UNIVERSIDADE DA CORUÑA

facultade de informática da coruña

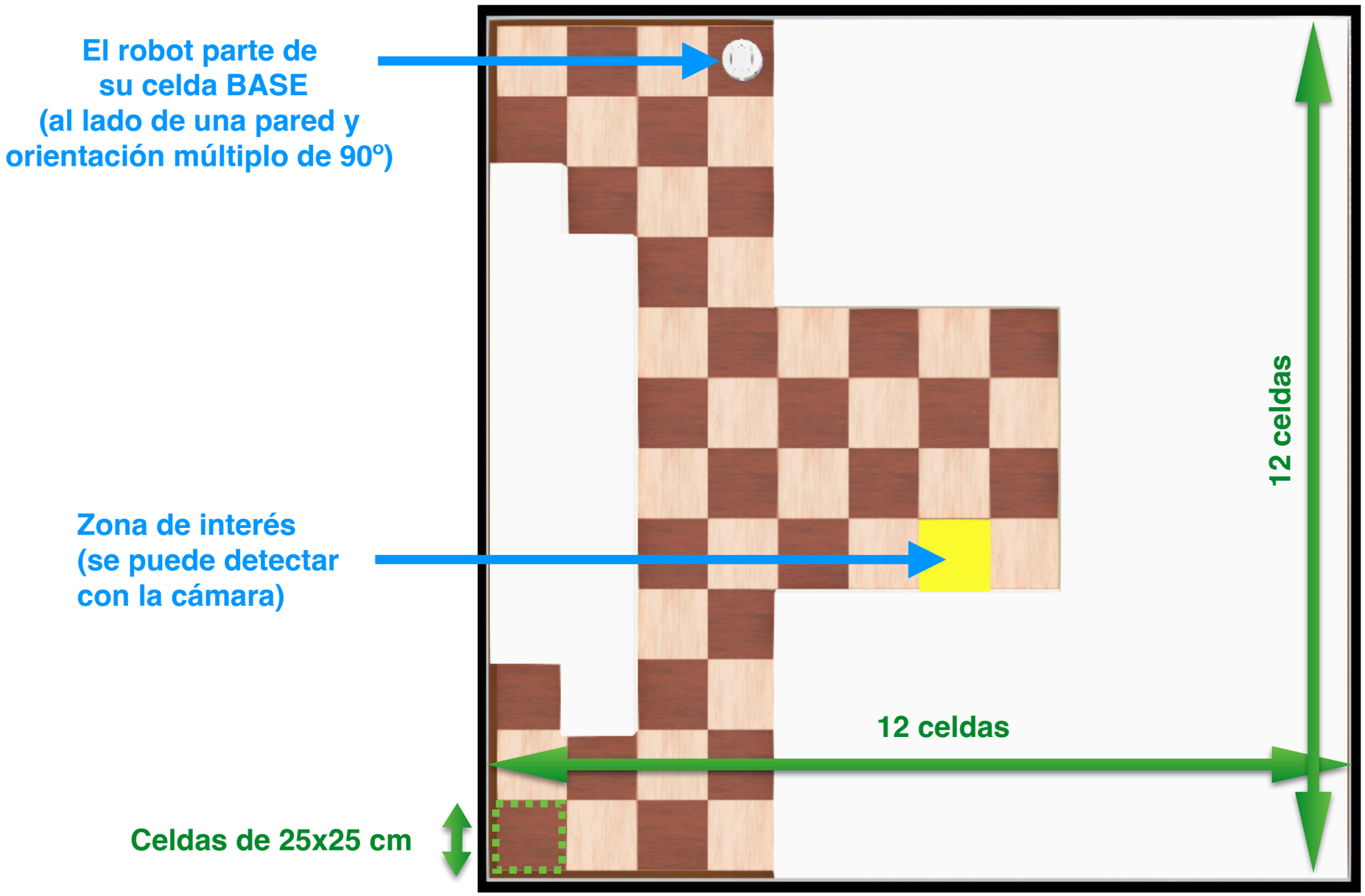
Práctica 2

Mapeado y localización

Grao en Enxeñaría Informática - 4º curso
ROBÓTICA

Enunciado

- Esta práctica consistirá en la implementación de una **arquitectura híbrida**, dónde el robot tendrá comportamiento **reactivo o deliberativo** dependiendo de su estado interno.



Enunciado

- La práctica se realizará **únicamente en simulación**, utilizando el **simulador Webots** y el **robot Khepera IV**.
- **Objetivo del robot:** patrullar un entorno siguiendo paredes con **odometría** y volver a la base inmediatamente en caso de detectar con la cámara un evento de interés (para el caso se marcará con una zona de color amarillo sólido).
    - **El entorno será discreto** (dividido en celdas de 25x25 cm) y de dimensiones conocidas (12x12 celdas, equivalente a 3x3 m). Por tanto, los obstáculos nunca estarán a medias entre dos celdas.
    - El comportamiento del robot debe ser válido para cualquier posición inicial de la base siempre que esté al lado de una pared (el robot no conoce la posición absoluta de su base).
    - El comportamiento del robot debe ser válido aunque se varíe la posición de una zona de interés (amarilla) sin afectar a su funcionamiento (incluso si se ubicase en otra posición una vez iniciada la ejecución).
- La ejecución debe realizarse en 2 etapas:
    1. Una etapa inicial reactiva de exploración del entorno **siguiendo paredes** y a la vez **creando un mapa interno del entorno**.
    2. Una segunda etapa en la que el robot aprovecha el mapa del entorno que ha creado (junto con un comportamiento deliberativo) para mejorar su desempeño:
        1. Patrullar el entorno siguiendo paredes (se sale de la base dejando siempre paredes al mismo lado).
        2. Si detecta con la cámara un evento de interés (zona amarilla), **utilizar el mapa interno** para regresar inmediatamente a la estación del robot de forma más eficiente (**planificación deliberativa de la ruta de regreso**). Dar un margen en centesimas porque é casi imposible que o retorno a casilla inicial coincida exactamente no mismo decimal que o float da posicion inicial

Enunciado

- **MIENTRAS EL ROBOT NO TIENE UN MAPA DEL ENTORNO CREADO:** se utilizará una funcionalidad que permita la actualización del mapa mientras el robot explora el entorno siguiendo paredes. Al no haber islas en el entorno, puede crear un mapa completo dando una vuelta completa hasta regresar a la base.

    - La **base del robot** estará al lado de una pared (la orientación de partida del robot es desconocida, una de entre norte, sur, este y oeste).

    - Una zona de interés (amarilla) siempre estará al lado de una pared. Mientras se está creando el mapa, el detectar una zona amarilla no interrumpe el comportamiento de seguir paredes. Igualmente podría no existir ninguna zona amarilla en ese momento sin que altere el comportamiento.

    - Dado que tenemos un entorno discreto (dividido en celdas) y necesitamos crear un mapa de ocupación, **el movimiento se realizará en avances y giros discretos utilizando odometría**, de forma que se conozca en todo momento en qué celda está el robot.

    - En cada celda el robot puede utilizar los **infrarrojos** para detectar los obstáculos a su alrededor y **actualizar el mapa de ocupación**.

    - **Detectar zona de interés (amarilla) con cámara:** se considerará detectada solo si el robot la ve desde muy cerca (aproximadamente a una o dos celdas de distancia). Una detección básica se puede hacer por el tamaño relativo de pixels amarillos respecto a la totalidad de píxeles.

    - Los infrarrojos también detectan las zonas amarillas, por lo que es pasar al lado como con cualquier otra pared.

- **CUANDO EL ROBOT YA DISPONE DE UN MAPA DEL ENTORNO:** se añadirá un comportamiento de planificación de trayectorias que utilice el mapa creado para que el robot regrese de forma eficiente a la base en el momento en que detecte una zona de interés (amarilla).

Enunciado

MAPA

- Se definirá un grid de ocupación binario (celda ocupada / celda libre).
    - El grid de ocupación se puede implementar como una matriz.
- Será actualizado en base a la información de los infrarrojos, que nos permitirán conocer las celdas adyacentes a la que ocupa el robot están libres u ocupadas.
    - Consejo: realizar la lectura de infrarrojos con el robot parado en el centro de cada celda (el robot tiene infrarrojos alrededor de todo su cuerpo).

Enunciado

MAPA

- Es necesario utilizar odometría para poder localizar al robot en el mapa.
    - Se utilizan los encoders de los motores para avanzar celda a celda en el mapa y así poder conocer las coordenadas discretas de posición en el entorno.
- Igualmente se pueden utilizar los encoders de los motores para realizar giros precisos múltiplos de 90º, esperando a que termine el movimiento.
    - Esto puede realizarse con mayor precisión en simulación.
    - En un entorno real sería preferible utilizar el giroscopio.
- Se acumulan menos errores, si se realizan movimientos discretos celda a celda (el robot avanza o gira, se detiene, sensoriza, vuelve a avanzar o girar).

Enunciado

NAVEGACIÓN

- Se utilizará un método de búsqueda heurístico, como A*, para planificar la navegación de regreso a la base utilizando el mapa.
- Si se quiere utilizar otro método de búsqueda, debe proponerse al profesor el método para su aceptación.

Odometría

- Realizando movimientos únicamente de avance recto de una celda y giros en parado múltiplos de 90º la odometría para controlar la posición del robot se simplifica.

- Necesitamos conocer ciertos parámetros para calcular el movimiento angular a aplicar a las ruedas:
  - RADIO_RUEDA = 21 mm
  - ESPACIO_ENTRE_RUEDAS = 105,4 mm (el valor real en el modelo de webots no se corresponde, ajustar entorno a 108,29 mm)
    - RADIO_ENTRE_RUEDAS = ESPACIO_ENTRE_RUEDAS / 2
  - TAMAÑO_CELDA_ENTORNO = 250x250 mm

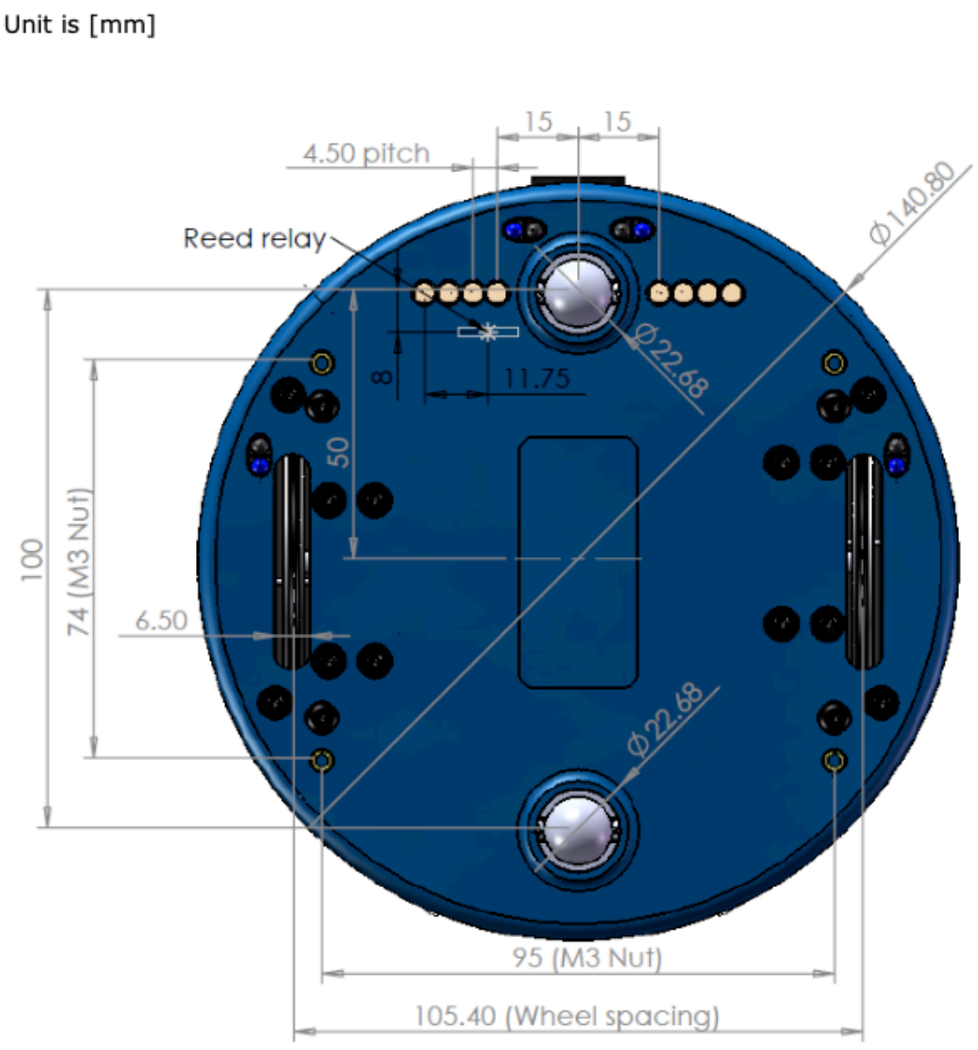
- Por tanto:

  1. Incremento posición angular para un **movimiento recto** (para ambas ruedas):

     delta = distancia_avance_deseada (long. celda) / radio_rueda

  2. Incremento posición angular para **giro de 'angulo_rad' sobre sí mismo** (el valor calculado será positivo para una rueda y negativo para la otra):

     delta = angulo_rad * RADIO_ENTRE_RUEDAS / RADIO_RUEDA



Odometría

- Lectura de la posición actual del los encoders de las ruedas:

```
leftWheel = robot.getDevice('left wheel motor')
rightWheel = robot.getDevice('right wheel motor')

encoderL = robot.getDevice("left wheel sensor”)
encoderR = robot.getDevice("right wheel sensor”)

leftWheel.setPosition(0)
rightWheel.setPosition(0)
leftWheel.setVelocity(0)
rightWheel.setVelocity(0)
```

```
posL = encoderL.getValue()
posR = encoderR.getValue()
```

- Movimiento del robot por posición angular (en vez de por velocidad):

```
leftWheel.setVelocity(CRUISE_SPEED)
rightWheel.setVelocity(CRUISE_SPEED)
leftWheel.setPosition(encoderL.getValue() + incrementL)
rightWheel.setPosition(encoderR.getValue() + incrementR)
```

- Cuando se establece un movimiento por posición, es necesario permanecer comprobando la posición de los encoders hasta que se alcance la nueva posición deseada (se ha finalizado el movimiento). Así se evita ejecutar otro movimiento antes de que el anterior termine.
    - **¡Atención!** Debe aplicarse un pequeño margen de error en la comparación con la posición esperada, ya que esta podría no alcanzarse con absoluta exactitud.

Evaluación

- La práctica se entregará en el Campus Virtual con fecha límite: **17 de abril a las 23:59**
- Un único miembro de cada grupo de prácticas entregará:
  - El código fuente de la arquitectura implementada.
  - Un breve informe que describa la arquitectura diseñada, muestre el mapa generado en las pruebas de funcionamiento y comente las ventajas e inconvenientes de la realización del mapa frente al uso de una arquitectura puramente reactiva.
- El profesor podrá ejecutar la arquitectura de cada grupo sobre un **entorno simulado desconocido** a priori que contendrá paredes y un punto de inicio (base), similar a los entornos proporcionados como ejemplo. El programa debe:
  1. Generar el mapa del entorno.
  2. A continuación, patrullar el entorno y detectar zonas de interés (amarillo).
  3. Planificar y ejecutar el regreso a la base utilizando el mapa generado.

Evaluación

- El controlador deberá implementarse en lenguaje Python o lenguaje C.
- Se valorarán aspectos de ingeniería del software, en particular la modularidad y el desacoplamiento entre comportamientos.
  - Se recomienda comentar con el profesor la solución que se va a implementar.
- Se valorará la generalidad y robustez de la solución.
- En esta práctica no es necesario implementar los comportamientos en threads diferentes.