Instruction

Prerequisite:

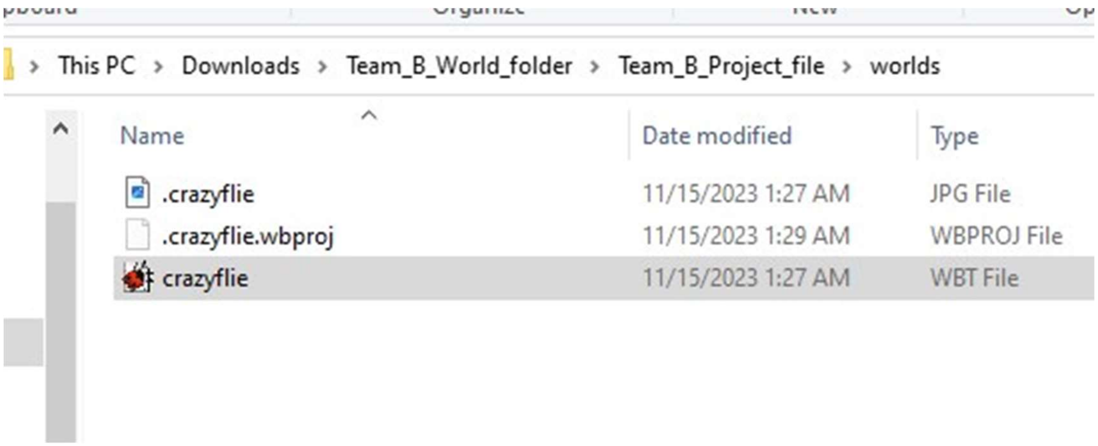
Webots installed on your system and Numpy library.

Follow the steps to install numpy in python installed windows system.

- Open Command Prompt in windows in administrator mode
- type command “pip install numpy”

Step 1

Open the world file as named crazyflie in world folder.

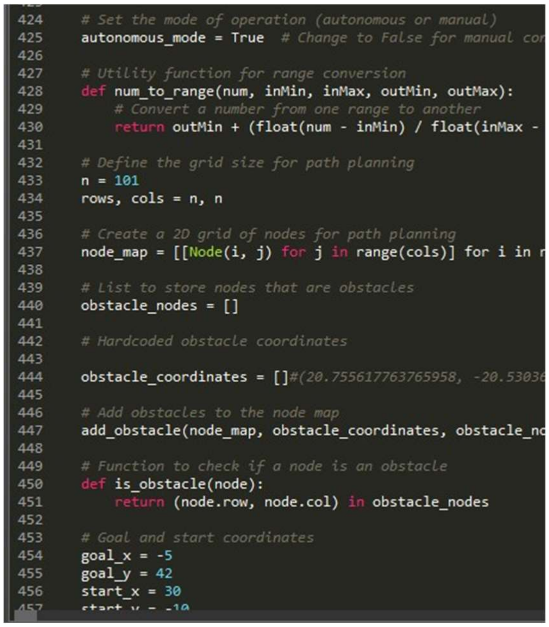


This PC › Downloads › Team_B_World_folder › Team_B_Project_file › worlds

Name	Date modified	Type
.crazyflie	11/15/2023 1:27 AM	JPG File
.crazyflie.wbproj	11/15/2023 1:29 AM	WBPROJ File
crazyflie	11/15/2023 1:27 AM	WBT File

Step 2

Update the Map size (n) at line 433 in controller file name (AutonomousDroneNavigation) in AutonomousDroneNavigation folder of controller folder, or update in controller text editor of webots simulator.

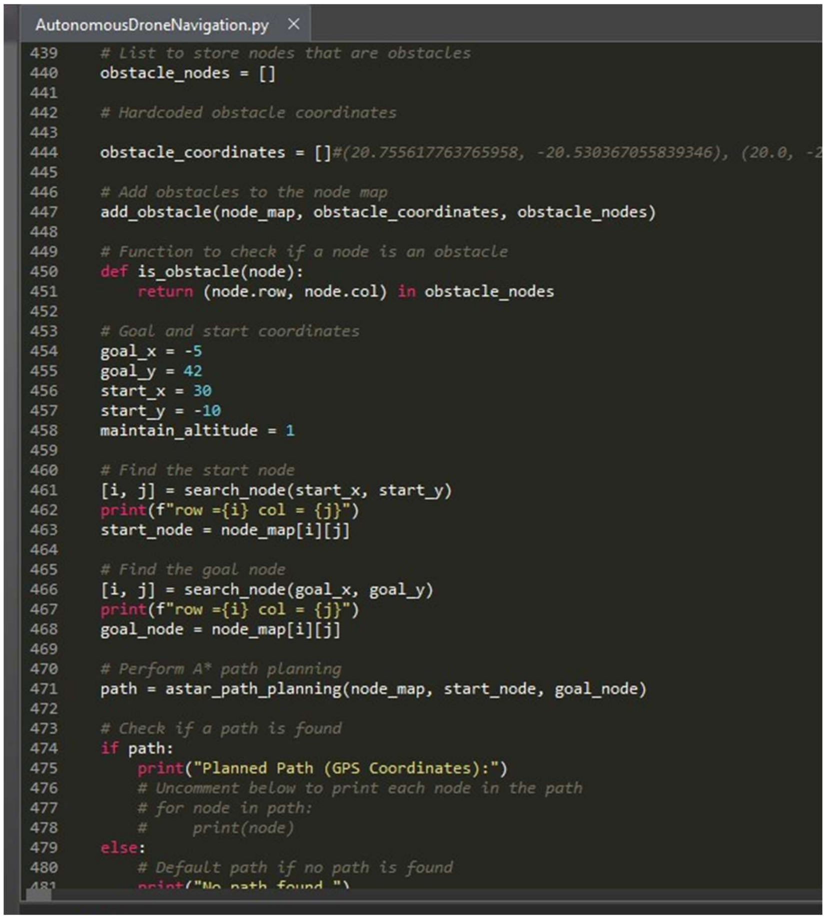


```
424    # Set the mode of operation (autonomous or manual)
425    autonomous_mode = True  # Change to False for manual cor
426
427    # Utility function for range conversion
428    def num_to_range(num, inMin, inMax, outMin, outMax):
429        # Convert a number from one range to another
430        return outMin + (float(num - inMin) / float(inMax -
431
432    # Define the grid size for path planning
433    n = 101
434    rows, cols = n, n
435
436    # Create a 2D grid of nodes for path planning
437    node_map = [[Node(i, j) for j in range(cols)] for i in r
438
439    # List to store nodes that are obstacles
440    obstacle_nodes = []
441
442    # Hardcoded obstacle coordinates
443
444    obstacle_coordinates = []#(20.755617763765958, -20.53036
445
446    # Add obstacles to the node map
447    add_obstacle(node_map, obstacle_coordinates, obstacle_nc
448
449    # Function to check if a node is an obstacle
450    def is_obstacle(node):
451        return (node.row, node.col) in obstacle_nodes
452
453    # Goal and start coordinates
454    goal_x = -5
455    goal_y = 42
456    start_x = 30
457    start_y = -10
```

Update n = Physical map size + 1.
example in the current provided configuration Physical map size is 100 x 100
so, n = 100 + 1

Step 3

Update start and goal GPS coordinates as required.



AutonomousDroneNavigation.py

```
439     # List to store nodes that are obstacles
440     obstacle_nodes = []
441
442     # Hardcoded obstacle coordinates
443
444     obstacle_coordinates = []#(20.755617763765958, -20.530367055839346), (20.0, -2
445
446     # Add obstacles to the node map
447     add_obstacle(node_map, obstacle_coordinates, obstacle_nodes)
448
449     # Function to check if a node is an obstacle
450     def is_obstacle(node):
451         return (node.row, node.col) in obstacle_nodes
452
453     # Goal and start coordinates
454     goal_x = -5
455     goal_y = 42
456     start_x = 30
457     start_y = -10
458     maintain_altitude = 1
459
460     # Find the start node
461     [i, j] = search_node(start_x, start_y)
462     print(f"row ={i} col = {j}")
463     start_node = node_map[i][j]
464
465     # Find the goal node
466     [i, j] = search_node(goal_x, goal_y)
467     print(f"row ={i} col = {j}")
468     goal_node = node_map[i][j]
469
470     # Perform A* path planning
471     path = astar_path_planning(node_map, start_node, goal_node)
472
473     # Check if a path is found
474     if path:
475         print("Planned Path (GPS Coordinates):")
476         # Uncomment below to print each node in the path
477         # for node in path:
478         #     print(node)
479     else:
480         # Default path if no path is found
481         print("No path found.")
```

1. x GPS coordinate of goal point to be updated in goal_x variable in line 453
2. Y GPS coordinate of goal point to be updated in goal_y variable in line 454
3. x GPS coordinate of start point to be updated in start_x variable in line 455
4. Y GPS coordinate of start point to be updated in start_y variable in line 456

Step 4

Start the simulation

Step 5 (Optional):

The Hardcoded obstacle can be added in obstacle_coordinate list in line 443, currently the list is empty.