这个代码的功能是软件（已知迷宫的情况下），可以根据A\*和DWA实现正确走到终点并成功返回起点，并且小车同样可以在迷宫中实现如上功能。是仿真的雷达数据来实现探索未知迷宫。以下是具体文件的功能：  
**各文件职责**

* A\_DWA.py  
  A\* 全局规划：构建可走节点与边，起终点“附着”，A\* 求解，再用 L 形拼接与共线化简；入口 plan\_path\_on\_4n\_lattice。
* bluetooth.py  
  蓝牙小车接口 BTCar：原子令牌发送、距离前进 M<d>、左/右转、掉头（两次 Lp）、雷达 TX 开关、里程/姿态查询；含 REPL/命令行入口。
* config.py  
  全局参数：尺寸与安全间隙、动作节拍与速度、SLAM/雷达地图大小与分辨率、可见半径、扫描节律（起始与周期）、仿真步进与转向时间。
* env.py  
  迷宫环境 MazeEnvironment：从 JSON 读墙段与起终点；点/线段几何判定；带安全边界的位姿碰撞检测与线段可通性；单位栅格尺寸与机器人几何参数集中封装。
* lidar\_sim.py  
  雷达仿真 LidarSimulator：按角分辨率生成 360° 光束，射线与墙段求交，返回带噪量测；支持最大量程裁剪。
* main.py  
  入口与编排：GUI 选 JSON→构建环境与蓝牙→关闭设备侧雷达 TX→用 规划→创建 Navigator 与 Visualizer→先出引导点与可选起始旋转扫描→逐段执行→到达后两次左转并按原路径反向返航→资源清理。
* navigator.py  
  执行与策略中枢 Navigator：维护姿态与轨迹；按段对齐运动；“先出引导点后移动”的扫描策略；周期性原地环扫更新 SLAM；基于 SLAM 栅格筛选引导点；支持返航模式、段阻塞告警与重规划钩子；对接 BTCar 或纯仿真步进。
* slam.py  
  简化占据栅格 SLAM：以机器人位姿为真值输入，Bresenham 游走标记自由，端点近距记占据，支持可见半径裁剪；维护轨迹与 2D 栅格。
* utils.py  
  角度归一化 wrap\_pi。
* viz.py  
  可视化：裁剪显示 SLAM 栅格，叠加占据着色与引导点/返航剩余路径；绘制机器人矩形与朝向；坐标轴按单元格刻度；增量刷新。