ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
MASTER EN INGENIERÍA DE SISTEMAS Y CONTROL

Memoria del trabajo realizado para la superación de la
Asignatura Proyecto

TITULO:

ORGANIZACIÓN Y DESARROLLO DE SOFTWARE PARA UN ROBOT MÓVIL CON NAVEGACIÓN ASISTIDA POR ETIQUETAS NFC

AUTOR:
Rubén Alonso García

DIRECTOR:
Carlos Cerrada Somolinos

SEPTIEMBRE 2012

Mi más profundo agradecimiento a Don Carlos Cerrada Somolinos, Catedrático de la UNED en el Departamento de Ingeniería de Software y Sistemas Informáticos por toda la ayuda prestada para la realización del presente trabajo.

Con cariño a José Ignacio Bermejo Martín, responsable de la parte hardware por toda la documentación prestada así como por la implicación y el apoyo pese a la difícil situación personal que ha atravesado durante el último año.

MASTER EN INGENIERÍA DE SISTEMAS Y CONTROL

TITULO:

ORGANIZACIÓN Y DESARROLLO DE SOFTWARE PARA UN ROBOT MÓVIL CON NAVEGACIÓN ASISTIDA POR ETIQUETAS NFC

CLASE DE PROYECTO:

Proyecto específico propuesto por un profesor

ESTUDIANTE:
Rubén Alonso García

DIRECTOR:
Carlos Cerrada Somolinos

Autorización

Autorizamos a la Universidad Complutense y a la UNED a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado.

Firmado: Rubén Alonso García

Firma del alumno

RESUMEN

En este trabajo se presentan los principales resultados del estudio realizado para el desarrollo de software dentro del contexto del desarrollo de plataformas robotizadas que incorporan dispositivos inalámbricos. Las actividades realizadas se enmarcan dentro del desarrollo de investigación en ambientes universitarios o privados dentro del marco I+D+I.

El enfoque inicial del trabajo tenía un carácter práctico orientado al desarrollo del software que permitiese la lectura/escritura de etiquetas NFC, este software se integra dentro de una plataforma cuya funcionalidad es su uso como demostrador de resultados de investigación en tareas de navegación de robots asistidos por tecnología RFID. Debido a complicaciones en la comunicación con el dispositivo hardware utilizado, el contenido final del trabajo se ha ido ampliando analizando algunos aspectos del resto de elementos de la plataforma como una aplicación de escritorio que dialogue con el robot y registre las simulaciones realizadas.

Como resultado se presenta una especificación de la plataforma y del software involucrado. Para la elaboración de algunos puntos de la especificación se ha considerado la posibilidad de utilizar UML. En este punto se ha hecho uso finalmente de la notación de los diagramas de estado ya que han resultado útiles para definir el comportamiento de los elementos del robot. A lo largo del proceso de modelado se comentan las decisiones de diseño tomadas en función de las restricciones y características de los elementos que conforman el sistema. Para la especificación de la aplicación de escritorio se ha incluido un diseño de la Base de Datos y un prototipo de las pantallas que constituirían el interfaz de usuario, describiendo las acciones permitidas y el dialogo establecido con el robot.

El contenido de la memoria está organizado en los siguientes apartados y anexos:

- **Apartado 1. Descripción del Proyecto.** Se realiza una introducción a las características del proyecto y se mencionan los objetivos que se pretenden alcanzar.

- **Apartado 2. Especificación de Requisitos**. Se describen los requisitos técnicos para el desarrollo del software.

- **Apartado 3. Descripción de la Plataforma**. Se describen los elementos que constituyen la plataforma de simulación.

- **Apartado 4. Modelado del Software**. Se presenta el proceso seguido para el modelado del software, indicando las herramientas utilizadas y las decisiones de diseño tomadas.

- **Apartado 5. Resultados.** Se exponen las conclusiones y las líneas de trabajo futuras.

- **Anexo1. RFID Desktop Reader CPR40.30.** Se presentan las características técnicas del lector y un extracto de los comandos utilizados.

- **Anexo 2. Placas Arduino.** Se exponen las características principales de las placas Arduino Mega y Arduino BT.

- **Anexo 3. Otros Elementos Hardware**. En este anexo se recopilan detalles de elementos hardware del robot como el circuito de alimentación, los drivers de potencia y los sensores de seguimiento de línea.

- **Anexo 4. Código Fuente**. Se presentan rutinas de código relacionadas con los elementos del robot (Comunicación con CPR40.30, comunicación I2C, bluetooth …)

PALABRAS CLAVE

Robot móvil, seguidor de línea, control de robots, NFC, UML, MicroUM, statecharts.

INDICE

FIGURAS Y TABLAS

APARTADO 1

Descripción del Proyecto

Introducción

Objetivos

Introducción

Con el presente proyecto se pretende analizar el software necesario para la realización de simulaciones utilizando una plataforma de bajo coste que incluye un robot móvil de seguimiento de línea y un circuito físico.

Las simulaciones se centrarán en el recorrido de trayectorias que abarquen dos tareas principales, la lectura/escritura de etiquetas NFC y la navegación por seguimiento de línea. Para facilitar la realización de las simulaciones el robot se comunicará con una aplicación externa de la que recibirá comandos y con la que intercambiará información relativa a las simulaciones.

El estudio abarcará tanto el software embebido como la aplicación de escritorio. A lo largo del análisis se considerará la viabilidad y conveniencia de utilizar la notación que proporciona el Lenguaje de Modelado Unificado para la especificación.

Se realizará especial hincapié en la parte NFC estudiando las posibilidades que ofrece el dispositivo RFID Desktop Reader CPR40.30.

Objetivos

En este punto se enunciarán los objetivos que se pretenden cumplir al realizar este proyecto. Se han subdividido en dos categorías, una que engloba los objetivos generales inherentes a la propia naturaleza del proyecto, y otros específicos referidos a las expectativas del autor.

Objetivos Generales

Los objetivos generales que se pretenden alcanzar con la realización de este proyecto se pueden resumir en los siguientes puntos:

- Realizar un estudio del software a implementar aplicando en la medida de lo posible la notación UML en aquellas partes que sean susceptibles de su uso.
- La funcionalidad que debe mostrar el prototipo será principalmente el seguimiento de línea guiado por el contenido de las etiquetas NFC. Se hará especial consideración a la funcionalidad correspondiente a la lectura/escritura de estas etiquetas.
- El sistema incluirá una aplicación externa que enviará comandos al robot y que registrará los datos relativos a las simulaciones realizadas.

Objetivos Específicos

Al seleccionar este proyecto el autor del mismo busca conseguir satisfacer una serie de objetivos específicos como los siguientes:

- Familiarizarse con la tecnología NFC.
- Comprobar la utilidad del Lenguaje de Modelado Unificado en el caso práctico que supone un sistema para el que inicialmente no estaba concebido su uso como supone un robot móvil.
- Comprobar de forma práctica algunos conocimientos adquiridos en las asignaturas del master.

APARTADO 2

Especificación de Requisitos

Introducción

Descripción General

Requisitos Específicos

Introducción

Esta sección provee un resumen de la especificación de requisitos técnicos para el desarrollo de un robot móvil con navegación asistida por etiquetas NFC y navegación guiada desde un PC.

Propósito

La presente especificación describe las necesidades tanto a nivel de hardware como de software para el desarrollo del robot. Se pretende proporcionar una enumeración concisa de requisitos que permita una comunicación fluida entre los miembros del equipo de desarrollo.

Esta especificación supondrá el punto de partida para el resto del estudio en el que se irá considerando la notación adecuada para las distintas partes de la especificación.

Podrá utilizarse también en trabajos posteriores como base para elaborar el plan de pruebas de aceptación.

Referencias

Consultar los anexos 1, 2 y 3 en los que se describen características hardware de los elementos de la plataforma.

Descripción General

Perspectiva del producto

El objetivo final y global del sistema será el estudio de la respuesta del robot con el uso de la tecnología NFC y simulación de distintos caminos.

Funcionalidad del sistema a desarrollar

La funcionalidad del sistema se distribuirá en los dos elementos principales de los que consta:

- Robot
    - Navegación por un circuito de línea.
    - Lectura/Escritura de etiquetas NFC.
    - Comunicación vía bluetooth con el PC.
- PC
    - Comunicación vía bluetooth con el Robot.
    - Representación gráfica de las simulaciones.
    - Elaboración de las trayectorias adecuadas.
    - Registro de Históricos.

Plan de desarrollo

La oportunidad de desarrollo de este proyecto está vinculada al desarrollo de investigación en ambientes universitarios o privados dentro del marco I+D+I.

Las tareas de desarrollo podrían separase en dos equipos. El primero se deberá encargar del desarrollo del hardware y el segundo del software. Comenzaría el primer equipo diseñando un prototipo sobre el que el segundo equipo pueda volcar el software de navegación, control y comunicación.

Requisitos Específicos

En este apartado se indicarán los requisitos iniciales de partida. Aunque se podría haber utilizado la notación UML de casos de uso se ha preferido utilizar las fichas vistas durante el master para recoger los requisitos iniciales de un sistema y utilizar la notación UML en otros puntos más avanzados del estudio.

Estas fichas utilizan una plantilla en la que se indicarán los datos principales, incluyendo número, tipo, nombre, prioridad y descripción. Posteriormente estas fichas podrán utilizarse como base para iniciar el documento de pruebas.

Requisitos Funcionales

Número de requisito	Fun_01
Nombre de requisito	Ciclo Principal
Tipo	☒Requisito ☐Restricción
Prioridad del Requisito	☒Alta/Esencial ☐Media/Deseado ☐Baja/Opcional
Descripción	Debe obedecer lo antes posible a los comandos recibidos desde el PC, conmutando al modo de operación según el siguiente Ciclo de Programa: • Recepción por bluetooth de comando. • Parser de comandos. • Transición al nuevo modo de operación. • Ejecución del modo. • Envío de respuesta por bluetooth.

Número de requisito	Fun_02
Nombre de requisito	Modo Navegación por Trayectoria NFC
Tipo	☒Requisito ☐Restricción
Prioridad del Requisito	☒Alta/Esencial ☐Media/Deseado ☐Baja/Opcional

Descripción	Al recibir un comando de modo navegación por trayectoria NFC el robot deberá realizar las siguientes acciones:
	• Navegar siguiendo la línea del circuito y las indicaciones grabadas en las etiquetas.
	• Leer el contenido de las etiquetas por las que vaya pasando y enviarlo por bluetooth al PC.
	• Pararse en el momento en que encuentre el final de la trayectoria o no pueda seguir.
	• Enviar el resultado:
	  ○ OK si ha completado la trayectoria leyendo correctamente todas las etiquetas NFC
	  ○ NOK si ha tenido algún problema en el recorrido como fallos en la lectura de una etiqueta.

Número de requisito	Fun_03
Nombre de requisito	Modo Grabación de Trayectoria
Tipo	☒Requisito ☐Restricción
Prioridad del Requisito	☒Alta/Esencial ☐Media/Deseado ☐Baja/Opcional
Descripción	Al recibir un comando de grabación de trayectoria el robot deberá realizar las siguientes acciones: • Navegar siguiendo la línea del circuito y la trayectoria recibida. • Escribir el contenido recibido en las etiquetas por las que va pasando. • Se parará en el momento en que encuentre el final de la trayectoria o no pueda seguir. • Enviar el resultado: o OK si ha completado la grabación de la trayectoria. o NOK si ha tenido algún problema en el recorrido como fallos en la escritura de una etiqueta.

Número de requisito	Fun_04
Nombre de requisito	Modo Navegación Guiada
Tipo	☒Requisito ☐Restricción
Prioridad del Requisito	☒Alta/Esencial ☐Media/Deseado ☐Baja/Opcional
Descripción	Al recibir un comando de navegación guiada el robot deberá realizar las siguientes acciones: • Navegar siguiendo la línea del circuito y la trayectoria recibida desde el PC. • Leer el contenido de las etiquetas por las que vaya pasando y enviarlo por bluetooth al PC. • Se parará en el momento en el que haya pasado por todos los nodos indicados en la trayectoria. • Enviar el resultado: o OK si ha completado la trayectoria leyendo correctamente todas las etiquetas NFC. o NOK si ha tenido algún problema en el recorrido como fallos en la lectura de una etiqueta.

Número de requisito	Fun_05
Nombre de requisito	Modo Posición Actual
Tipo	☒Requisito ☐Restricción
Prioridad del Requisito	☒Alta/Esencial ☐Media/Deseado ☐Baja/Opcional
Descripción	Al recibir un comando de posición actual el robot realizará las siguientes acciones: • Leer el contenido de la etiqueta en la que se encuentra y enviarlo por bluetooth al PC. En caso de fallo en la lectura enviará un NOK. • Enviar por bluetooth el nodo previo en el que se encontraba. Este dato permitirá al PC conocer la orientación del robot.

Número de requisito	Fun_06
Nombre de requisito	Movimiento del robot
Tipo	☐Requisito ☒Restricción
Prioridad del Requisito	☒Alta/Esencial ☐Media/Deseado ☐Baja/Opcional
Descripción	El robot podrá realizar los siguientes movimientos: • Avance • Giro de hasta 90º a izquierda y derecha No podrá realizar los siguientes movimientos: • Marcha atrás • Media vuelta (giros de 180ª)

Requisitos de comunicación

El robot tiene comunicaciones con el PC y con los sensores internos que utiliza.

Número de requisito	Com_01
Nombre de requisito	Comunicación con el lector NFC
Tipo	☒Requisito ☐Restricción
Prioridad del Requisito	☒Alta/Esencial ☐Media/Deseado ☐Baja/Opcional
Descripción	La comunicación con el dispositivo de lectura/escritura NFC se realizará vía serie utilizando un adaptador serie/usb siguiendo los protocolos de comunicación y comandos indicados en la especificación del dispositivo. Para más detalle ver Anexo1

Número de requisito	Com_02
Nombre de requisito	Comunicación con el PC
Tipo	☒Requisito ☐Restricción
Prioridad del Requisito	☒Alta/Esencial ☐Media/Deseado ☐Baja/Opcional
Descripción	La comunicación con el PC se realizará mediante bluetooth.

Número de requisito	Com_03
Nombre de requisito	Comunicación con el sensor de infrarrojos
Tipo	☒Requisito ☐Restricción
Prioridad del Requisito	☒Alta/Esencial ☐Media/Deseado ☐Baja/Opcional
Descripción	La comunicación con los sensores de seguimiento de línea se realizará vía serie. Para más detalle ver anexo 3.

Requisitos de interfaz de usuario

El interfaz de usuario estará presente en la aplicación ubicada en el PC que se comunicará con el robot.

Número de requisito	IntUsu_01
Nombre de requisito	Grabación de Trayectoria
Tipo	☒Requisito ☐Restricción
Prioridad del Requisito	☒Alta/Esencial ☐Media/Deseado ☐Baja/Opcional
Descripción	La aplicación mostrará una pantalla para especificar los datos de la trayectoria que el robot deberá grabar en el circuito. Para ello presentará los campos necesarios para especificar el contenido de las celdas involucradas en la trayectoria.

Número de requisito	IntUsu_02
Nombre de requisito	Navegación por Trayectoria NFC
Tipo	☒Requisito ☐Restricción
Prioridad del Requisito	☒Alta/Esencial ☐Media/Deseado ☐Baja/Opcional
Descripción	La aplicación mostrará una pantalla para especificar el nodo de inicio donde debe comenzar el robot la trayectoria.

Número de requisito	IntUsu_03
Nombre de requisito	Recorrido Completo
Tipo	☒Requisito ☐Restricción
Prioridad del Requisito	☒Alta/Esencial ☐Media/Deseado ☐Baja/Opcional
Descripción	La aplicación mostrará un botón que permitirá realizar el recorrido completo del circuito. Para ello la aplicación con los datos de la posición del robot, posición previa del robot y el mapa del circuito elaborará una trayectoria adecuada para recorrer todos los nodos al menos una vez.

Número de requisito	IntUsu_04
Nombre de requisito	Representación del Circuito
Tipo	☒Requisito ☐Restricción
Prioridad del Requisito	☐Alta/Esencial ☒Media/Deseado ☐Baja/Opcional
Descripción	La aplicación mostrará en todo momento una representación del circuito en el que se podrá visualizar el último contenido leído de cada etiqueta. Cuando el robot esté navegando por el circuito irá resaltando de forma dinámica los nodos que va atravesando.

Número de requisito	IntUsu_05		
Nombre de requisito	Histórico de simulaciones		
Tipo	☒Requisito	☐Restricción	
Prioridad del Requisito	☐Alta/Esencial	☒Media/Deseado	☐Baja/Opcional
Descripción	La aplicación mostrará una pantalla en la que se podrá consultar el listado de las simulaciones realizadas. Para ello se deberá guardar datos como el tipo de simulación, la fecha, la trayectoria etc.		

APARTADO 3

Descripción de la Plataforma

Introducción

Robot

Circuito

Dispositivos para Lectura/Escritura

PC

Introducción

Esta sección provee una descripción de la plataforma física que se utilizará para la realización de las simulaciones. Los elementos básicos que componen la plataforma son:

- Robot Móvil.
- Circuito físico con etiquetas NFC.
- PC de sobremesa que albergará la aplicación de escritorio.
- Teléfono Móvil para la grabación de etiquetas NFC.

Estos elementos interaccionan entre si mediante distintos protocolos de comunicación:

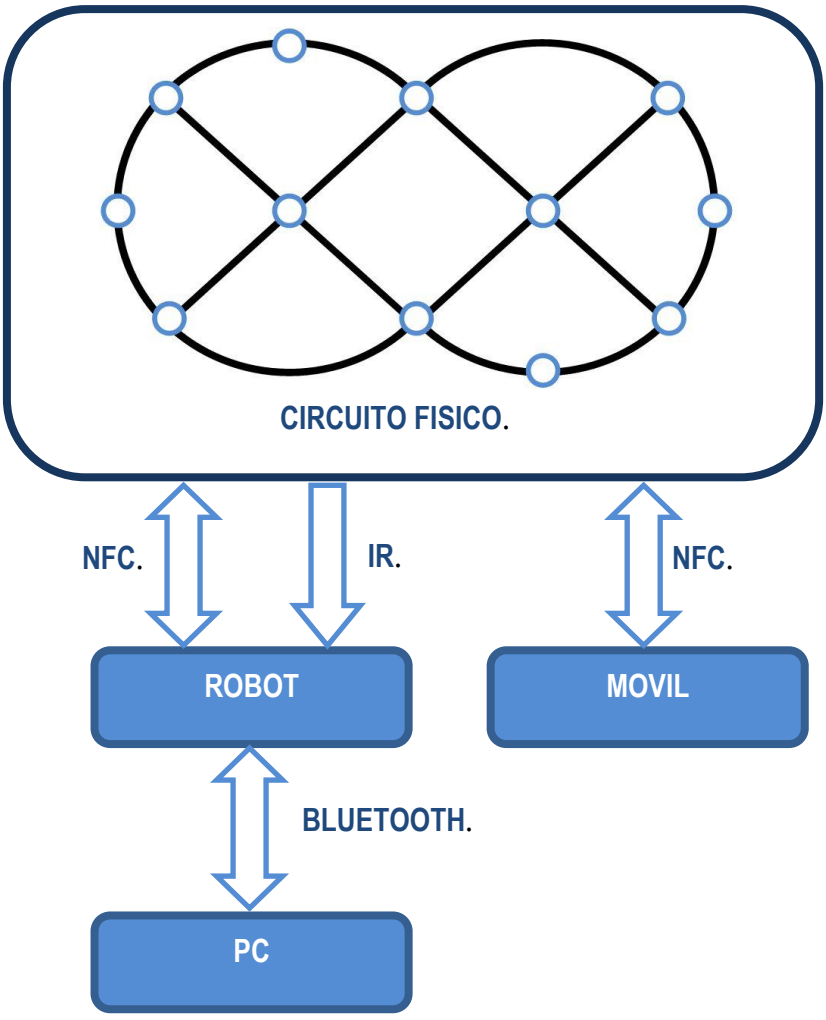


Figura 1: Esquema de la plataforma.

Robot

El robot con el que se ha realizado este proyecto es un robot móvil seguidor de línea. Un robot seguidor de línea es capaz de moverse en una trayectoria marcada en el suelo. Mediante un fotosensor y un led sigue una trayectoria marcada en negro sobre un fondo blanco.

Arquitectura

Elementos Hardware

La composición del robot incluye todos los elementos hardware necesarios para la realización de las simulaciones contempladas:

La distribución e integración de los distintos elementos se puede apreciar en la siguiente figura:

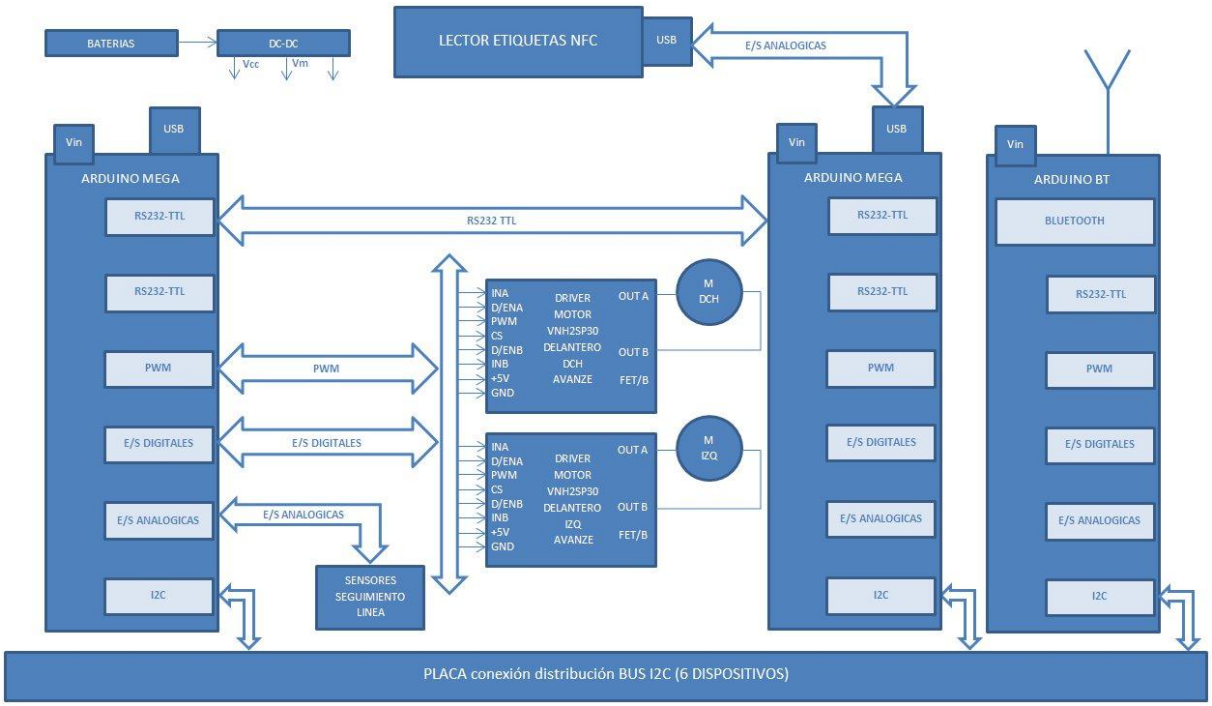


Figura 2: Esquema del Robot.

Los elementos principales de los que consta son los siguientes:

- Placa para navegación en línea Arduino Mega: Placa encargada de la navegación en línea y pensada inicialmente como placa de control mediante I2C del resto de Placas arduino.

- Placa gestión NFC Arduino Mega: Placa encargada del control del lector de etiquetas NFC para la lectura y escritura.
- Placa comunicación ArduinoBT: Placa encargada de la comunicación via bluetooth con el PC.
- Lector de Etiquetas NFC: Dispositivo que permitirá leer y escribir en las etiquetas NFC dispuestas a lo largo del circuito.
- Circuito de alimentación. Alimentación mediante baterías de litio –polímero.
- Drivers de potencia para el control de tracción. Etapa de potencia para el control de los motores DC.
- Sensores de seguimiento de línea.

(Para mayor información consultar los anexos 1, 2 y 3.)

Configuración cinemática

Como puede apreciarse en la siguiente figura, la configuración cinemática se basa en dos ejes paralelos convenientemente separados. Uno de tracción diferencial (el delantero) y otro (el trasero) donde ambas ruedas giran libremente. También cabe destacar la adición de un eje vertical de giro, situado entre ambos ejes de tracción. Este sistema dota al robot de una capacidad de giro ampliada para curvas de radio pequeño.

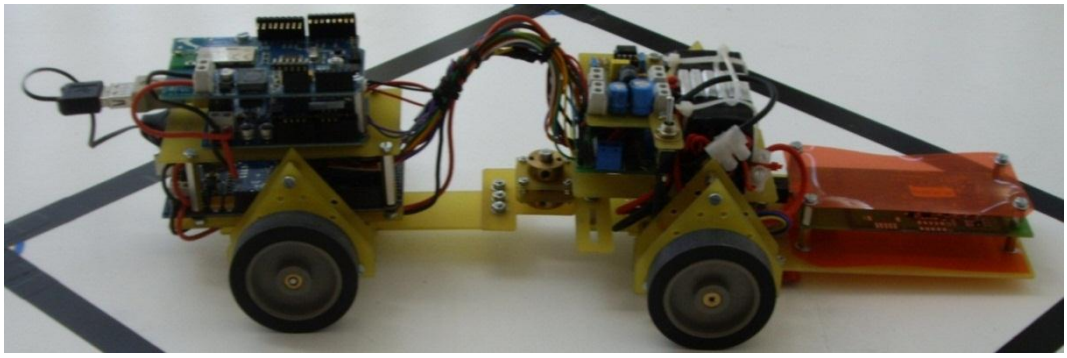


Figura 3: Aspecto General del Robot.

Circuito

Se ha diseñado un circuito básico instalado en un tablero de dimensiones 2x1m, con la base de su superficie en fondo blanco y los caminos trazados en color negro. Debido a las limitaciones espaciales del tablero, en la elaboración del circuito se ha procurado facilitar la labor de detección de etiquetas realizada por el robot. Por ello, se han elegido unos trazados tales que el robot pueda moverse con mayor facilidad en la plataforma. El resultado es el siguiente circuito de formas redondeadas.

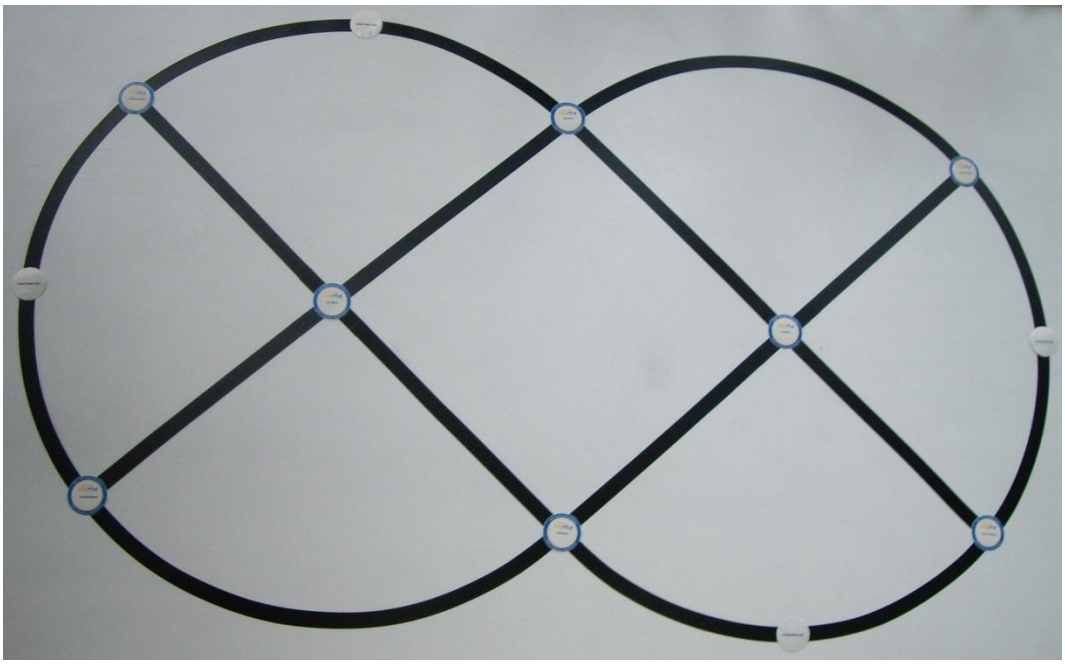


Figura 4: Circuito real trazado en la plataforma

Nodos Etiquetas NFC

Como se puede apreciar en el circuito, existen una serie de nodos que corresponden al conjunto de etiquetas instaladas en el escenario. Dichas etiquetas están ubicadas en las bifurcaciones o puntos clave de la red de trayectorias con el fin de poder configurar un número elevado de posibles recorridos.

Figura 5: Etiqueta NFC

Las etiquetas utilizadas son de tipo pasivo (MiFare) donde el campo generado para la transferencia de información es sólo el producido por el lector. La capacidad de almacenamiento será de un 1Kb.

Los datos introducidos en la etiqueta indicarán al robot en el modo de navegación por trayectoria NFC el movimiento que tiene que realizar. Serán los siguientes:

- ID. Número de identificación. Cada una de las etiquetas dispondrá de n valor único que la distinguirá del resto. Dicho valor estará vinculado a una posición en el plano (x,y) que permitirá conocer la ubicación del robot cada vez que se encuentre una etiqueta.
- Tipo Nodo. Este parámetro indica el número de vías que salen (o llegan) del nodo. De esta manera, se introduce al dispositivo de escritura el tamaño de los vectores entrada/salida.
- Entradas/Salidas. Estos datos indican la trayectoria en cada nodo. Se vincula la vía de entrada con el tipo de desplazamiento que es necesario realizar, izquierda (I), derecha(D), recto (R) o pararse (0).

Dispositivos para lectura/escritura

Para facilitar la realización de simulaciones, la plataforma dispone de un móvil Nokia 6212 con el software adecuado para la grabación y lectura de las etiquetas NFC.

Figura 6: Móvil Nokia 6212

El dispositivo que se integrará en el robot para la lectura/escritura será un dispositivo CPR40. (Ver anexo 1 para más detalle).

Figura 7: Dispositivo para lectura/escritura NFC

PC

La aplicación de control y gestión de las simulaciones se instalará en un PC con capacidad de comunicación por Bluetooth, por lo que su ubicación deberá estar en una distancia adecuada al circuito (< 10m).

Será en el PC en el que se llevarán a cabo los cálculos de las trayectorias que deberá seguir el robot. Será el elemento desde el que se podrán probar diversos algoritmos de recorrido de grafos.

APARTADO 4

Modelado del Software

Introducción

Robot

Aplicación de Escritorio

Introducción

En este apartado se describirá el software que se implementará en el robot. Considerando la especificación de requisitos y la plataforma descrita en apartados anteriores se realizará un modelo de software que permita representar las características deseadas.

El software estará distribuido en función de la capacidad de cálculo de los distintos dispositivos. Así las tarjetas arduino se encargarán de operaciones que exijan poca capacidad de cómputo mientras que en el PC se ubicarán las tareas más exigentes relacionados principalmente con el cálculo de trayectorias.

El modelo se centrará principalmente en la parte concerniente al robot considerando las acciones desencadenadas desde el PC. Así por ejemplo, una acción de grabación de trayectoria solicitada desde el PC, puede desencadenar varias acciones en el robot como lectura de la posición actual, navegación guiada, navegación NFC etc. Y estas a su vez desencadenarán primitivas de más bajo nivel como lectura/escritura de datos en bus I2C, escritura/lectura de datos NFC etc.

Herramientas de diseño

En esta sección se indicarán las herramientas y los elementos de diseño que se han utilizado para modelar las distintas partes del sistema.

UML

El lenguaje de modelado unificado UML es ampliamente usado para el modelado de software y se ha convertido en un estándar en el mundo de la industria. Su uso se ha extendido al desarrollo de sistemas embebidos y de tiempo real.

UML proporciona la posibilidad de presentar el modelo de software mediante el uso de diagramas estandarizados. Al representar las ideas de desarrollo en un modelo visual se facilita la comprensión del sistema por parte de personal ajeno al proyecto y además facilita la comunicación entre los miembros del equipo de desarrollo a la hora de tomar decisiones de diseño.

UML está especialmente indicado para sistemas orientados a objetos y aunque el lenguaje de programación arduino no lo es, es viable utilizar UML para describir el comportamiento de los objetos del robot, considerando estos objetos como los elementos que lo conforman (tarjeta BT, tarjeta de navegación …)

De los diferentes tipos de diagramas que ofrece UML para el modelado nos hemos decantado por los diagramas de estado. La literatura consultada, así como la gran mayoría de las herramientas disponibles relacionadas con arduino están enfocadas al uso de estos diagramas. El resto de diagramas también son susceptibles de ser usados pero los que más van a facilitar el modelado en este caso concreto son los statecharts.

Las herramientas que se han consultado para la integración de UML y arduino son las siguientes:

MicroUML

MicroUML ha sido la primera herramienta que se empezó a usar durante este trabajo. Se trata de un servicio web que permite a partir de un modelo obtener la imagen compilada para subir a la tarjeta arduino. Los modelos se podían realizar desde el entorno Eclipse Indigo utilizando las herramientas de modelado UML del plugin Papyrus. MicroUML se basa en los diagramas de estado que podían modelarse desde Eclipse, partiendo de ellos a través del servicio web proporcionaban el código compilado.

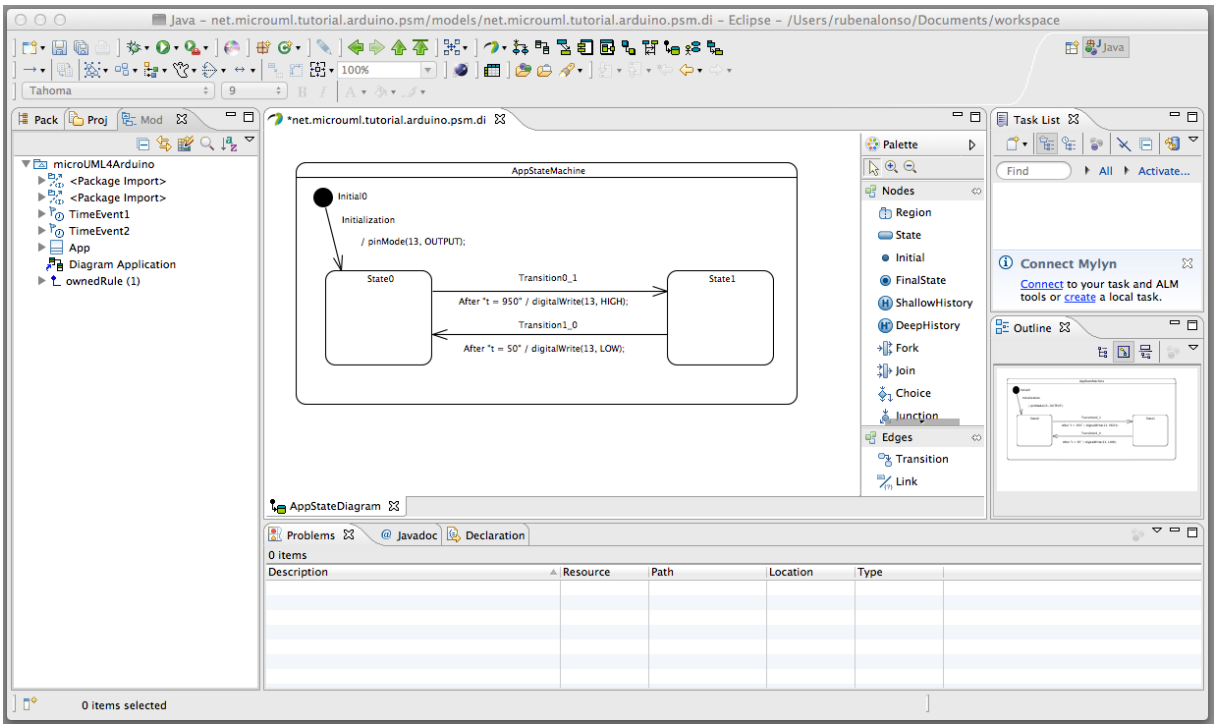


Figura 8: Eclipse con herramientas de modelado UML

Lamentablemente al poco de empezar a trabajar con MicroUML la página web dejó de funcionar por lo que se buscó otra herramienta del mismo estilo.

QM

QM es una herramienta similar a MicroUML pero que incluye su propio software de modelado. Está centrado exclusivamente en los diagramas de estado. Ha sido la herramienta utilizada finalmente para el modelado del sistema.

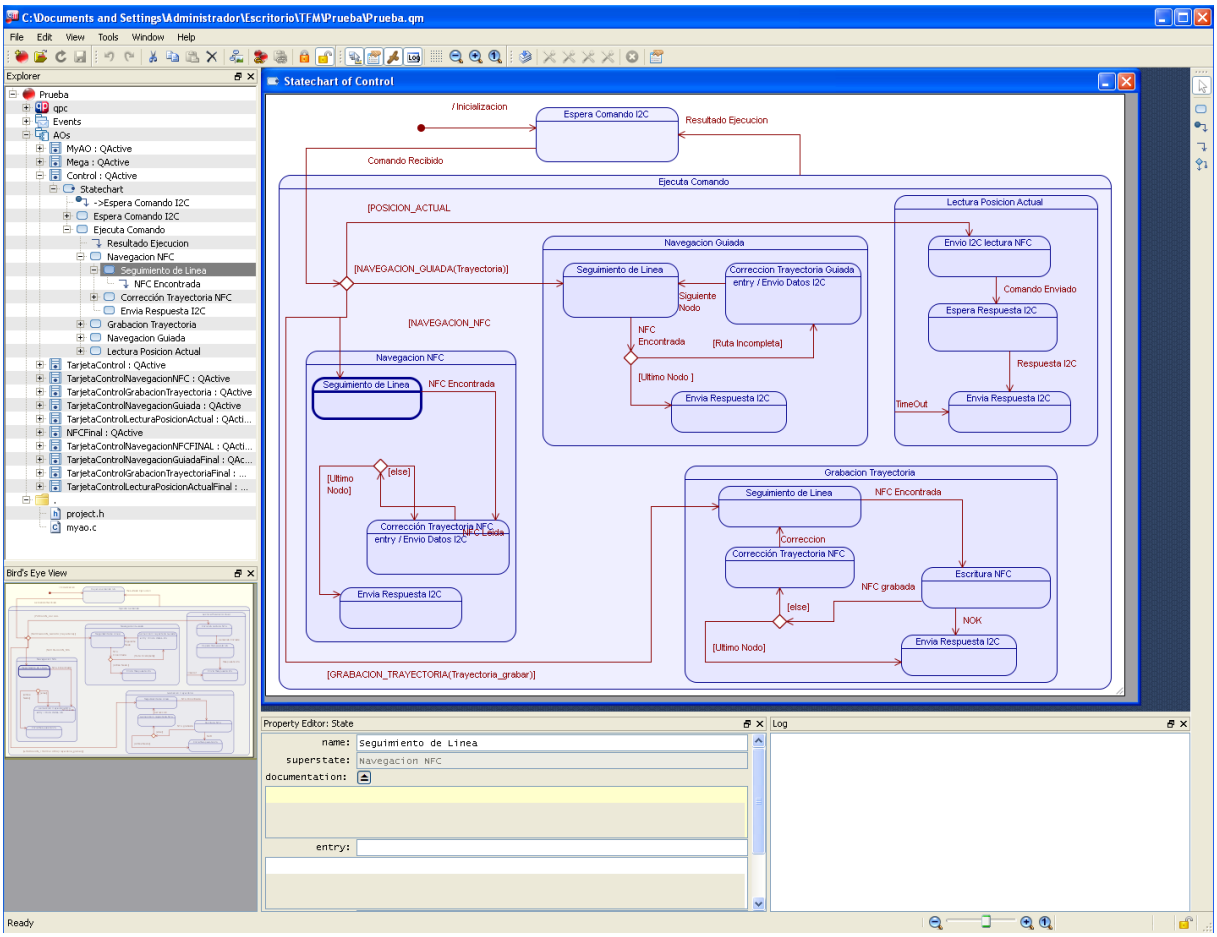


Figura 9: Entorno de Modelado QM

Diseño de interfaz de usuario

El interfaz de usuario se ubicará en la aplicación de escritorio. Para su especificación se presentarán las pantallas diseñadas describiendo los elementos que las forman y las acciones asociadas.

Diseño de BBDD

La única base de datos que contendrá el sistema se encuentra vinculada a la aplicación de escritorio ubicada en el PC. Para su diseño se utilizará un diagrama E/R que muestre las tablas y los datos a almacenar.

Diseño Preliminar

Para realizar una primera aproximación al diseño del software del robot se prescindirá de las posibles restricciones de los dispositivos físicos concretos utilizados en el robot. Se pretende crear un punto de partida donde empiecen a surgir las necesidades de comunicación y funcionalidad y su distribución entre los diferentes elementos del robot. Posteriormente esta primera aproximación se refinará y optimizará realizando las correcciones oportunas para adaptarlo a los elementos del robot y a las restricciones de comunicación.

El primer punto a considerar es decidir cuál de las tarjetas se encargará de llevar el control del funcionamiento global del robot, gestionando el comportamiento del resto de tarjetas. Este robot estaba pensado inicialmente para la lectura de etiquetas NFC y su envío por Bluetooth concentrando las tareas de control en la tarjeta de navegación. En esta primera aproximación de diseño se mantendrá la distribución original y se otorgará el control a la tarjeta de navegación incluyendo el resto de funcionalidades deseadas para la plataforma (grabación de trayectorias, comunicación bidireccional con el PC, nuevos comandos …).

Tarjeta de Navegación

Como se ha mencionado, en esta primera aproximación se mantendrá la distribución original y se otorgarán las funciones de control a esta tarjeta, de forma que se encargará de gestionar el comportamiento del resto de tarjetas a partir de los comandos recibidos. Se comunicará con el resto de tarjetas mediante I2C.

El funcionamiento de la tarjeta se resumiría en el siguiente diagrama de estado:

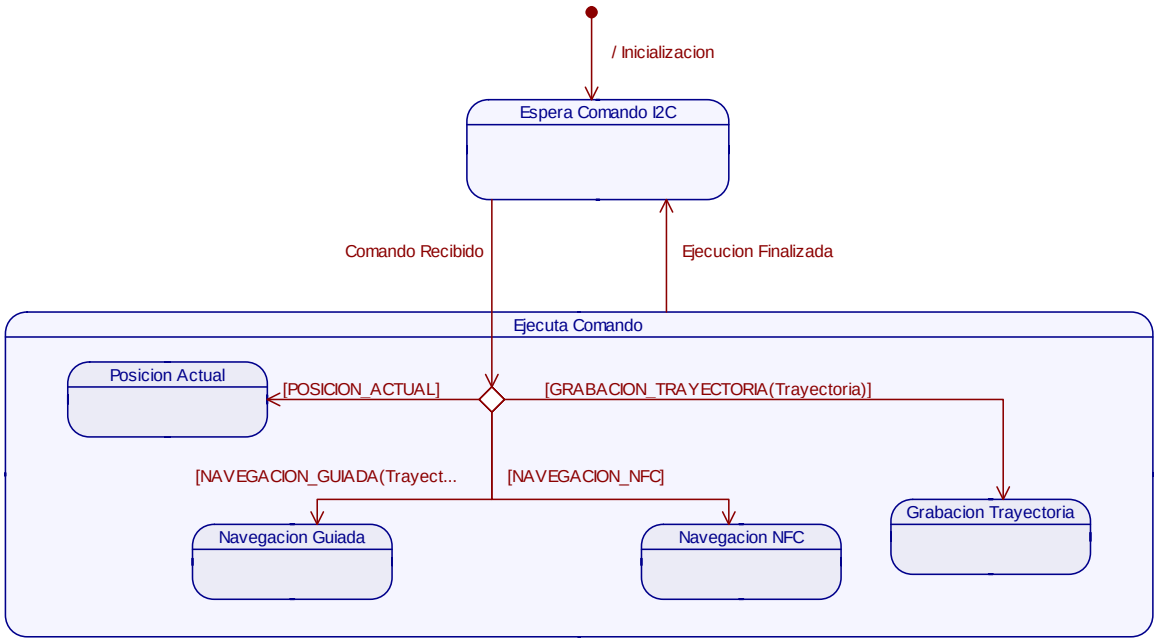


Figura 10: Diagrama de Estado. Tarjeta de control

Inicialización

En la inicialización de la tarjeta se configurarán las comunicaciones y se establecerá el nodo previo como nodo 3. La variable del nodo previo servirá para conocer la orientación del robot. La condición de inicialización del sistema consistirá en tener posicionado el robot en el nodo 1 y considerar que el nodo previo era el 3, el robot por tanto deberá estar orientado hacia el nodo 2 (ver esquema del circuito físico para más detalle).

Espera Comando I2C

El estado principal en el que se encontrará la tarjeta será la espera de comunicación I2C con instrucciones. En este estado el robot permanecerá parado. Si se produce la llegada de un comando se pasará al estado **Ejecuta Comando**.

Ejecuta Comando

En este estado se parseará el comando recibido. Si el comando se encuentra entre los comandos soportados se realizará la transición al estado asociado. Una vez finalizada la ejecución se retornará al estado **Espera Comando I2C**. En caso de que no se reconozca el comando se ignorará y se retornará al estado de **Espera Comando I2C**.

Los subestados que engloban este estado se indican a continuación:

Grabación Trayectoría

La tarjeta pasará a este modo al recibir el comando GRABACION_TRAYECTORÍA. El comando debe incluir la trayectoria a grabar.

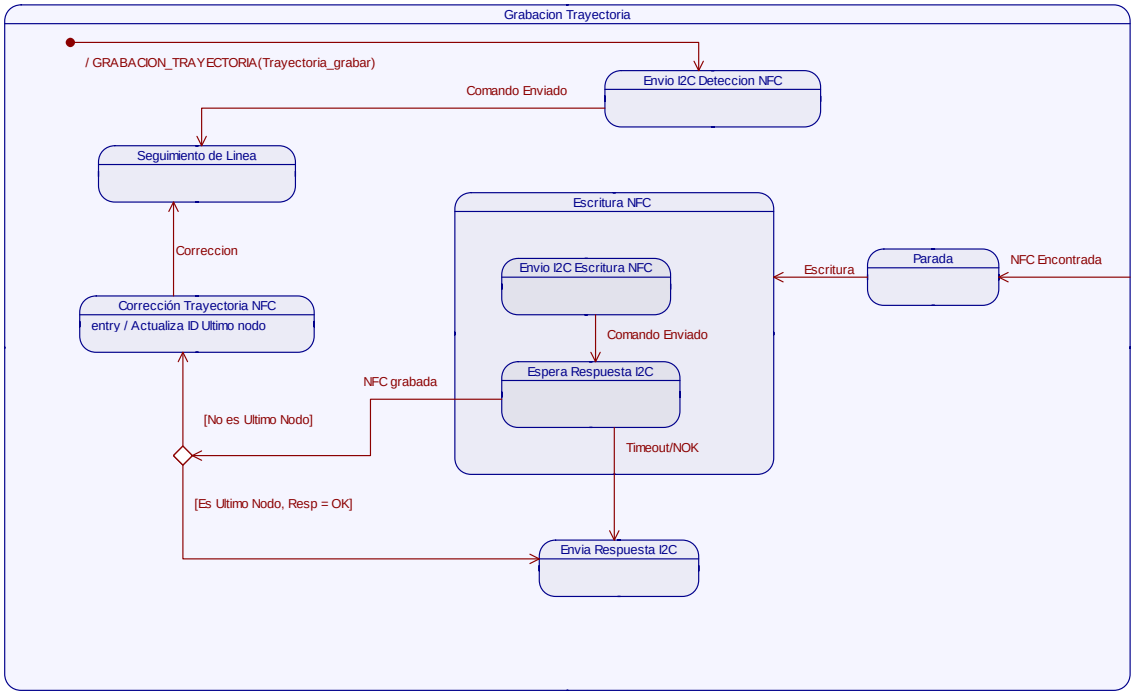


Figura 11: Diagrama de Estado. Grabación Trayectoria

Envío I2C Detección NFC
El primer paso que se realizará en este estado será el envío por I2C del comando de detección de etiquetas NFC a la tarjeta conectada al dispositivo NFC. Se realizará una espera inicial para dar tiempo a la ejecución del comando. Pasado ese tiempo si no se recibe respuesta se interpretará que el robot no ha localizado ninguna etiqueta en su posición y se pasará al estado **Seguimiento de Línea** en el que el robot comenzará a moverse.

La tarjeta conectada al dispositivo NFC al recibir el comando de detección se pondrá en modo de búsqueda y si detecta una etiqueta enviará un comando I2C a la tarjeta de control. Esto desencadenará el evento 'NFC Encontrada'. En este caso se pasará al estado **Parada**.

Parada
En este estado se detendrán los motores que accionan las ruedas (si estuviesen en funcionamiento) y se provocará el paso al estado de **Escritura NFC**.

Escritura NFC
Para la escritura de los datos NFC se enviará a la tarjeta conectada al dispositivo NFC el comando de escritura NFC con los datos a grabar. La tarjeta quedará a la espera de recibir la respuesta de la operación en el estado **Espera Respuesta I2C**.

Espera RespuestaI2C
Este estado se abandonará cuando se reciba el resultado de la operación de escritura por parte de la tarjeta conectada al dispositivo NFC. Si ha habido algún problema en la escritura o salta el timeout de espera se pasará al estado **Envía Respuesta I2C** para enviar un NOK por I2C a la tarjeta Bluetooth. Si la escritura se ha realizado correctamente se verificará si el nodo escrito es el último de la trayectoria en cuyo caso se pasará al estado **Envía Respuesta I2C** para enviar un OK por I2C a la tarjeta Bluetooth. Si no es el último nodo pasará al estado **Corrección Trayectoria NFC**.

Envía RespuestaI2C
En este estado se enviará por I2C un NOK o un OK a la tarjeta Bluetooth para que lo envíe al PC. Será el estado final con lo que se considerará completado el comando de **Grabación Trayectoría**.

Corrección Trayectoria NFC
En este estado se realizará una corrección de la trayectoria seguida en función de las indicaciones asociadas a la etiqueta NFC. El robot deberá girar a la derecha, izquierda o mantener su dirección actual. Además actualizará el valor del último nodo visitado que permitirá posteriormente conocer la orientación del robot. Se activará el **Seguimiento de Línea**

Seguimiento de Línea
El robot se desplazará activando los motores que transmiten movimiento a las ruedas, realizando los ajustes adecuados en función de los datos leídos del sensor de infrarrojos.

Navegación NFC

La tarjeta pasará a este MODO al recibir el comando NAVEGACION_NFC.

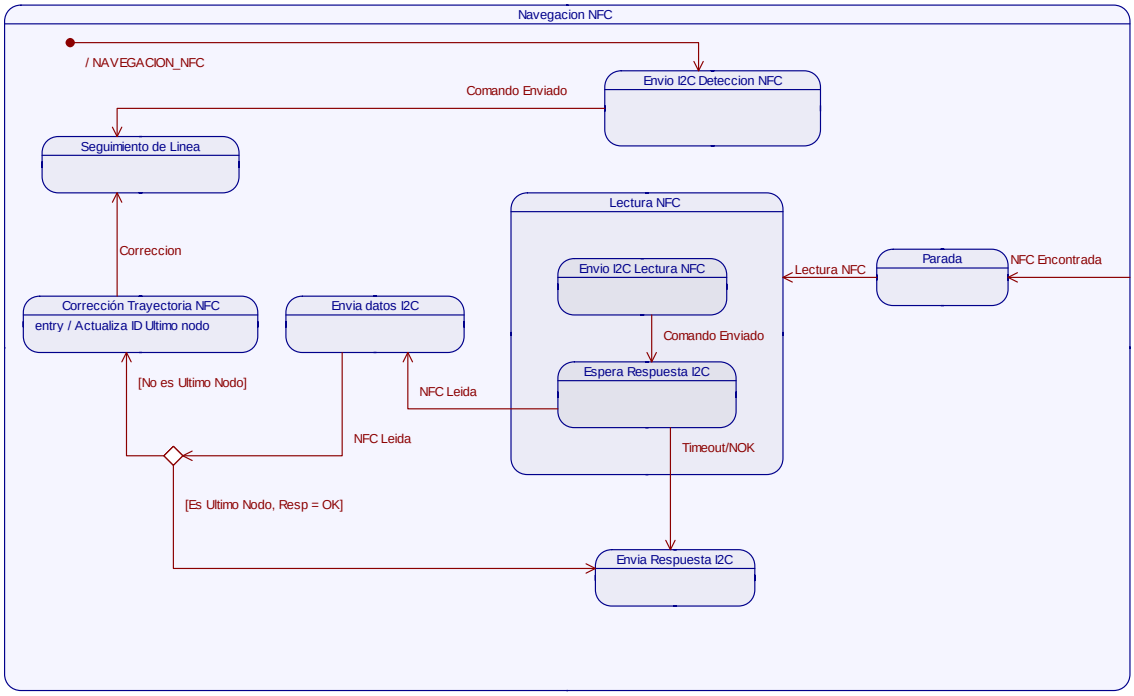


Figura 12: Diagrama de Estado. Navegación NFC

El comportamiento es muy similar al de **Grabación Trayectoria** salvo que en vez de enviar el comando de grabación se envía un comando de Lectura NFC por I2C a la tarjeta asociada al dispositivo NFC. De igual forma una vez leídos los datos de una etiqueta se enviarán por I2C a la tarjeta Bluetooth para su envío al PC.

Navegación Guiada

La tarjeta pasará a este modo al recibir el comando NAVEGACION_GUIADA. El comando deberá incluir la trayectoria a seguir.

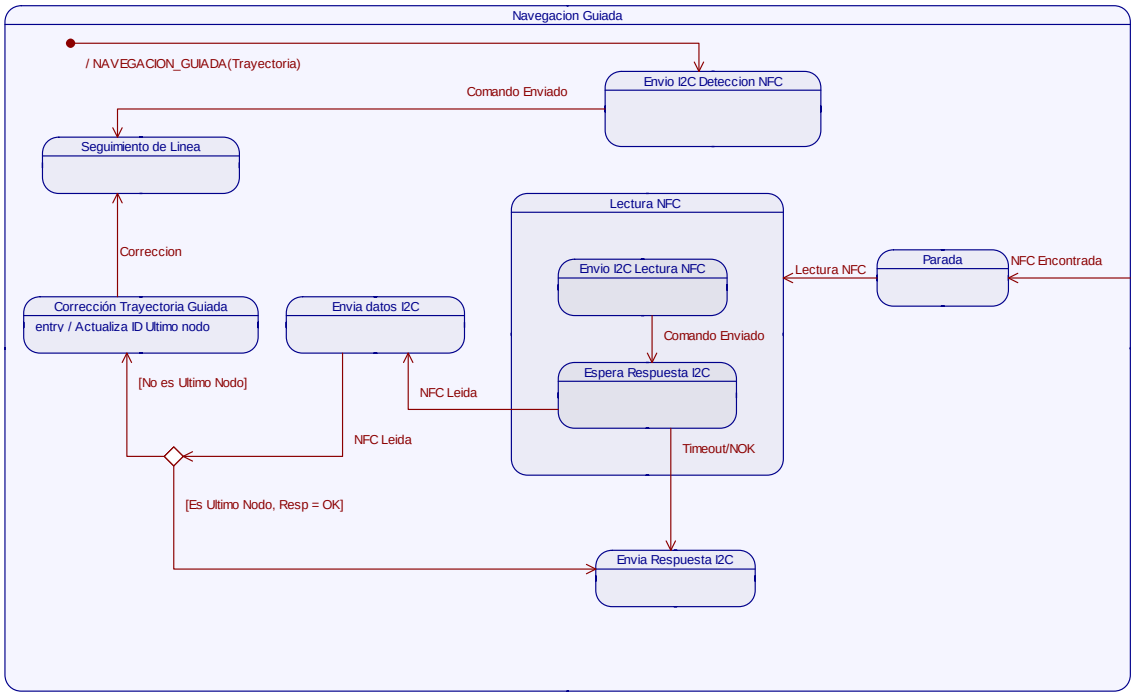


Figura 13: Diagrama de Estado. Navegación Guiada

El comportamiento es muy similar al de **Navegación NFC** con una diferencia en la corrección de la trayectoria, que se realizará en función de la trayectoria recibida y no del contenido de las etiquetas NFC.

Lectura Posición Actual

La tarjeta pasará a este modo al recibir el comando POSICION_ACTUAL.

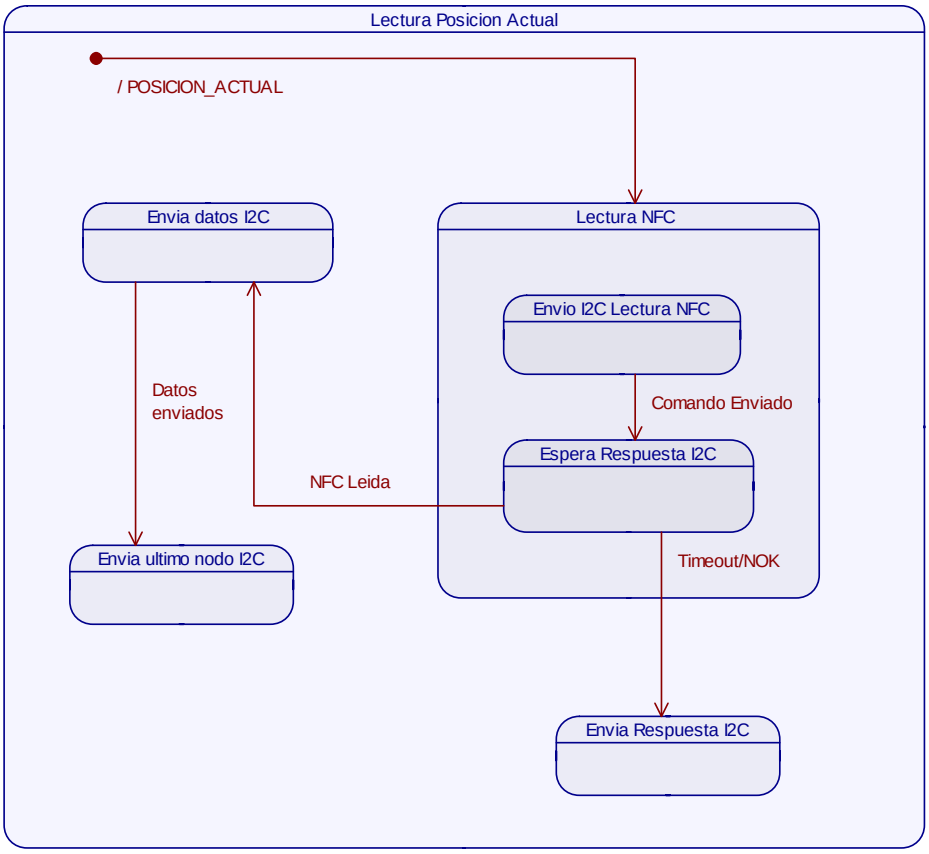


Figura 14: Diagrama de Estado. Lectura Posición Actual

La operación realizada es un intento de lectura NFC, posteriormente se enviará el resultado leído por I2C y el valor del último nodo visitado. En caso de fallo en la lectura no se enviará este dato, solo un NOK.

Tarjeta de lectura/escritura NFC

Esta tarjeta se encargará de gestionar el dispositivo lector/grabador NFC en función de las órdenes recibidas desde la tarjeta de control.

El funcionamiento de la tarjeta se resume en el siguiente diagrama de estado:

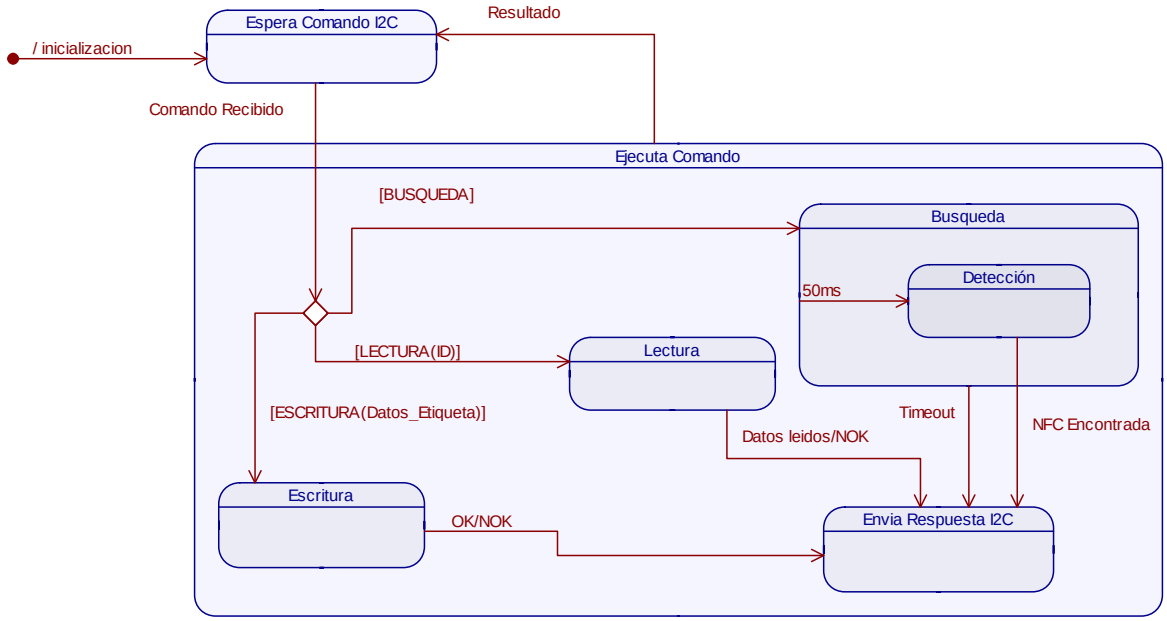


Figura 15: Diagrama de Estado. Tarjeta de lectura/escritura NFC

Inicialización

En el proceso de inicialización se realizarán las siguientes acciones:

- Configuración de las comunicaciones I2C.
- Configuración del dispositivo lector/grabador NFC.

Espera Comando I2C

El estado principal en el que se encontrará la tarjeta será la espera de la llegada de algún comando I2C de la tarjeta de control. En este estado no se realizará ninguna interacción con el lector/grabador NFC. Si se produce la llegada de un comando se pasará al estado **Ejecuta Comando**

Ejecuta Comando

En este estado se parseará el comando recibido. Si el comando se encuentra entre los comandos soportados se realizará la transición al estado asociado. Una vez finalizada la ejecución se retornará al estado **Espera Comando I2C**. En caso de que no se reconozca el comando se ignorará y se retornará al estado de **Espera Comando I2C**.

Los subestados que engloban este estado se indican a continuación.

Búsqueda

Si el comando recibido por I2C es BUSQUEDA se pasará al modo de búsqueda en el cual cada 50ms se solicita un comando de detección al dispositivo NFC. Cuando el dispositivo

indique que ha localizado una etiqueta en su radio de antena se finalizará la ejecución y se enviará por I2C el identificador de la etiqueta encontrada.

Lectura

Si el comando recibido por I2C es LECTURA se solicitará un comando de lectura al dispositivo NFC. Se enviará por I2C a la tarjeta control los datos leídos o en caso de error un NOK.

Escritura

Si el comando recibido por I2C es LECTURA se solicitará un comando de escritura al dispositivo NFC indicando los datos recibidos a escribir. Si la operación de escritura se realiza correctamente se enviará por I2C a la tarjeta de control un OK en caso de error un NOK

Tarjeta de Bluetooth

Esta tarjeta funcionará como puente entre el PC y el robot. Su labor principal será el envío de los comandos recibidos de la tarjeta de control por I2C a Bluetooth y viceversa.

El funcionamiento de la tarjeta se resume en el siguiente diagrama de estado:

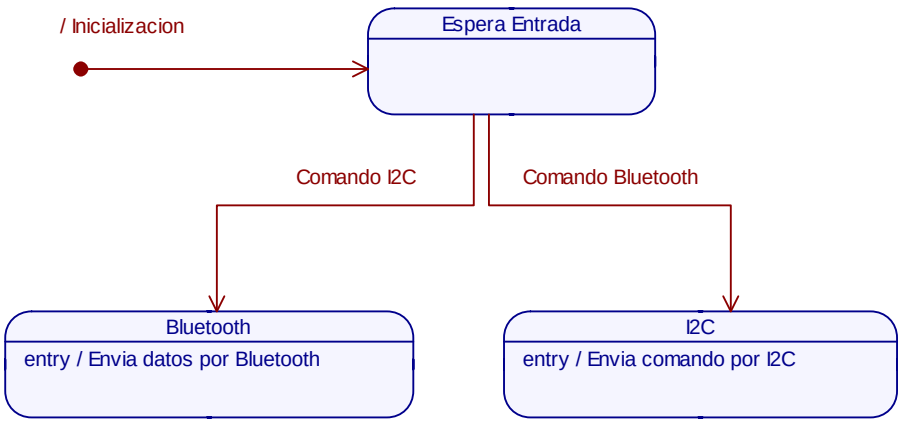


Figura 16: Diagrama de Estado. Tarjeta de Bluetooth

Optimización y adaptación

Se han mostrado los diagrama de estado preliminares que suponen una primera aproximación al comportamiento deseado del robot. A la hora de llevar estos diagramas a la práctica es necesario considerar las particularidades de los dispositivos físicos concretos que se utilizarán así como las características de las comunicaciones utilizadas. Es el momento también para

intentar optimizar el diseño intentando buscar un comportamiento más óptimo, reducción de los datos intercambiados etc.

Dos de los elementos que más condicionarán el diseño considerando la funcionalidad que se desea soportar son el lector CPR40.30 y las comunicaciones. A continuación mostramos las características de estos elementos que podrían influir en el diseño y las decisiones de cambio de diseño tomadas.

Lector CPR40.30

Uno de los dispositivos que más condicionará el comportamiento del robot será el dispositivo lector/grabador de etiquetas NFC. El lector comercial que se utilizará será el modelo CPR40.30 de OBID. El funcionamiento de la tarjeta que se comunica con este lector y las comunicaciones relacionadas deberán adaptarse a las condiciones particulares y restricciones de este dispositivo. En este apartado mostraremos las peculiaridades de este lector y las implicaciones en el funcionamiento del robot. Para más información del lector consultar el Anexo 1.

El lector CPR40.30 dispone de tres modos para el intercambio de datos entre las etiquetas y el dispositivo (tarjeta arduino) que se comunica con el lector. Estos modos condicionarán el comportamiento de la tarjeta. La comunicación solo es efectiva mientras la etiqueta permanece dentro del rango de detección del lector, este aspecto es especialmente importante en el caso de una operación de escritura, en la que la etiqueta deberá permanecer dentro del rango de detección durante todo el proceso. Si la etiqueta sale de este rango durante el proceso se producirá una pérdida de datos.

Modo direccionamiento

En este modo es necesario conocer el identificador de la etiqueta antes de realizar una operación de lectura o escritura. Para ello es necesario utilizar previamente el comando "[0x01] Inventory" para obtener dicho identificador. Con este comando si una o varias etiquetas se encuentran dentro del rango de detección en ese momento, responderá con sus identificadores Las operaciones siguientes de lectura y escritura deberán realizarse indicando el identificador de la etiqueta deseada.

Modo selección

En este modo el lector solo se comunica con la etiqueta que haya seleccionado. Antes de realizar una operación de lectura/escritura es necesario ejecutar el comando "[0x01] Inventory" para obtener el UID y posteriormente seleccionar la etiqueta con el comando "[0x25] Select" indicando la UID deseada. Posteriormente todas las operaciones de lectura/escritura solo se realizarán con esa etiqueta sin necesidad de especificar el identificador.

Modo escaneo

En este modo en cuanto el lector detecta una etiqueta dentro de su radio de detección envía datos al dispositivo al que esté conectado. El lector puede configurarse en este modo

modificando los parámetros del registro CFG1. Este modo presenta una restricción considerable y es que no es posible utilizarlo para operaciones de escritura, solo es válido para lectura.

El lector solo volverá a enviar los datos de una etiqueta cuando esta salga de su rango de acción y vuelva a entrar.

Consideraciones de diseño

Considerando los modos de funcionamiento disponibles, las características del circuito físico y buscando optimizar los tiempos de respuesta se utilizarán los modos direccionamiento y modo escaneo, las razones por las que se decide estos modos son las siguientes:

- **Modo escaneo**: Este modo será adecuado para la navegación guiada y la navegación NFC en las que la respuesta será inmediata por parte del lector y facilitará que la navegación sea más fluida. Como este modo implica cambio de configuración será necesario modificar la configuración del lector al recibir un comando y desactivar el modo de escaneo modificando de nuevo la configuración al finalizar el recorrido de la trayectoria NFC o Guiada. El número de operaciones a realizar con el lector se reducirá considerablemente al no tener que enviar operaciones de lectura cada vez que se pase por una etiqueta. Además si el robot permanece un tiempo encima de una etiqueta (p.ej. durante un giro en la corrección de la trayectoria) no recibirá datos duplicados ya que el lector no enviará de nuevo los datos mientras la etiqueta permanezca en el rango de detección (este apartado será necesario verificarlo experimentalmente y realizar las correcciones oportunas).

- **Modo direccionamiento**: Este modo será adecuado para la solicitud de posición y para la operación de escritura.

    - Solicitud de posición. El modo de direccionamiento es más adecuado que el de escaneo porque el número de comandos a intercambiar es el mismo, dos. En el caso del modo escaneo uno para activarlo y otro para desactivarlo (cambios de configuración) y en el caso de direccionamiento uno para localizar la etiqueta (UID) y otro para la lectura. La desventaja del modo escaneo es que en una hipotética situación en la que el robot parado no tenga ninguna etiqueta dentro del radio de acción no obtendríamos respuesta salvo que se estableciese un timeout. Sin embargo en el caso del modo direccionamiento la respuesta sería inmediata por lo que sería factible detectar esa situación.

    - Operación de escritura. El modo escaneo es inviable para las operaciones de escritura por restricción del lector CPR40.30.

- **Modo selección:** Este modo se descarta ya que por las características del circuito físico las etiquetas están suficientemente separadas para que en un comando de direccionamiento solo se obtenga una única etiqueta y por tanto no es necesario una nueva operación de selección.

También se aprecia que debido a las capacidades del lector podría asumir más carga de trabajo liberando a la tarjeta de control.

Comunicaciones

Las comunicaciones que soporta el robot son las siguientes:

I2C

En la comunicación I2C un dispositivo maestro puede realizar la transmisión de datos a un dispositivo esclavo o solicitar datos a dicho dispositivo. El inicio por tanto de las comunicaciones vendría determinado siempre por el dispositivo maestro. Sería por tanto necesario decidir qué dispositivo actuará como maestro:

- **Tarjeta de Navegación**: En nuestro caso particular el primer enfoque lógico podría consistir en considerar la tarjeta de navegación como Maestro ya que focaliza la mayor parte de las funciones de control y establecer las tarjetas NFC y Bluetooth como esclavas.

- **Tarjeta Bluetooth**. En el enfoque anterior la tarjeta de Navegación debería estar solicitando constantemente datos a la tarjeta Bluetooth para verificar la presencia de comandos. Una alternativa consistiría en que el maestro fuese la tarjeta Bluetooth y trasladarle las funciones de control ya que es en ella donde se inician los comandos al recibirlos por bluetooth.

- **Multimaster**: En el segundo enfoque sería necesaria más carga de comunicación I2C por cuanto sería necesario controlar desde la tarjeta bluetooth los movimientos del robot a través de la tarjeta de navegación. Una alternativa consistiría en un entorno multimaster en el que cualquiera de las tres tarjetas pudiese en un momento determinado actuar como master. De esta forma a lo largo del ciclo de ejecución de un comando cada tarjeta iría adquiriendo el rol de maestro/esclavo según las necesidades.

Aunque cualquiera de los tres enfoques sería válido, consideraremos el enfoque multimaster por ser el más flexible.

Bluetooth

La comunicación bluetooth con el PC no implica ningún cambio en el diseño inicial. La placa arduino BT maneja la comunicación bluetooth con la librería serie y la única consideración importante a tener en cuenta es la inicialización de la línea para emparejar correctamente el PC con la placa.

Comunicación con sensores/actuadores

La comunicación de las placas con los sensores/actuadores (sensor de infrarojos, lector CPR40, motores …) con los que están conectados no implica ningún cambio en el diseño inicial.

Modificación de diagramas de estado

Con las consideraciones vistas será necesario realizar algunas adaptaciones que se comentan a continuación comenzando por la Tarjeta de lectura/escritura NFC que implicará cambios en el resto de tarjetas.

Tarjeta de lectura/escritura NFC

En esta tarjeta es donde se originarán los cambios que repercutirán en el resto, ya que será necesario considerar las peculiaridades de los comandos soportados por el dispositivo CPR40.30 para adaptarse a los modos de funcionamiento adecuados para cada situación. El nuevo diagrama de estado sería el siguiente:

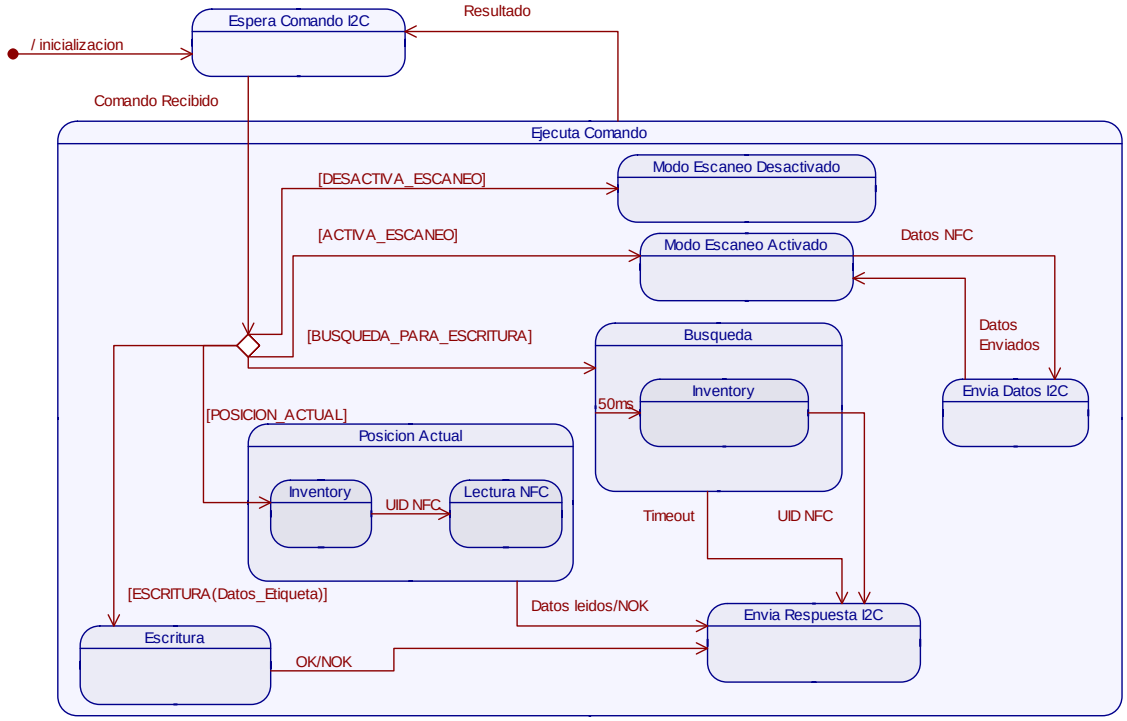


Figura 17: Diagrama de Estado Modificado. Tarjeta de lectura/escritura NFC

Como se puede apreciar se incluye el soporte para la activación del modo Escaneo propio del dispositivo utilizado. La tarjeta por tanto aceptará un comando para su activación ACTIVA_ESCANEO y otro para su desactivación DESACTIVA_ESCANEO. En el estado de escaneo cada vez que se detecte una etiqueta se enviarán los datos por I2C tanto a la tarjeta de bluetooth para que los envíe al PC como a la tarjeta de control para que realice las actualizaciones oportunas. El comando de BUSQUEDA se restringe únicamente al caso de la escritura BUSQUEDA_PARA_ESCRITURA. En este caso el comportamiento será muy parecido al imaginado inicialmente en el que cada 50ms se realiza una solicitud de chequeo de presencia de etiqueta NFC, que para el caso particular del dispositivo CPR40 se traduce en la solicitud del comando Inventory. Si la ejecución de este comando devuelve un identificador

de etiqueta se devolverá dicho identificador por I2C a la tarjeta de control. Se mantendrá un timeout para evitar la demora excesiva del comando. Se soportará también un nuevo comando POSICION_ACTUAL que realizará las dos operaciones comentadas para la lectura, un Inventory y una Lectura NFC. El comportamiento de la escritura no varía.

Tarjeta de Navegación

La tarjeta de Navegación seguirá soportando los mismos comandos contemplados inicialmente, pero la ejecución de alguno de ellos variará simplificándose en la mayoría de los casos. Esta simplificación puede apreciarse por ejemplo en el comportamiento del estado Navegación NFC.

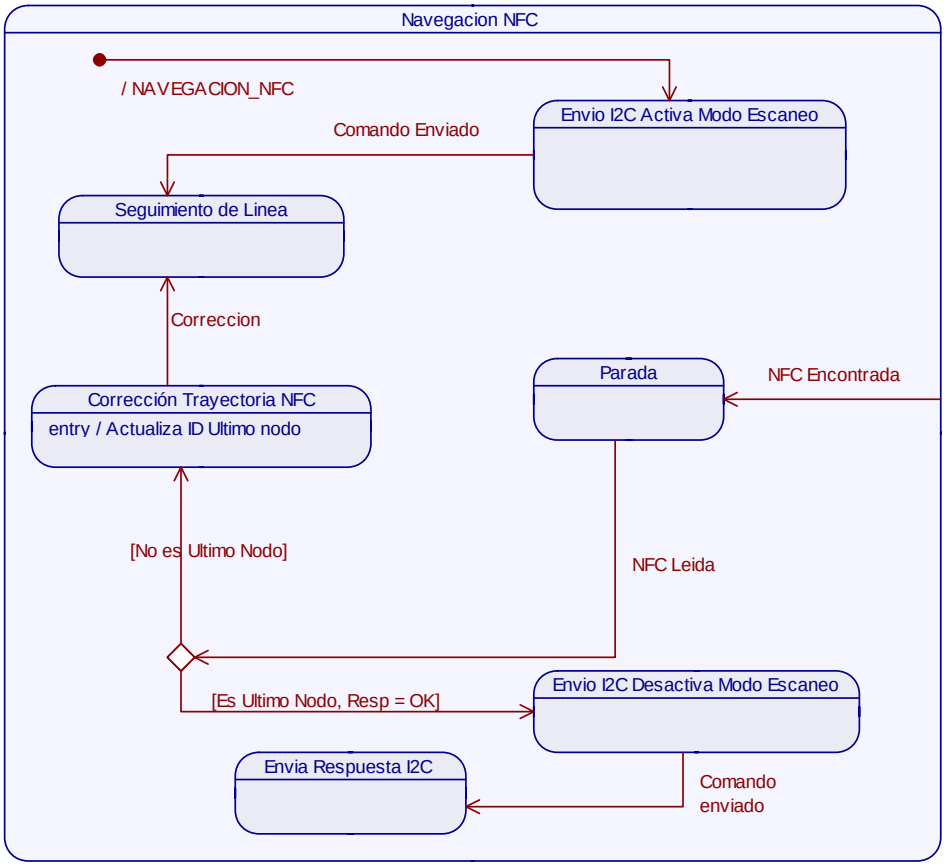


Figura 18: Diagrama de Estado Modificado . Navegación NFC

La mayor simplificación radica en la supresión de la solicitud de las operaciones de lectura y el envío de los datos por I2C ya que de esto se encarga directamente la tarjeta NFC. Cuando esta tarjeta envíe el último nodo la tarjeta de control le enviará el comando para desactivar el modo de escaneo.

Estos mismos cambios son aplicables al estado **Navegación Guiada**. Respecto al estado **Grabación Trayectoría** solo se modificará el subestado **Envio I2C Detección NFC** que pasará a ser el estado **Envío I2C Búsqueda para Escritura** que enviará un comando I2C BUSQUEDA_PARA_ESCRITURA a la tarjeta NFC.

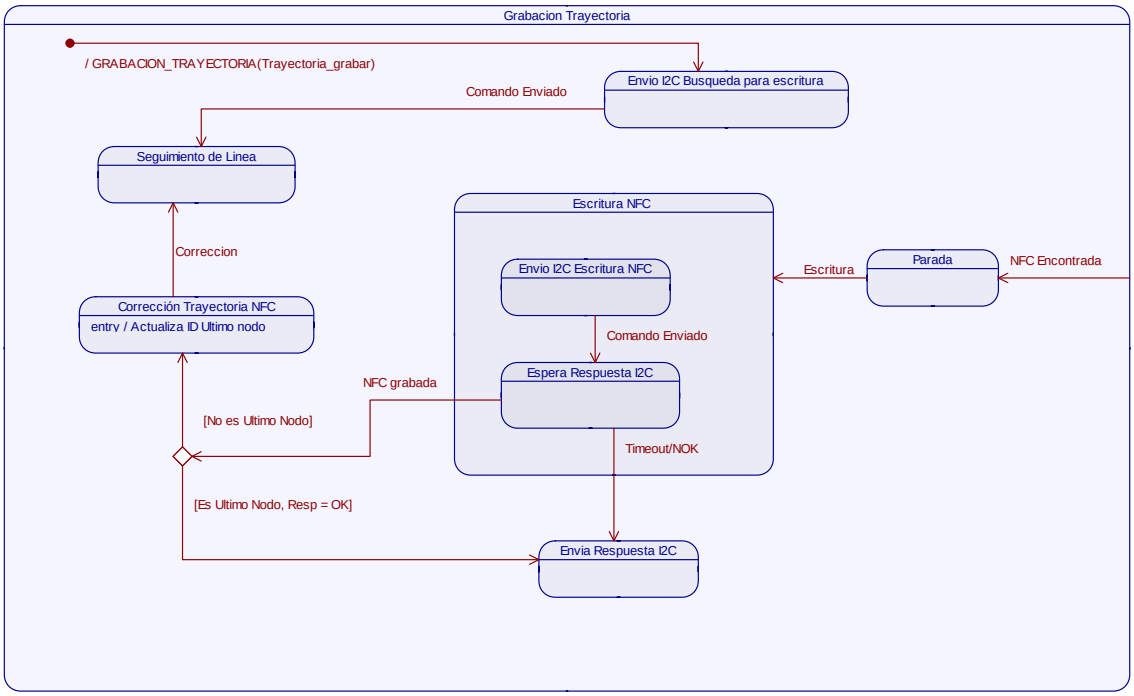


Figura 19: Diagrama de Estado Modificado. Grabación Trayectoria

Respecto al estado **Lectura Posicion Actual** también se adaptará para indicar que se debe enviar un comando POSICION_ACTUAL a la tarjeta NFC en vez de una petición directa de Lectura.

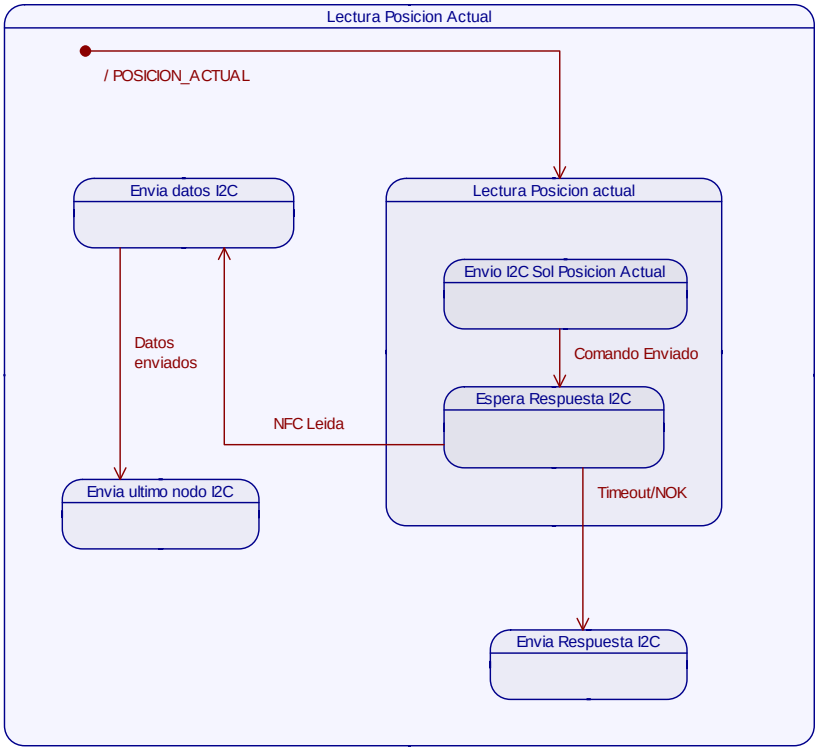


Figura 20: Diagrama de Estado Modificado. Lectura Posición Actual

Tarjeta Bluetooth

Los cambios en la tarjeta bluetooth no son significativos, seguirá funcionando como un puente entre comandos recibidos I2C y bluetooth pero en este caso podrá recibir también datos I2C de la tarjeta NFC.

Formato mensajes

Una vez analizados los diagramas de estado se concretará el formato de los mensajes a intercambiar por Bluetooth con el PC y por I2C entre las placas del robot.

Comunicación Bluetooth

La comunicación bluetooth se realizará entre el PC y la tarjeta arduíno BT. El formato de los comandos y respuestas es el siguiente:

(Ver apartado de Interfaz de Usuario para ejemplos de diálogos entre PC y Robot)

Comando Solicitud de Posición

Este comando se enviará desde el PC para solicitar al robot la posición en la que se encuentra. Desencadenará la lectura de la etiqueta NFC sobre la que se encuentre el robot a partir de cuya información se obtendrá el nodo en el que se encuentra posicionado y por tanto la posición del robot. El formato será

Comando Solicitud Posición	
Campo	Descripción
Comando	Orden para solicitar la posición actual: **POSICION_ACTUAL**
LFCR	Final de línea y salto de carro

Figura 21: Bluetooth. Formato comando solicitud posición

Respuesta Comando Solicitud de Posición

El robot al recibir el comando de posición actual ejecutará una operación de lectura y devolverá el resultado de la lectura, así como el nodo previo en el que se encontraba, este dato permitirá conocer la orientación del robot. Los dos valores se enviarán en líneas distintas.

Respuesta Comando Solicitud Posición Correcta	
Campo	Descripción
Identificador	Identificador de la Etiqueta. (p.ej. ***1***)
|	Carácter Separador.
Nombre	Nombre de la etiqueta. (p.ej. ***NFC01***)

|	Carácter Separador.
posX	Coordenada X de la posición absoluta de la etiqueta sobre el tablero. (p.ej. ***40***)
|	Carácter Separador.
posY	Coordenada Y de la posición absoluta de la etiqueta sobre el tablero. (p.ej.***50***)
|	Carácter Separador.
Tipo	Indica el número de segmentos que conecta el nodo. (p.ej.***2***)
|	Carácter Separador.
Vector de Entradas	Valores de los nodos conectados como entrada al nodo separados por comas. (p.ej. ***2,3***)
|	Carácter Separador.
Vector de Salidas	Valores separados por comas de las acciones a realizar en cada salida asociada a cada entrada conectada. Los posibles desplazamientos serían izquierda (I), derecha(D), recto (R) o pararse (0) (p.ej ***0,0***). En este caso se indicaría que si el robot viene de la entrada 2 o de la entrada 3 debe pararse.
|	Carácter Separador.
LFCR	Final de línea y salto de carro
Nodo previo	Identificador de la Etiqueta del nodo previo. (p.ej. ***3***)
LFCR	Final de línea y salto de carro

Figura 22: Bluetooth. Formato respuesta solicitud posición correcta

En caso de error el dato devuelto será

Respuesta Comando solicitud Posición Fallido	
Campo	Descripción
Indicador de error	**NOK**
LFCR	Final de línea y salto de carro

Figura 23: Bluetooth. Formato respuesta solicitud posición fallido

Comando Navegación Guiada

Este comando se enviará desde el PC para indicar al robot el recorrido de una trayectoria ignorando el contenido de las etiquetas NFC del tablero y siguiendo las indicaciones de la trayectoria incluida en el comando. El formato será:

Comando Navegación Guiada	
Campo	Descripción
Comando	Orden para solicitar la Navegación Guiada: **NAVEGACION_GUIADA**
|	Carácter Separador.
Indicaciones nodo 1	Indicaciones de ruta del primer nodo de la trayectoria. El formato será el siguiente: • Identificador de la etiqueta • **:** Carácter separador • Vector de entradas. Nodos conectados como entrada separados por coma. • : Carácter separador • Vector de salidas. Acciones a realizar de salida asociada a cada entrada. Los valores separados por coma. p.ej. **1:2,3:0,R** este caso indica que en el nodo 1 al llegar por el nodo 2 el

	robot debe pararse pero sin embargo si viene por el nodo 3 deberá girar a la derecha.
|	Carácter Separador.
…	
|	Carácter Separador.
Indicaciones último nodo	Indicaciones de ruta del último nodo de la trayectoria. El valor lógico en el vector de salidas debería ser un 0 para todas indicando al robot la parada al llegar al final de la trayectoria.
|	Carácter Separador.
LFCR	Final de línea y salto de carro

Figura 24: Bluetooth. Formato comando navegación guiada

Respuesta Comando Navegación Guiada

La respuesta al comando de Navegación Guiada consistirá en el contenido de cada etiqueta y un OK en la finalización.

Respuesta Comando Navegación Guiada Correcta	
Campo	Descripción
Contenido Etiqueta nodo 1	Contenido de la primera etiqueta leída. El formato será el mismo que el devuelto en la respuesta para la posición actual sin incluir el nodo previo.
…	
Contenido Etiqueta último nodo	Contenido de la etiqueta del último nodo de la trayectoria. El formato será el mismo que el devuelto en la respuesta para la posición actual sin incluir el nodo previo.
Indicador de ejecución correcta	**OK**
LFRC	Final de línea y salto de carro.

Figura 25: Bluetooth. Formato respuesta navegación guiada correcta

En caso de error el dato devuelto será el siguiente, pudiéndose recibir durante la recepción de las lecturas realizadas:

Respuesta Comando Navegación Guiada Fallida	
Campo	Descripción
Indicador de error	**NOK**
LFCR	Final de línea y salto de carro

Figura 26: Bluetooth. Formato respuesta navegación guiada fallida

Comando Grabación de Trayectoria

Este comando ordenará al robot que debe realizar la grabación de la trayectoria indicada en el comando.

Comando Grabación Trayectoria	
Campo	Descripción
Comando	Orden para solicitar la grabación de trayectoria: **GRABACIÓN_TRAYECTORIA.**
|	Carácter Separador.
Indicaciones nodo 1	Indicaciones de ruta del primer nodo de la trayectoria. El formato será el mismo que el utilizado en las indicaciones para la navegación guiada.
|	Carácter Separador.
…	
|	Carácter Separador.
Indicaciones último nodo	Indicaciones de ruta del último nodo de la trayectoria. El valor lógico en el vector de salidas debería ser un 0 para todas indicando al robot la parada al llegar al final de la trayectoria.
|	Carácter Separador.
LFCR	Final de línea y salto de carro

Figura 27: Bluetooth. Formato comando grabación trayectoria

Respuesta Comando Grabación Trayectoria

La respuesta al comando de Grabación de Trayectoria consistirá en un OK si se ha realizado correctamente o un NOK en caso contrario. A diferencia de las respuestas para otros comandos no se envía el contenido de las etiquetas.

Respuesta Comando Grabación Trayectoria Correcta	
Campo	Descripción
Indicador de ejecución correcta	**OK**
LFCR	Final de línea y salto de carro

Figura 28: Bluetooth. Formato respuesta grabación trayectoria correcta

Respuesta Comando Grabación Trayectoria Fallida	
Campo	Descripción
Indicador de error	**NOK**
LFCR	Final de línea y salto de carro

Figura 29: Bluetooth. Formato respuesta grabación trayectoria fallida

Comando Navegación NFC

Este comando se enviará desde el PC para solicitar al robot que realice un recorrido siguiendo las indicaciones grabadas en las etiquetas NFC. El formato será:

Comando Navegación NFC	
Campo	Descripción
Comando	Orden para solicitar la navegación NFC: **NAVEGACION_NFC**
LFCR	Final de línea y salto de carro

Figura 30: Bluetooth. Formato comando navegación NFC

Respuesta Navegación NFC

El formato de la respuesta a la solicitud de Navegación NFC será el mismo que el de la respuesta a la solicitud de Navegación guiada.

Comunicación I2C

La comunicación I2C se realizará entre las tres tarjetas arduino que constituyen el robot. Como se ha visto en el apartado anterior se considerará un enfoque multimaster en el que la condición de maestro la adquirirá la tarjeta que inicie la comunicación.

Tal como se muestra en los diagramas de estado se enviarán datos de las etiquetas por I2C (desde la tarjeta NFC a la de Navegación y BT), el formato de los datos envíados es el mismo que el utilizado para transmitir el contenido de la posición actual. Además de estos datos se intercambiarán una serie de comandos que se mencionan a continuación:

Comando	Descripción
POSICION_ACTUAL	Mismo formato que el enviado por bluetooth. Se reenviará desde la tarjeta BT a la de control al recibir el comando por bluetooth. Se enviará también desde la tarjeta de control a la tarjeta NFC para solicitar la lectura de la posición actual.
GRABACION_TRAYECTORIA	Incluirá la trayectoria a grabar. Tiene el mismo formato que el enviado por bluetooth. Se reenviará desde la tarjeta BT a la tarjeta de control en el momento que se reciba por bluetooth
NAVEGACION_GUIADA	Incluirá la trayectoria a seguir. Mismo formato que el enviado por bluetooth. Se reenviará desde la tarjeta BT a la de control al recibir el comando por bluetooth.
NAVEGACION_NFC	Mismo formato que el enviado por bluetooth. Se reenviará desde la tarjeta BT a la de control al recibir el comando por bluetooth.
DESACTIVA_ESCANEO	Será enviado desde la tarjeta de control a la tarjeta NFC para que desactive el modo de escaneo.
ACTIVA_ESCANEO	Será enviado desde la tarjeta de control a la tarjeta NFC para que active el modo de escaneo.
BUSQUEDA_PARA_ESCRITURA	Será enviado desde la tarjeta de control a la tarjeta NFC para que realice una búsqueda de NFC mediante el modo de direccionamiento. La respuesta I2C a este comando será el identificador de la etiqueta encontrada.
OK	Respuesta a un comando realizado correctamente
NOK	Respuesta a un comando fallido.

Figura 31: Comandos I2C

Aplicación de Escritorio

En este apartado se hará una referencia a la aplicación de escritorio que se desarrollará para el control del robot.

La aplicación contará con una pequeña base de datos donde se almacenarán el histórico de las simulaciones. Además presentará un interfaz adecuado para poder visualizar las simulaciones en tiempo real y enviar comandos de operación al robot.

Modelo de Base de Datos

La base de datos contendrá las tablas necesarias para guardar las simulaciones realizadas de tal forma que puedan consultarse en cualquier momento.

El diseño de la base de datos sería el siguiente:

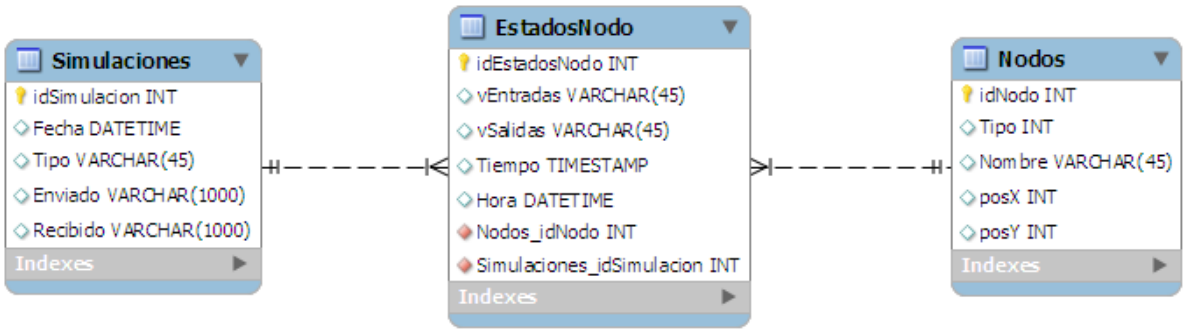


Figura 32: Tablas de Base de Datos

El contenido de cada tabla sería el siguiente:

Simulaciones	
idSimulacion	• INT • PRIMARY KEY • Identificador Interno
Fecha	• DATETIME • Fecha en la que se ha grabado la simulación
Tipo	• VARCHAR(45) • Contendrá el tipo de simulación realizada. Se corresponderá con los modos de operación del robot. Podrá contener alguno de los siguientes valores: ○ GRABACION_TRAYECTORIA. ○ NAVEGACION_NFC. ○ NAVEGACION_GUIADA. ○ SOLICITUD_POSICION.
Enviado	• VARCHAR(1000) • Contendrá los datos enviados desde el PC durante la simulación.

Recibido	• VARCHAR(1000) • Contendrá los datos enviados por el robot.

Figura 33: BBDD. Tabla Simulaciones

Nodos	
idNodo	• INT • PRIMARY KEY • Identificador Interno
Tipo	• INT • Indica el número de segmentos que conecta el nodo. Coincide con el tamaño de los vectores de entrada/salida.
Nombre	• VARCHAR(45) • Nombre descriptivo del nodo. Incluirá el número de nodo. (NFC01 p.ej.).
posX	• INT • Coordenada X de la posición absoluta de la etiqueta sobre el tablero.
posY	• INT • Coordenada Y de la posición absoluta de la etiqueta sobre el tablero.

Figura 34: BBDD. Tabla Nodos

EstadosNodo	
idEstadosNodo	• INT • PRIMARY KEY • Identificador Interno
vEntradas	• VARCHAR(45) • Vector de valores separados por comas. Identificador de los nodos conectados como entrada al nodo
vSalidas	• VARCHAR(45) • Vector de valores separados por comas. Acciones a realizar en cada salida asociada a cada entrada conectada. Los posibles desplazamientos serían izquierda (I), derecha(D), recto (R) o pararse (0)
Hora	• DATETIME • Registrará el momento en el que fue leído el contenido de la etiqueta asociada al nodo.
Nodos_idNodo	• INT • CLAVE AJENA. • Identificador del nodo al que se asocia la información.
Simulaciones_idSimulacion	• INT • CLAVE AJENA • Identificador de la Simulación al que está asociado el estado del nodo.

Figura 35: BBDD. Tabla Simulaciones

Interfaz de Usuario

Pantalla Principal

Al arrancar se mostrará la pantalla inicial que estará dividida en tres secciones:

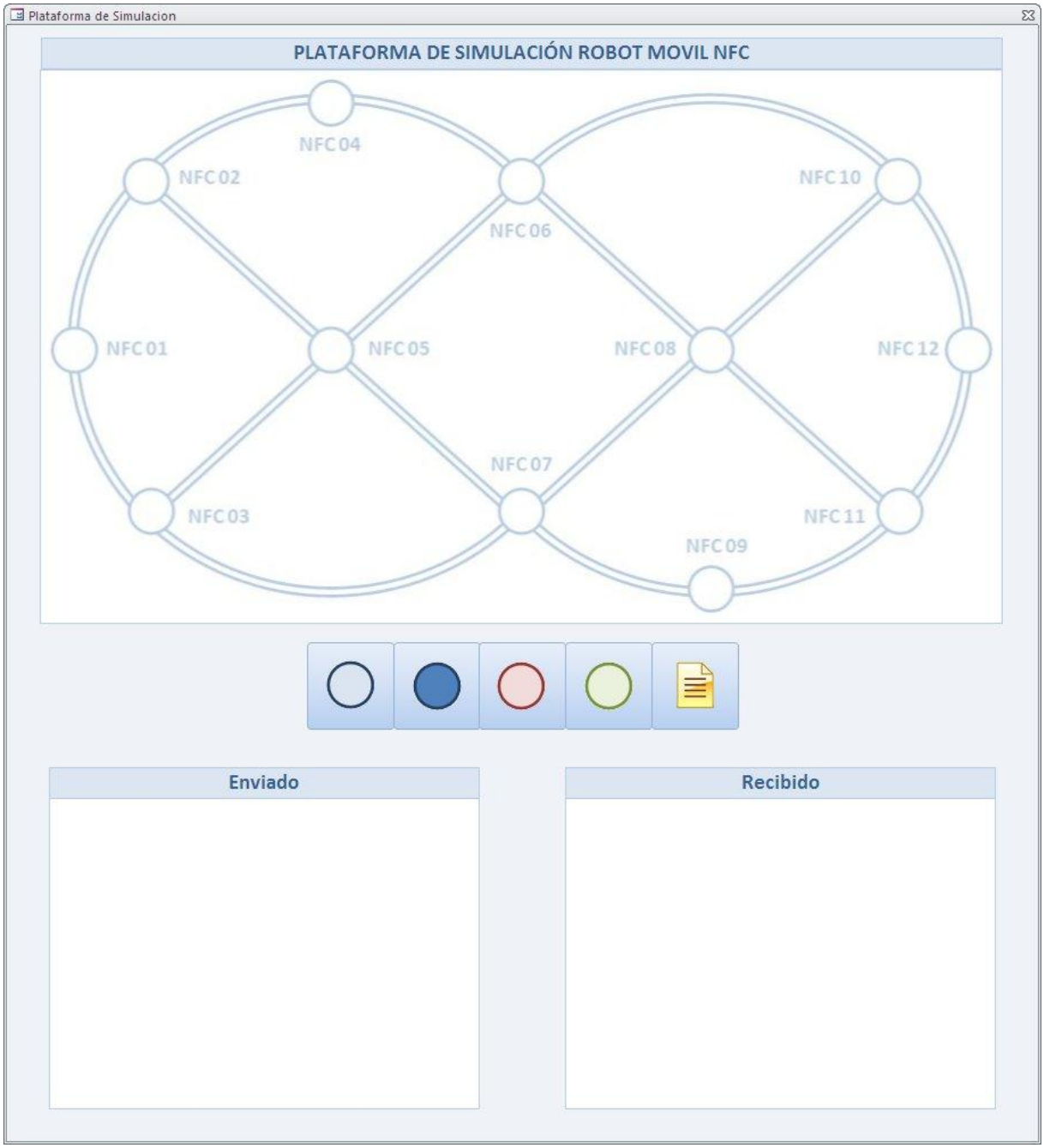


Figura 36: Interfaz de Usuario. Pantalla Principal

En la parte superior se mostrará una representación del circuito en la que se podrán visualizar las distintas simulaciones que se realicen.

En la parte central se mostrará una hilera de botones con las distintas acciones que se podrán llevar a cabo

Acciones Pantalla Principal	
	• Posición del Robot. • Permite resaltar en el circuito el nodo en el cual se encuentra el robot.
	• Navegación completa. • Envía al robot los comandos necesarios para que recorra todos los nodos del circuito mediante Navegación Guiada.
	• Grabación de Trayectoria. • Envía al robot los comandos necesarios para que grabe en las etiquetas adecuadas la trayectoria indicada.
	• Navegación por NFC. • Envía al robot los comandos necesarios para recorrer una trayectoria siguiendo las indicaciones grabadas en las etiquetas.
	• Histórico. • Se mostrará el histórico de simulaciones realizadas.

Figura 37: Tabla Simulaciones

En la parte inferior se mostrarán dos cuadros de texto que irán presentando los mensajes intercambiados por bluetooth entre el PC y el robot.

Posición del Robot

Mediante esta acción se podrá visualizar la posición actual del robot que se traducirá en el nodo en el cual se encuentra actualmente. Para ello será necesario enviar el comando de lectura de etiqueta al robot. El robot al recibir este comando intentará leer el contenido de la etiqueta en la que se encuentra y devolverá dicho contenido o un NOK en caso de fallo. Con la información leída la aplicación resaltará el nodo en el cual se encuentre el robot.

El diálogo intercambiado sería el siguiente:

PC->Robot

```
POSICION_ACTUAL
```

Robot->PC

```
1|NFC01|40|50|2|2,3|0,0|
```

```
3
```

Con esta información se resaltará el nodo 1 donde se encuentra la etiqueta NFC01.

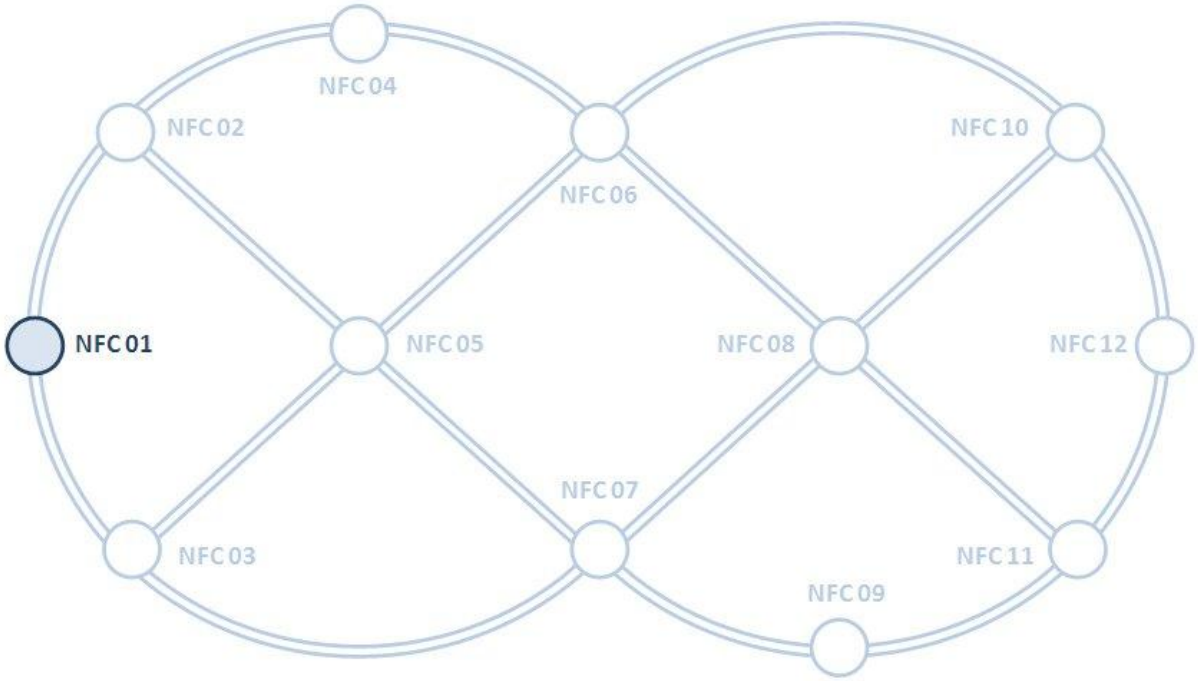


Figura 38: Interfaz de Usuario. Posición del Robot

Recorrido Completo

Con esta acción se indicará al robot que recorra todo el circuito pasando al menos una vez por cada nodo. Para ello será necesario enviar la secuencia de comandos adecuada que consistirá en el comando de recorrido completo más la ruta que deberá seguir el robot. La elaboración de la ruta dependerá del nodo en el que se encuentre actualmente el robot por lo que el primer paso consistirá en obtener dicho nodo.

En esta navegación es la aplicación la que indica al robot la trayectoria a seguir, el robot por tanto ignorará el contenido de las etiquetas para decidir qué camino seguir. Cada vez que pase por una etiqueta enviará su contenido y si completa la trayectoria enviará un OK. En caso de algún problema en la navegación enviará un NOK .

El dialogo intercambiado sería el siguiente:

PC->Robot: Se solicita el nodo actual

```
POSICION_ACTUAL
```

Robot->PC: Envía el contenido de la etiqueta sobre la que se encuentra posicionado y el nodo del que viene (en la inicialización se considerará el nodo 3) . En caso de error se recibirá un NOK.

```
1|NFC01|40|50|2|2,3|0,0|
3
```

PC->Robot: Se solicita el recorrido completo indicando al robot la ruta que debe seguir. Es en este punto donde la aplicación utilizará los algoritmos adecuados para elaborar dicha ruta partiendo del nodo inicial.

```
NAVEGACION_GUIADA|1:2,3:0,R|2:1,4,5:R,0,0|4:2,6:R,,0|6:4,5,8,10:D,0,0,0|5:2,3,6,7:
0,0,R,0|3:1,5,7:0,I,0|7:3,5,8,9:R,0,0,0|8:6,7,10,11:0,R,0,0|10:6,8,12:0,D,0|12:10,11:R,0|11:8,9,12:0,0,R
|9:7,11:0,0|
```

Robot->PC: Al ir recorriendo las distintas etiquetas irá enviando su contenido. Al alcanzar el nodo final enviará un OK, en caso de ocurrir cualquier problema se enviará un NOK.

```
1|NFC01|40|50|2|2,3|0,0|
2|NFC02|49|73|3|1,4,5|0,0,0|
4|NFC04|75|85|2|2,6|0,0,0|
6|NFC06|100|75|4|4,5,8,10|0,0,0,0|
5|NFC05|75|50|4|2,3,6,7|0,0,0,0|
3|NFC03|49|27|3|1,5,7|0,0,0|
7|NFC07|100|25|4|3,5,8,9|0,0,0,0|
8|NFC08|127|50|4|6,7,10,11|0,0,0,0|
10|NFC10|151|73|3|6,8,12|0,0,0|
12|NFC12|161|50|2|10,11|0,0|
11|NFC11|151|27|3|8,9,10,12|0,0,0,0|
9|NFC09|127|15|2|7,8,11|0,0,0|
OK
```

La traducción visual correspondería a la siguiente figura:

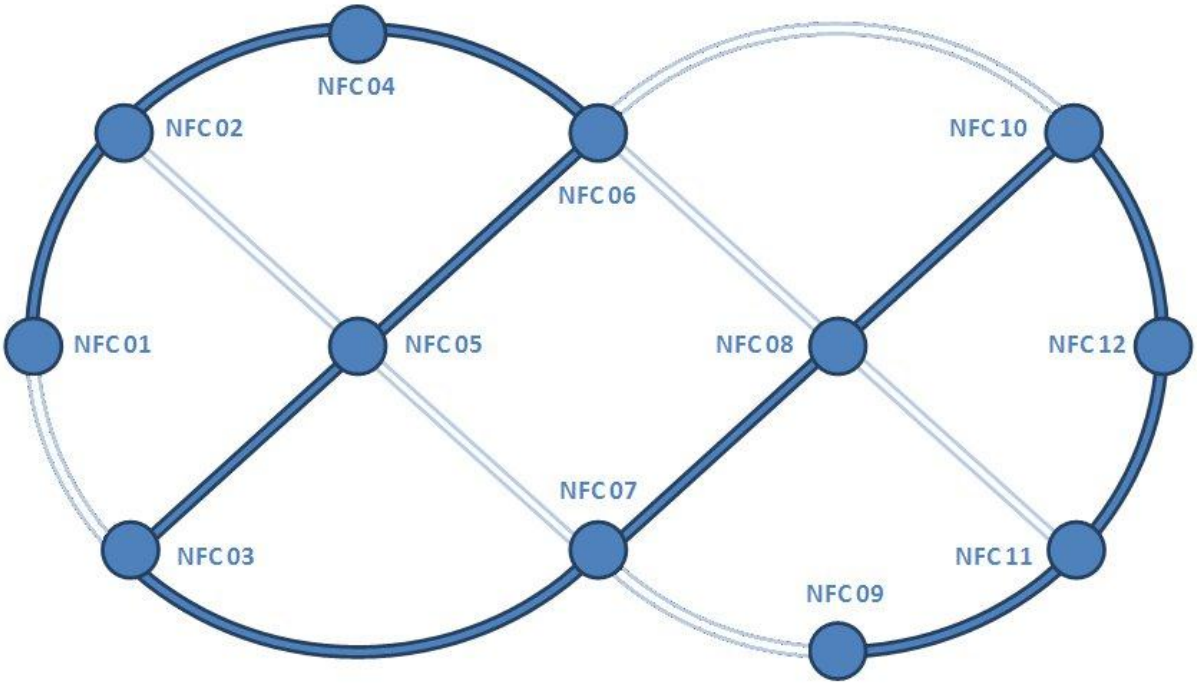


Figura 39: Interfaz de Usuario. Recorrido Completo

En este estado al solicitar la posición actual del robot nos debería mostrar el nodo 9.

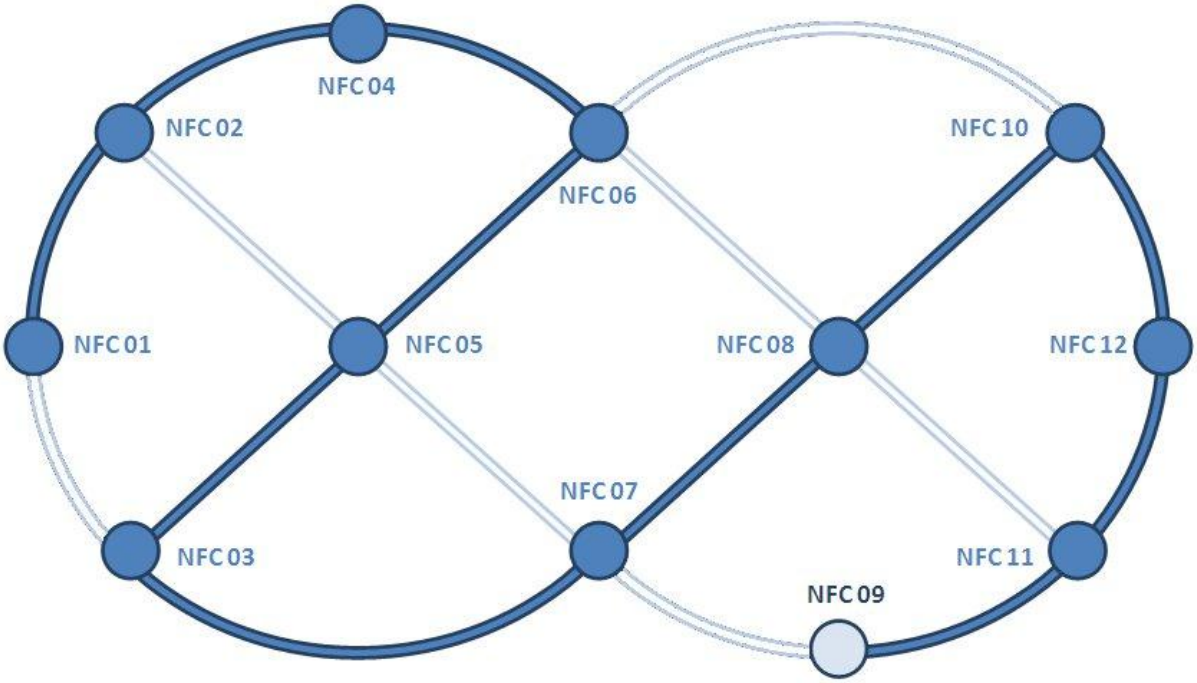


Figura 40: Interfaz de Usuario. Recorrido Completo con Posición actual del robot

Grabación de Trayectoria

Mediante esta acción se podrá grabar contenido en las etiquetas NFC del circuito. Para generar los comandos necesarios se presentará un asistente visual que irá solicitando el contenido de los nodos que compondrán la trayectoria, indicando el nodo inicial, los nodos intermedios y el nodo final. La aplicación generará los comandos necesarios para que el robot recorra la trayectoria indicada grabando los datos adecuados en las etiquetas NFC. El primer paso consistirá en saber la posición actual del robot para preparar la ruta.

El dialogo entre el robot y la aplicación sería el siguiente:

PC->Robot: Se solicita el nodo actual

```
POSICION_ACTUAL
```

Robot->PC: Envía el contenido de la etiqueta sobre la que se encuentra posicionado. En caso de error se recibirá un NOK.

```
9|NFC09|127|15|2|7,8,11|0,0,0|
11
```

PC->Robot: Se solicita al robot la ruta que debe seguir. Es en este punto donde la aplicación utilizará los algoritmos adecuados para elaborar dicha ruta partiendo del nodo inicial. Como el nodo actual es el 9 y el comienzo de la ruta a grabar es el 8, podría considerarse como ruta la 9, 11, 8. Pero como el robot no puede realizar giros de 180º la ruta a realizar tendrá que incluir el nodo 7. Una ruta válida sería la 9, 7, 5, 6, 8.

```
NAVEGACION_GUIADA|9:7,11:0,R|7:3,5,8,9:0,0,0,R|5:3,2,6,7:0,0,0,I|3:1,5,7:0,D,0|1:3,2:0,0|
```

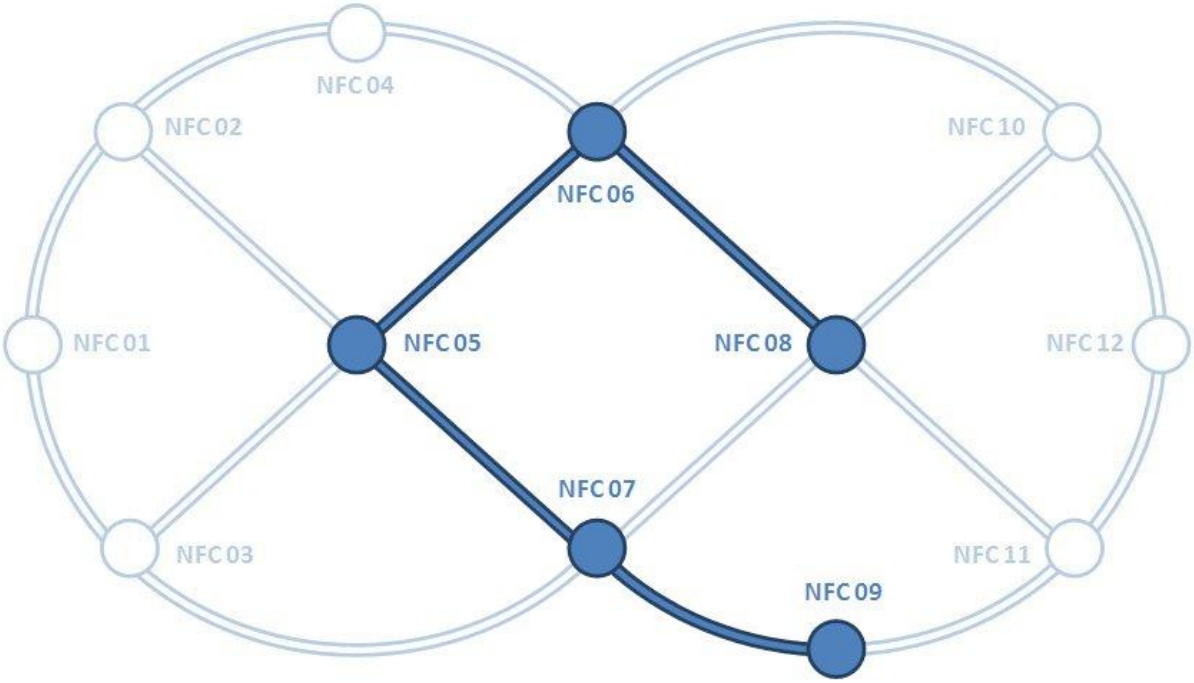


Figura 41: Interfaz de Usuario. Grabación de Trayectoria. Movimiento al nodo inicial

Robot->PC: Al ir recorriendo las distintas etiquetas irá enviando su contenido. Al alcanzar el nodo final enviará un OK, en caso de ocurrir cualquier problema se enviará un NOK.

```
9|NFC09|127|15|2|7,8,11|0,0,0|
7|NFC07|100|25|4|3,5,8,9|0,0,0,0|
5|NFC05|75|50|4|2,3,6,7|0,0,0,0|
3|NFC03|49|27|3|1,5,7|0,0,0|
1|NFC01|40|50|2|2,3|0,0|
OK
```

PC->Robot: Se solicita la grabación de la trayectoria pasada como parámetro.

```
GRABACION_TRAYECTORIA|8:6,7,10,11:D,0,R,I|7:3,5,8,9:0,0,D,R|5:2,3,6,7:0,0,R,I|3:1,5,7:0,D,R|1:2,3:0,0|
```

Robot->PC: Envía el resultado de la grabación

```
OK
```

La traducción visual correspondería a la siguiente figura en la que se aprecia en azul el recorrido previo que debe realizar el robot ignorando el contenido de las etiquetas para llegar al nodo inicial.

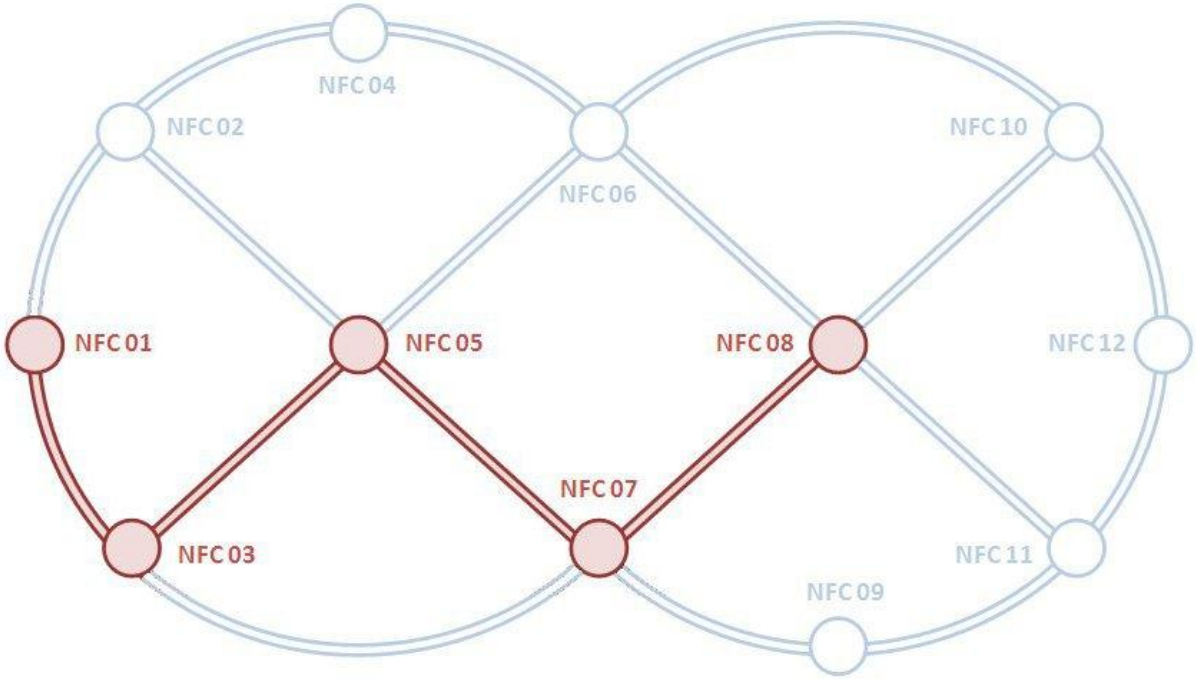


Figura 42: Interfaz de Usuario. Grabación de Trayectoria

En este estado al solicitar la posición actual del robot nos debería mostrar el nodo 1.

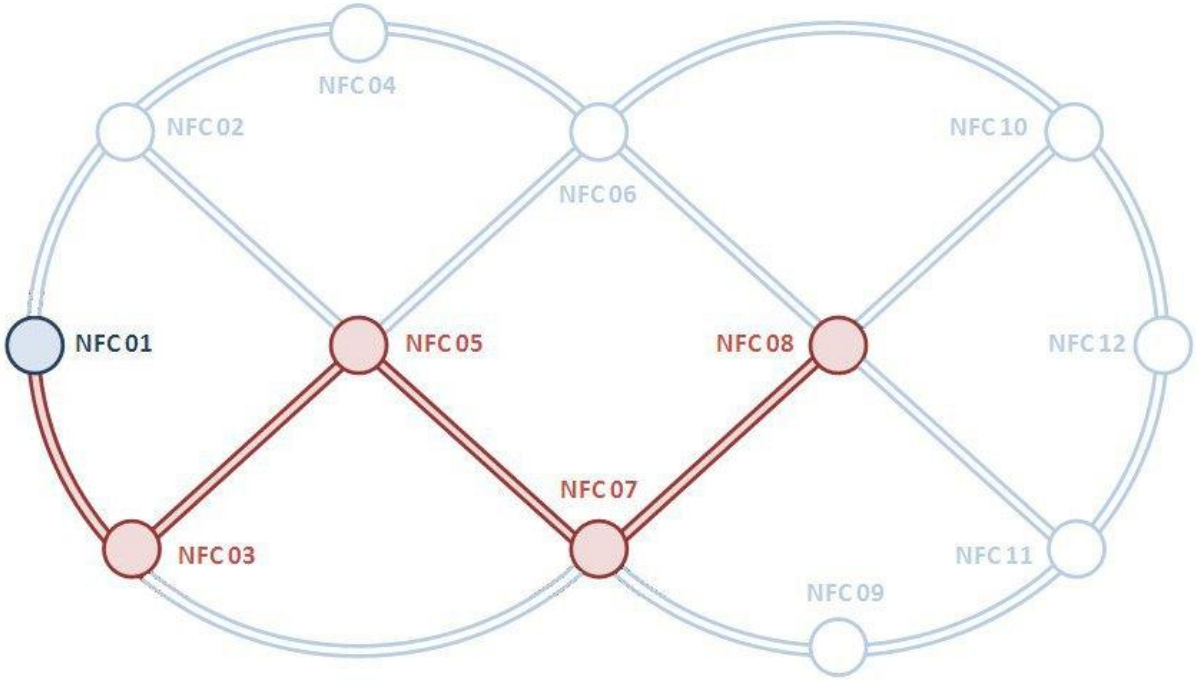


Figura 43: Interfaz de Usuario. Grabación de Trayectoria con Posición actual del robot

Navegación NFC

Con esta acción el robot recorrerá la trayectoria indicada leyendo el contenido de las etiquetas del circuito. La aplicación solicitará el nodo desde el que comienza la trayectoria, calculará la trayectoria desde la posición actual del robot y la trayectoria para llegar al nodo inicial. Una vez posicionado en el nodo inicial, el robot empezará sus desplazamientos siguiendo las indicaciones grabadas en las etiquetas.

El diálogo entre el robot y la aplicación sería el siguiente:

PC->Robot: Se solicita el nodo actual

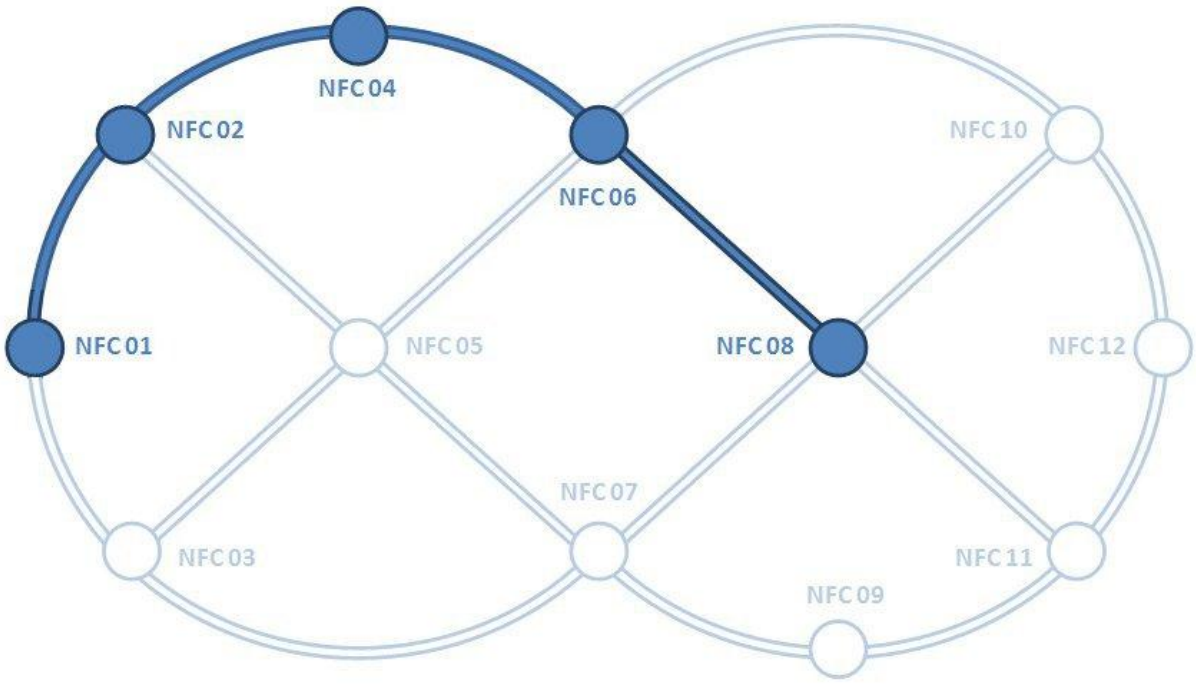
```
POSICION_ACTUAL
```

Robot->PC: Envía el contenido de la etiqueta sobre la que se encuentra posicionado y el nodo del que viene. En caso de error se recibirá un NOK.

```
1|NFC01|40|50|2|2,3|0,0|
3
```

PC->Robot: Se envía la navegación guiada hasta el nodo inicial de la trayectoria a recorrer, 8.

```
NAVEGACION_GUIADA|1:2,3:0,R|2:1,4,5:R,0,0|4:2,6:R,0|6:4,5,8,10:R,0,0,0|8:2,3,6,7: 0,0,0,0|
```



Robot->PC: Al ir recorriendo las distintas etiquetas irá enviando su contenido. Al alcanzar el nodo final enviará un OK, en caso de ocurrir cualquier problema se enviará un NOK.

```
1|NFC01|40|50|2|2,3|0,0|
```

```
2|NFC02|49|73|3|1,4,5|0,0,0|
4|NFC04|75|85|2|2,6|0,0,0|
6|NFC06|100|75|4|4,5,8,10|0,0,0,0|
8|NFC08|127|50|4|6,7,10,11|D,0,R,I|
OK
```

PC->Robot: Se envía la orden de navegación considerando el contenido de las etiquetas NFC.

```
NAVEGACION_NFC
```

Robot->PC: Al ir recorriendo las distintas etiquetas irá enviando su contenido. Al alcanzar el nodo final enviará un OK, en caso de ocurrir cualquier problema se enviará un NOK.

```
8|NFC08|127|50|4|6,7,10,11|D,0,R,I |
7|NFC07|100|25|4|3,5,8,9|:0,0,D,R |
5|NFC05|75|50|4|2,3,6,7|0,0,R,I |
3|NFC03|49|27|3|1,5,7|:0,D,R||
1|NFC01|40|50|2|2,3|0,0|
OK
```

La traducción visual sería la siguiente figura en la que el recorrido en verde debe corresponder con el recorrido anteriormente grabado.

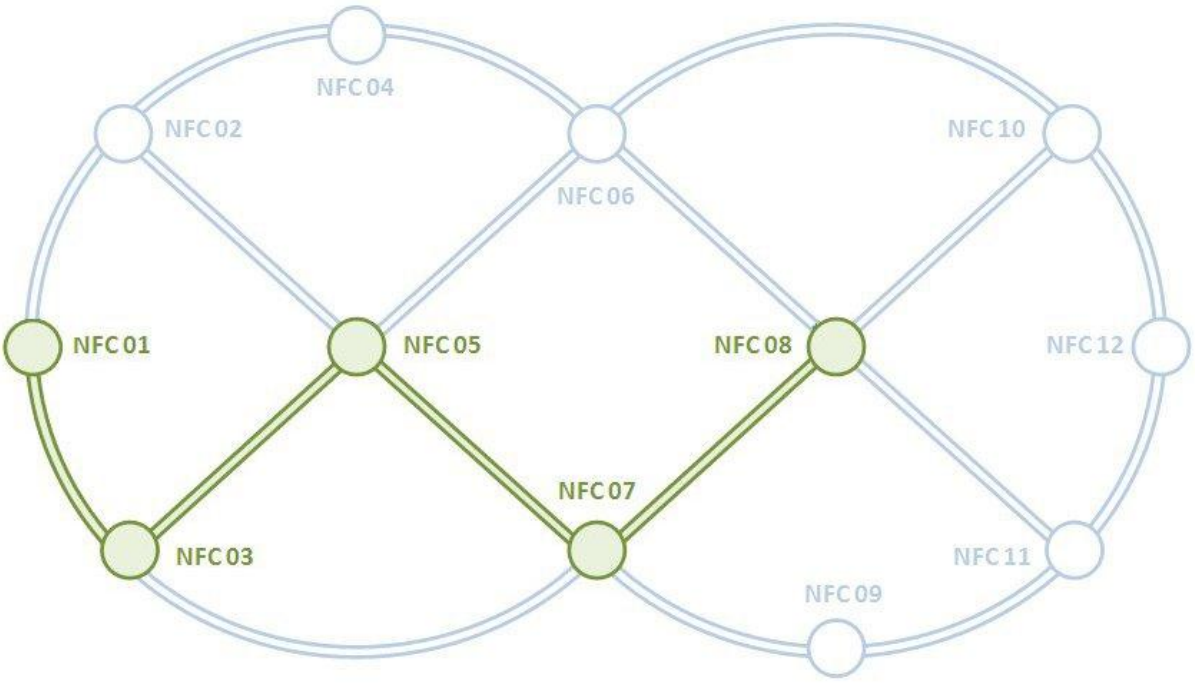
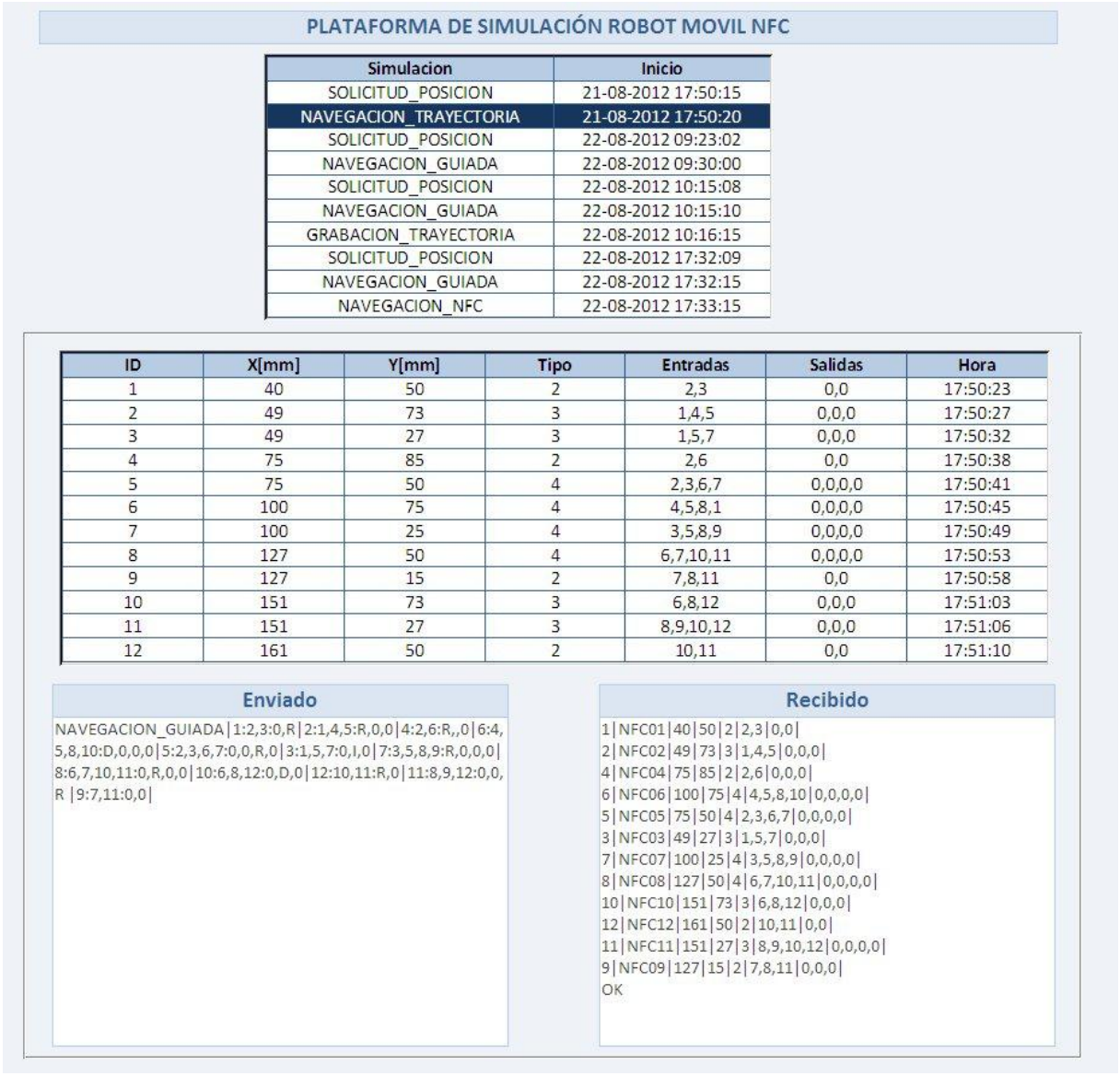


Figura 44: Interfaz de Usuario. Navegación NFC

Histórico

Con esta acción se podrá visualizar un histórico de las simulaciones realizadas. Para ello se mostrará una ventana en la que se listarán todas las simulaciones realizadas con el tipo y la fecha. Al seleccionar una de las simulaciones se mostrará el detalle incluyendo la información intercambiada entre PC y Robot.



Simulacion	Inicio
SOLICITUD_POSICION	21-08-2012 17:50:15
NAVEGACION_TRAYECTORIA	21-08-2012 17:50:20
SOLICITUD_POSICION	22-08-2012 09:23:02
NAVEGACION_GUIADA	22-08-2012 09:30:00
SOLICITUD_POSICION	22-08-2012 10:15:08
NAVEGACION_GUIADA	22-08-2012 10:15:10
GRABACION_TRAYECTORIA	22-08-2012 10:16:15
SOLICITUD_POSICION	22-08-2012 17:32:09
NAVEGACION_GUIADA	22-08-2012 17:32:15
NAVEGACION_NFC	22-08-2012 17:33:15

ID	X[mm]	Y[mm]	Tipo	Entradas	Salidas	Hora
1	40	50	2	2,3	0,0	17:50:23
2	49	73	3	1,4,5	0,0,0	17:50:27
3	49	27	3	1,5,7	0,0,0	17:50:32
4	75	85	2	2,6	0,0	17:50:38
5	75	50	4	2,3,6,7	0,0,0,0	17:50:41
6	100	75	4	4,5,8,1	0,0,0,0	17:50:45
7	100	25	4	3,5,8,9	0,0,0,0	17:50:49
8	127	50	4	6,7,10,11	0,0,0,0	17:50:53
9	127	15	2	7,8,11	0,0	17:50:58
10	151	73	3	6,8,12	0,0,0	17:51:03
11	151	27	3	8,9,10,12	0,0,0	17:51:06
12	161	50	2	10,11	0,0	17:51:10

Figura 45: Interfaz de Usuario. Histórico

APARTADO 5

Resultados

Conclusiones

Líneas de trabajo futuras

Conclusiones

En este trabajo se ha sentado la base para completar el software necesario para realizar simulaciones utilizando una plataforma física en la que se podrá efectuar el estudio de la respuesta de un robot con el uso de la tecnología NFC.

Se ha verificado que dentro del abanico de notaciones que ofrece UML existen diagramas adecuados para la especificación de un sistema como el de este trabajo, en concreto los diagramas de estado facilitan la labor de plasmar el comportamiento de las distintas tarjetas que conforman el robot.

Aunque no se ha podido utilizar el dispositivo NFC directamente con el robot, se ha podido verificar su uso con las aplicaciones comerciales desde PC, comprobando que las posibilidades de esta tecnología pueden ampliar el campo de estudio para la robótica móvil.

Líneas de Trabajo Futuras

Las tareas que podrían iniciarse a partir de este trabajo serían:

- Finalización del software. Siguiendo las especificaciones se podrá completar el software. Dentro de este apartado mencionar que hay partes ya implementadas como la interacción con el sensor de línea. Otras partes no han podido completarse por problemas hardware, como los procesos de lectura y escritura de etiquetas NFC y otras partes están por implementar, como la aplicación externa. En el caso de los procesos de lectura/escritura será necesario solucionar los problemas hardware considerando la posibilidad de utilizar otro tipo de dispositivo.
- Pruebas del software. Se realizarán pruebas unitarias y de integración para verificar el correcto funcionamiento de la plataforma. Para ello se elaborará un plan de pruebas a partir de esta especificación.
- Optimización del diseño. En este trabajo se ha presentado un diseño que no tiene por qué ser el más óptimo. Con las pruebas realizadas se podrán realizar correcciones en el diseño como por ejemplo repartir de otra forma la carga de trabajo entre las tarjetas, un posible cambio puede ir enfocado en dotar aún mas de mayor autonomía a la tarjeta encargada de la lectura y escritura de las etiquetas NFC, guardando más información de estado (como último nodo visitado), atendiendo directamente comandos etc. Siempre se buscará una optimización en los tiempos de respuesta que generalmente estará asociada a la reducción de los comandos intercambiados y a la paralelización de tareas.

Además de la finalización de las tareas mencionadas, el abanico de posibilidades para trabajos futuros es bastante amplio, como ejemplo algunos aspectos en los que podrían centrarse serían:

- Implementación y prueba de distintos tipos de algoritmos de recorrido de grafos.
- Ampliación de las funcionalidades de la aplicación de escritorio.
- Estudio de la viabilidad del uso de otros medios de comunicación como wifi que permitan un radio más amplio de comunicación entre el pc y el robot.
- Desarrollo de un servicio web que permita el acceso remoto a la aplicación de control.

BIBLIOGRAFÍA

- Barrón, M. *"**Curso de Programación de Sistemas Embebidos con Statecharts"*** IEEE-RITA Vol. 4, Núm. 1, Feb. 2009
- Cerrada, C. *"**Diseño y construcción de robots de bajo coste para experimentación con tecnologías RFID".*** Depto. de Ingeniería de Software y Sistemas Informáticos. UNED, Madrid.
- Cerrada, C. ***"Plataforma experimental para la navegación asistida por etiquetas NFC de un robot móvil".*** Depto. de Ingeniería de Software y Sistemas Informáticos. UNED, Madrid.
- Lundqvist, M. ***"UML for embedded software development: An evaluation of Rhapsody in C".*** Master's Thesis, November 2002.
- MicroUML. http://microuml.net/
- Quantum Leaps. QP Development Kits for Arduino. http://www.state-machine.com/arduino/
- Arduino. http://www.arduino.cc/
- Eclipse http://www.eclipse.org/
- FEIG Electronic. http://www.feig.de/home.html
- Papyrus UML Editor. http://www.papyrusuml.org/
- NFCForum: http://www.nfc-forum.org/aboutnfc
- FEIG Electronic: http://www.feig.de

DEFINICIONES

CFG: Configuration Parameter Block

DB: Data Block

I2C: Inter Integrated Circuits

NFC: Near Field Communication

PWM: Pulse Width Modulation

RF: Radio Frequency

RFID: Radio Frequency Identification

SAM: Security Access Module

SPI: Serial Peripheral Interface

UID: Unique Identifier

UML: Unified Modeling Language

ANEXO1

RFID Desktop Reader CPR40.30

Technical Summary

Command Summary

Description

The CPR40.30 is designed as a desktop device for contactless data exchange with common ISO14443-Aand ISO14443-B compliant transponders.

Power supply and data exchange with a computer or other equipment is carried out via the USB interface.

Features

- Multi-tag Reader for ISO tags14443-A & -B tags
- Supports NFC applications
- Data rates up to 847 kBit/s
- PC/SC driver USB
- Desktop Reader to be used in offices or at the PoS for applications like Ticketing,Payment Systems, eDocument
- Command Queue for considerable reduction of communication cycles between reader and host
- Optional: 2 SAM sockets

Technical Data

Technical Data	
Dimensions	144 x 84 x 18 mm
Housing	Plastic ASA, Front: acrylic glass
Color	White and black
Weight	Approx.. 105g
Protectionclass	IP 42
Operatingfrequency	13.56 MHz
RF transmittingpower	100mW
Supplyvoltage	5V, USB Bus powered
Currentconsumption	max. 140 mA
Supportedtransponders	ISO 14443-A ISO 14443-B (106, 212, 424, 847 kbits/s)
Operationmodes	Polling-Mode&ScanMode
Antenna	Integrated
Interface	USB Full-Speed (12 MBits/s)
Opticalindicators	2 LED green: power and online/offline blue: Transpondercommunication
Temperaturerangeoperation	-20ºC up to 60ºC

Temperaturerange Storage	-40 ºC up to 85ºC
Relative air humidity	95 % (non-condensing)
MTBF	500.000 h
Driver Support	PC/SC driver and OBID® USB driver for Windows® 2000 SP4, Windows® Server 2003, Windows® XP SP2, Vista® 32 Bit

Figura 46: CPR40-30. Technical Data

Estándar Conformity

- Radio approval
  - Europe EN 300 330
  - USA FCC 47 CFR Part 15
- EMC EN 301 489
- Safety
  - LowVoltage EN 60950
  - Human Exposure EN 50364

Command for Reader Control

[0x52] Baud Rate Detection

This protocol serves to determine the actual baud rate of the Reader's asynchronous interface.

Host→Reader

1	2	3	4	5...6
6	COM-ADR	[0x52]	0x00	CRC16

Host ← Reader

1	2	3	4	5...6
6	COM-ADR	[0x52]	0x00	CRC16

NOTICE:
The return protocol will only be sent if the inquiry is executed with the baud rate and actual parity of the Reader.

[0x63] CPU Reset

This protocol allows you to reset the CPU on the Reader.

Host→Reader

1	2	3	4...5
5	COM-ADR	[0x63]	CRC16

Host ← Reader

1	2	3	4	5...6
6	COM-ADR	[0x63]	STATUS	CRC16

NOTICE:
The RF-field will be switched off while a CPU Reset.

ISO Host Commands for Transponder Communication

[0xB0] ISO Standard Host Commands

This command sends standard RF commands to the Transponder.

Host→Reader

1	2	3	4...n-2	n-1,n
n	COM-ADR	[0xB0]	REQUEST-DATA	CRC16

Host ← Reader

1	2	3	4	5…n-2	n-1, n
n	COM-ADR	[0xB0]	STATUS	RESPONSE-DATA	CRC16

REQUEST-DATA:
Command specific request

RESPONSE-DATA:
Command specific response

Notes:

- ***Data are only transferred if STATUS = 0x00, 0x83, 0x94, 0x95.***
- ***These commands are not available if Scan-Mode is active.***

[0x01] Inventory

This command reads the UID of all Transponders inside the detection range. The reply behavior of this command can be configures by the ONT parameter of configuration block.

REQUEST-DATA

4	5	(6)
[0x01]	MODE	NTFC_TIME

Notes:
The operating behavior of the inventory command depends on some settings in CFG5, parameter ONT and on settings in MODE byte.

- ***If the CFG5.ONT Bit ONT=b1 only the response of those Transponders are read which came into the antenna field since the last inventory command.***

  In this case The Reader sends back a response including an UID only if:

  - ***If the Transponder has left the antenna and reentered the antenna field or***

- ***The command 5.6. [0x69] RF Reset was send to the Reader meanwhile or***
- ***The Transponder in the antenna field is a Jewel***

- ***If the CFG5.ONT Bit ONT = b0 a RF-Reset is performed to read the UID of all Transponders inside the antenna field.***
- ***If CFG5.ONT Bit ACOLL = b0 (anticollision function is disabled) the Reader selects the Transponder itself.***
- ***If MODE bit PRESC is set to b1 the response includes the Transponders inside the antenna field without performing a RF-Reset.***

MODE:

Bit:	7	6	5	4	3	2	1	0
Function	MORE	NTFC	PRESC	-	-	-	-	-

PRESC:

Setting of this bit activates the presence check mode of the Inventory command. This setting is suitable to perform a presence check of all Transponder in detection range of the reader without influencing the UID of Transponder with a random UID.

b0: Presence check is deactivated
b1: The response of the Inventory command [0x01] includes the UID of all detected Transponders in the reader detection range.

NOTICE:
The PRESC = b1 can only used if CFG5.ONT, ONT bit and ACOLL bit is set to b1 (see CFG5.ONT)

NTFC:

Setting of this bit activates the notification mode of the Inventory command.

b0: Standard Inventory command
b1: Inventory with notification:
In this case the optional parameter NTFC_TIME must send to the reader.

In notification mode the Inventory command runs internally while on ore more Transponders are detected by the reader or while the time defined by NTFC_TIME elapsed.

MORE:

This bit can be used, to read out the whole UIDs, after the Reader had signalized more data sets with status 0x94.

b0: new Inventory requested
The reader carries out a new inquiry, which Transponder are in his detection range.
b1: more data requested

The reader response contains the UIDs which are not transferred with the last response because of the status 0x94.

NOTICE:
The MORE and NTFC function can be used only exclusive.

NTFC_TIME:

This optional parameter defines the maximum duration of the Inventory command in notification mode (see NTFC bit in MODE Byte).

	max. response duration
NTFC_TIME	0..255 * 100ms

NOTICE:

- ***The NTFC_TIME overwrites the TR-RESPONSE-TIME which is defined in CFG1.***
  The receive block timeout of the host computer must set to a value ≥ NTFC_TIME.
- ***A running Inventory command with NTFC option couldn't be interrupted by any other command while NTFC_TIME.***

[0x23] Read Multiple Blocks

This command reads one or more data blocks.

REQUEST-DATA

4	5	(6…13)	6/(14)	7/(15)
[0x23]	MODE	UID	DB_ADR	DB-N

RESPONSE-DATA (STATUS = 0x95)

5
ISO-ERROR

RESPONSE-DATA

5	6	7	8 … n
DB-N	DB-SIZE	SEC-STATUS	DB
		Repeated DB-N times	

MODE:

Bit:	7	6	5	4	3	2	1	0

Function	0	0	0	0	SEC	ADR

ADR:

b000 non-addressed
b001 addressed
b010 selected

SEC:

Requests optional the security status of the followed data block
b0: security status not requested (SEC-STATUS always = 0x00)
b1: security status is requested

UID:

Read-only UID of the Transponder. The UID is required only in the addressed mode.

DB_ADR:

First block number to be read. First block can be any value between 0 and 255.

DB-N:

Number of data blocks to be read from the Transponder, starting at DB_ADR. The maximumnumber of DB-N, depends on DB-Size. The maximum number of bytes is 128 byte

DB-Size	Max.DB-N
1	128
4	32
8	16
x	= 128/x

ISO-ERROR:

Additional error code if STATUS = 0x95.

DB-SIZE:

Number of bytes of one data block.

SEC-STATUS:

Block security status of followed data block.
If SEC-STATUS is not requested or not supported, this value will return 0x00.

DB:

Requested data block. The block size is defined by DB-SIZE.

[0x24] Write Multiple Blocks

This command writes one or more data blocks.

REQUEST-DATA

4	5	(6 …13)	6/(14)	7/(15)	8/(16)	9…n/(17…n)
[0x24]	MODE	UID	DB_ADR	DB-N	DB-SIZE	DB
						Repeated DB-N times

RESPONSE-DATA (STATUS = 0x03)

5
DB_ADR-E

RESPONSE-DATA (STATUS = 0x95)

5	6
ISO-ERROR	DB_ADR-E

MODE:

Bit:	7	6	5	4	3	2	1	0
Function	0	0	0	0	WR-NE	ADR		

ADR:

b000 non-addressed
b001 addressed
b010 selected

WR-NE (Only JEWEL):

b0 JEWEL Write-Erase
b1 JEWEL Write-No-Erase
This setting is necessary for write operations on OTP bytes.

UID:

Read-only UID of the Transpoder. The UID is required only in the addressed mode.

DB_ADR:

Address of the first data block to be written to the Transponder. First block can be any value between 0 and 255.

DB-N:

Number of data blocks to be writtento the Transponder, starting at DB_ADR. The maximumnumber of DB-N, depends on DB-Size. The maximum number of bytes is 128 byte.

DB-Size	Max.DB-N
1	128
4	32
8	16

x	= 128/x

DB-SIZE:

Number of bytes of one data block.

DB:

Data of the data block to be written to the Transponder. The required block size is defined by DB-SIZE. The number of the expected bytes are DB-N * DB-SIZE.

ISO-ERROR:

Additional error code if STATUS = 0x95.

DB_ADR-E:

Block number where the error occurred.

ANEXO 2

Placas Arduino

Arduino Mega

Arduino BT

Arduino Mega

The Arduino Mega is a microcontroller board based on theATmega1280. It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4UARTs(hardware serial ports), a 16MHzcrystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the ArduinoDuemilanove or Diecimila.

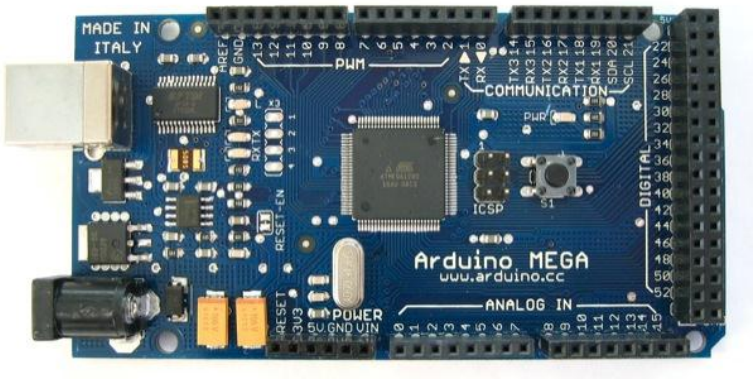


Figura 47: Arduino Mega Front

Summary

Technical Data	
Microcontroller	ATmega1280
OperatingVoltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	128 KB of which 4 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
ClockSpeed	16 MHz

Figura 48: Arduino Mega. Technical Data

Power

The Arduino Mega can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- VIN.The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- 5V.The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- 3V3.A 3.3 volt supply generated by the on-board FTDI chip. Maximum current draw is 50 mA.
- GND.Ground pins.

Memory

TheATmega1280has 128 KB of flash memory for storing code (of which 4 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 54 digital pins on the Mega can be used as an input or output, usingpinMode(),digitalWrite(), anddigitalRead()functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX).**Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the FTDI USB-to-TTL Serial chip.

- **External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2).**These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See theattachInterrupt()function for details.
- **PWM: 2 to 13 and 44 to 46.**Provide 8-bit PWM output with theanalogWrite()function.
- **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).**These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Duemilanove and Diecimila.
- **LED: 13.**There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- **I2C: 20 (SDA) and 21 (SCL).**Support I2C (TWI) communication using theWire library. Note that these pins are not in the same location as the I2C pins on the Duemilanove or Diecimila.

The Mega has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analogReference() function.

There are a couple of other pins on the board:

- **AREF.**Reference voltage for the analog inputs. Used withanalogReference().
- **Reset.**Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Comunication

The Arduino Mega has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. TheATmega1280provides four hardwareUARTsfor TTL (5V) serial communication. An FTDIFT232RLon the board channels one of these over USB and theFTDIdrivers(included with the Arduino software) provide a virtual com port to software on the computer. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TXLEDson the board will flash when data is being transmitted via the FTDI chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

ASoftwareSeriallibraryallows for serial communication on any of the Mega's digital pins.
TheATmega1280also supportsI2C(TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of theI2Cbus; see thedocumentation on the Wiring websitefor details. To use the SPI communication, please see theATmega1280datasheet.

Physical Characteristics

The maximum length and width of the Mega PCB are 4 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes

allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

The Mega is designed to be compatible with most shields designed for the Diecimila or Duemilanove. Digital pins 0 to 13 (and the adjacent AREF and GND pins), analog inputs 0 to 5, the power header, and ICSP header are all in equivalent locations. Further the main UART (serial port) is located on the same pins (0 and 1), as are external interrupts 0 and 1 (pins 2 and 3 respectively). SPI is available through the ICSP header on both the Mega and Duemilanove / Diecimila.

Arduino BT

The Arduino BT is a microcontroller board originally was based on theATmega168, but now is supplied with the 328. and the BluegigaWT11bluetooth module. It supports wireless serial communication over bluetooth (but is not compatible with Bluetooth headsets or other audio devices). It has 14 digital input/output pins (of which 6 can be used as PWM outputs and one can be used to reset theWT11module), 6 analog inputs, a 16MHzcrystal oscillator, screw terminals for power, an ICSP header, and a reset button. It contains everything needed to support the microcontroller and can be programmed wirelessly over the Bluetooth connection.

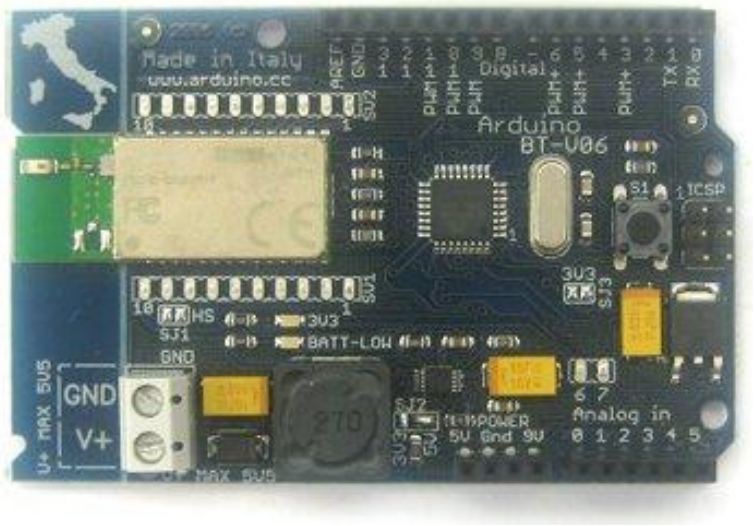


Figura 49: Arduino BT

Summary

Technical Data	
Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage	1.2-5.5 V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (of which 2 KB used by bootloader)
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz
Microcontroller	ATmega328

Figura 50: Arduino BT. Technical Data

Power

The Arduino BT can be powered via the V+ and GND screw terminals. The board contains a DC-DC convector that allows it to be powered with as little as 1.2V, but a maximum of 5.5V.Higher voltages or reversed polarity in the power supply can damage or destroy the board.

The power pins are as follows:

- 9V.The input voltage to the Arduino board (i.e. the same as the V+ screw terminal). You can supply voltage through this pin, or, if supplying voltage via the screw terminals, access it through this pin.Warning: despite the label, do not attach 9V to this pin. It will damage the board.
- 5V.This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.
- GND.Ground pins.

Memory

TheATmega328has 32 KB of flash memory for storing code (of which 2 KB is used for the bootloader). It has 1 KB of SRAM and 512 bytes of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 14 digital pins on the BT can be used as an input or output, usingpinMode(),digitalWrite(), anddigitalRead()functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).**Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the BluegigaWT11module.
- **External Interrupts: 2 and 3.**These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
- **PWM: 3, 5, 6, 9, 10, and 11.**Provide 8-bit PWM output with theanalogWrite()function.
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).**These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.

- **BT Reset: 7.**Connected to the reset line of the BluegigaWT11module, which is active high.
- **LED: 13.**There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The BT has 6 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and some low-level code. Additionally, some pins have specialized functionality:

- **I2C: 4 (SDA) and 5 (SCL).**Support I2C (TWI) communication using theWire library.
- There are a couple of other pins on the board:
- **AREF.**Reference voltage for the analog inputs. Used withanalogReference().

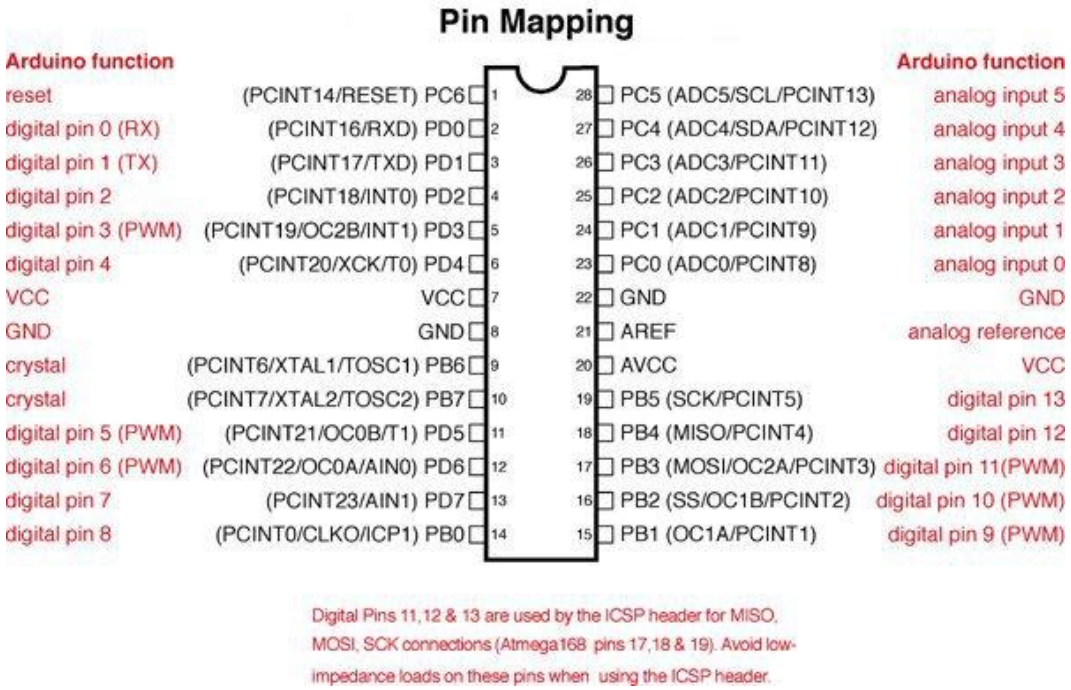


Figura 51: Arduino BT. Pin Mapping

Bluetooth Comunication

The BluegigaWT11module on the Arduino BT provides Bluetooth communication with computers, phones, and other Bluetooth devices. TheWT11communicates with theATmega328via serial (shared with the RX and TX pins on the board). It comes configured for 115200 baud communication. The module should be configurable and detectable by your operating system's bluetooth drivers, which should then provide a virtual com port for use by other applications. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board over this bluetooth connection. The board can also be reprogrammed using this same wireless connection.

TheWT11is specially configured for use in the Arduino BT. Its name is set to ARDUINOBT and passcode to 12345.

Comunication

The Arduino BT has a number of other facilities for communicating. TheATmega328's UART TTL (5V) serial communication is available on digital pins 0 (RX) and 1 (TX) as well as being connected to theWT11module.

ASoftwareSeriallibraryallows for serial communication on any of the BT's digital pins. TheATmega328also supportsI2C(TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of theI2Cbus; see thedocumentation on the Wiring websitefor details..

Physical Characteristics

The maximum length and width of the BT are approximately 3.2 and 2.1 inches respectively. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

ANEXO 3

Otros Elementos Hardware

Circuito de Alimentación

Drivers de Potencia

Sensores de Seguimiento de Línea

Circuito de Alimentación

El circuito de alimentación permite disponer de distintas alimentaciones en corriente continua con baterías de litio – polímero mediante convertidor DC-DC stepdown, capaz de suministrar a su salida distintas tensiones, con una corriente máxima de 3 Amperios de pico entre 3,3 y 12 V para alimentar los distintos módulos del robot.

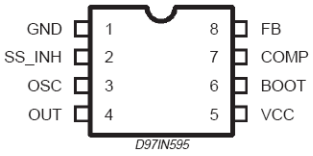


Figura 52: Chip convertidor DC-DC step down

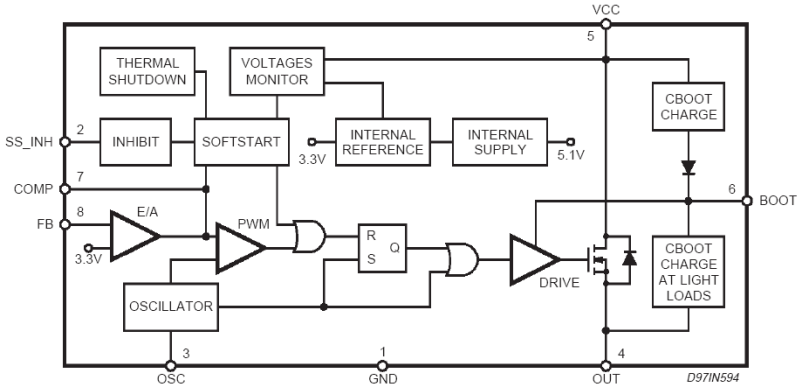
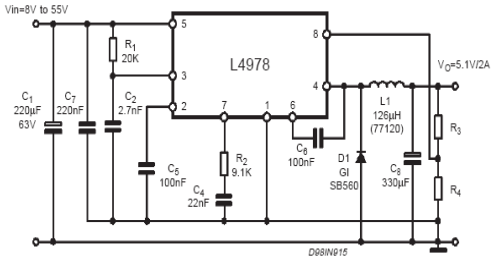


Figura 53: Esquema general interno del chip



C1=220µF/63V EKE
C2=2.7nF
C5=100nF
C6=100nF
C7=220nF/63V
C8=330µF/35V CG Sanyo
L1=126µH KoolMu 77120 - 55 Turns - 0.5mm
R1=20K
R2=9.1K
D1=GI SB560

L4978

V_O(V)	R3(KΩ)	R4(KΩ)
3.3	0	
5.1	2.7	4.7
12	12	4.7
15	16	4.7
18	20	4.7
24	30	4.7

Figura 54: Circuito de alimentación

Drivers de potencia

El robot utiliza una etapa de potencia comercial de la marca Pololu para el control de los motores DC, asociados al sistema de tracción del robot móvil.

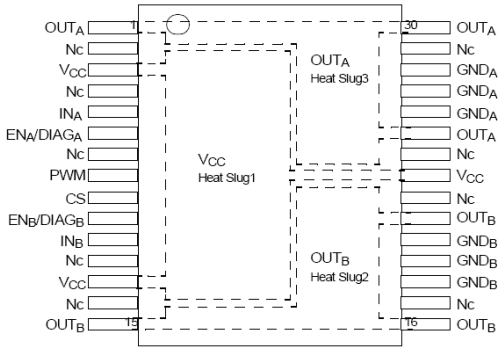


Figura 55: Chip VNH2SP30

Caracteristicas VNH2SP30:

Operating supply voltage (Vcc) = 5.5 – 16 V
Maximum current rating = 30 A
MOSFET on-resistance (per leg) = 19 mΩ
Maximum PWM frequency = 20 kHz
Current sense = approximately 0.13 V/A
Over-voltage shutoff = 16 V minimum (19 V typical)
Time to overheat at 20 A** = 35 seconds
Time to overheat at 15 A** = 150 seconds
Current for infinite run time** = 14 A

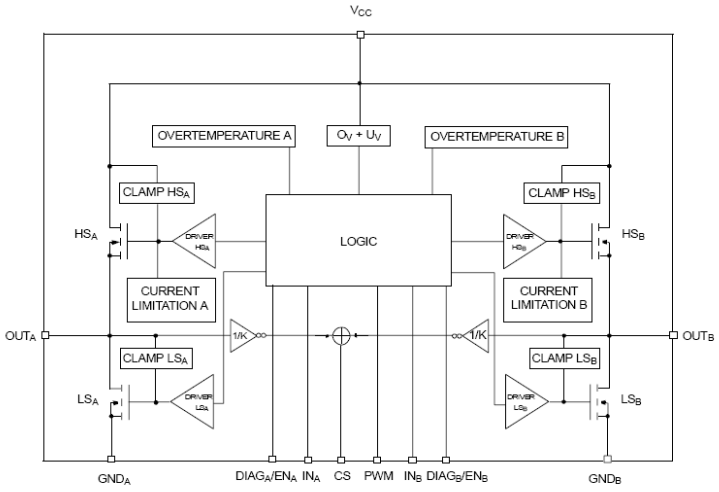


Figura 56: Esquema general interno de chip

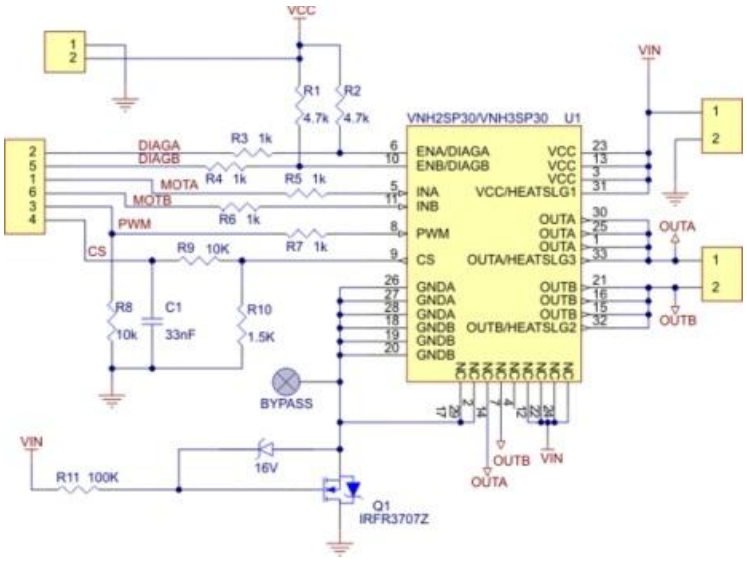
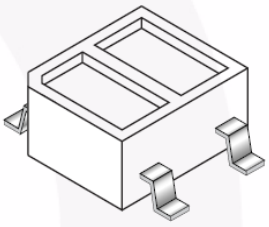


Figura 57: Circuito driver de potencia

Sensores de seguimiento de línea

El robot utiliza una placa de sensores comercial de la marca Pololu que dispone de 8 sensores con diodo emisor y receptor de luz infrarroja con salida analógica y que dotan al robot de la capacidad de detección de la línea negra sobre fondo blanco que seguirá el robot para evolucionar por el circuito de ensayos experimental.



EMITTER			
I_F	Continuous Forward Current	50	mA
V_R	Reverse Voltage	5	V
I_{FP}	Peak Forward Current(5)	1	A
P_D	Power Dissipation(1)	75	mW
SENSOR			
V_{CEO}	Collector-Emitter Voltage	30	V
V_{ECO}	Emitter-Collector Voltage	5	V
I_C	Collector Current	20	mA
P_D	Power Dissipation(1)	50	mW

Figura 58: Sensor QRE1113GR

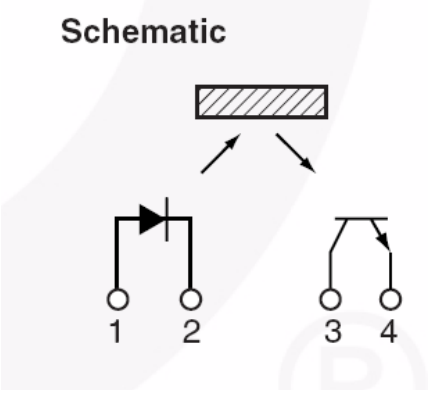


Figura 59: Esquema general interno del sensor

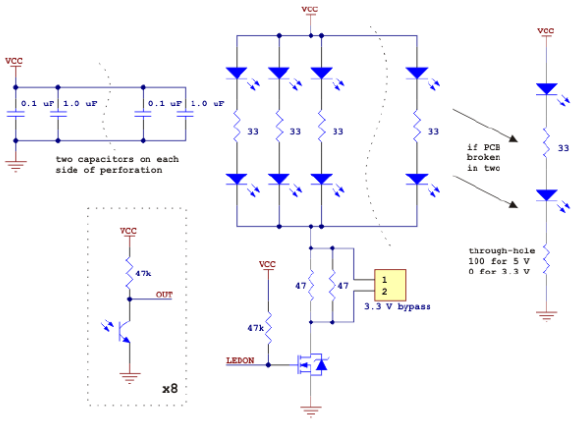


Figura 60: Esquema del circuito de seguimiento de línea

ANEXO 4

Código Fuente

Introducción

Rutinas de código

Introducción

En este anexo se recopila código fuente relacionado con el trabajo. Se incluyen rutinas para la realización de pruebas básicas de los elementos involucrados en el sistema (Comunicación con CPR40.30, I2C, Bluetooth, motores …). El código incluido solo hace referencia al robot y servirá de punto de partida para el desarrollo final del software del sistema.

El código se implementa utilizando la herramienta de edición y compilación del paquete Arduino descargado desde su página. Se deberá seleccionar dentro de las opciones del entorno de desarrollo la placa con la que se esté trabajando para cargar el código compilado.

Rutinas de código

Comunicación con el lector CPR40.30

```
/*
  Rutinas para verificar la correcta comunicacion con el lector CPR40.30
*/

#define CRC_POLYNOM 0x8408
#define CRC_PRESET  0xFFFF

void setup() {
   Serial.begin(9600);
   Serial1.begin(9600); //CPR40
   cmdGetSoftwareVersion();
   cmdResetCPU();
 }

void loop() {
   // la posible respuesta del lector en Serial1 lo mostramos por Serial:
   if (Serial1.available()) {
     int inByte = Serial1.read();
     Serial.write(inByte);
  }
 }

/*
  cmdGetSoftwareVersion
  [0x65] Get Software Version
  Solicita la version del firmware del lector

  [05 FF 65 CRC16]:
     05 LENGTH
     FF COM-ADR direccion del lector (con 255 podemos referenciarlo directamente)
     65 comando para solicitar la version del software
     CRC16 (dos bytes)
*/
void cmdGetSoftwareVersion(){
   Serial1.write(0x05);
   Serial1.write(0xFF);
   Serial1.write(0x65);
   Serial1.write(crc(3));
}
```

```
/*
  cmdResetCPU
  [0x63] CPU Reset
  Reseteo de la CPU del lector

  [05 FF 63 CRC16]:
     05 LENGTH
     FF COM-ADR direccion del lector (con 255 podemos referenciarlo directamente)
     63 comando para resetear la CPU del lector
     CRC16 (dos bytes)
*/
void cmdGetSoftwareVersion(){
   Serial1.write(0x05);
   Serial1.write(0xFF);
   Serial1.write(0x63);
   Serial1.write(crc(3));
}

/*
  crc
  Calcula el CRC de la trama segun el polinomio x^16+x^12+x^5+1

  Parametros:
        int cnt - numero de bytes sin el CRC
*/
unsigned int crc(int cnt){
   unsigned int crc = CRC_PRESET;
   for (i = 0; i < cnt; i++) {
      crc ^= DATA[i];
      for (j = 0; j < 8; j++) {
        if (crc & 0x0001)
          crc = (crc >> 1) ^ CRC_POLYNOM;
        else
          crc = (crc >> 1);
        }
   }
   return crc;
 }
```

Comunicación I2C

```
/*
 Ejemplo de uso de la libreria Wire para la comunicación I2C entre dos tarjetas

*/
include <Wire.h>

#define THIS_ADDRESS 0x8
#define OTHER_ADDRESS 0x9

void setup() {
  Wire.begin(THIS_ADDRESS);
  Wire.onReceive(receiveEvent);
}

byte x = 0;

void loop() {
  Wire.beginTransmission(OTHER_ADDRESS);
  Wire.write("Comando ");
  Wire.write(x);
  Wire.endTransmission();
  X++;
  delay(500);
 }

void receiveEvent(int howMany){
  while (Wire.available() > 0){
    char c = Wire.receive();
    Serial.print(c);
  }
  Serial.println();
}
```

Driver de Potencia y sensor IR

```
// Robot sigue lineas  con lector/grabador NFC
// Version para Driver VNH2SP30

// Constantes




// Variables sensores

int Val_A_0=0;  /* Declaramos e inicializamos */
int Val_A_1=0;
int Val_A_2=0;
int Val_A_3=0;
int Val_A_4=0;
int Val_A_5=0;

float Pos_Numerador;
float Pos_Denominador;
float Posicion_Linea;   /* valores entre 0 y 500 o 0 y 5000 segun factor */



// Variables PID

 int Kp=1.5;
 int Kd=1.5;
 int Ki=1;
 int Potencia_Max=20;

 int Error;
 int Error_Anterior;

 int Consigna=2500;
 int S_Proporcional;
 int Derivativo;
 int  S_Derivativo;
 int  Integral;
 int S_Integral;
 float Salida;
```

```
 int Potencia_Izq;
 int Potencia_Dch;

// Definicion de puertos utilizados

/* Sensores de Linea */

 //int Pin Led ON = 0;     Activa/desactiva todos los led infrarojos del sensor QTR-8A
analogivo
 int Pin_Analog_IR_0 = 0;   // Pin sensor posicion 0
 int Pin_Analog_IR_1 = 1;
 int Pin_Analog_IR_2 = 2;
 int Pin_Analog_IR_3 = 3;
 int Pin_Analog_IR_4 = 4;
 int Pin_Analog_IR_5 = 5;   // Pin sensor posicion 5

//MOTORES

/* Motor Derecho */
    int Pin_Pwm_Dch = 3;         //Pin  senal PWM
    int Pin_Control_INA_D1=2;   //Senales de control
    int Pin_Control_INB_D1=4;   //Senales de control

/* Motor Izquierdo */

    int Pin_Pwm_Izq = 5;         //Pin  seÃ±al PWM
    int Pin_Control_INA_I1=7;   //Senales de control
    int Pin_Control_INB_I1=6;   //Senales de control




void setup()
{
 Serial.begin(9600); // Abrimos el puerto serie

  /* Configurancion puertos */
  //Motor derecho senales de control
  pinMode(Pin_Control_INA_D1, OUTPUT);      // Pin como salida
  pinMode(Pin_Control_INB_D1, OUTPUT);      // Pin como salida
  //Motor izquierdo senales de control
  pinMode(Pin_Control_INA_I1, OUTPUT);      // Pin como salida
  pinMode(Pin_Control_INB_I1, OUTPUT);      // Pin como salida

 /* Motores */
```

```
    /* Motores Avanzan */
        //Motor derecho
        digitalWrite(Pin_Control_INA_D1,LOW);
        digitalWrite(Pin_Control_INB_D1,HIGH);
        //Motor Izquierdo
        digitalWrite(Pin_Control_INA_I1,HIGH);
        digitalWrite(Pin_Control_INB_I1,LOW);


}


void loop()
{


  // Leemos sensores linea


  /* Lee valores entre 0 y 1023 */
  Val_A_0 = analogRead(Pin_Analog_IR_0);   // Lee valores entre 0 y 1023
  Val_A_1 = analogRead(Pin_Analog_IR_1);
  Val_A_2 = analogRead(Pin_Analog_IR_2);
  Val_A_3 = analogRead(Pin_Analog_IR_3);
  Val_A_4 = analogRead(Pin_Analog_IR_4);
  Val_A_5 = analogRead(Pin_Analog_IR_5);



  Pos Numerador=
((Val_A_1*1000.0)+(Val_A_2*2000.0)+(Val_A_3*3000.0)+(Val_A_4*4000.0)+(Val_A_5*5000.0));
  Pos_Denominador =(Val_A_0 +Val_A_1 + Val_A_2 + Val_A_3 + Val_A_4 + Val_A_5);


  Posicion_Linea=(Pos_Numerador / Pos_Denominador);


  if ((Val A 0<50) &&(Val A 1<50) && (Val A 2<50)&& (Val A 3<50) && (Val A 4<50) &&
(Val_A_5<50))
      {
       //todos en blanco
         Posicion_Linea=0;
      }




  Serial.println("----POS_--");
  Serial.println(int(Posicion_Linea));
 Serial.println("----------");


  /* Fin lectura sensores */



  /* PID */
```

```
  Error=(Consigna-int(Posicion_Linea));

  //Serial.println("--ERROR---");
  //Serial.println(Error);
  //Serial.println("----------");

  S_Proporcional=Error;
  Derivativo=Error;
  Derivativo=Derivativo-Error_Anterior;
  Error_Anterior=Error;
  S_Derivativo=Derivativo;
  Integral=Integral+Error;

     if (Integral>300)

      {
        Integral=1;  // action A
      }
     else
      {
        // action B
        // No hacer nada
      }

  S_Integral=Integral;



  Salida=(S_Proporcional *(Kp)) + (S_Derivativo *(Kd/3)) + (S_Integral * (Ki/1000));



  if ((Val A 0<70) &&(Val A 1<70) && (Val A 2<70)&& (Val A 3<70) && (Val A 4<70) &&
(Val_A_5<70))
      {
       //todos en blanco
          Posicion_Linea=0;
      }



   if (int(Salida) > 255)

     {
       Salida=255;
     }



  /* Motor Izquierdo */
```

```
    Potencia_Izq= Potencia_Max+(int(Salida));
    analogWrite(Pin_Pwm_Izq, Potencia_Izq);    /* Valores de 0 a 255 */

  /* Motor Derecho */
    Potencia_Dch= Potencia_Max-(int(Salida));
    analogWrite(Pin_Pwm_Dch,Potencia_Dch); //

  //delay(2000);                  //Retardo 1s

  //...
} // Fin bucle infinito
```

Inclusión de Bluetooth

```
/* Robot_NFC03_BT_pololu
* ----------------------
* JIBM junio 2010
*
*/

// Robot sigue lineas  con lector/grabador NFC
// Version para Driver VNH2SP30

// Constantes simbolicas
#define Consigna   2500

// Variables sensores

int Val_A_0=0;  /* Declaramos e inicializamos */
int Val_A_1=0;
int Val_A_2=0;
int Val_A_3=0;
int Val_A_4=0;
int Val_A_5=0;

float Pos_Numerador;
float Pos_Denominador;
int Posicion_Linea=2500;   /* valores entre 0 y 500 o 0 y 5000 segun factor */

int Potencia_Izq=0;
int Potencia_Dch=0;
```

```
// Protitipos de funciones
unsigned int Lectura_Sensores();
void para_motores();
// Variables PID
 float kp=25.0;
 float kd=3/2;
 float ki=10000.0;

 int V_max=24;
 int Proporcional_Anterior=0;
 long Integral=0.0;

// Definicion de puertos utilizados

/* Sensores de Linea */

  //int Pin Led ON = 0;     Activa/desactiva todos los led infrarojos del sensor QTR-8A
analogivo
  int Pin_Analog_IR_0 = 0;   // Pin sensor posicion 0
  int Pin_Analog_IR_1 = 1;
  int Pin_Analog_IR_2 = 2;
  int Pin_Analog_IR_3 = 3;
  int Pin_Analog_IR_4 = 4;
  int Pin_Analog_IR_5 = 5;   // Pin sensor posicion 5

//MOTORES

/* Motor Derecho */
    int Pin_Pwm_Dch = 3;         //Pin  senal PWM
    int Pin_Control_INA_D1=2;   //Senales de control
    int Pin_Control_INB_D1=4;   //Senales de control

/* Motor Izquierdo */

    int Pin_Pwm_Izq = 5;         //Pin  senal PWM
    int Pin_Control_INA_I1=7;   //Senales de control
    int Pin_Control_INB_I1=6;   //Senales de control

void setup() {

  //Configuracion bluetooth

  Serial.begin(19200);   // connect to the serial port
  Serial.println("SET BT PAGEMODE 3 2000 1");
  Serial.println("SET BT NAME ARDUINOBT");
  Serial.println("SET BT ROLE 0 f 7d00");
  Serial.println("SET CONTROL ECHO 0");
```

```
  Serial.println("SET BT AUTH * 12345");
  Serial.println("SET CONTROL ESCAPE - 00 1");
  Serial.println("SET CONTROL BAUD 19200,8n1");      //first release 19200, 115200

  // Configuracion robot

      //Serial.begin(9600); // Abrimos el puerto serie
       /* Configurancion puertos */
     //Motor derecho senales de control
     pinMode(Pin_Control_INA_D1, OUTPUT);      // Pin como salida
     pinMode(Pin_Control_INB_D1, OUTPUT);      // Pin como salida
     //Motor izquierdo senales de control
     pinMode(Pin_Control_INA_I1, OUTPUT);      // Pin como salida
     pinMode(Pin_Control_INB_I1, OUTPUT);      // Pin como salida

     /* Motores */

    /* Motores Avanzan */
        //Motor derecho
        digitalWrite(Pin_Control_INA_D1,LOW);
        digitalWrite(Pin_Control_INB_D1,HIGH);
        //Motor Izquierdo
        digitalWrite(Pin_Control_INA_I1,HIGH);
        digitalWrite(Pin_Control_INB_I1,LOW);

} // Fin void setup


void loop() {

      // Mi main
      // Leemos sensores linea

      /* PID */
      unsigned int Posicion = Lectura_Sensores();
      Serial.println("----Posicion_--");
      Serial.println(int(Posicion));
      Serial.println("----------");

      //Calculamos error
      int Proporcional =((int)Posicion - Consigna);
      int Derivativo = Proporcional-Proporcional_Anterior;
      Integral+=Proporcional; //Integral=Integral+Proporcional

      // Almacenamos valor
      Proporcional_Anterior = Proporcional;
```

```
      Serial.println("--ERROR---");
      Serial.println(Proporcional);
      Serial.println("----------");

      int V_dif=(((int)Proporcional/kp) + ((int)Integral/ki) + ((int)Derivativo*kd));

      Serial.println("-------");
      Serial.println(V_dif);
      Serial.println("----------");

      if (V_dif > V_max)
         {
           V_dif=V_max;
         }

     if (V_dif < -V_max)
        {
          V_dif=-V_max;
        }

     if (V_dif < 0)
        {
          /* Motor Izquierdo */
          Potencia_Izq= V_max +(int(V_dif));
          analogWrite(Pin_Pwm_Izq, Potencia_Izq);    /* Valores de 0 a 255 */
          /* Motor Derecho */
          Potencia_Dch= V_max;
          analogWrite(Pin_Pwm_Dch,Potencia_Dch); //
         }
   else
         {
           /* Motor Izquierdo */
           Potencia_Izq= V_max;
           analogWrite(Pin_Pwm_Izq, Potencia_Izq);    /* Valores de 0 a 255 */
           /* Motor Derecho */
           Potencia_Dch= V_max-(int(V_dif));;
           analogWrite(Pin_Pwm_Dch,Potencia_Dch);
         }

} // Fin bucle infinito funcion void loop


//**************************************************************************
// funcion lectura sensores IR linea
//**************************************************************************
unsigned int Lectura_Sensores()
```

```
    {
      // Leemos sensores linea


      /* Lee valores entre 0 y 1023 */
      Val_A_0 = analogRead(Pin_Analog_IR_0);   // Lee valores entre 0 y 1023
      Val_A_1 = analogRead(Pin_Analog_IR_1);
      Val_A_2 = analogRead(Pin_Analog_IR_2);
      Val_A_3 = analogRead(Pin_Analog_IR_3);
      Val_A_4 = analogRead(Pin_Analog_IR_4);
      Val_A_5 = analogRead(Pin_Analog_IR_5);


      // Calculamos posicion
      Pos Numerador= (
(Val_A_1*1000.0)+(Val_A_2*2000.0)+(Val_A_3*3000.0)+(Val_A_4*4000.0)+(Val_A_5*5000.0) );
      Pos_Denominador =(Val_A_0 +Val_A_1 + Val_A_2 + Val_A_3 + Val_A_4 + Val_A_5);
      Posicion_Linea=(Pos_Numerador / Pos_Denominador);



      if ( (Val A 0<50) &&(Val A 1<50) && (Val A 2<50)&& (Val A 3<50) && (Val A 4<50) &&
(Val_A_5<50) )
          {
            //todos en blanco
            Posicion_Linea=0;
          }


       return Posicion_Linea;


    } // Fin fuuncion



//*************************************************************************
// funcion para motores
//*************************************************************************
 void para_motores()
    {
          /* Motor Izquierdo */
          analogWrite(Pin_Pwm_Izq, 0);    /* Valores de 0 a 255 */
          /* Motor Derecho */
          analogWrite(Pin_Pwm_Dch,0); //
    } // Fin funcion
```