Documento en proceso de elaboración

**UNIVERSIDAD SANTO TOMÁS**

**FACULTAD DE INGENIERÍA ELECTRÓNICA**

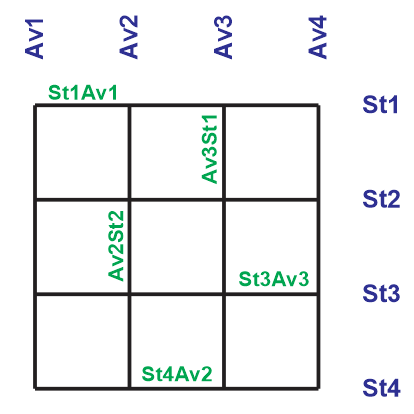
**LABORATORIO FINAL**

**Visión Artificial**

**Introducción a la Inteligencia Artificial**

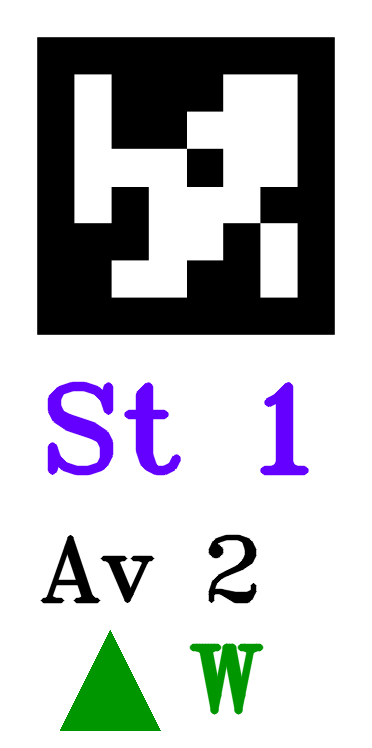
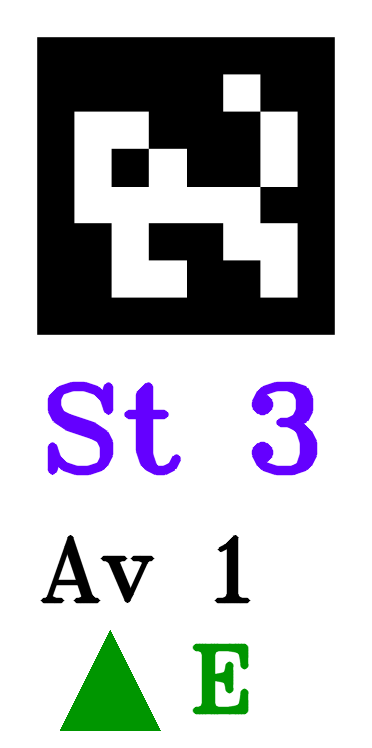
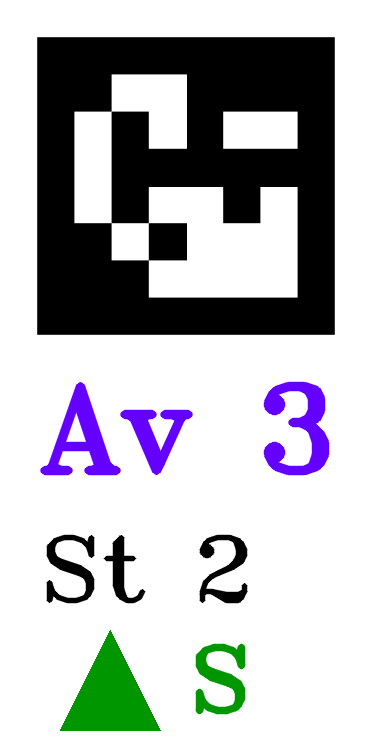
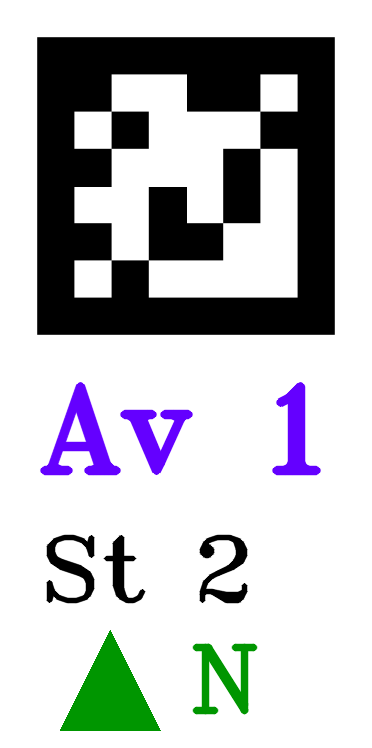
Este laboratorio está basado en el simulador para la plataforma de enseñanza de inteligencia artificial *Duckietown*, disponible en <https://github.com/duckietown>, sin embago, ha sido modificado para el propósito de las asignaturas en cuestión.

Los mapas crean ciudades con la siguiente estructura:



Las avenidas (av) siempre van hacia el norte N o hacia el sur S, y las calles (st) siempre van hacia el oriente E o hacia el occidente W. Una cuadra se denomina según la avenida o calle sobre la cual está, y sobre su perpendicular de menos valor (veáse la imágen de ejemplo).

El vehículo debe circular por la derecha de la calle, por lo cual, a la derecha de la calle de cada cuadra, encontrará una señal indicando la dirección de la misma. A continuación algunos ejemplos:



1. Está sobre la avenida 1, entre calles 2 y 3, mirando hacia el norte.
2. Está sobre la avenida 3, entre calles 2 y 3, mirando hacia el sur.
3. Está sobre la calle 3, entre avenidas 1 y 2, mirando hacia el oriente.
4. Está sobre la calle 1, entre avenidas 2 y 3, mirando hacia el occidente.

Cada señalización cuenta en la parte superior con una Etiqueta April tag16h11 (<https://april.eecs.umich.edu/software/apriltag>), cuyo ID es calculado así:

Mirando hacia el norte: idx = 100 + 5\*(av-1) + (st-1)

Mirando hacia el sur: idx = 200 + 5\*(av-1) + (st-1)

Mirando hacia el oriente: idx = 300 + 5\*(av-1) + (st-1)

Mirando hacia el occidente: idx = 400 + 5\*(av-1) + (st-1)

El vehículo arrancará en una ubicación aleatoria en el mapa, para el caso de la solución para *visión artificial*, éste debe ubicarse a partir de la señalización. Para *inteligencia artificial*, la ubicación, entre otros parámetros, serán entregados cada fotograma.

El objetivo es crear el código necesario para lograr que el vehículo llegue a una dirección predefinida.

Restricciones:

* Andar por la derecha
* No salirse de las calles

**CÓDIGO**

Para las pruebas y la calificación, siempre se ejecutará el archivo *usta\_test.py*, así (después de haber inicializado el entorno de conda y las variables de entorno necesarias):

python usta\_test.py --map-name city\_3x3\_s1 --mode ai --target av2st1

El nombre del mapa puede cambiar, los disponibles son:

* city\_3x3\_s1 : Ciudad de 3x3 cuadras, con 1 espacio por cuadra.
* city\_4x4\_s1 : Ciudad de 4x4 cuadras, con 1 espacio por cuadra.
* city\_5x5\_s1 : Ciudad de 5x5 cuadras, con 1 espacio por cuadra.
* city\_3x3\_s2 : Ciudad de 3x3 cuadras, con 2 espacios por cuadra.
* city\_4x4\_s2 : Ciudad de 4x4 cuadras, con 2 espacios por cuadra.
* city\_5x5\_s2 : Ciudad de 5x5 cuadras, con 2 espacios por cuadra.

Existen dos modos de ejecución: ***ai*** inteligencia artificial, ***av*** visión artificial, la diferencia se discutirá más adelante.

El nombre del destino (*target*) puede ser, por ejemplo, av5st4, st2av4, etc.

Dicho archivo no se puede modificar, así como ningún otro archivo de la librería. Todo el código a realizar por los estudiantes debe escribirse en el archivo *usta\_sol.py*, el cual contiene la implementación de la clase *UstaSolution*. Se pueden crear los métodos que se consideren necesario, sin embargo es necesario que existan los siguientes:

***\_\_init\_\_*:** Constructor, recibirá como parámetros un string correspondiente al destino, y un diccionario con la información completa del archivo *yaml* del mapa.

***step\_av:*** en el modo ‘visión artificial’, es el método que se ejecuta en cada fotograma, recibe como parámetro dicho fotograma, y debe retornar la acción a tomar por parte del vehículo.

***step\_ai:*** en el modo ‘inteligencia artificial’, es el método que se ejecuta en cada fotograma, debe retornar la acción a tomar por parte del vehículo y recibe los siguientes parámetros:

* *obs*: fotograma capturado por la cámara del vehículo.
* *coord:* coordenada actual en el mapa.
* *dist:* distancia al centro del carril **(más no de la vía)**
* *angle:* ángulo respecto al centro del carril (en grados).
* *global\_angle:* ángulo respecto al mapa (E:0°, N:90°, W:180°, S:270°)

Se puede agregar la bandera *--manual-control* con el fin de controlar el robot con el teclado e ignorar la acción producida por solución:

python usta\_test.py --map-name city\_3x3\_s1 --mode ai --target av2st1 --manual-control

Si considera que el constructor o el alguno de los métodos *step* deberían incluir algún argumento adicional, informe al profesor, y si se considera acertado, éste será incluido y todo el curso será informado.

Si encuentra algún bug o error en el código, informar al profesor de inmediato.

**CALIFICACIÓN**

Para calificar el laboratorio, se ejecutará en el computador del profesor, reemplazando únicamente el archivo ***usta\_sol.py***.

Pendiente...

**REGISTRO DE CAMBIOS**

**v0.3:**

* Máximo número de pasos pasa de ser 1500 a 50000
* Cambian levemente las dimensiones del robot, para evitar tantas colisiones y salidas de la carretera
* Agregado parámetro *global\_angle* (revisar documentación) al método *step\_ai* de la clase *UstaSolution*.

**v0.2**

* El nombre del mapa como argumento es obligatorio
* Creada bandera para control manual
* Parámetro para modo de operación, según la asignatura
* Parámetro para el destino
* En la clases ***UstaSolution*** se crearon los métodos ***step\_ai*** y ***step\_av***, y se eliminó ***step***.
* Creados nuevos mapas: city\_3x3\_s1, city\_4x4\_s1, city\_5x5\_s1, city\_3x3\_s2, city\_4x4\_s2, city\_5x5\_s2.

**v0.1**

* Archivos innecesarios para el laboratorio eliminados
* Creado archivo de prueba
* Creado archivo de solución
* Creadas nuevas texturas para señales de dirección
* Creado primer mapa de prueba