Duckietown Engineering

Map conventions for DP6a

This is the set of conventions we use to create and save a map of Duckietown for localization (DP6a) purposes.

Contents

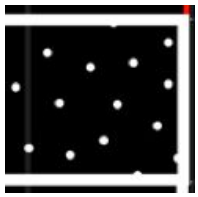
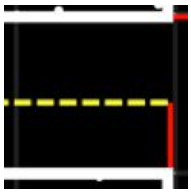
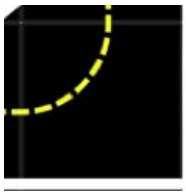
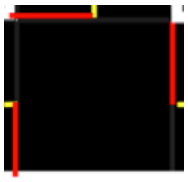
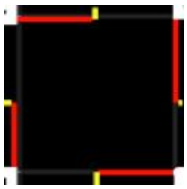
Overall information structure

All the information about a particular Duckietown that is needed for localization is saved in two separate .csv files:
duckietown_tile_map.csv contains information about the dimensions (in terms of tiles) of Duckietown; for every tile, it contains the tile's position, type, and orientation.
duckietown_tag_map.csv contains information about all the april tags (currently only intersection tags with ID's 1-124) in Duckietown: (the tag's position and orientation with respect to a tile).
Neither of these contain metric information. The idea is that someone building a (physical) Duckietown can fully describe it using these two files without having to measure anything. The metric information (which matches the Duckietown convention) is saved in the param file `catkin_ws/src/duckietown/config/baseline/duckietown_description/default.yaml`

duckietown_tile_map.csv

- Comma-separated
- 4 columns: [int x, int y, int rotation, string tile_type]
- The first row contains the headers "x, y, tile_type, rotation" (this is for clarity only, the code skips this line)
- Each subsequent row defines a tile in the map.
- Every map is rectangular (if it isn't we define empty tiles as needed) with dimensions **n rows** by **m columns** of tiles
- The **origin** (0, 0) is the bottom left of the map

- The point (**x, y**) is the vertex defined by the boundary between up to 4 tiles and has x total tiles to its left and y total files under it.
- Each coordinate pair (**x, y**) describes the tile in the first quadrant wrt (**x, y**) i.e. above and to the right. So, for example (0, 0) describes the left-most, bottom tile.
    - **x** ranges from **0** to **m**
    - **y** ranges from **0** to **n**
- The **tile_type** column describes what type of tile it is. The possible types are:

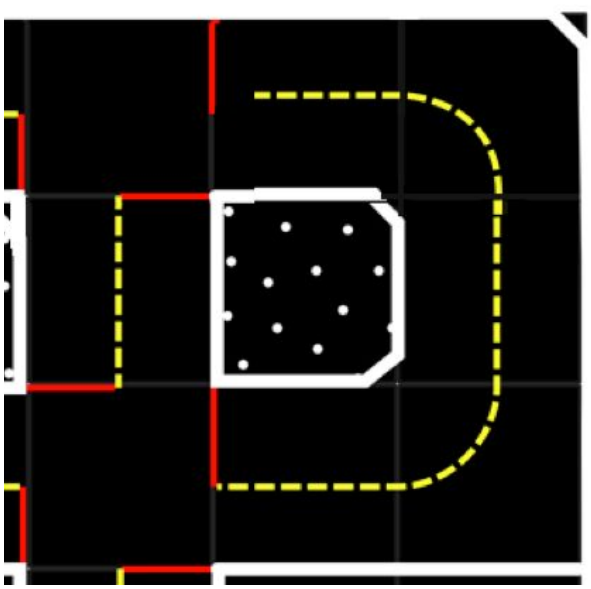
‘empty’	‘straight’	‘turn’	‘3way’	‘4way’
				

- The **rotation** column specifies how many degrees the tile is rotated **counterclockwise** according to the following convention:
    - For ‘empty’ the rotation is **always** 0.
    - For ‘straight’ the rotation is 0 if a car can traverse it **horizontally** or 90 if **vertically**
    - For ‘turn’ the rotation is
        - 0 if it is a **left turn** for a car coming **from the left**.
        - 90 if it is a **left turn** for a car coming **from the bottom.**
        - 180 if it is a **left turn** for a car coming **from the right.**
        - 270 if it is a **left turn** for a car coming **from the top.**
    - For a ‘3way’ the rotation is 0 if cars can enter from the **left, right**, and **top** like an inverted T.
    - For a ‘4way’ the rotation is **always** 0
- The 4 possible rotations are 0, 90, 180, and 270

Example:

As an example, the following table is the **duckietown_tile_map.csv** for the Duckietown section below

x	y	tile_type	rotation
0	0	4way	0
0	1	straight	90
0	2	3way	180
1	0	straight	0
1	1	empty	0
1	2	straight	0
2	0	turn	0
2	1	straight	90
2	2	turn	90

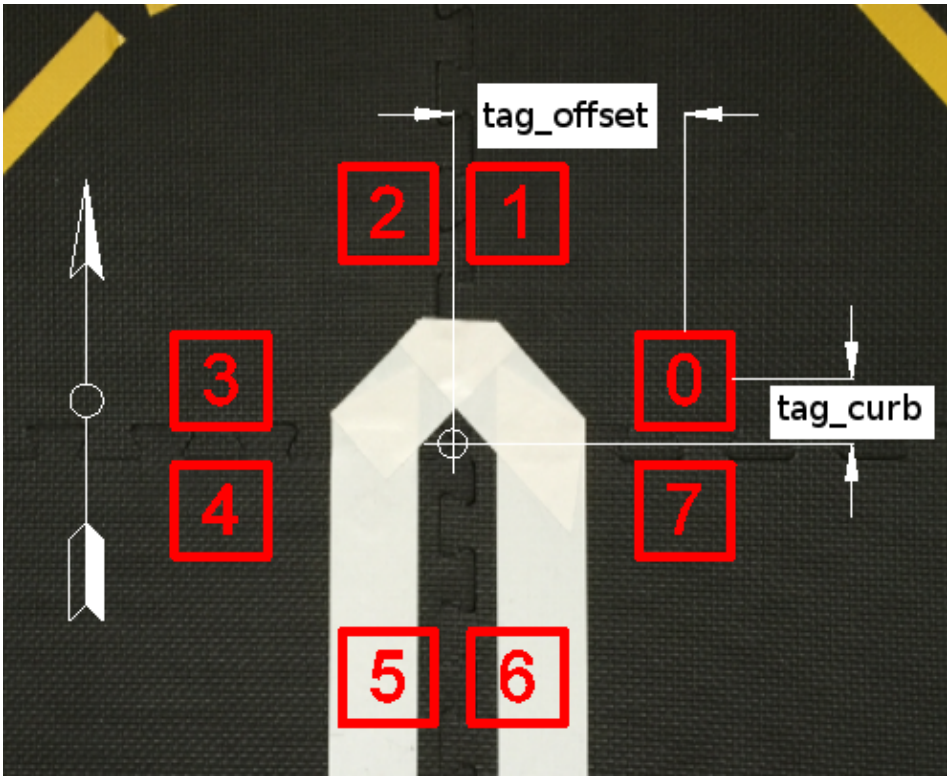


duckietown_tag_map.csv

Note: currently the duckietown description code only uses intersection tags (those with ID's 1-124) so although other tags such as street names can be seen by the robot, the csv file should only contain the intersection tags.

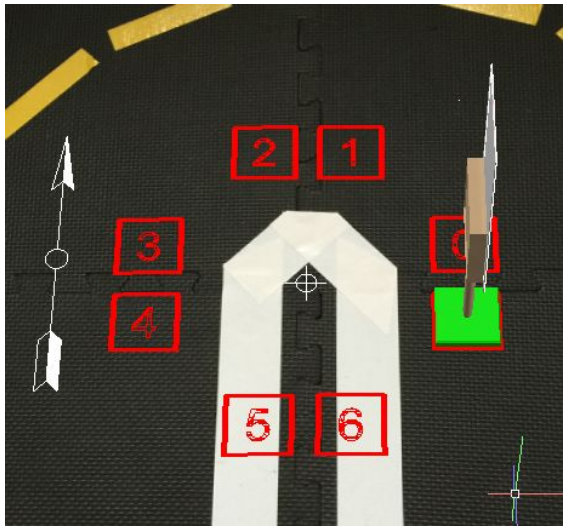
- Comma-separated
- 5 columns: [int tag_ID, int x, int y, int position, int rotation]
- The first row contains the headers " tag_ID, x, y, position, rotation" (this is for clarity only, the code skips this line)
- Each subsequent row defines one tag
- **tag_ID** is the unique (within a Duckietown) id number every tag has printed in it.
- The columns **x** and **y** define the global position of the tag in Duckietown.
    - The meaning of the pair (**x, y**) here matches that of (**x, y**) in duckietown_tile_map.csv but the maximum value of **x** and **y** here is one more than the maximum value of **x** and **y** in duckietown_tile_map.csv to allow for tags on the last edge to the right and top
        - **x** ranges from **0** to **m+1**
        - **y** ranges from **0** to **n+1**
    - Note: This is convenient because most tags (**all**, as of the date of creation of this document) are placed right on or next to a vertex in the boundary between four tiles so it is obvious which (**x, y**) the tag belongs to since the pairs (**x, y**) describe vertices.
- **Position** is an integer that defines where the tag is with respect to the vertex (**x, y**)
  The current duckietown convention is that **tag_offset**=0.09 and **tag_curb**=0.035. Refer to the pictures below. These values are set in:
  `catkin_ws/src/duckietown/config/baseline/duckietown_description/default.yaml`

- The eight possibilities are:

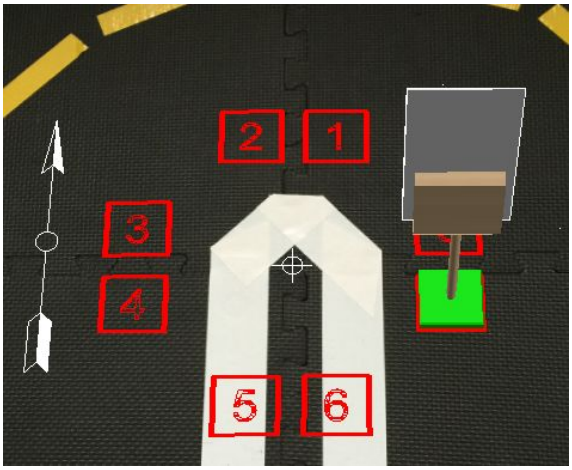


- This can easily be augmented to more positions if it becomes necessary

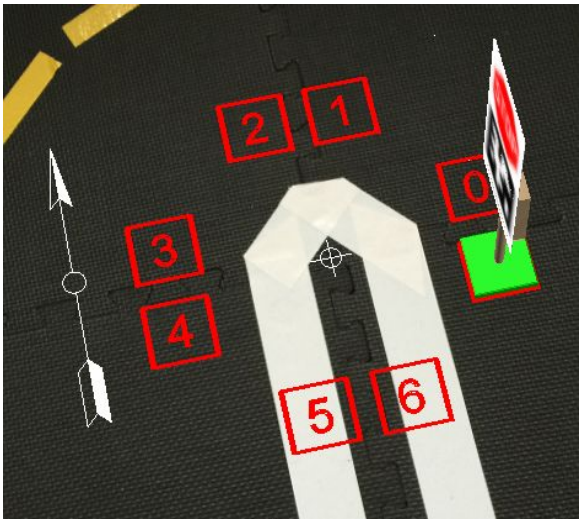
- **Rotation** is an integer that describes which way the tag is facing
  - The four possible values are:
    - 0 if the tag is facing towards the **right**



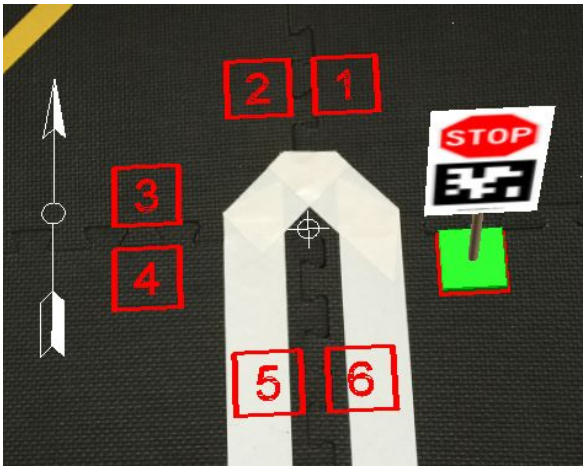
- 90 if the tag is facing towards the **top**



- 180 if the tag is facing towards the **left**

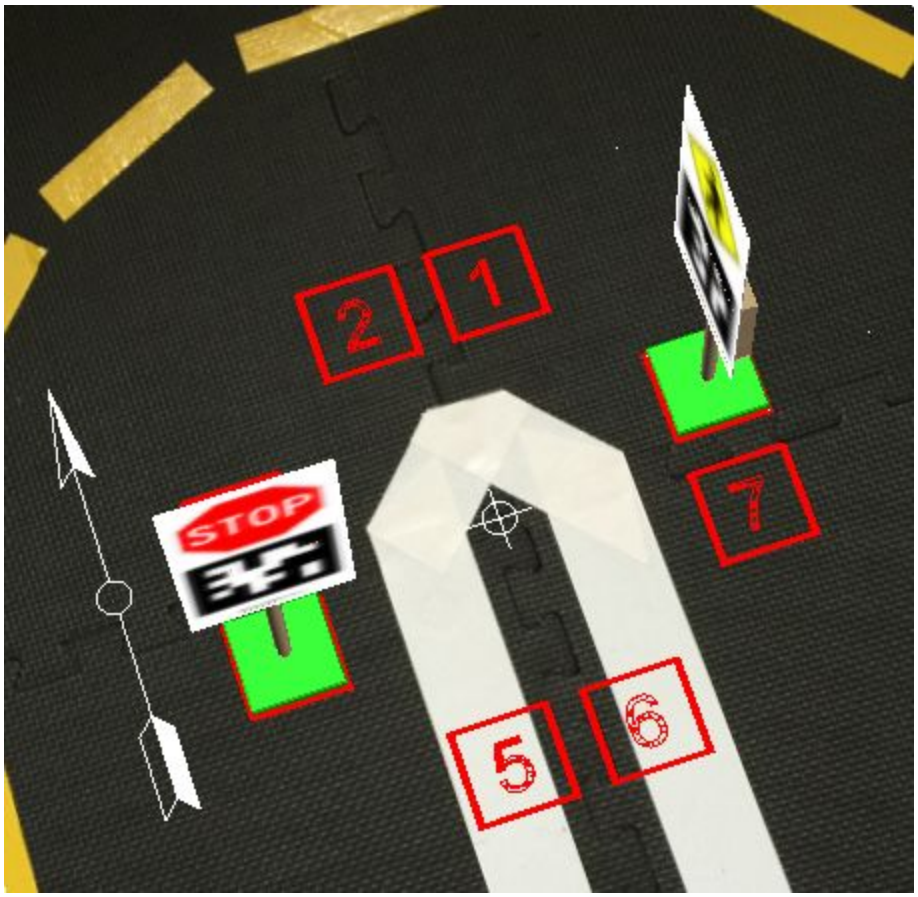


- 270 if the tag is facing towards the **bottom**



Example:

As an example, consider the following picture:



Assume that the vertex we're seeing is **(1, 5)**, the tag ID of the Stop sign is **100** and the tag ID of the pedestrian sign is **150**. Then these two tags appear in duckietown_tag_map.csv like this:

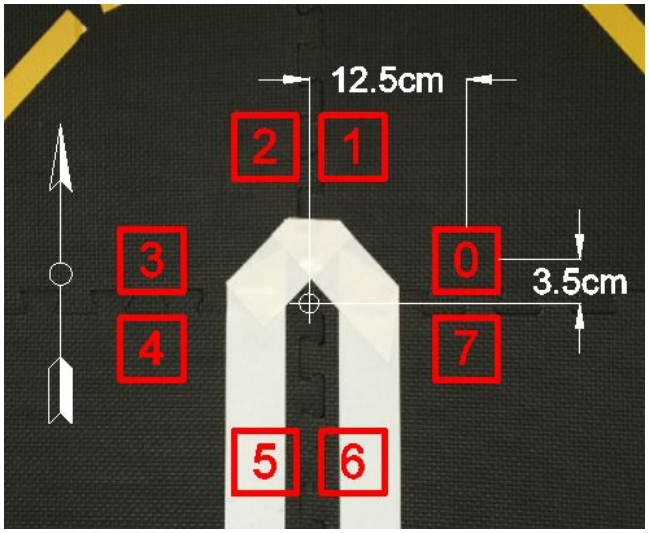
tag_ID	x	y	position	rotation
100	1	5	4	270
150	1	5	0	180

Note that the positions of the tags with respect to the tape in this case is obviously **not valid.** They should be outside the lane in an actual Duckietown. Otherwise, accidents will happen and duckies will get injured.

Generating the Map File (on laptop)

After creating the two CSV files described in the previous section, generate a map file with the following steps:

1. Make sure the values **`tag_offset`** and **`tag_curb`** in the config file `catkin_ws/src/duckietown/config/baseline/duckietown_description/default.yaml` match the offsets of the tags.
   As an example, if the tag positions are placed according to the distances below the config file will have: **`tag_offset: 0.125`** and **`tag_curb: 0.035`**



2. Run the following command:
   `laptop $ make openhouse-dp6a-generate-map-<map_name>`
   where `<map_name>` is a simple name that describes the current duckietown e.g. `map226`

   a. This should prompt you for the paths to the CSV files created and create the map file in `catkin_ws/src/duckietown_description/urdf/<map_name>.urdf.xacro`

   Note: It is normal to see a “Node has died” message after successful completion of map generation

3. To verify that the map was generated correctly, run:
   `laptop $ roslaunch duckietown_description duckietown_description_node.launch veh:=<some_vehicle> map_name:=<map_name>`

   a. It should display the correct map in rviz. Verify that all the tiles and tags are in the right place.

Note: The generation step only needs to be done once. The resulting .xacro file can be pushed and used by duckietown_description to publish all the transforms for all the tiles and tags described in the .xacro file. It is also used by rviz for the visualization.