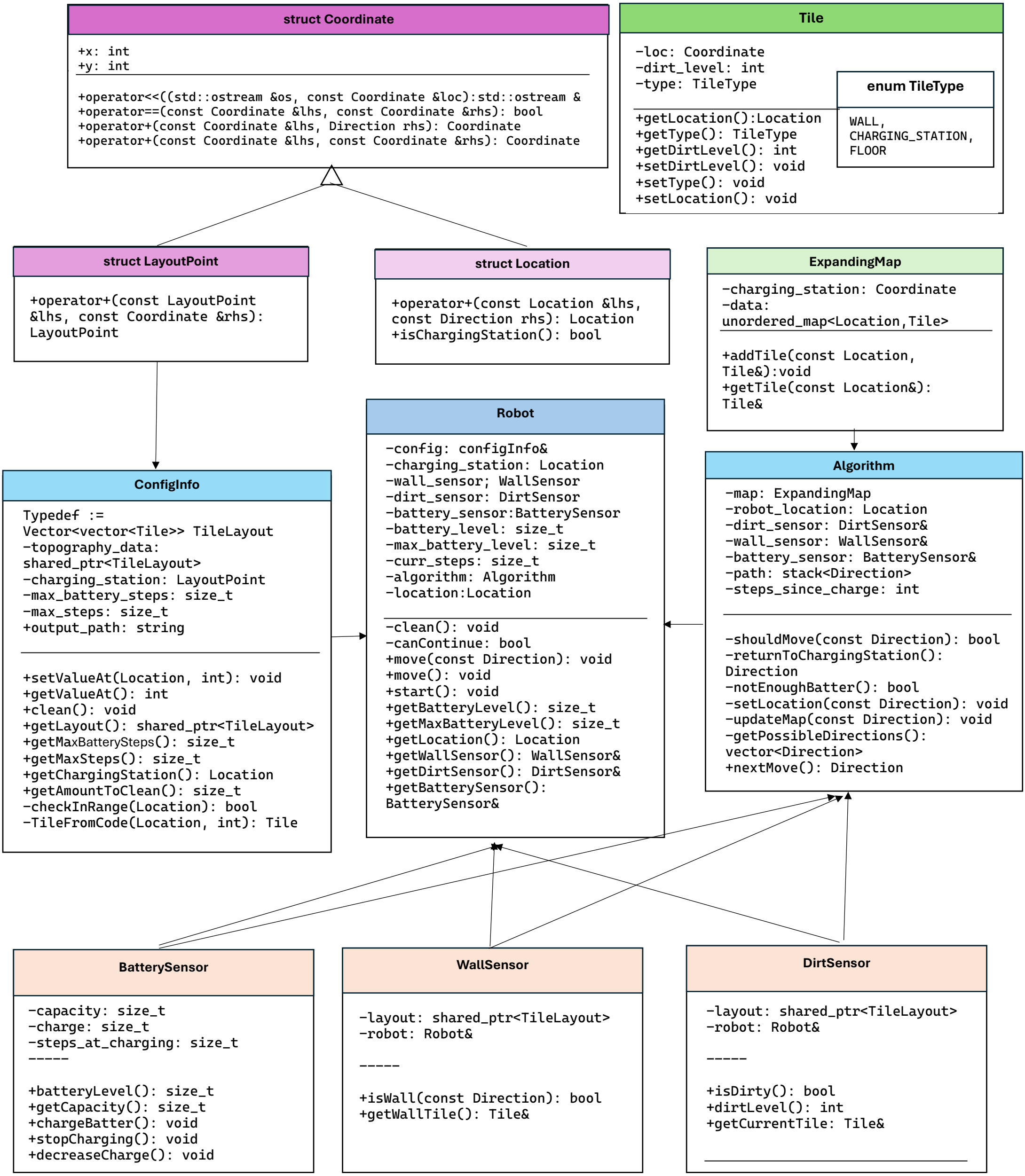
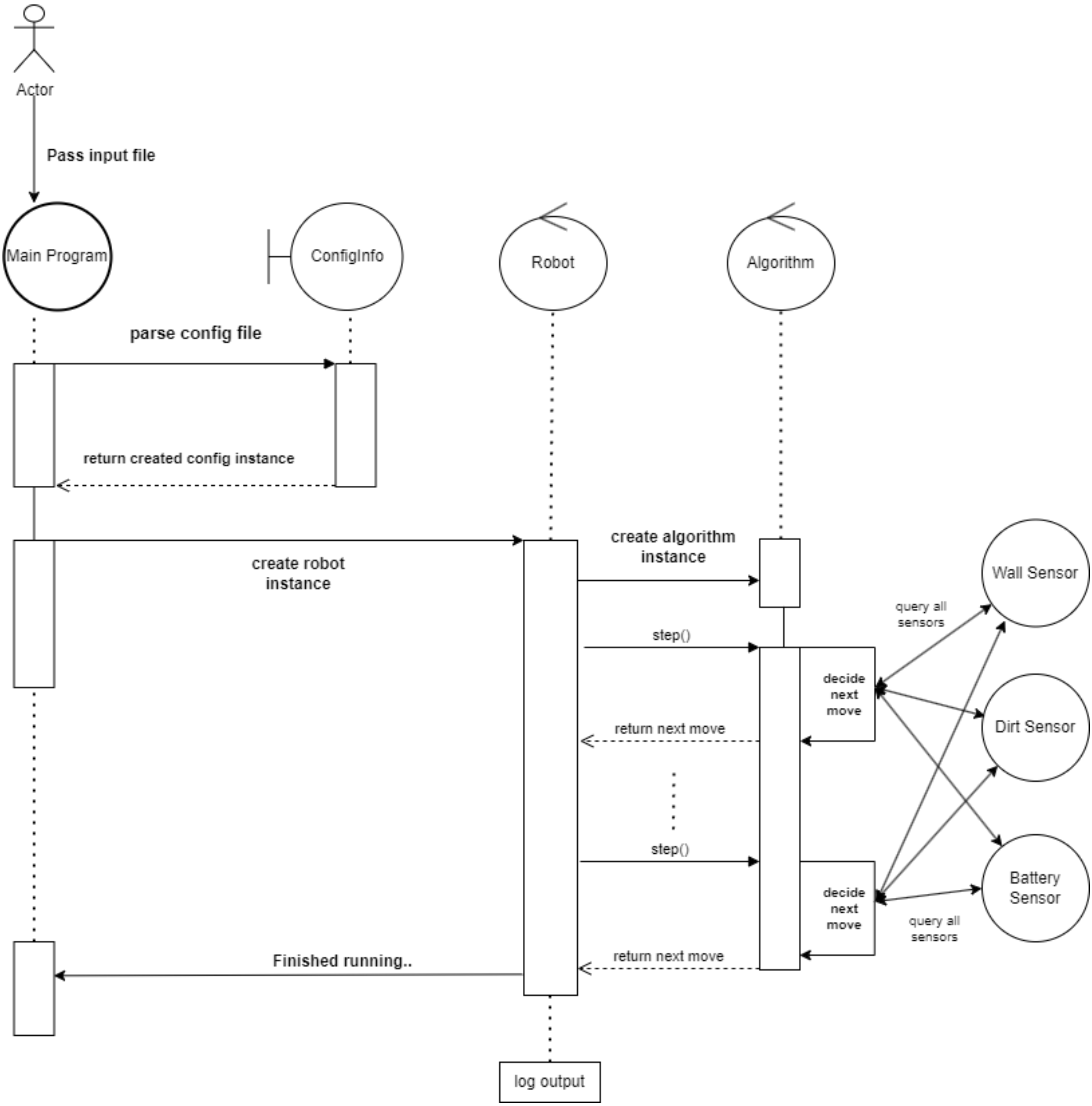

struct Coordinate
+x: int
+y: int
+operator<<((std::ostream &os, const Coordinate &loc):std::ostream &
+operator==(const Coordinate &lhs, const Coordinate &rhs): bool
+operator+(const Coordinate &lhs, Direction rhs): Coordinate
+operator+(const Coordinate &lhs, const Coordinate &rhs): Coordinate
Tile
-loc: Coordinate
-dirt_level: int
-type: TileType
+getLocation():Location
+getType(): TileType
+getDirtLevel(): int
+setDirtLevel(): void
+setType(): void
+setLocation(): void
enum TileType
WALL,
CHARGING_STATION,
FLOOR
struct LayoutPoint
+operator+(const LayoutPoint &lhs, const Coordinate &rhs): LayoutPoint
struct Location
+operator+(const Location &lhs, const Direction rhs): Location
+isChargingStation(): bool
ExpandingMap
-charging_station: Coordinate
-data: unordered_map<Location,Tile>
+addTile(const Location, Tile&):void
+getTile(const Location&): Tile&
Robot
-config: configInfo&
-charging_station: Location
-wall_sensor; WallSensor
-dirt_sensor: DirtSensor
-battery_sensor:BatterySensor
-battery_level: size_t
-max_battery_level: size_t
-curr_steps: size_t
-algorithm: Algorithm
-location:Location
-clean(): void
-canContinue: bool
+move(const Direction): void
+move(): void
+start(): void
+getBatteryLevel(): size_t
+getMaxBatteryLevel(): size_t
+getLocation(): Location
+getWallSensor(): WallSensor&
+getDirtSensor(): DirtSensor&
+getBatterySensor(): BatterySensor&
Algorithm
-map: ExpandingMap
-robot_location: Location
-dirt_sensor: DirtSensor&
-wall_sensor: WallSensor&
-battery_sensor: BatterySensor&
-path: stack<Direction>
-steps_since_charge: int
-shouldMove(const Direction): bool
-returnToChargingStation(): Direction
-notEnoughBatter(): bool
-setLocation(const Direction): void
-updateMap(const Direction): void
-getPossibleDirections(): vector<Direction>
+nextMove(): Direction
ConfigInfo
Typedef := Vector<vector<Tile>> TileLayout
-topography_data: shared_ptr<TileLayout>
-charging_station: LayoutPoint
-max_battery_steps: size_t
-max_steps: size_t
+output_path: string
+setValueAt(Location, int): void
+getValueAt(): int
+clean(): void
+getLayout(): shared_ptr<TileLayout>
+getMaxBatterySteps(): size_t
+getMaxSteps(): size_t
+getChargingStation(): Location
+getAmountToClean(): size_t
-checkInRange(Location): bool
-TileFromCode(Location, int): Tile
BatterySensor
-capacity: size_t
-charge: size_t
-steps_at_charging: size_t
+batteryLevel(): size_t
+getCapacity(): size_t
+chargeBatter(): void
+stopCharging(): void
+decreaseCharge(): void
WallSensor
-layout: shared_ptr<TileLayout>
-robot: Robot&
+isWall(const Direction): bool
+getWallTile(): Tile&
DirtSensor
-layout: shared_ptr<TileLayout>
-robot: Robot&
+isDirty(): bool
+dirtLevel(): int
+getCurrentTile: Tile&

UML SEQUENCE DIAGRAM



Explanation of our design considerations and alternatives:

Design considerations:

Separating logic to different components in a way that best reflects the real world.

The robot is aware of its surroundings only through communication with the sensors. Each sensor represents a true 'physical' sensor. Currently the 'input' room is still held by the robot but the disjoint nature of the sensors and the robot will allow it to be omitted in the future, and properly integrate with a non-fake sensor roster.

About the algorithm - it exposes an API that takes real-world data (sensors) and outputs the next move the robot should take. Theoretically the current Algorithm can implement such interface then it is hot-swappable with other algorithm. Alternatively, the algorithm itself can be constructed with the decision logic (this is all not implemented, but is a design consideration to allow such thing to be supported in the future).

Testing approach:

We used GTest as our testing tool for this assignment, as described in the bonus.txt file. We used test suites to test each class for functionality and correctness. Using this, we verified that the algorithm decisions are legal and that during the run of the robot, we don’t overstay in the docking station, walk into walls and so on. We also added tests to test the robots, sensors and configs methods to verify they are working and implemented correctly.