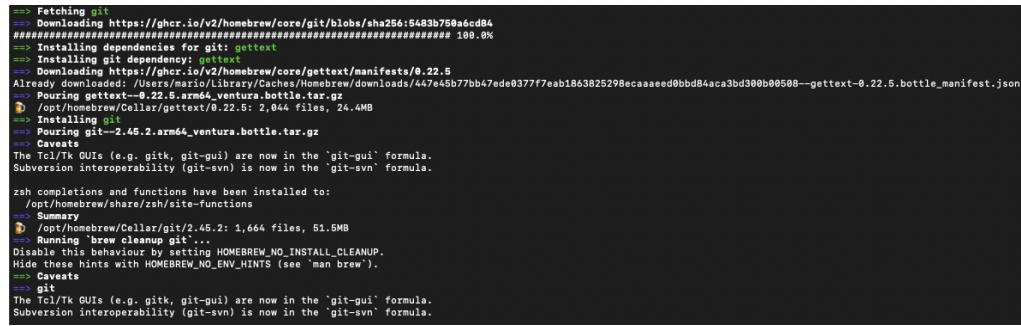
Para hacer uso del repositorio se necesita tener instalado el gestor de versiones Git. Este programa permite clonar el repositorio, movernos dentro de él... etc.

Se puede obtener desde: [GIT](#)

Cuando esté instalado, trabajaremos con Git Bash.

### Obtención del código fuente

Para el desarrollo de los algoritmos empleados en PrimeBot se ha utilizado el repositorio Git hospedado en GitHub, lo primero es obtener una copia de la siguiente manera:



```

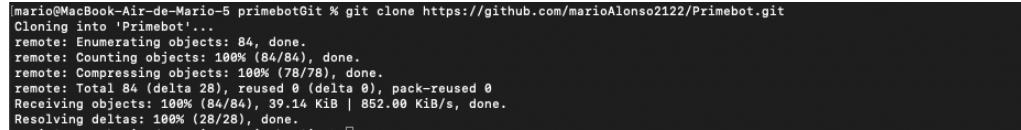
=> Fetching git
==== Downloading https://ghcr.io/v2/homebrew/core/git/blobs/sha256:5483b78a6cd84
=> Installing dependencies for git: gettext
=> Installing dependency gettext
=> Downloading https://ghcr.io/v2/homebrew/core/gettext/manifests/0.22.8
Already downloaded: /Users/mario/Library/Caches/Homebrew/downloads/447e45b77bb47ede0377f7eab1863825298ecaaaaed0bbd84aca3bd300b00508--gettext-0.22.8.bottle_manifest.json
=> Pouring gettext--0.22.8.arm64_ventura.bottle.tar.gz
=> Installing git
=> Pouring git--2.45.2.arm64_ventura.bottle.tar.gz
=> Caveats
The Tcl/Tk GUIs (e.g. gitk, git-gui) are now in the 'git-gui' formula.
Subversion interoperability (git-svn) is now in the 'git-svn' formula.

zsh completions and functions have been installed to:
/opt/homebrew/share/zsh/site-functions
=> Summary
=> /opt/homebrew/Cellar/git/2.45.2: 1,664 files, 51.BMB
=> Removing unused 'cleanup' git hints.
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
=> Caveats
=> git
The Tcl/Tk GUIs (e.g. gitk, git-gui) are now in the 'git-gui' formula.
Subversion interoperability (git-svn) is now in the 'git-svn' formula.

```

Figura D.3: Instalación de Git

1. Abrir la terminal Git Bash.
2. Desplazarse al directorio donde se desee copiar el repositorio (utilizando el comando `cd`).
3. Introducir el siguiente comando:  
`git clone https://github.com/marioAlonso2122/Primebot.git`
4. Se iniciará la descarga del repositorio, cuando finalice se dispondrá de una copia completa de este.



```

mario@MacBook-Air-de-Mario-5 primebotGit % git clone https://github.com/marioAlonso2122/Primebot.git
Cloning into 'Primebot'...
remote: Enumerating objects: 84, done.
remote: Counting objects: 100% (84/84), done.
remote: Compressing objects: 100% (78/78), done.
remote: Total 84 (delta 28), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (84/84), 39.14 KiB | 852.00 KiB/s, done.
Resolving deltas: 100% (28/28), done.

```

Figura D.4: Obtener repositorio

## Añadir nuevas funcionalidades

Una vez dispongamos de una copia del repositorio de PrimeBot en nuestro equipo ya podremos realizar la incorporación de nuevas funcionalidades al código de la siguiente manera:

1. Abrir el archivo `main.ino` con el editor de código Arduino.
2. Debemos dejar las definiciones iniciales intactas ya que son los pines de las conexiones de la PCB que deben estar estáticas o PrimeBot podría dejar de funcionar.
3. Tenemos que localizar la función llamada `primeBotAction`.

4. Dentro de esta función disponemos de un recurso switch-case donde podremos incorporar nuevas funcionalidades.
5. En el main.ino que se descargar del repositorio encontramos los 4 primeros case ya ocupados, cada valor de case corresponde a uno de los valores obtenidos a través del switch dip integrado en el PCB.
6. Dentro del case podemos añadir nuevos valores donde incorporar nuevas funcionalidades a PrimeBot. Se recomienda dejar un valor de case para cada funcionalidad nueva.
7. Hay que incorporar dentro de la nueva funcionalidad un bucle while(true) que solo salga cuando se pulse el botón para poder ejecutar constantemente la funcionalidad hasta que se pulse de nuevo el botón de cambio de función.

```
// Definición de pines
const int ENCODER_LEFT_CLK = 2;
const int ENCODER_RIGHT_CLK = 3;
const int ENCODER_LEFT_B = 4;
const int ENCODER_RIGHT_B = 5;
const int MOTOR_LEFT_DIR = 7;
const int MOTOR_RIGHT_DIR = 8;
const int MOTOR_LEFT_PWM = 9;
const int MOTOR_RIGHT_PWM = 10;
const int LED_RIGHT = 6;
const int LED_LEFT = 11;
const int Emitter = 12;
const int SENSOR_RIGHT_MARK = A0;
const int SENSOR_1 = A1;
const int SENSOR_2 = A2;
const int SENSOR_3 = A3;
const int SENSOR_4 = A4;
const int SENSOR_LEFT_MARK = A5;
const int FUNCTION_PIN = A6;
const int BATTERY_VOLTS = A7;
const int Serial1_RX = 12;
const int Serial1_TX = 13;
```

Figura D.5: Definicion de los pines de la placa en el Main

```

void primeBotAction(int function) {
    switch (function) {
        case 0:
            calibrarSensores();
            Serial.println("¡Bienvenido a PrimeBot!");
            Serial.println("Este Proyecto ha sido desarrollado por: Mario Alonso Pulgar.");
            Serial.println("Selecciona la posición en el dip-switch y pulsa el botón para comenzar.");
            Serial1.println("¡Bienvenido a PrimeBot!");
            Serial1.println("Este Proyecto ha sido desarrollado por: Mario Alonso Pulgar.");
            Serial1.println("Selecciona la posición en el dip-switch y pulsa el botón para comenzar.");

            break;
        case 1:
            Serial.println("Comenzando Siguelineas de PrimeBot");
            Serial1.println("Comenzando Siguelineas de PrimeBot");
            while (true) {
                seguirLinea();
                controlador();
                if (getFunctionSwitch() == 16) {
                    setMotorPWM(0, 0);
                    break;
                }
            }
            break;
        case 2:
            Serial.println("Prueba de Cuadricula con PrimeBot");
            Serial1.println("Prueba de Cuadricula con PrimeBot");
            baseSpeed = 50;
            while (true) {
                pedirBloqueos();
                pedirEstaciones();
                calcularTrayectoria();
                if (getFunctionSwitch() == 16) {
                    setMotorPWM(0, 0);
                    break;
                }
            }
        case 3:
            Serial.println("Prueba de Laberinto con PrimeBot");
            Serial1.println("Prueba de Laberinto con PrimeBot");
            while (true) {
                resolucionlaberinto();
                if (getFunctionSwitch() == 16) {
                    setMotorPWM(0, 0);
                    break;
                }
            }
        default:
            setMotorPWM(0, 0);
            Serial.println("Función no implementada.");
            Serial1.println("Función no implementada.");
            break;
    }
}

```

Figura D.6: Lugar donde añadir nuevas funcionalidades

### Ejecutar código

Al estar trabajando con Arduino, el código se recomienda ejecutarlo siempre en un dispositivo real, para ello:

1. Conectamos la placa Arduino al equipo vía USB
2. Dentro de las opciones del IDE de arduino en el apartado Herramientas, debemos seleccionar la placa donde vamos a cargar el código y el puerto al que está conectada esa placa.
3. Pulsaremos en la opción subir para compilar y enviar el código a la placa arduino que vayamos a emplear.

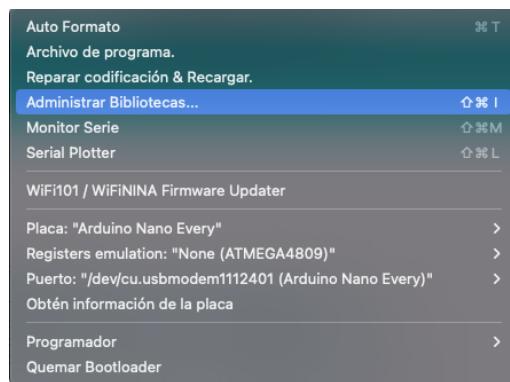


Figura D.7: Seleccion de placa en Arduino IDE



## *Apéndice E*

---

# **Documentación de usuario**

---

## **E.1. Introducción**

En este manual se detallan los requerimientos para utilizar todas las funcionalidades de PrimeBot y poder aprovechar cada uno de los parámetros que hay en la programación del robot.

## **E.2. Requisitos de usuarios**

Los requisitos mínimos para poder utilizar el proyecto son:

- Disponer de un kit de PrimeBot con main.ino cargado
- Disponer de un móvil Android o un equipo con conexión Bluetooth
- Disponer de una aplicación como Monitor Serial

## **E.3. Instalación**

Si el usuario ya dispone del programa main.ino cargado en la placa de Arduino correspondiente y una aplicación de monitor serial instalada en el dispositivo a conectar, no deberá realizar ninguna instalación adicional.

## **E.4. Manual del usuario**

En esta sección se describe el uso de las diferentes funcionalidades de PrimeBot

## Conexión Bluetooth

Antes de ejecutar cualquier programa, debemos encender PrimeBot y realizar la conexión Bluetooth. Cuando esta conexión esté hecha, el LED rojo del módulo Bluetooth permanecerá fijo.

## Modo Calibración

El modo calibración corresponde a la posición 0000 del dip switch que está integrado en la placa de control.

Cuando se quiere utilizar PrimeBot, este es el primer modo que debemos ejecutar ya que en este modo se realiza la calibración de los sensores siguelíneas QTR y se establece la conexión Bluetooth con el robot. El orden correcto de ejecución será:

1. Seleccionamos la posición 0000 en el switch.
2. Se pulsa el botón de selección de programa.
3. Mientras la luz LED de la placa arduino esté encendida de forma fija, debemos hacer pasadas suaves sobre la línea negra que debe seguir, de esta forma estaremos calibrando los sensores.
4. Cuando acabe la calibración de los sensores, los motores se moverán durante un segundo indicando que la calibración ha terminado.
5. Ahora podemos realizar la conexión Bluetooth con PrimeBot desde nuestro dispositivo.



Figura E.1: Posición Modo Calibración

## Modo Siguelíneas

Esta funcionalidad corresponde a la posición 0001 del dip switch integrado en la placa de control. El modo siguelíneas consiste en el seguimiento de una línea negra sobre un fondo blanco de cara a realizar las vueltas al circuito cerrado en el menor tiempo posible y con la mayor estabilidad posible del robot.

1. Tras realizar la calibración y la conexión por Bluetooth, colocaremos a PrimeBot sobre la línea negra a seguir.
2. Colocaremos el switch en la posición 0001.
3. Se pulsa el botón de selección de programa.
4. Si hemos hecho bien los pasos en la aplicación de Monitor Serial veremos un mensaje indicando que se ejecutará el código correspondiente a la prueba siguelíneas.
5. PrimeBot comenzará a moverse por la línea negra

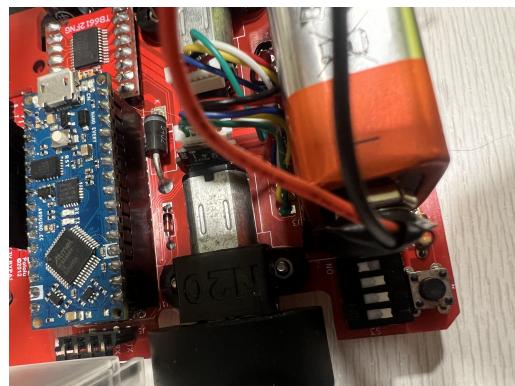


Figura E.2: Posición Prueba Siguelíneas

A la vez que se está ejecutando este algoritmo y PrimeBot se va moviendo, en el Monitor Serial veremos impresos diferentes parámetros los cuales son configurables enviando caracteres a través de Monitor Serial de la siguiente forma:

- P: Aumentará la variable Kp en 0.1
- p: Reducirá la variable Kp en 0.1
- I: Aumentará la variable Ki en 0.01
- i: Reducirá la variable Ki en 0.01
- D: Aumentará la variable Kd en 0.1

- d: Reducirá la variable Kd en 0.1
- V: Aumentará la variable Kv en 0.01
- v: Reducirá la variable Kv en 0.01
- B: Aumentará la variable Velocidad base en 1
- b: Reducirá la variable Velocidad base en 1

## Modo Cuadrícula

Esta funcionalidad corresponde a la posición 0010 del dip switch integrado en la placa de control.

El modo cuadrícula está pensado para la navegación de PrimeBot a través de una cuadrícula ya definida en dimensiones y otorgando a PrimeBot una estación de origen y una de destino. La ejecución correcta de este programa será:

1. Seleccionamos la posición 0010 en el switch.
2. Se pulsa el botón de selección de programa.
3. Colocamos a PrimeBot en la estación de origen que corresponda.
4. A través del monitor serial nos comunicaremos con PrimeBot.
5. El primer parámetro que tenemos que poner es el número de puntos bloqueados, en caso de poner un valor mayor que 0, PrimeBot solicitará las coordenadas X e Y de los puntos bloqueados.
6. Seguido nos solicitará la estación de Origen donde ya está situado PrimeBot.
7. Por último nos solicitará la estación de destino.
8. PrimeBot comenzará a navegar por la cuadrícula hasta llegar a la estación de destino.

En el caso de que no sea posible calcular una ruta debido a los puntos bloqueados, PrimeBot nos lo comunicará por el Monitor serial y se deberá pulsar de nuevo el botón de selección de programa. Además, a través del monitor serial PrimeBot imprimirá una cuadrícula igual a la real indicando tanto el recorrido que está realizando como los puntos bloqueados.

Esta prueba también emplea un algoritmo PID para navegar a través de la cuadrícula por tanto podemos modificar los parámetros que hemos nombrado en el apartado anterior si lo vemos necesario.

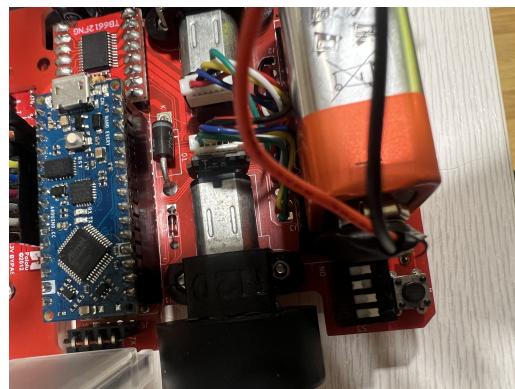


Figura E.3: Posición Prueba Cuadrícula

## Modo Laberinto

Esta funcionalidad corresponde a la posición 0011 del dip switch integrado en la placa de control.

1. Seleccionamos la posición 0011 en el switch.
2. Colocamos a PrimeBot en el origen del laberinto.
3. PrimeBot recorrerá el laberinto hasta encontrar la salida.

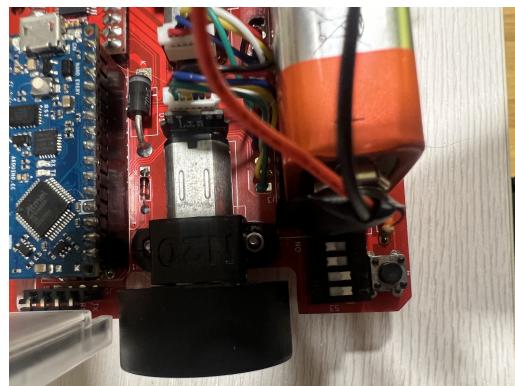


Figura E.4: Posición Prueba Laberinto



## *Apéndice F*

---

# **Anexo de sostenibilización curricular**

---

## **F.1. Introducción**

El desarrollo de PrimeBot no solo se centró en la creación de un robot eficiente y competitivo, sino que también se consideraron aspectos importantes de sostenibilidad. Este anexo proporciona una reflexión personal sobre las competencias de sostenibilidad adquiridas durante el desarrollo del proyecto y cómo se han aplicado en el Trabajo de Fin de Grado. La sostenibilidad en la ingeniería es crucial para asegurar que las soluciones tecnológicas no solo sean efectivas, sino también responsables con el medio ambiente y la sociedad.

## **F.2. Elección de materiales**

Durante el desarrollo de PrimeBot, una de las primeras consideraciones de sostenibilidad fue la elección de materiales y componentes. Se priorizaron componentes electrónicos de bajo consumo energético, como el Arduino Nano y los sensores de Pololu, que son eficientes y reducen el impacto ambiental. Además, se utilizó un PCB personalizado para minimizar el uso de cables y componentes adicionales, reduciendo así los residuos electrónicos.

Además los componentes realizados a través de impresión 3D han sido fabricados con los materiales menos contaminantes que hay en el mercado.

### **F.3. Optimización energética**

El algoritmo PID y otros algoritmos de control fueron diseñados para ser eficientes en términos de consumo de energía. PrimeBot fue programado para optimizar el uso de energía de los motores y sensores, lo que no solo mejora el rendimiento del robot, sino que también extiende su vida útil y reduce la necesidad de recargas frecuentes.

### **F.4. Reciclaje y Reutilización**

Otro aspecto de sostenibilidad abordado en el proyecto fue el reciclaje y la reutilización de componentes. Muchos de los componentes utilizados en PrimeBot fueron reutilizados de proyectos anteriores, reduciendo la necesidad de adquirir nuevos materiales.

### **F.5. Gestión de Residuos**

La gestión de residuos fue un aspecto clave en todas las fases del proyecto. Durante la fase de prototipado, se implementaron prácticas para minimizar los residuos generados, como la reducción de impresiones y el uso eficiente de materiales durante el montaje del PCB. Los componentes obsoletos o defectuosos fueron desechados de acuerdo con las normativas de gestión de residuos electrónicos.

### **F.6. Conclusión**

El desarrollo de PrimeBot me permitió adquirir y aplicar diversas competencias de sostenibilidad, desde el uso responsable de recursos y la optimización energética, hasta la promoción de la economía circular y la educación social. Estas competencias no solo son relevantes para el proyecto específico de PrimeBot, sino que también serán esenciales en mi futura carrera como ingeniero. La integración de prácticas sostenibles en proyectos de ingeniería es crucial para contribuir a un futuro más sostenible y responsable.

---

## Bibliografía

---

- [1] Arduino. Arduino nano every. <https://store.arduino.cc/products/arduino-nano-every>, 2023. [Internet; descargado 10-junio-2024].
- [2] JetBrains. Youtrack. <https://www.jetbrains.com/youtrack/>, 2023. [Internet; descargado 10-junio-2024].
- [3] Katsuhiko Ogata. *Modern Control Engineering*. Prentice Hall, 2010.
- [4] Pololu. Opt3101 distance sensor. <https://www.pololu.com/product/3101>, 2023. [Internet; descargado 10-junio-2024].
- [5] Pololu. Qtr-8a reflectance sensor array. <https://www.pololu.com/product/960>, 2023. [Internet; descargado 10-junio-2024].
- [6] Ken Schwaber and Jeff Sutherland. The scrum guide. <https://www.scrumguides.org/scrum-guide.html>, 2020. [Internet; descargado 10-junio-2024].