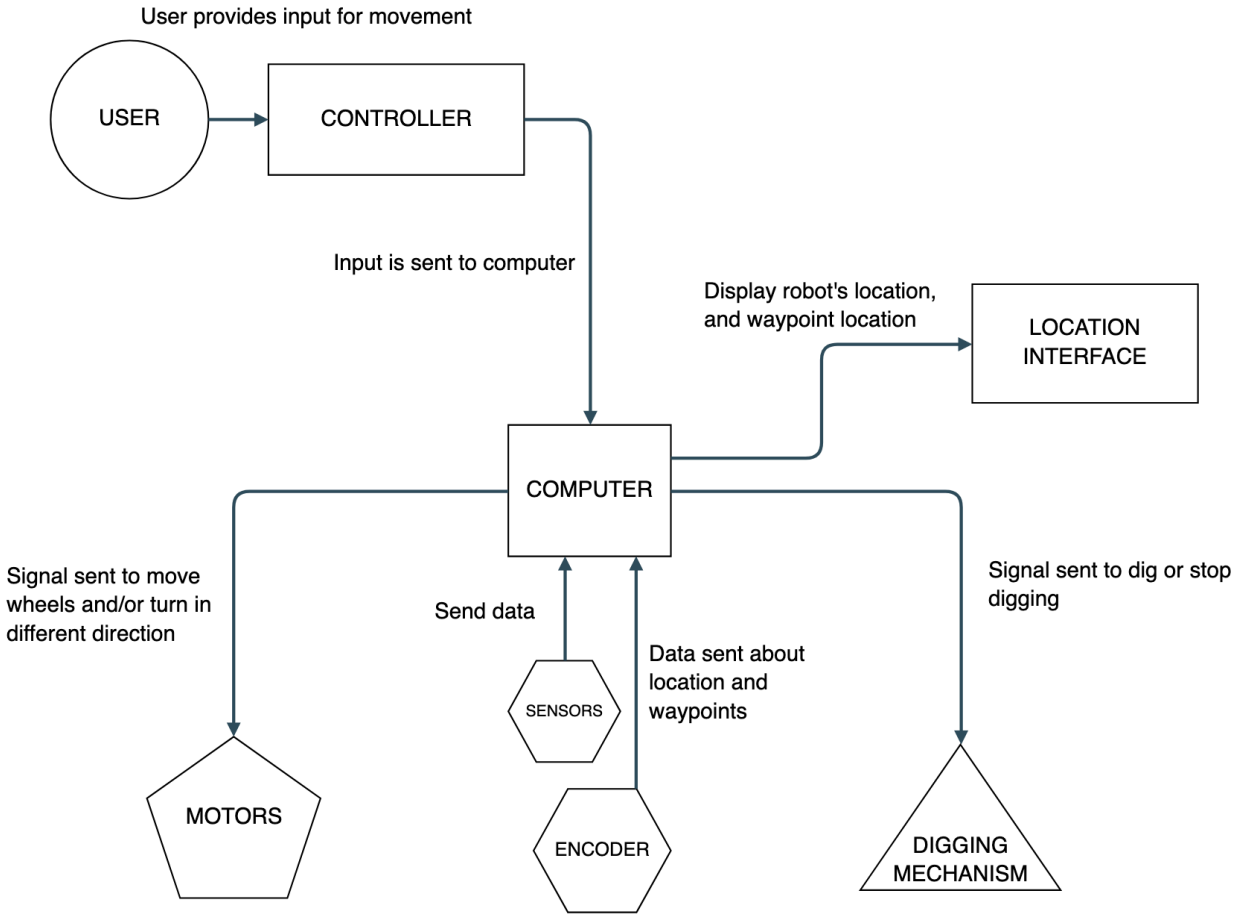
Robotic Mining Capstone - CSE - 2024

DEVELOPER MANUAL

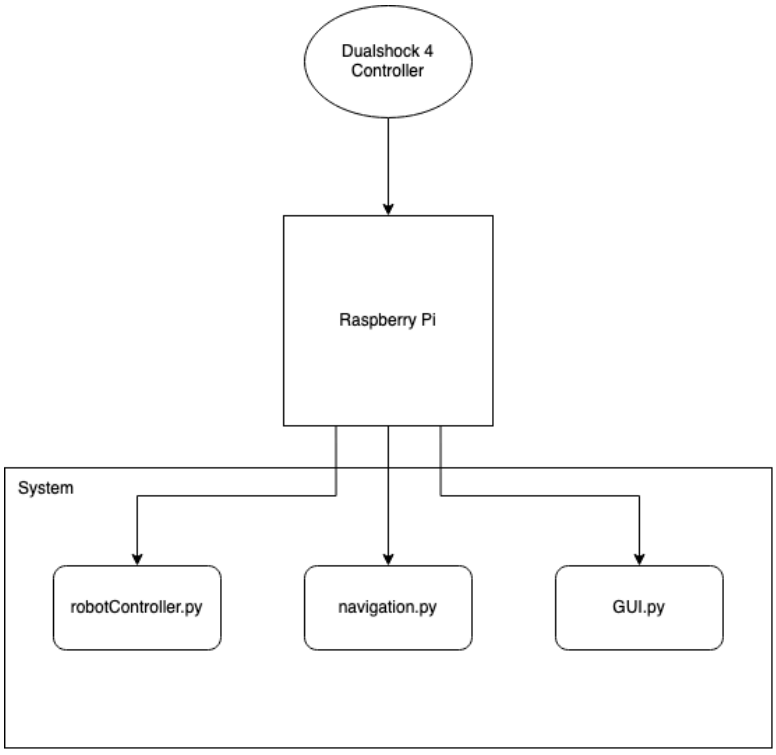
Liam Sapper

System Architecture

Interaction Diagram



Computer System



Source Files

GUI.py

This file contains the code for initiating the GUI, or graphical user interface. It sets up the main window to contain four separate sections: the waypoint list, the map, the autonav signal, and the origin point setter.

navigation.py

This file contains the code for the robot's navigation. It contains classes and functions that initialize a waypoint list, keep track of x and y coordinates, add new waypoints, and autonomously navigate along the waypoint path we've set.

robotController.py

robotController.py contains the libraries and functions needed to help the physical robot move. In it we import pyPS4Controller, and can customize the button inputs by overriding the functions associated with each button.

Classes and Functions

Note: Any references to *self* are necessary for classes to refer to variables between methods.

GUI.py

GUI - Class containing the code needed for startup.

__init__(self) - Initializes the GUI, calls load_gui()

load_gui(self) - Sets up the GUI window, setting various values for section sizes for map, waypoint list, autonav signal, and origin point set.

add_origin(self, x, y) - When the ENTER button is pressed, the imputed starting x and y coordinates will be entered as the first waypoint, and the robot's position on the map will appear.

add_waypoint(self, x, y) - When spacebar is pressed, a new waypoint will be stored in the waypoint list. (Not currently working)

keyPressEvent (self, event) - This section uses Qt's input functions, making adjustments so that the user can move left, right, up, and down with the arrow keys, as well as use the spacebar for marking waypoints.

main() - The method that instantiates the GUI by calling on the QApplication package. Main is the only method called to run by the file itself, everything else stems from within main().

navigation.py

Waypoint - Class for storing waypoint information. Includes variables for stored data, as well as links to the next and previous waypoints, should there be any.

DoublyLinkedList - Aka the navigation module, this class houses the functions necessary to store waypoint data, as well as keep track of the current waypoint list. It also contains functions necessary for movement during autonomous navigation.

addWaypoint(self, x, y) - Adds a new waypoint to the list, inputting the given x and y values as data.

getWaypoints(self) - Only used within the console for testing purposes, returns the current list of waypoints.

robotForward(self, speed, t) - During autoNav, a speed is given for the wheels to move, and the function is told to run for a certain amount of time.

robotTurn(self, speed, direction, t) - During autoNav, a speed is given for the wheels to move, and the function is told to run for a certain amount of time. It is also told which direction to turn based from *direction,* going either *left* or *right*.

autoNav(self) - Once this function is called, it starts a loop, taking data from a single waypoint and calculating angle of trajectory, as well as time it takes to travel the distance, towards the next waypoint in the list. autoNav calls robotTurn() and

robotForward() to move once calculations are complete. Once at the next waypoint, it loops to do the same thing until it reaches the last waypoint in the list.

robotController.py

MyController - This is used to override the button functions from the imported pyPS4package.

on_up_arrow_press(self) - Turn wheel motors forward

on_down_arrow_press(self) - Turn wheel motors backwards

on_left_arrow_press(self) - Left wheel motors backwards, right wheel motors forwards

on_right_arrow_press(self) - Right wheel motors backwards, left wheel motors forwards

on_up_down_release(self) - Stop wheel motors

on_left_right_release(self) - Stop wheel motors

on_L1_press(self) - Raise arms.

on_L1_release(self) - Halt signal to raise arms.

on_R1_press(self) - Lower arms.

on_R1_release(self) - Halt signal to lower arms.

on_circle_press(self) - Spin drums to dump contents.

on_square_press(self) - Spin drums to dig.

main() - The method that instantiates the controller. It connects to the ada hat attached to the raspberry pi, allowing us to set the signal for all motors to 0. This way nothing is moving starting off. It then calls a listener to listen for incoming inputs from the PS4 controller.