Presentación Programación de arquitecturas encastadas

Xavi Cano

Claudio Cárcamo

Grupo B

Introducción

¿ Que queremos hacer ?

Lo que se pretende conseguir en este proyecto es hacer que nuestro robot siga una pared, ya sea a izquierda o a derecha y además en este proceso evitar los obstáculos que se vaya encontrando a lo largo del recorrido.

Objetivos:

- ✓ Conseguir mover sus ruedas mediante sus 4 motores.
- ✓ Conseguir evitar obstáculos mediante su sensor.
- ✓ Conseguir seguir una pared a derecha o izquierda.
- ✓ Conseguir mostrar los valores de los sensores en su pantalla.

Ingeniería de detalle

- **Herramientas de trabajo:**

  - IDE: Code Composer

  - Lenguaje de programación: C

Ingeniería de detalle

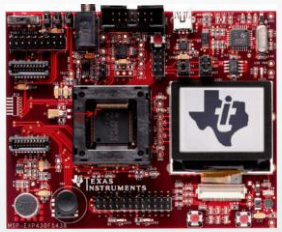
➢ Tipo de robot:

Nuestro robot esta compuesto por:

- 4 módulos Dynamixel AX-12

- 1 modulo Dynamixel AX-S1

- 1 placa superior MSP-EXP430F5438



- 1 placa inferior de 8 Leds

Ingeniería de detalle

➢ Módulos del robot:

- **Módulos Dynamixel AX-12:**

  Cada uno es estos módulos esta compuesto por un microcontrolador que gestiona las comunicaciones y acciona el motor adecuado.
  Para poder saber que modulo estamos usando, cada uno de ellos tiene un identificador, en nuestro caso corresponden al 1,2,3 y 4.

- **Módulos Dynamixel AX-s1:**

  Este modulo esta compuesto por tres sensores infrarrojos para detectar distancias, ángulos i luz. También incorpora un micrófono, una alarma y control remoto por infrarrojos, aunque estos últimos no los hemos utilizado en nuestro proyecto.
  Este modulo solo tiene un único identificador, que en nuestro caso es el 100.

Ingeniería de detalle

➢ Placas del robot:

- **Placa superior MSP-EXP430F5438 :**

❑ Pantalla LCD Dot-Matrix.
❑ Joystick de 5 direcciones.
❑ Pulsadores S1 y S2.
❑ Leds 1 y 2.
❑ Acelerómetro.
❑ Entrada de Audio.
❑ Salida de Audio.
❑ Conectores a puertos x,y.
❑ Conexión USB.
❑ Conexión RF.

- **Placa inferior de 8 leds:**

Esta placa esta conectada a la anterior con la siguiente configuración:

❑ LED 0: Puerto 4.0
❑ LED 1: Puerto 4.1
❑ LED 2: Puerto 4.2
❑ LED 3: Puerto 4.3
❑ LED 4: Puerto 4.4
❑ LED 5: Puerto 4.5
❑ LED 6: Puerto 4.6
❑ LED 7: Puerto 4.7

Ingeniería de detalle

➢ Librerías:

En nuestro proyecto hemos implementado una única librería, la cual contiene todas las funciones principales necesarias para los objetivos de nuestro proyecto, es decir:

- Poder avanzar y retroceder.
- Girar a izquierda y derecha.
- Detectar y evitar un obstáculo.

Estas funciones, no se podrían haber implementado sin haber hecho antes otras muy importantes , que son las siguientes:

- Configuración de la UCS: *Configuración del sistema de reloj del microcontrolador.*
- Configuración de la UART: *Configuración de la transmisión-recepción asíncrona de 8 bits.*
- Configuración de Timers: *Configuración de un temporizador para gestionar interrupciones.*
- Configuración del RxPacket: *Configuración de la recepción de una trama.*
- Configuración del TxPacket: *Configuración de la transmisión de una trama.*

Ingeniería de detalle

➢ Algoritmo de seguimiento de paredes(I):

1. Para comenzar el algoritmo se necesita seleccionar que pared debe seguir el robot. Esta selección se hará por medio de los botones s1 y s2 de la placa MSP-EXP430F5438.

2. Primero de todo comprobamos si hay un obstáculo adelante del robot. Si al leer del sensor obtenemos un resultado de 255, quiere decir que hay un obstáculo y giramos para evitarlo. Posteriormente iniciamos un flag obstáculo a 1.

3. De nuevo, comprobaremos si la variable del obstáculo esta a 1. Si es así, comprobamos que la distancia del sensor delantero sea cero. Si se cumplen estas condiciones, cambiamos el valor del flag obstáculo a cero y movemos el robot hacia adelante.

4. Después volvemos a comprobamos que el flag obstáculo este a cero. Si es así tendremos tres comprobaciones que nos permitirán seguir la pared derecha o izquierda.

Ingeniería de detalle

➢ Algoritmo de seguimiento de paredes(II):

5. Si el valor del sensor derecho o izquierda, se encuentra entre 100 y 20, entonces, el robot se moverá hacia adelante y los flags de comprobación de la pared derecha e izquierda volverán a ser cero.

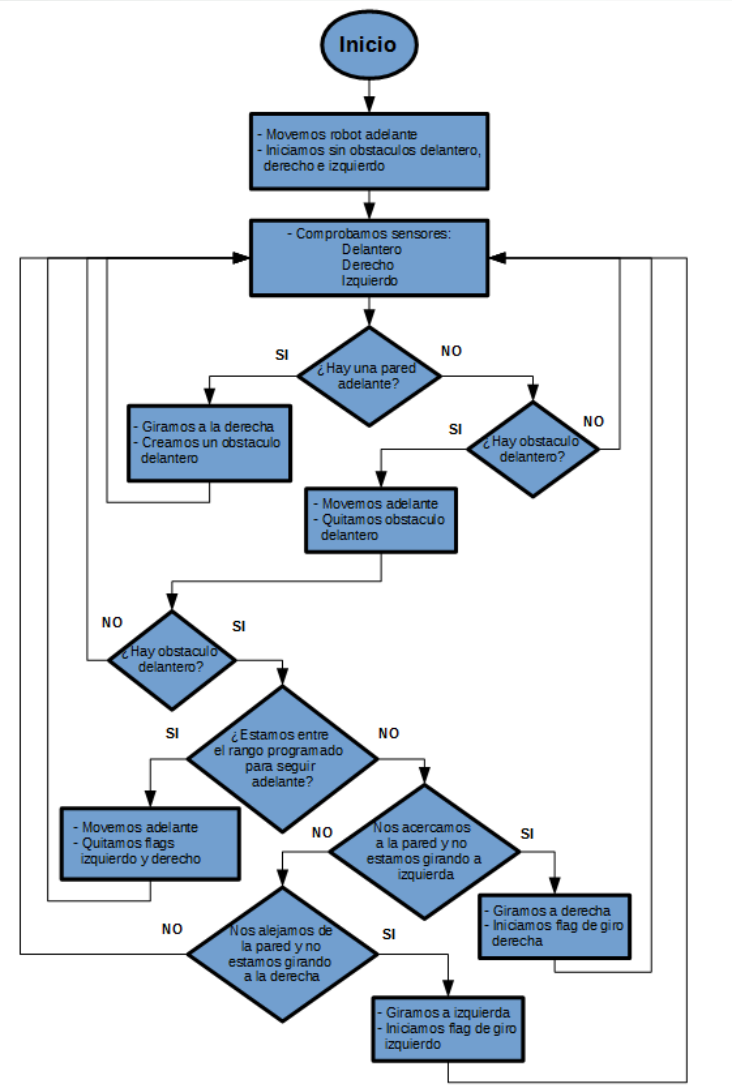
6. Si el valor del sensor derecho o izquierdo es mayor que 100, indica que estamos cerca de la pared y el flag de la pared contraria es cero *(lo que significa que el robot no esta girando). Giraremos en el sentido contrario y cambiamos el flag que indica que el robot esta girando a 1.

7. Si el valor del sensor derecho o izquierdo es menor que 20, que indica que nos alejamos de la pared, y la variable de la pared contraria es cero *(lo que significa que el robot no esta girando). Giraremos en el sentido contrario y cambiamos la variable que indica que el robot esta girando a 1.

** Si el robot esta girando y no tenemos flags que indiquen que esta girando. Realizara ambas condiciones y al girar a la derecha e izquierda (cambio de sentido en las ruedas), ira al revés.*

Ingeniería de detalle

- **Diagrama de flujo:**



Problemas

- En ocasiones el robot cambiaba de sentido, moviéndose en al revés. Esto se debía a que en ocasiones se cumplían dos condiciones y mientras giraba hacia un sentido, giraba hacia otro moviéndose al revés. Ya que el giro cambia el sentido de las ruedas.

- El sensor devolvía un cero, creándose valores falsos en algunas ocasiones.

- Cuando el robot se encontraba en una esquina, el valor que devolvía el sensor era mayor que el programado por lo que el robot hacia un giro inesperado. La solución fue aumentar el rango programado.

Conclusiones

- Debido al escaso tiempo de programación nos hubiera gustado mejorar el código , no incluyendo tantas variables de comprobación.

- Hemos conseguido realizar los giros del robot sin tener que detener ninguna de las 4 ruedas, de esta forma el giro es mas eficiente.

- Nos hubiera gustado haber podido trabajar con todos los recursos del robot, como por ejemplo: sonido, leds, etc… Para haber demostrado todos nuestros conocimientos.

Demostración

https://www.youtube.com/watch?v=TR54CwV3lo4