ROBÓTICA Y PERCEPCIÓN COMPUTACIONAL

ENTREGA: CONTROL

Autores: Jason Felipe Vaz

ÍNDICE

1.Código

1.1. Cómo seguir una línea

Para ello utilizamos uno de los métodos vistos en clase, es decir, usando una función para simplificar todos los casos, la cual, se basa en el análisis del error, cuando mayor error haya, la velocidad disminuirá, mientras que los giros del robot serán mayores, y cuanto menos error haya, la velocidad aumentará mientras que el giro del robot disminuirá.

$$\text{T.V.} = -1 * \text{Error F.V.} = \max(0, 1 - \text{abs}(\text{T.V.} * 1.5)$$

1.2. Cómo evitar un obstáculo

- Utilizamos una variable "esquivaObs" que estará inicializada en False, e irá cambiando a true o false en función de si ha encontrado o no un objeto en el camino.

- Se considerará que hay obstáculo cuando los sensores 3 y 4 ("Frente") detecten un objeto a una distancia inferior de 0.5.

También se ha considerado que habrá obstáculo cuando la agrupación de sensores Front-Left y Front-Right tenga una distancia inferior a 0.5 y 0.6 respectivamente. Esto se ha decidido así, porque en algunos casos cuando el obstáculo se encuentra en la siguiente situación los sensores no detectan el obstáculo y sufrirán un choque.

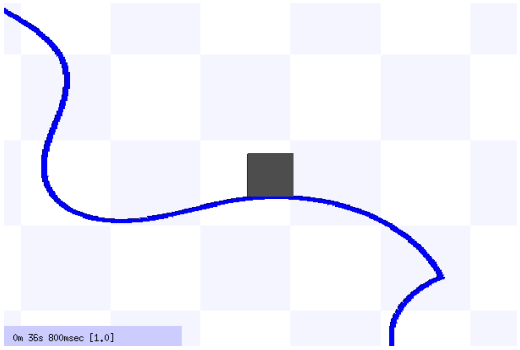
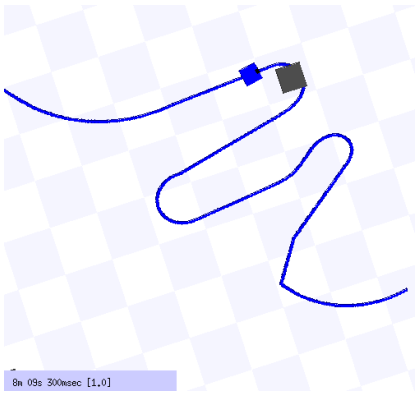


Figura 1: Los sensores 3 y 4 no detectarán el obstáculo

Y también en situaciones de curvas abiertas o cerradas, es necesario contemplar esto:



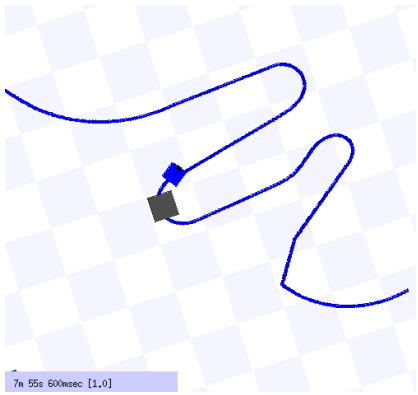


Figura 2 y 3: Sensores 3 y 4 no detecta el obstáculo

Si no se contempla las agrupaciones de Front-Left y Front-Right de los sensores, el robot chocará sin detectar el obstáculo (Figura 2 y 3).

- En el momento que encuentra dicho obstáculo entramos en un estado de "esquivar obstáculo" que consiste en rodearla por la "izquierda" hasta que encuentra otra vez la línea. Esto se explica en 3 pasos:

    1. Cuando empezamos a esquivar primero girará el robot hacia la izquierda sin avanzar.
    2. Una vez consiga colocarse, avanzará girando hacia la derecha con una velocidad y giro regular.

        Además, en este movimiento habrá otros 2 detalles:

        - La variable "esquivaObs" se pondrá a True si no se encuentra la línea, y cambiará a False en caso contrario. Si consigue ponerse a True esta variable saldremos del estado de "esquivar obstáculo" y pasará al estado de "seguir línea".

        - La variable "reposicion" iniciada en False, se activará en este paso únicamente si "esquivaObs" pasa a ser False. Donde, el objetivo de esta variable será reposicionar el robot al salir del estado "esquivar obstáculo" y pasar al estado "seguir línea". Sólo se realizará una vez después de salir de este estado.

3. El último movimiento está pensado especialmente para los bordes de los objetos(obstáculo), y el objetivo es girar el robot hacia la derecha, avanzando lentamente, para no desviarnos del objeto a esquivar.

1.3. Seguir una línea

- A continuación, se activará el estado de "seguir línea" donde se aplicará lo ya explicado anteriormente (*Punto 1*).

Pero además tendrá el detalle de una variable "pierdeLinea", inicializada en False, donde el objetivo de esta variable es guardarse el estado de la variable "foundLine" para casos donde pierde la línea, como el siguiente:

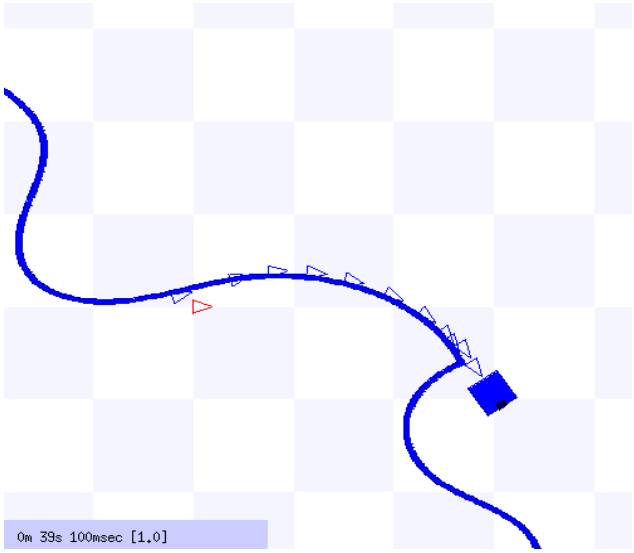


Figura 4: El robot pierde la línea

De tal manera que podrá recuperarse gracias a que se ha guardado el estado de "foundLine", y realizará el movimiento de ir hacia atrás (Figura 5).

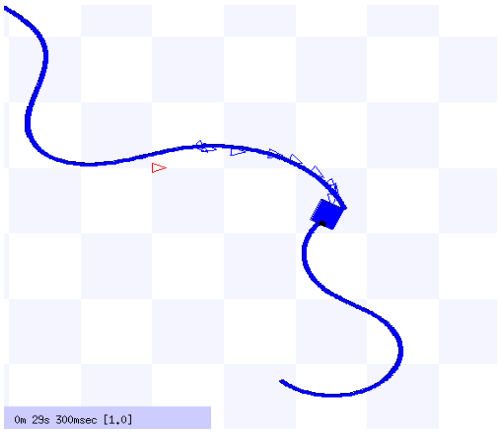


Figura 5: Logra recuperar y no pierde la línea

- Por último, tendremos el caso donde el robot no empieza en ninguna línea, por tanto, efectuará simples movimientos hacia delante sin realizar giros como método de búsqueda de línea.

Comentario: Para este último paso se tenía pensado desarrollar un algoritmo más desarrollado, que consistía en realizar una búsqueda espiral de tal manera que el radio de la espiral crecería hasta encontrar una línea.

El algoritmo se basaba en la siguiente función:

```
x=r*t*sin(t*2*pi*n);
y=r*t*cos(t*2*pi*n);
```

Donde 'n' sería el número de giros, 't' variable auxiliar que variaría entre 0 y 1. Y por último 'r', el radio que iría creciendo.

Pero finalmente, no hubo éxito en esta implementación, por lo que se encuentra en el código como forma de intento, pero comentado.

2.Pruebas

2.1. Primer circuito

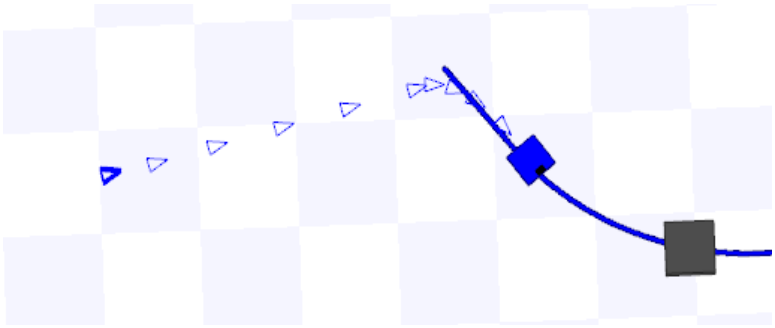


Figura 1

Figura 1. Como podemos ver el robot es capaz de partir de una posición que no tenga línea y avanzar hacia delante hasta encontrar una línea en el camino.

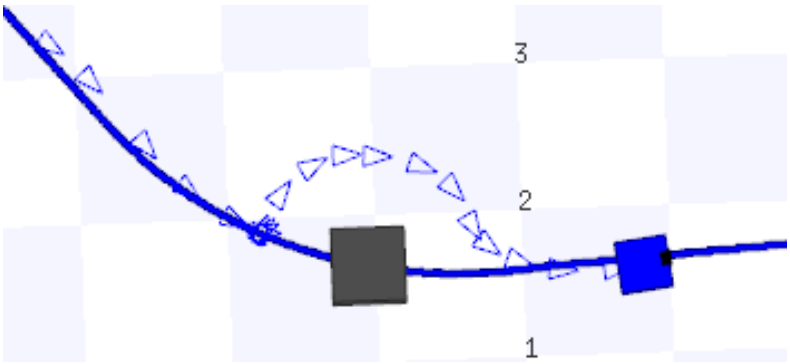


Figura 2

Figura 2. Cuando el robot se encuentra un objeto empieza a girar hacia la izquierda sobre sí mismo hasta que los sensores dejan de detectar dicho objeto, a continuación, avanza y poco a poco va girando hacia la derecha hasta que esquiva el obstáculo y encuentra el camino.

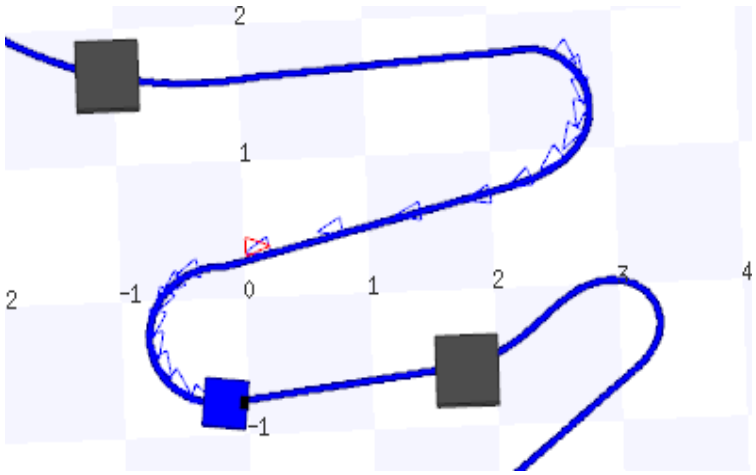


Figura 3

Figura 3. En esta figura podemos observar como el robot toma las curvas sin ningún problema. En función de la curvatura aumenta o decrementa la velocidad como hemos explicado en el apartado uno.

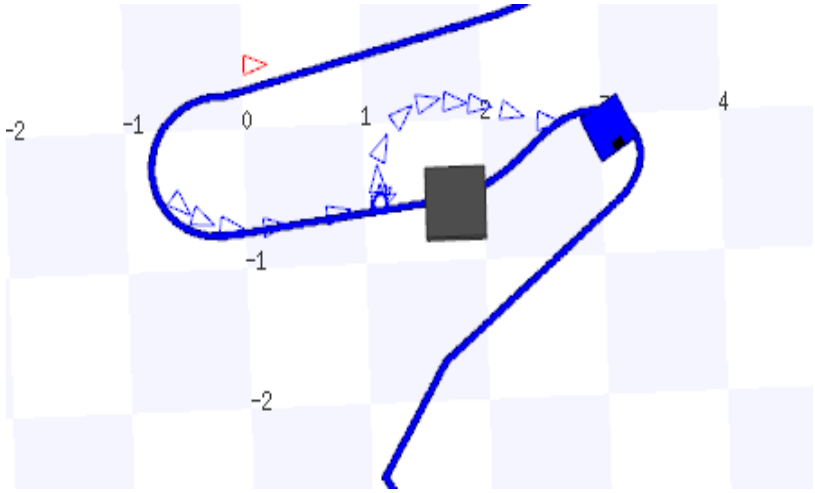


Figura 4

Figura 4. Esta vez el robot esquiva un obstáculo teniendo una línea encima deél, no se confunde de línea cuando está en la parte superior del objeto y sigue el recorrido esperado.

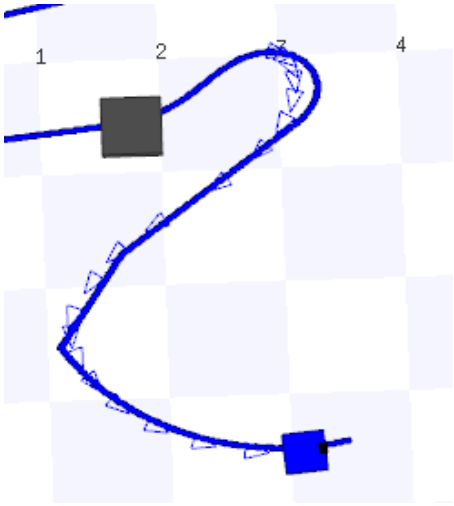


Figura 5

Figura 5. En esta situación, cuando llega una curva cerrada, es capaz de reducir la velocidad y seguir el camino esperado.

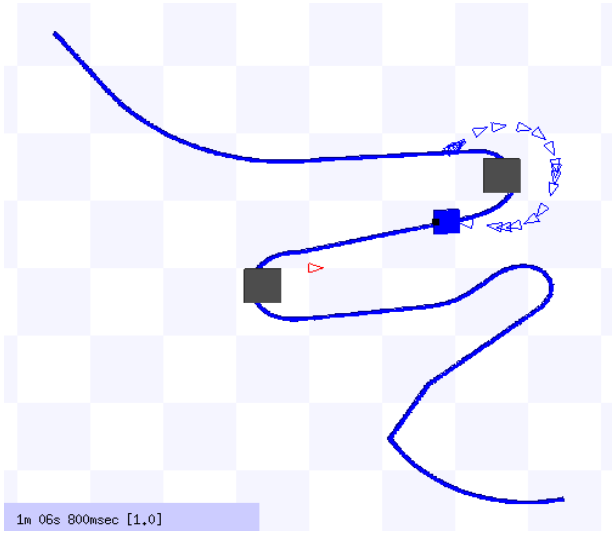


Figura 6

Figura 6. Esta situación ha sido la más complicada, ya que el sensor a veces no detectaba el obstáculo, por lo que hemos tenido que tener en cuenta también los sensores 4,5 y 6. Por tanto, el robot en esta situación detectará elobstáculo, y pasará a esquivar el obstáculo por la izquierda, para finalmentereposicionarse y seguir la línea.

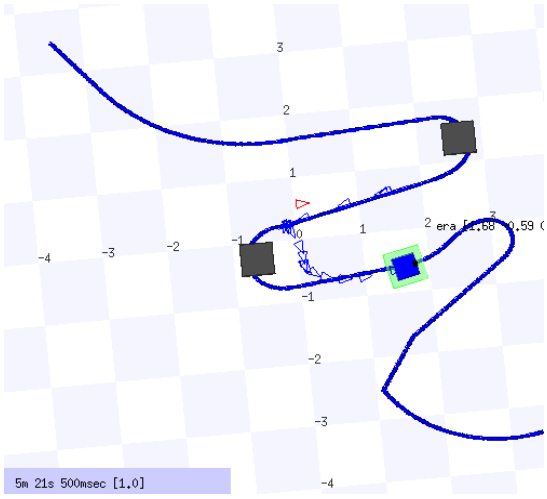


Figura 7

Figura 7. La situación es la misma que la anterior, pero esta vez los sensores 1,2 y 3 serán los encargados de detectar el obstáculo por la izquierda del robot. En esta circunstancia, el robot evade el obstáculo por la izquierda, yantes de llegar a la línea, el robot queda en posición perpendicular a la línea provocando que el robot realice varios movimientos hacia atrás para finalmente recolocarse y poder seguir la línea nuevamente.

2.2. Segundo circuito

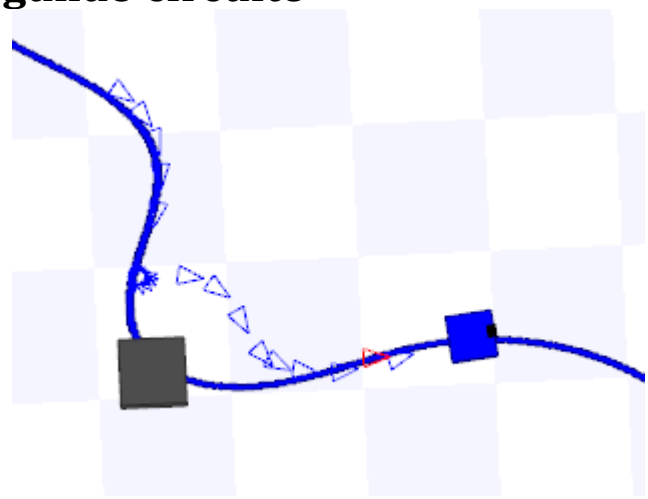


Figura 1

Figura 1. Como podemos observar el robot es capaz también de esquivar un obstáculo, aunque esté en una curva. Empieza a girar sobre sí mismo hacia su izquierda hasta que deja de detectar el objeto y entonces empieza a avanzar esquivando el obstáculo.

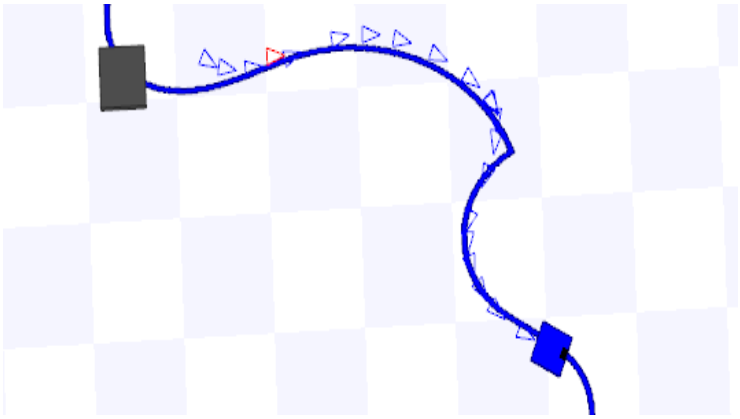


Figura 2

Figura 2. Esta vez, tenemos un giro muy cerrado, cuando el robot llega al vértice, retrocede un poco hacia atrás para reposicionarse y poder tomar la curva con mayor facilidad.

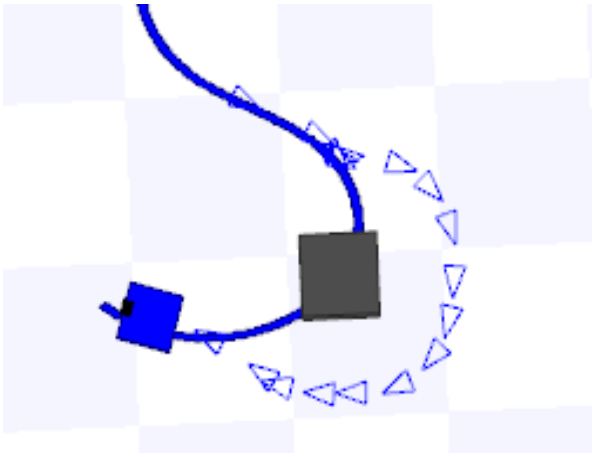


Figura 3

Figura 3. Esta vez ponemos un objeto en una curva en la cual el robot tiene que hacer un mayor recorrido en el momento de bordear el objeto, como podemos observar sigue correctamente el algoritmo implementado

2.3. Tercer circuito

Comentario: Este circuito ha sido creado por el grupo, con el objetivo de comprobar casos más extremos como giros con ángulos muy cerrados y obstáculos de mayor tamaño.

En el zip adjunto se encontrará en "practica1Robotica-v4.2/lineBackground-3.JPG"

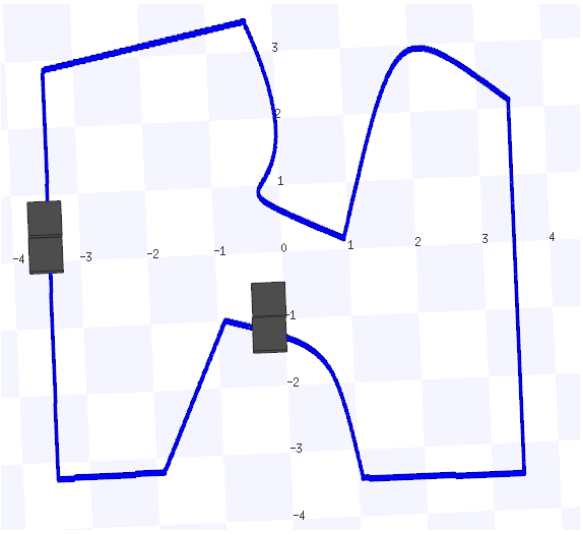


Figura 1

Figura 1. Imagen del circuito cerrado creado, el robot es capaz de dar varias vueltas sin problema

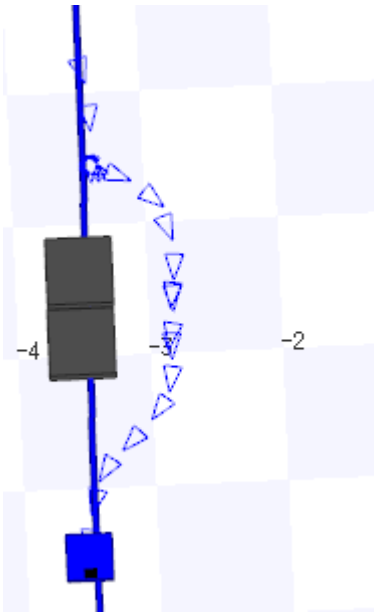


Figura 2

Figura 2. Dos obstáculos seguidos en la línea, el robot es capaz de esquivarlos aplicando el algoritmo diseñado.

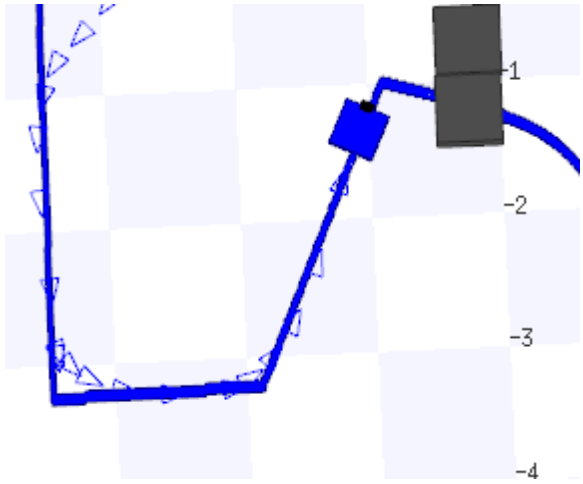


Figura 3

Figura 3. El robot es capaz de hacer giros de 90º, cuando llega al vértice retrocede para poder tomar la curva con mayor facilidad.

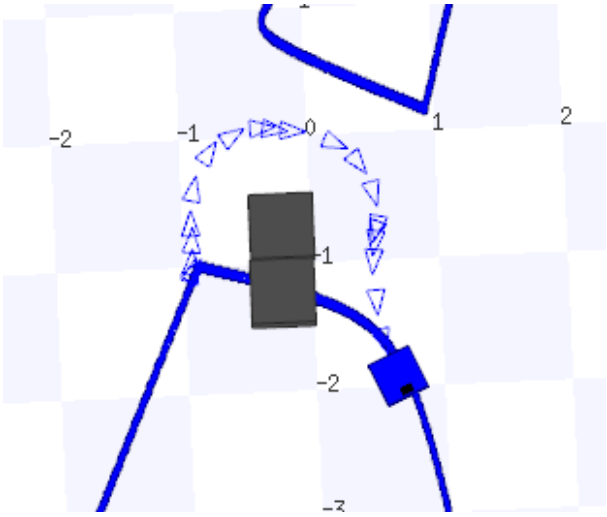


Figura 4

Figura 4. Dos objetos uno encima de otro después de una curva. El robot esquiva los dos objetos y además no se confunde con la línea superior.

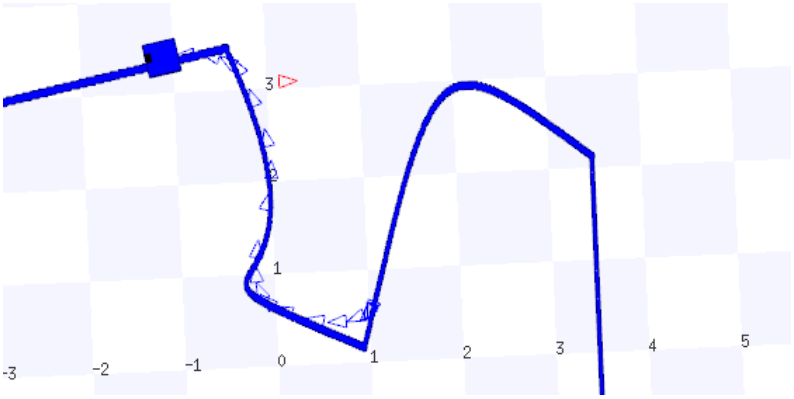


Figura 5

Figura 5: Varias curvas seguidas cerradas, en los casos en que la curva es muy cerrada retrocede y hace la curva de manera más sencilla.

3. Conclusión Final

Como conclusión final cabe destacar que se ha tenido dificultades con el tema de los giros en los obstáculos, sobre todo cuando las curvas eran muy cerradas o muy abiertas y estas contenían obstáculos.

Aunque también ha sido de gran ayuda que se haya trabajado previamente en "seguir la pared" con el código de avoid.py, es más, incluso la manera en que hemos realizado esta práctica es siguiendo la misma metodología de avoid.py, es decir, centrándonos en realizar variables que sirvan de estado, y comprobando siempre si el robot provocará algún choque, antes de seguir la línea.

Cabe destacar que se ha intentado que el código sea lo más simple y legible posible, así como la realización de comentarios que se veían necesarios, además hemos dejado comentados algunos "prints" que han sido fundamentales para la realización de esta práctica y que creemos que es necesarios que se mantengan.

En cuanto, a las pruebas realizadas, hemos probado el máximo posible de situaciones con el robot y obstáculos, e incluso giros más extremos en los circuitos, quizás lo único que falta por cubrir es para obstáculos que vayan más allá de la forma de un cuadrado, como un círculo o incluso un triángulo.

Por último, quiero destacar que habrá una parte en el código que se encontrará comentada, que es la parte de realización de la búsqueda espiral, la cual, no se ha logrado completar, debido a que se había metido un bucle largo en la función de "step", y esto provocaba problemas con las detecciones de la línea, ya que no se estaba realizando el movimiento espiral por trozos sino de golpe.