Mapviz

https://swri-robotics.github.io/mapviz/
(if you're compiling Mapviz from source using git, then you likely will have to go through the commit history to checkout a build that passes all compile checks)

Getting Orthoimagery

Download NYS orthoimagery from:
https://orthos.dhses.ny.gov/#

Bulk Downloads (not recommended if you only need a handful of tiles, use above link to download only needed tiles):
https://gis.ny.gov/st-lawrence-county-orthoimagery-downloads

Misc NYS orthoimagery info:
https://gis.ny.gov/orthoimagery
https://gis.ny.gov/orthoimagery-faqs

.jp2 files contain imagery and location information

tile_map

Install GDAL:
https://github.com/OSGeo/GDAL

(needed if using .jp2 files for orthoimagery)
https://github.com/uclouvain/openjpeg

Follow this guide to set up a local tile_map.
https://swri-robotics.github.io/mapviz/guides/local_tile_map_imagery/

Guide notes:

- Max zoom value for Mapviz can be found in the .mapml file generated by gdal2tiles.py
- Set the local xy origin to the location you will be working in
    - (x = longitude, y = latitude)
    - (keep note of the value set, needed when creating keepout filter later)
    - (doesn't need to be exact, ideally somewhere in the center of your map)
    - (coords can be found with something like google maps)
- If you're working with .jp2 files, then you don't need to worry about removing transparency

Gazebo

Gazebo ROS (fortress is recommended for ROS 2 Humble):
https://gazebosim.org/docs/fortress/ros_installation/

Gazebo Ubuntu install instructions:
https://gazebosim.org/docs/fortress/install_ubuntu/

Install ros_gz (or compile from source):
https://github.com/gazebosim/ros_gz
(ros_gz github humble branch):
https://github.com/gazebosim/ros_gz/tree/humble

TurtleBot3

Github for gazebo turtlebot with GPS
https://github.com/arcater/turtlebot_gps

TurtleBot3 gazebo manual
https://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/

TurtleBot3 simulations github
https://github.com/ROBOTIS-GIT/turtlebot3_simulations/tree/humble

Nav2 currently outputs Twist messages for cmd_vel to control the robot. Turtlebot3 in gazebo only accepts TwistStamped messages for robot control. Use this package to stamp the twist messages coming out of Nav2. Follow tutorial for implementing Nav2 collision monitor, but pass velocity commands through the twist_stamper instead
https://github.com/joshnewans/twist_stamper
https://docs.nav2.org/tutorials/docs/using_collision_monitor.html

Control gazebo turtlebot from command line
https://index.ros.org/r/teleop_twist_keyboard/

```
ros2 run teleop_twist_keyboard teleop_twist_keyboard --ros-args -p stamped:=true
```

Misc info for how to add gps to turtlebot

Gazebo sensors (for adding gps to turtlebot)
https://gazebosim.org/libs/sensors/

Spherical Coordinates
https://gazebosim.org/api/sim/9/spherical_coordinates.html

https://github.com/gazebosim/gz-sim/blob/ign-gazebo6/examples/worlds/spherical_coordinates.sdf
https://gazebosim.org/api/sim/8/classgz_1_1sim_1_1systems_1_1NavSat.html

Get gazebo launch params
`ign gazebo --help`

Changing sensor frame_id:
https://robotics.stackexchange.com/questions/29242/is-is-possible-to-change-frame-id-of-a-sensor

```
<ignition_frame_id>custom_name_here</ignition_frame_id>
```

robot_localization

Robot localization
https://docs.ros.org/en/noetic/api/robot_localization/html/index.html

Robot localization initial position
https://docs.ros.org/en/noetic/api/robot_localization/html/state_estimation_nodes.html#initial-state

Misc links for automatically getting initial position:

Geonav_transform (helpful python script to convert lat/lon coordinates to xy coordinates)
https://github.com/bsb808/geonav_transform
https://github.com/bsb808/geonav_transform/blob/master/src/geonav_transform/geonav_conversions.py

Python minimal subscriber example
https://github.com/ros2/examples/blob/humble/rclpy/topics/minimal_subscriber/examples_rclpy_minimal_subscriber/subscriber_lambda.py

Nav2

GPS tutorial
https://docs.nav2.org/tutorials/docs/navigation2_with_gps.html

GPS demo files
https://github.com/ros-navigation/navigation2_tutorials/tree/rolling/nav2_gps_waypoint_follower_demo

Controller Server
https://docs.nav2.org/configuration/packages/configuring-controller-server.html

costmap params
https://docs.nav2.org/configuration/packages/costmap-plugins/obstacle.html
https://docs.nav2.org/configuration/packages/configuring-costmaps.html
https://docs.nav2.org/tuning/index.html

Nav2 Keepout filters

https://docs.nav2.org/tutorials/docs/navigation2_with_keepout_filter.html

Creating a Keepout filter that lines up with orthoimagery

Install QGIS
https://www.qgis.org/en/site/forusers/download.html

Create new Project
Project > New

Set Project Reference System (CRS) to "EPSG:4326 - WGS 84"
Project > Properties…

Import your orthoimagery:
Layer > Add Layer > Add Raster Layer…

Source Type: File
Source: orthoimagery files
For jp2 options, leave everything as default

Export image with georeference info
(ensure project CRS is still WGS 84)
Project > Import/Export > Export Map to Image…

Extent: Draw on canvas the area you want to export
Resolution: adjust DPI until the output image is at a resolution where you can reasonably draw your paths on it
Disable Draw annotations and decorations
Enable "append georeference information"
Save as png file
This gives you a .png image and a .pgw world file.

Drawing the Keepout Filter

Use an image editing software (one that allows you to scale the image horizontally to a specific pixel count) to draw your keepout filter over the map image. Save the keepout filter as a (seperate) png file.

yaml parameters and scaling the image

World files:
https://en.wikipedia.org/wiki/World_file

The world file can be used to calculate the origin and resolution info needed for Nav2. The map image output by QGIS needs to be scaled down horizontally so it lines up in nav2.

nav2yaml_calculator.py can be used to output the resolution and origin parameters for the yaml file needed by Nav2, as well as how much you need to scale your image. Download https://github.com/bsb808/geonav_transform/blob/master/src/geonav_transform/geonav_conversions.py into the same directory as nav2yaml_calculator.py to run.

Yaml parameters explanation:

- Resolution: how much distance a pixel of the image covers. Resolution 1.0 means one pixel covers a 1x1 area on the map
- Origin [a, b, c]
  - a: positive = east, negative = west
  - b: positive = north, negative = south
  - c: always 0

RVIZ satellite

RVIZ plugin to add satellite overlay.

https://github.com/nobleo/rviz_satellite

AerialMap Settings:
Object URI: same as mapviz base URL but replace “{level}” with “{z}”
Zoom: same as from mapviz
Timeout: 0

Nav2 Humble development progress

As of the time of writing, the Nav2 developers are currently updating the Humble branch of Nav2 to include all the features currently on the main branch. One of these features that could be used to keep the robot on the road/sidewalk is the route_server. Since the route_server was not yet implemented for ROS 2 Humble, a keepout_filter was used instead.

https://github.com/ros-navigation/navigation2/commits/humble_main/
https://github.com/ros-navigation/navigation2/issues/4998
https://github.com/ros-navigation/navigation2/pull/5017

Nav2 Route Server

route_server
https://docs.nav2.org/tutorials/docs/route_server_tools.html

route_example_launch.py
https://github.com/ros-navigation/navigation2/blob/main/nav2_simple_commander/launch/route_example_launch.py

Route server configuration guide
https://docs.nav2.org/configuration/packages/configuring-route-server.html

Navigator API
https://docs.nav2.org/commander_api/index.html

Creating Route Graphs

Nav2 QGIS tutorial
https://docs.nav2.org/tutorials/docs/route_server_tools/route_graph_generation.html#route-graph-generation

Tutorial notes:
- Set CRS to "EPSG:4326 - WGS 84" so coordinates lineup with Navsatfix coordinates
- Step2 only needed when not working with orthoimagery
- Step5 seems completely unnecessary for exporting geoJSONs

QGIS scripts used by above tutorial
https://github.com/ros-navigation/navigation2/tree/main/nav2_route/graphs/scripts

Web tool for creating route graphs (just use QGIS)
https://docs.nav2.org/tutorials/docs/route_server_tools/route_graph_generation_lif_editor.html#route-graph-generation-lif-editor

Open license street data
https://www.openstreetmap.org/