Contenido

[1. INTRODUCCIÓN 2](#_Toc435097519)

[1.1. Objetivo 2](#_Toc435097520)

[1.2. Hardware empleado 2](#_Toc435097521)

[1.2.1. Arduino Leonardo 2](#_Toc435097522)

[1.2.2. Sensores 2](#_Toc435097523)

[1.2.3. Actuadores 2](#_Toc435097524)

[1.2.4. Elementos de comunicación 2](#_Toc435097525)

[1.2.5. Alimentación 2](#_Toc435097526)

[1.3. Software empleado 2](#_Toc435097527)

[1.3.1. Arduino IDE 2](#_Toc435097528)

[1.3.2. Python 2.7 2](#_Toc435097529)

[2. ESPECIFICACIÓN DE REQUISITOS 2](#_Toc435097530)

[2.1. Requisitos funcionales 2](#_Toc435097531)

[2.2. Requisitos no funcionales 3](#_Toc435097532)

[3. PLANIFICACIÓN 3](#_Toc435097533)

[4. PRESUPUESTO 3](#_Toc435097534)

[5. ANÁLISIS 4](#_Toc435097535)

[5.1. Casos de uso 4](#_Toc435097536)

[5.2. Diagrama de flujo 4](#_Toc435097537)

[6. DISEÑO 4](#_Toc435097538)

[6.1. Estructura 4](#_Toc435097539)

[6.2. Plan de pruebas 4](#_Toc435097540)

[7. IMPLEMENTACIÓN 4](#_Toc435097541)

[7.1. Librerías 4](#_Toc435097542)

[7.2. Apuntes sobre el código 4](#_Toc435097543)

[8. MONTAJE 4](#_Toc435097544)

[9. PRUEBAS 4](#_Toc435097545)

[10. MEJORAS 4](#_Toc435097546)

# INTRODUCCIÓN

## Objetivo

Robot móvil capaz de recorrer un laberinto de 5x5 celdas tratando de encontrar la salida. Las acciones llevadas a cabo por el robot para conseguir su objetivo son:

* Movimientos: Movimientos hacía adelante y atrás, además de movimientos rotatorios combinando de el giro de las ruedas independientemente
* Detección: Detección de marcas negras en el suelo mediante 3 sensores CNY, detección de distancia en un ángulo de 190º gracias la unión de un servo y un sensor de ultrasonido.
* Recogida de información: Se comprueba la existencia de paredes a menos de 20 cm en los angulos -90º (izq) 0º (frente) y 90º (der), se comprueban los 2 sensores de suelo traseros para alinear el robot en la celda y para saber que hemos llegado a ella, se comprueba el sensor delantero de suelo una vez el robot se ha alineado con la celda para determinar si hemos llegado a la celda objetivo. También se utiliza el sensor de ultrasonidos para comprobar si hay una pared a menos de 8cm cada vez que se alinea el robot en la celda, con el objetivo de girar el robot y evitar el coche con la pared.
* Algoritmo de resolución del laberinto: Se basa en la resolución de laberinto mediante la regla de la mano izq, esto significa ir avanzando por las celdas girando a la izquierda siempre que sea posible, si no hacia delante y como última opción derecha. Cuando se llega a un camino sin salida se da un giro de 180º y se sigue con la rutina.
* Monitorización de información: El robot envía información cada vez que llega a una casilla, manda las paredes de esa casilla , la distancia recorrida, el tiempo transcurrido y el numero de casillas recorridas.

## Hardware empleado

<Listado y descripción básica de los elementos finalmente utilizados en el diseño final>

## Arduino Leonardo

Para ello, se utilizará una placa basado en microcontrolador, Arduino Leonardo, y un ordenador personal. <Describir brevemente>

## Sensores

<rellenar con información referente a los sensores>

## Actuadores

## Elementos de comunicación

## Alimentación

## Software empleado

## Arduino IDE

## Python 2.7

# ESPECIFICACIÓN DE REQUISITOS

## Requisitos funcionales

| **ID** | **CATEGORIA** | **DESCRIPCIÓN** |
| --- | --- | --- |
| RF01 | Movimiento | El robot debe ser capaz de moverse |
| RF02 | Movimiento | El robot debe ser capaz de pivotar 90º a derecha |
| RF03 | Movimiento | El robot debe ser capaz de pivotar 90º a izquierda |
| RF04 | Movimiento | El robot debe ser capaz de avanzar en línea recta |
| RF05 | Movimiento | El robot avanza adecuadamente hasta la siguiente celda |
| RF06 | Detección | El robot debe ser capaz de detectar una pared frontal |
| RF07 | Detección | El robot debe ser capaz de detectar una pared lateral derecha |
| RF08 | Detección | El robot debe ser capaz de detectar una pared lateral izquierda |
| RF09 | Detección | El robot debe ser capaz de detectar la transición entre celdas |
| RF10 | Detección | El robot debe ser capaz de detectar la celda de salida |
| RF11 | Resolución | El robot almacena información sobre las celdas del laberinto |
| RF12 | Resolución | El robot es capaz de decidir el siguiente movimiento en base a la información sobre la celda |
| RF13 | Resolución | El robot es capaz de recorrer varias celdas del laberinto siguiendo el algoritmo empleado |
| RF14 | Resolución | El robot es capaz de salir del laberinto |
| RF15 | Información | El robot envía al PC información sobre el número de celdas recorridas |
| RF16 | Información | El robot envía al PC información sobre obstáculos en cada celda |
| RF18 | Información | El robot envía al PC información sobre la distancia que lleva recorrida |
| RF19 | Información | El robot envía al PC información sobre el tiempo transcurrido desde la entrada al laberinto |
| RF20 | Información | El robot envía al PC información sobre la trayectoria ejecutada |
| RF21 | Información | El robot envía al PC información sobre el número de celdas recorridas |
| RF22 | Información | El PC muestra una representación gráfica del laberinto |
| RF23 | Usuario | Interfaz gráfica en PC que recopile información |

## Requisitos no funcionales

| **ID** | **CATEGORIA** | **DESCRIPCIÓN** | **ACCIÓN** |
| --- | --- | --- | --- |
| RNF01 | Tamaño | Condicionado por las dimensiones del laberinto. Laberinto de 5 x 5 celdas, celda de 20 cm |  |
| RNF02 | Consumo | Condicionado por la batería disponible. Batería de 9v, compuesta por 6 pilas AAA |  |
| RNF03 | Correcciones | El robot choca contra una pared frontal | Se evita chocar mediante una detección temprana observando si hay un obstáculo frontal a menos de 8cm, en ese caso se retrocede durante 500 ms y sigue con la rutina como si hubiera llegado. Se manda un mensaje indicándolo |
| RNF04 | Correcciones | El robot está muy cerca lateralmente de alguna pared | Se gira en el sentido opuesto al de la pared. |

# PLANIFICACIÓN

<Incluir un diagrama de Gantt que contemple las tareas a ejecutar para cumplir los requisitos funcionales y no funcionales. Incluir estimación de tiempo para cada una de las tareas.>

# PRESUPUESTO

<Localizar precios a través de internet. Se facilitará el precio de la PCB electronica>.

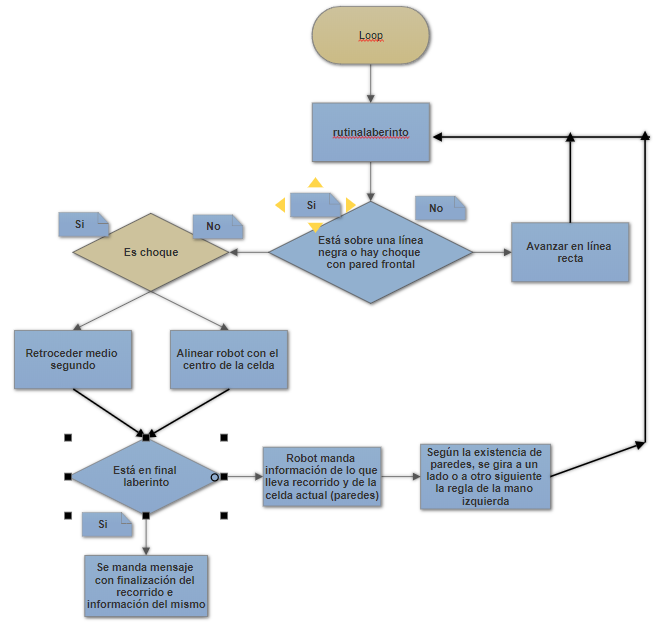
# ANÁLISIS

## Casos de uso

<Interfaz gráfica en PC>

## Diagrama de flujo

**Diagrama de flujo de la aplicación Arduino:**



**Diagrama de flujo de la aplicación Pc**

# DISEÑO

## Estructura

El programa principal hace uso de 8 librerías para llevar a cabo toda la lógica necesaria:

* Motor
* Giro
* ServoMotor
* SensorDistancia
* SensorDistanciaMovil
* Cny
* BotLaberinto
* Batería

Y para resolver el laberinto, llama en su bucle principal al método rutinaLaberinto de la clase BotLaberinto. En esta función se lleva a cabo el algoritmo de resolución de laberintos, haciendo uso de las demás librerías.

Adicionalmente cuenta con una función llamada testDeComponentes, donde hay varias pruebas de los componentes del robot.

## Plan de pruebas

Se han realizado pruebas unitarias por cada librería:

**BotLaberinto:**

*botLaberinto->avanzarFrente(50);*

*botLaberinto->retroceder(50);*

*Serial1.println("Hay pared izquierda?:");*

*Serial1.println(botLaberinto->hayParedIzquierda(20));*

*delay(250);*

*Serial1.println("Hay pared Derecha?:");*

*Serial1.println(botLaberinto->hayParedDerecha(20));*

*delay(250);*

*Serial1.println("Hay pared Frente?:");*

*Serial1.println(botLaberinto->hayParedFrente(20));*

*botLaberinto->avanzarFrente(50);*

**Motor:**

*motorIzq->acelerar(175);*

*motorDer->acelerar(175);*

**Giro:**

*giro->giroBiMotor(90);*

*delay(5000);*

*giro->giroBiMotor(-90);*

*delay(5000);*

*giro->giroBiMotor(-180);*

**ServoMotor:**

*servo->giro(-90);*

*delay(1000);*

*servo->giro(90);*

*delay(1000);*

*servo->giro(90);*

*delay(1000);*

*servo->giro(0);*

*delay(1000);*

*servo->giro(-90);*

*delay(1000);*

*servo->giro(90);*

*delay(1000);*

*servo->giro(0);*

*delay(1000);*

*servo->giroSuave(-90, 10);*

*servo->giroSuave(0, 10);*

*servo->giroSuave(90, 10);*

**SensorDistancia:**

*Serial1.println(sensorDistancia->getDistancia());*

**SensorDistanciaMovil:**

*Serial1.println("Distancia al frente:");*

*Serial1.println(sensorDistanciaMovil->getDistancia(0));*

*delay(250);*

*Serial1.println("Distancia a la izquierda:");*

*Serial1.println(sensorDistanciaMovil->getDistancia(-90));*

*delay(250);*

*Serial1.println("Distancia al frente:");*

*Serial1.println(sensorDistanciaMovil->getDistancia(0));*

*delay(250);*

*Serial1.println("Distancia a la derecha:");*

*Serial1.println(sensorDistanciaMovil->getDistancia(90));*

*delay(250);*

**Cny:**

*Serial1.println("-----------------------");*

*Serial1.println("Cny delantero derecho");*

*Serial1.println(cnyDelDer->esNegro(4));*

*Serial1.println(cnyDelDer->getVoltaje());*

*Serial1.println("Cny trasero derecho");*

*Serial1.println(cnyTraDer->esNegro(4.99));*

*Serial1.println(cnyTraDer->getVoltaje());*

*Serial1.println("Cny trasero izquierdo");*

*Serial1.println(cnyTraIzq->esNegro(4));*

*Serial1.println(cnyTraIzq->getVoltaje());*

**Batería:**

*Serial1.println((String)bateria->getVoltaje());*

# IMPLEMENTACIÓN

## Librerías

* Motor: Proporciona funciones para el manejo del motor
* Giro: Proporciona funcionalidades para efectuar giros con dos ruedas
* ServoMotor: Proporciona funciones para el manejo del servo motor
* SensorDistancia: Proporciona funciones para la utilización del sensor ultrasonidos
* SensorDistanciaMovil: proporciona funcionalidades para el manejo conjunto del servo motor y el sensor de ultrasonidos
* Cny: Proporciona utilidades con respecto a el uso de los CNY para identificación binaria
* BotLaberinto: Clase que modela el robot laberinto, haciendo uso de todos los componentes, ofrece funciones útiles para resolver el laberinto
* Batería: Proporciona información sobre la batería

Todas estas librerías se pueden encontrar dentro de la carpeta “codigos”

## Apuntes sobre el código

Se han encontrado muchos problemas a la hora de realizar las tareas que se precisan, a continuación se resumen los principales problemas:

**Problemas con los Giros:**

El mayor problema de los giros es que si no se hace nada para evitarlo, cuando calcular el tiempo que dejas encendido los motores para girar un determinado ángulo, a medida que se agota la batería y baja la tensión, las ruedas se mueven más lentamente y se necesita más tiempo para realizar el giro. Para resolver esta problemática se intento analizar la correlación entre voltaje y retardo necesario para girar unos determinados grados. Pero esta relación no es lineal, por lo que generar una función de la recta para resolver esta relación no resulto muy eficar. Para solucionarlo, finalmente se acudió a los encoders de las ruedas, con los cuales se puede determinar la cantidad que gira cada rueda y así hacer el giro más preciso

**Problemas con los CNY:**

Los módulos CNY, han dado muchos problemas, pero casi todos relacionados a su conexión con la placa y fallos en las soldaduras del socket.

**Problemas con el servo motor:**

En un principio utilizamos la librería servo que trae por defecto el ide de Arduino, pero generaba problema con la rueda izq, concretamente con el pin D9, hasta donde sabemos, la librería servo utilizaba uno de los timmers del procesador. Eso hacía que no funcionara correctamente el pin D9. Tuvimos que hacer nosotros mismos una librería para el manejo del servo motor

**Problemas con los encoders:**

El encoder de la rueda izq no funciona, porque el pin D2 no funciona correctamente, hemos solventado este problema haciendo un puente entre el encoder y el pin D7.

# MONTAJE

<Incluir un diagrama de bloques que refleje los elementos interconectados para construir el robot>

<Incluir alguna foto del robot final>

<Incluir capturas de pantalla de la aplicación desarrollada>

# PRUEBAS

Se ha construido un laberinto, de 2 x 4, emulando en que hay en clase, y hemos probado el funcionamiento del robot, en ese escenario, con respecto a los requisitos funcionales relacionados con la comunicación, hemos conectado con el smartphone con el robot y visto los mensajes mandados en el transcurso de la resolución del laberinto.

# MEJORAS

Dado a lo poco preciso que son los giros en cuando a ángulo, se han llevado a cabo unas mejoras para resolver algunos casos que suceden en el transcurso de la resolución del laberinto.

**Alineamiento con la línea de la celda**

Dado que cada celda están marcadas por unas líneas negras, y teniendo en cuenta que tenemos 2 sensores cny traseros, hemos realizado una pequeña corrección del rumbo del robot, basándonos el alineamiento de estos sensores con la línea

**Centrado lateral en la celda**

Se usa el ultrasonido para determinar si en los laterales se está cerca de una pared, si es así se realiza un pequeño giro a la dirección opuesta, esto se realiza una vez se haya realizado el alineamiento con la línea que delimita la celda.