



## The Scenario

The robot has to identify and count the numbers of trapped and injured workers (represented by green and red tokens respectively) in a disaster-stricken factory (the maze course). To do this it needs to autonomously enter, navigate and exit the maze while searching for the victims.

## Gameplay, Course and Scoring

- Each robot has 2 minutes to complete the maze.
- There is a combined maximum of 5 victims.
- Points are given for locating and identifying each victim.
- Additional points are given for correctly counting the number of each type of victim.
- Robots must avoid black holes in the ground.
- If the robot becomes stuck is returned to the location of the previous victim without penalty.
- The robot can be restarted at any time, but all points will be lost, and the timer will continue running.

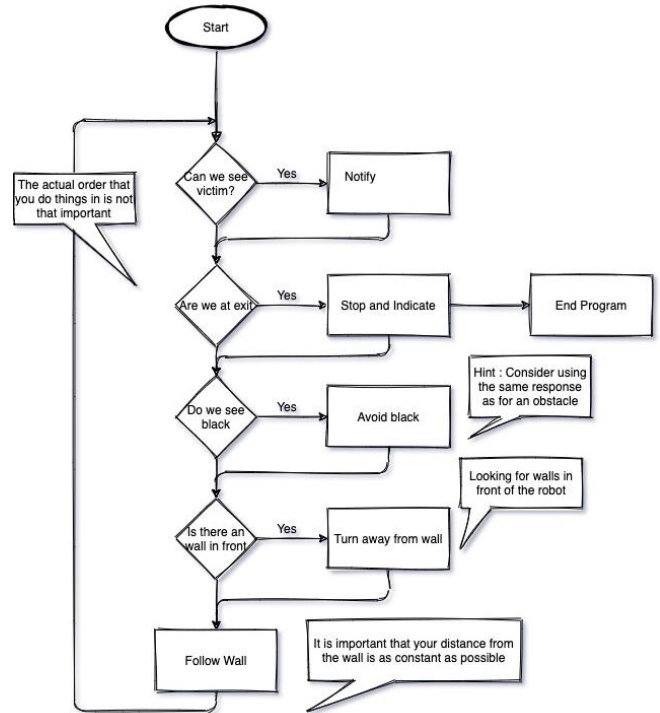
## Suggested Sensor Configuration

- 1 x Colour sensor on the front of the robot (pointing down). A second colour sensor may be useful on EV3 robots
- 1 x Ultrasonic sensor (facing the maze wall)
- 1 x Touch sensor on the front of the robot

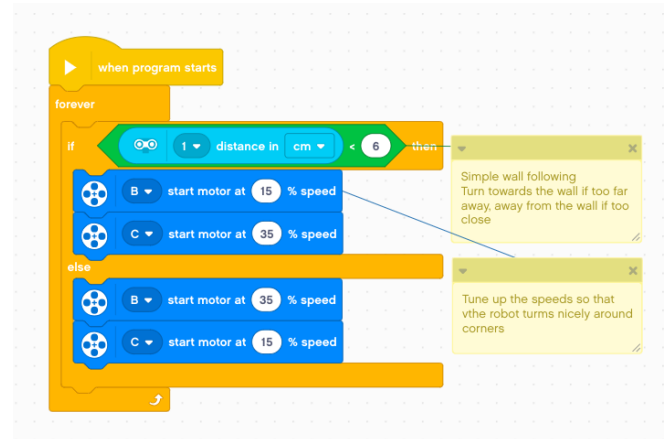
## Developing Code for the Robot

- It is a good idea to code elements of the program one at a time. Coding them in the following order is recommended:
  1. Following walls
  2. Detecting and avoiding a black hole
  3. Finding a victim
  4. Finding the exit
  5. Reporting the number of victims
- The refresh rates on Lego sensors usually require motor speed to reduced significantly (20 to 25% is a good starting point)

## Flow Diagram of Suggested Code Logic



## Wall Following



The robot should turn gradually toward or away from the wall depending on how far away it is. The speeds of the motors determine the turn radius. Adjusting the turn radius so that the robot is near central on the tile (from side to side) after a 90 or 180-degree turn is recommended.



## Avoiding Black Holes and Forward Collisions with Walls

If sensors detect a black hole or a wall in front of the robot a good approach is to reverse a little and then make a small turn. With this code, it may take multiple of these small adjustments to avoid the wall or black hole which is perfectly acceptable.

## Locating Victims, Identifying their Type and Counting them

If a victim is located, the robot should stop and indicate that a victim has been identified (this can be done in multiple ways but a visual indication such as flashing the button backlighting or printing on the screen is preferred over an auditory indication as these can be hard to hear at events). The robot should then identify the type of victim and add 1 to the variable storing the count of the type of victim found, move forward off the victim and return to the primary loop in the code.

## Finding the exit:

The robot will need to detect the exit. It is highly reflective allowing it to be detected by the colour sensors. When the exit is found the number of each type of victim should be printed on the screen.

Hint: Detecting Silver on a Spike Robot will need to show Block Extensions and enable More Sensors. Then use Raw Light Sensor Block.

## Resources

- Workshop Materials: <https://tinyurl.com/RCJWARescue>
- RCJA Website: <https://www.robocupjunior.org.au/>
- Rescue Maze: <https://www.robocupjunior.org.au/rescue-maze/>
- David's Robotics Portal: <https://WinthropDC.com/Robotics>
- EV3 Standard Design: <https://www.robocupjunior.org.au/wp-content/uploads/2021/08/Maze-EV3-Standard-Design.pdf>
- Spike Standard Design Video: <https://youtu.be/7wiqk-wr-wo>

## Getting Help

Please feel free to reach out at any time if you have any questions. Students are welcome to email me directly with any questions too.

David Musgrave – WA & National Rescue Maze Challenge Co-ordinator – [david.musgrave@robocupjunior.org.au](mailto:david.musgrave@robocupjunior.org.au)

Tim Ronchi – Victorian Rescue (Maze focus) Challenge Co-ordinator – [tim.ronchi@robocupjunior.org.au](mailto:tim.ronchi@robocupjunior.org.au)