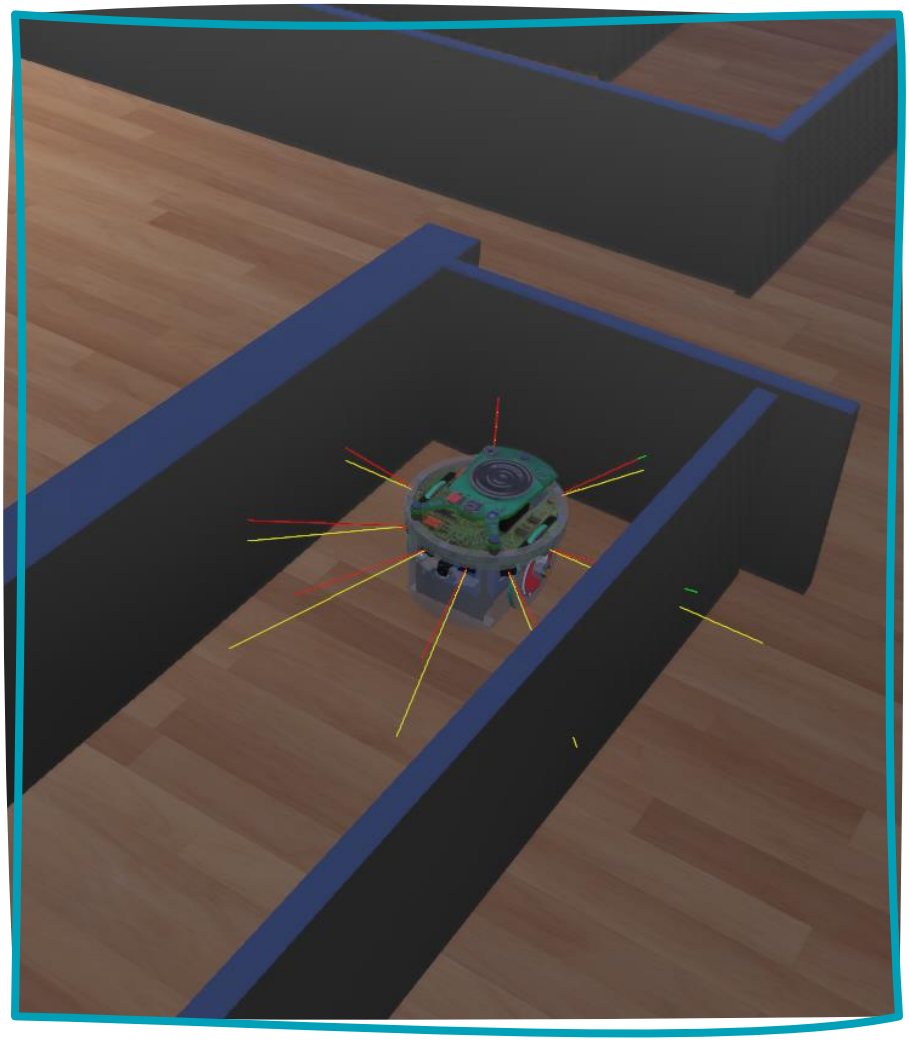

Webots Mobile Robot Application: Maze Solving

CSCI291 PROJECT: GROUP 4

Name: Ahmed Ismail | ID: 8541747 | amaea440@uowmail.edu.au
Name: Muthana Al Sirhan| ID: 8785995 | msma997@uowmail.edu.au

Date: 24/11/2024

UNIVERSITY OF WOLLONGONG IN DUBAI

Robot Structure and Configuration

The implementation utilizes an e-puck robot equipped with two sets of sensors: (Cyberbotics.com, 2019)

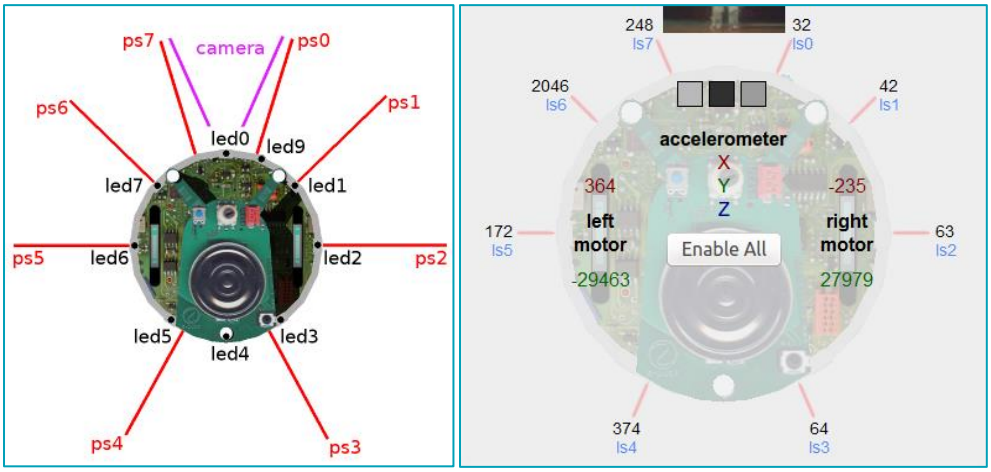

Figure 1 E-puck Robot motors, Proximity Sensors and Light Sensors Diagrams

- 8 light sensors (lso-ls7) for detecting light intensity at stations
- 8 proximity sensors (pso-ps7) for wall detection and navigation
- Left and right wheel motors for differential drive movement
- Three primary proximity sensors are actively used:
    - ps5: Left wall detection
    - ps6: Left corner detection
    - ps7: Front wall detection

This sensor configuration provides sufficient information for wall-following while maintaining cost-effectiveness.

Implementation Design

MAZE-SOLVING ALGORITHM

The implementation uses a modified left-wall following algorithm, chosen for its:

- Simple implementation requirements
- Guaranteed coverage of all dead-ends in a connected maze

The algorithm maintains contact with the left wall while exploring, ensuring systematic coverage of all possible paths and stations. (Bienias, Szczepański and Duch, 2016)

IMPLEMENTATION OF BIRGHTEST POINT DETECTION

1. First Loop:
    - Robot follows left walls
    - Detects dead ends and records it as a station
    - Records light intensity at each station
    - Creates array for station light intensities
    - Identifies brightest station
2. Second Loop:
    - Navigates to brightest station
    - Uses same wall-following algorithm
    - Stops upon reaching target station noted in array

CORE FUNCTIONS AND THEIR INTERACTIONS

1. Main Function's While Loop

```
while (wb_robot_step(TIME_STEP) != -1) {
```

Figure 2 While loop within Main Function

- Contains wall detection and navigation algorithm, light intensity measurement at stations and motor speed control for specific scenarios

2. Movement Control

```
void turn_right(WbDeviceTag left_motor, WbDeviceTag right_motor) {
  double turn_speed = MAX_SPEED / 2;
  wb_motor_set_velocity(left_motor, turn_speed);
  wb_motor_set_velocity(right_motor, 0.0);
  int turn_duration = (int)(1500 / TIME_STEP);
  int c = 0;
  while (c < turn_duration) {
    wb_robot_step(TIME_STEP);
    c++;
  }
}
```

Figure 3 turn_right Function

- turn_right(): Implements controlled right turns
- "left_speed" and "right_speed" Separate speed controls for left and right motors

3. Station Detection and Light Measurement

```
// Light sensors
const char *light_sensors_names[8] = {
  "ls0", "ls1", "ls2", "ls3", "ls4", "ls5", "ls6", "ls7"
};
WbDeviceTag light_sensors[8];
for (int i = 0; i < 8; i++) {
  light_sensors[i] = wb_robot_get_device(light_sensors_names[i]);
  wb_light_sensor_enable(light_sensors[i], TIME_STEP);
}
```

Figure 4 Light Measurement Code

- Identifies stations using wall sensor configurations, then calculates average light intensity from all light sensors. Finally, Stores and tracks the brightest station

Test Evidence

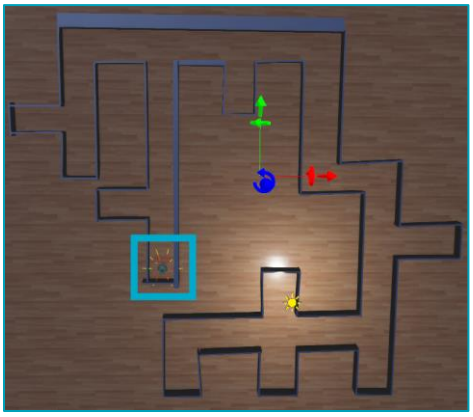

Figure 5 Starting Position Evidence

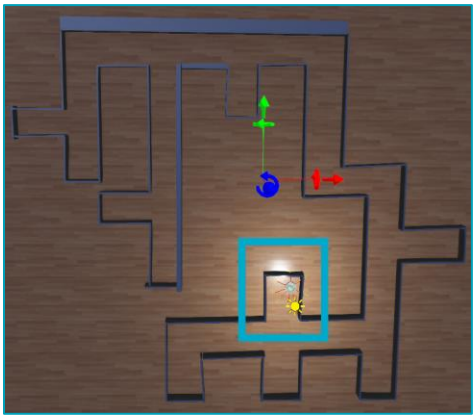

Figure 6 End Position At Dead End With Brightest Light

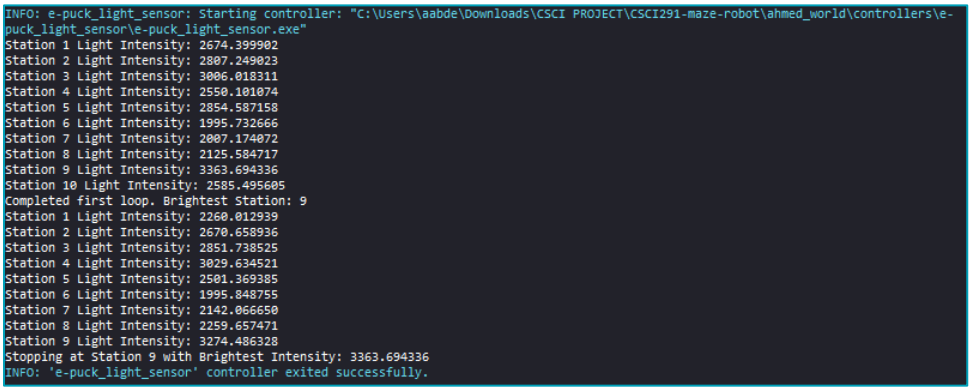

Figure 7 Console View of Light Intensities At Each Dead End

Areas for Improvement

- Current implementation may not take shortest path to brightest station as it travels across the maze twice
- Could implement path memory or GPS for optimization
- E-puck may miscalculate if placed incorrectly which is a user error

Weekly Progress

Week 2-5: Git Hub and Demo Video of Navigation, **Week 6:** Navigation Tweaks**, Week 7:** Light Detection, **Week 8-9:** Dead End Detection, **Week 10:** Final Code and Report.

GIT Hub Link

https://github.com/yaleihen/CSCI291-maze-robot.git

References

- Cyberbotics.com. (2019). Cyberbotics: [online] Available at: https://cyberbotics.com/doc/guide/epuck?version=R2021a [Accessed 24 Nov. 2024].
- Bienias, Ł., Szczepański, K. and Duch, P. (2016). Maze Exploration Algorithm for Small Mobile Platforms. Image Processing & Communications, 21(3), pp.15–26. doi:https://doi.org/10.1515/ipc-2016-0013.