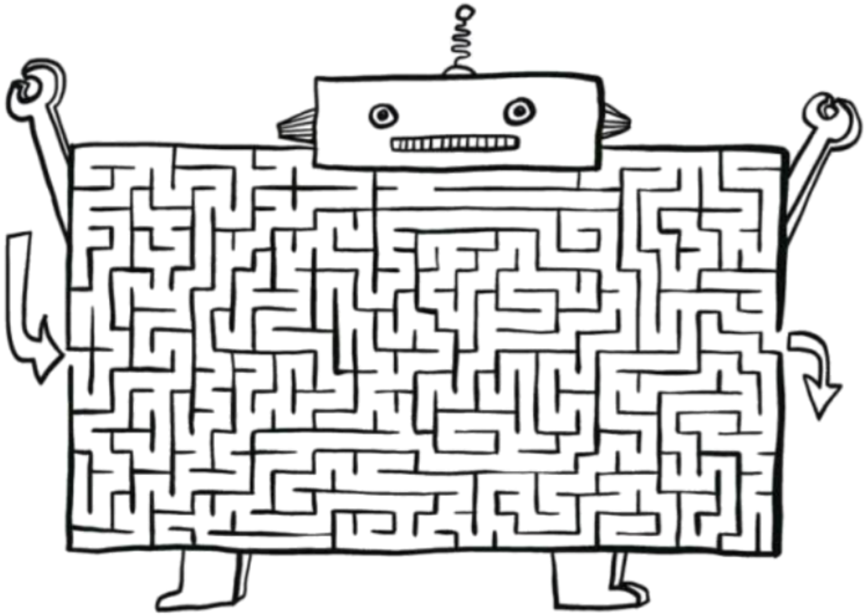
Robot Maze Solver



Prepared by: Qossay Abu Rida
IEEE Robotics and Automation Society

Agenda

Hardware

- Micro-Controller
- Wheels
- H-Bridge
- Sensors
- Battery
- Chassis

Software

- What is required ?
- Left Hand Rule
- Right Hand Rule
- Flood-Fill

Control

Robot Hardware

IEEE Robotics & Automation Society

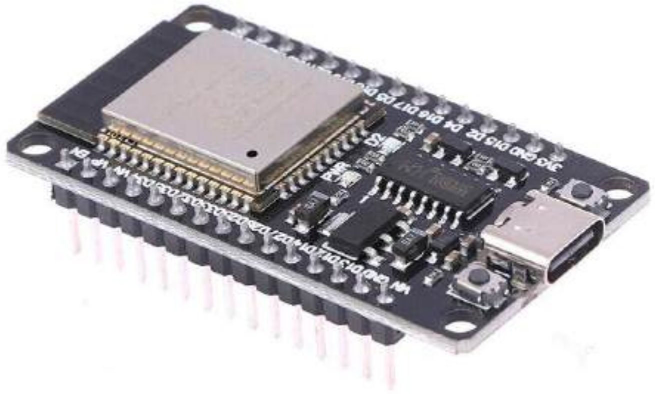
Micro-Controller

ESP32 is a low-cost, low-power system on a chip (SoC). It is widely used in IoT, embedded systems, and smart devices.

Key features include:

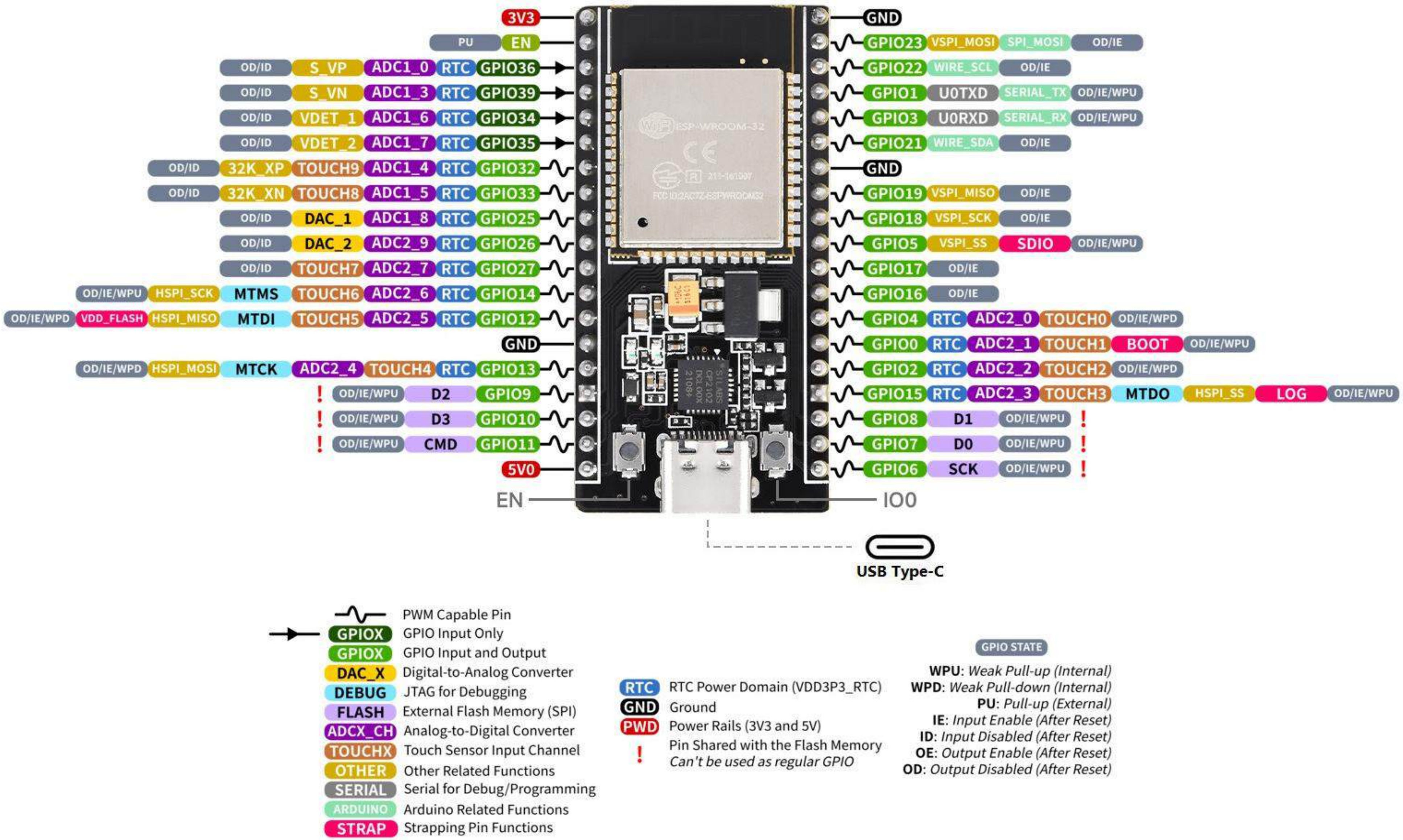
- Dual-core processor:
  Clock speeds up to 240 MHz.
- Connectivity:
  Integrated Wi-Fi (802.11 b/g/n)
  Bluetooth (BLE and Classic).
- Peripherals:
  GPIO, ADC, DAC, UART, SPI, I2C, PWM.
- Rich development ecosystem:
  Supported by Arduino IDE.

It’s ideal for applications like robotics.

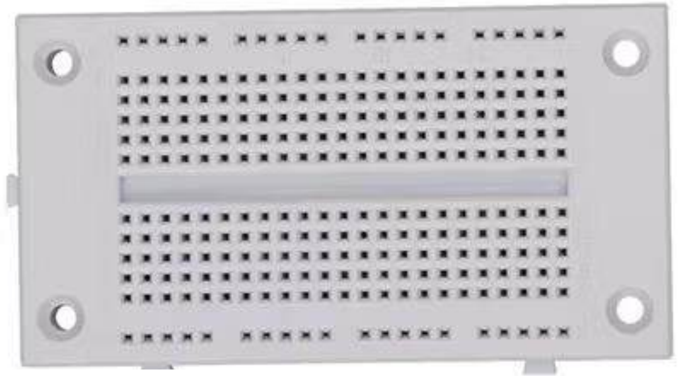


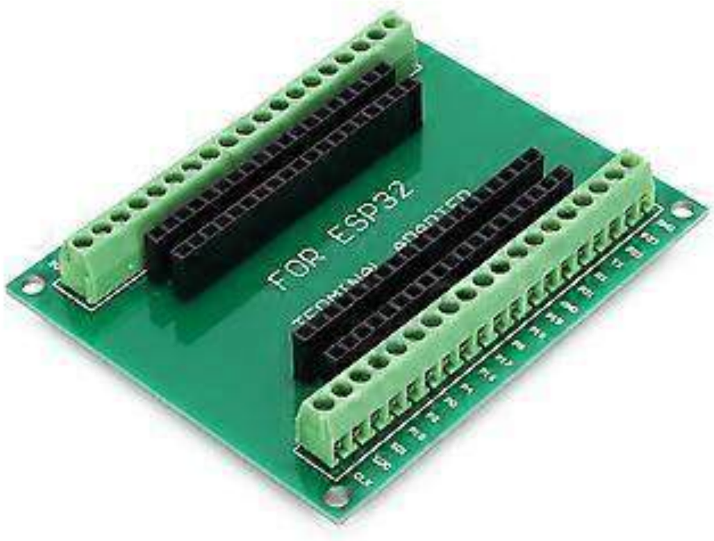
IEEE Robotics & Automation Society

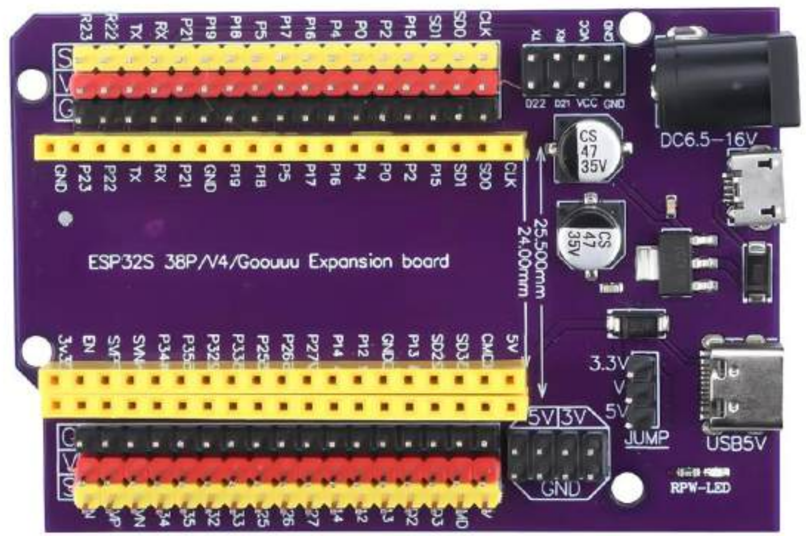
Brief About ESP32 (ESP32 CP2102 TYPE-C 38-Pin)



ESP32 shield







Wheels

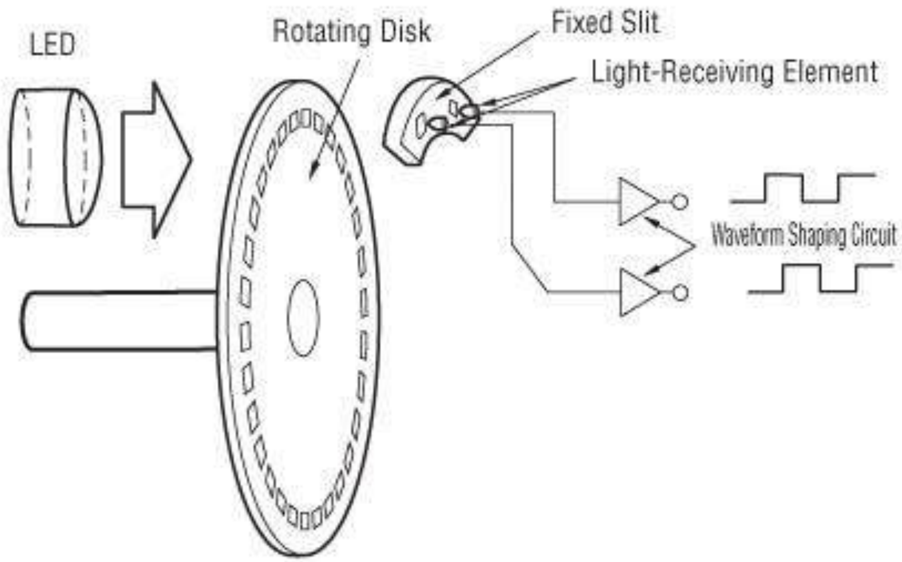
Motor with Encoder:
Has a sensor to provide position, speed, and direction feedback for precise control.

Motor without Encoder:
Lacks feedback, offering simpler, cost-effective control but less accuracy.

IEEE Robotics & Automation Society

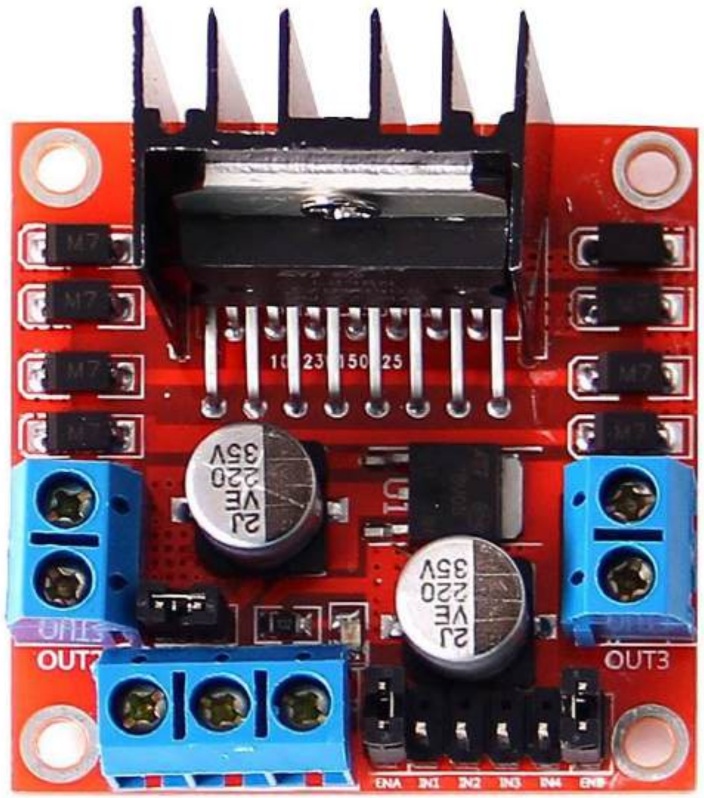
How encoder works:

Optical encoder. It uses an LED and a rotating disk. and light-receiving elements generate electrical signals. These signals represent the position, speed, and direction of the motor.



H-Bridge (Motor Driver L298N)

L298N H-Bridge is a motor driver module that allows control of two DC motors. It supports forward/reverse rotation, speed control via PWM.



How to connect:

Method 1:

Direction Control:

Two input pins (IN1, IN2) for each motor determine the motor's direction.

IN1= High , IN2 = Low → Motor rotates forward.

IN1= Low , IN2 = High → Motor rotates backward.

Speed Control:

Speed is adjusted using Pulse Width Modulation (PWM) applied to the Enable pin (EN).

Higher duty cycle → Faster speed.

Lower duty cycle → Slower speed

Method 2:

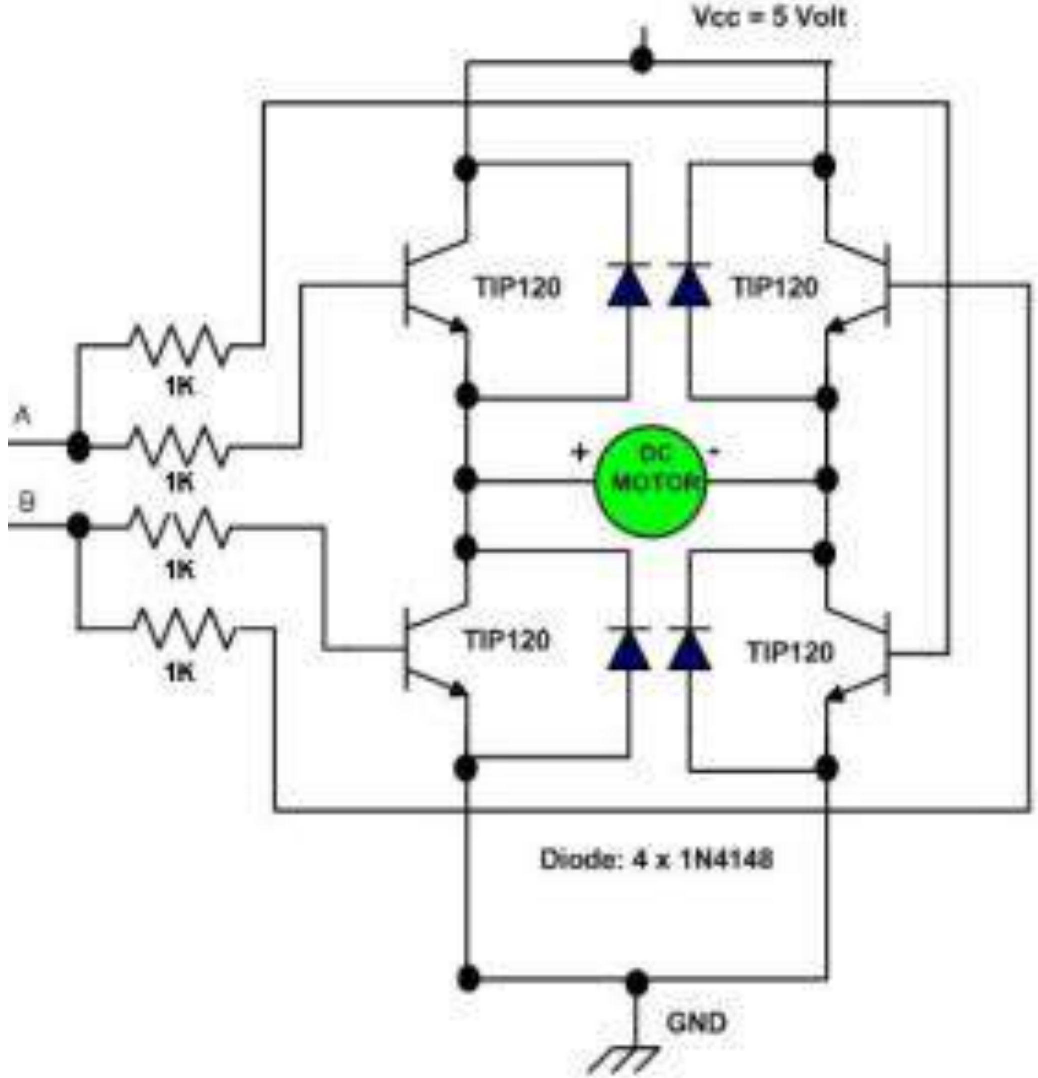
Direction and Speed Control:

Connect EN to 5 volt use jumper then

IN1 = PWM, IN2 = LOW → Forward speed controlled by IN1's PWM.

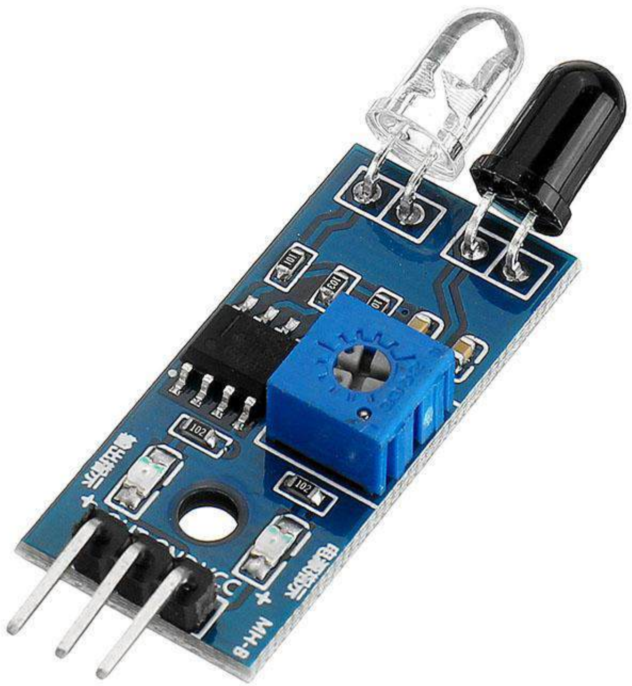
IN1 = LOW, IN2 = PWM → Reverse speed controlled by IN2's PWM.

How H-Bridge works:



Sensors (IR)

حاولوا ما تجيبوه لانه بشتغل على مزاجه وحساس للظروف الخارجية زيادة عن اللزوم

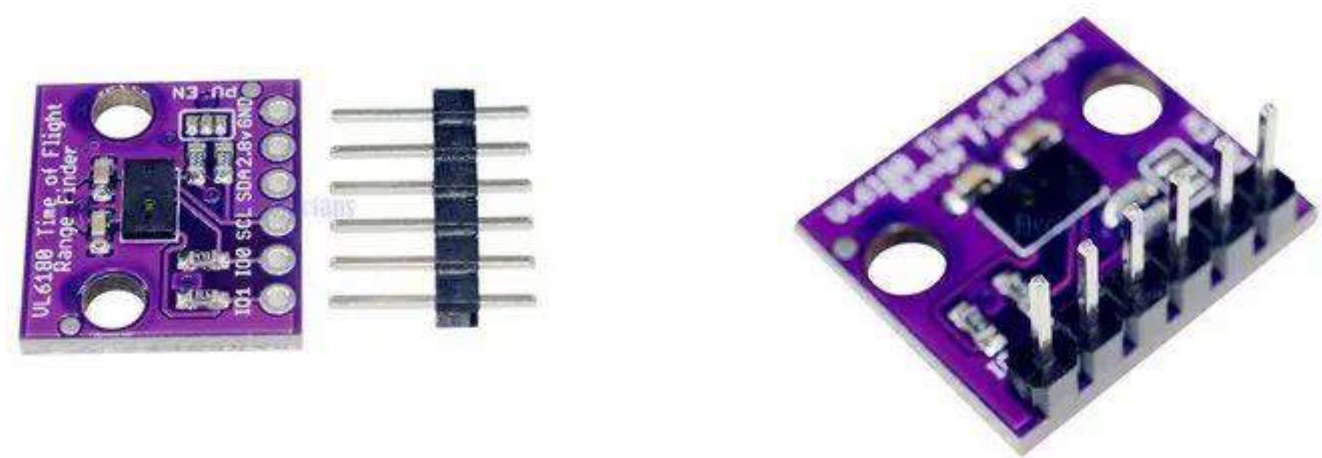


Sensors (ultrasonic)

جيد بس مشكلته حجمه كبير شوي

Sensors (Laser Range Finder Distance Sensor Time of Flight VL6180X)

خيار جيد وهسا بنشرح عنه بالتفصيل



Sensors

It is a good idea to connect one sensor to a servo motor.

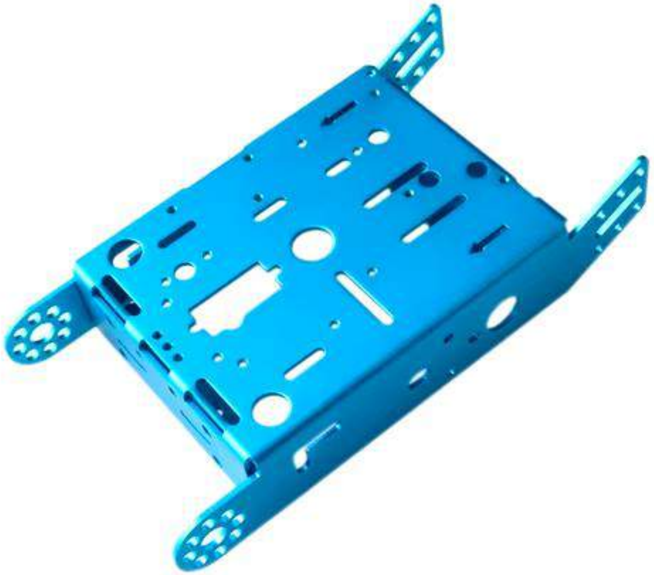
https://www.youtube.com/watch?v=Ro7T3q14uDY

Battery (9 volt battery)

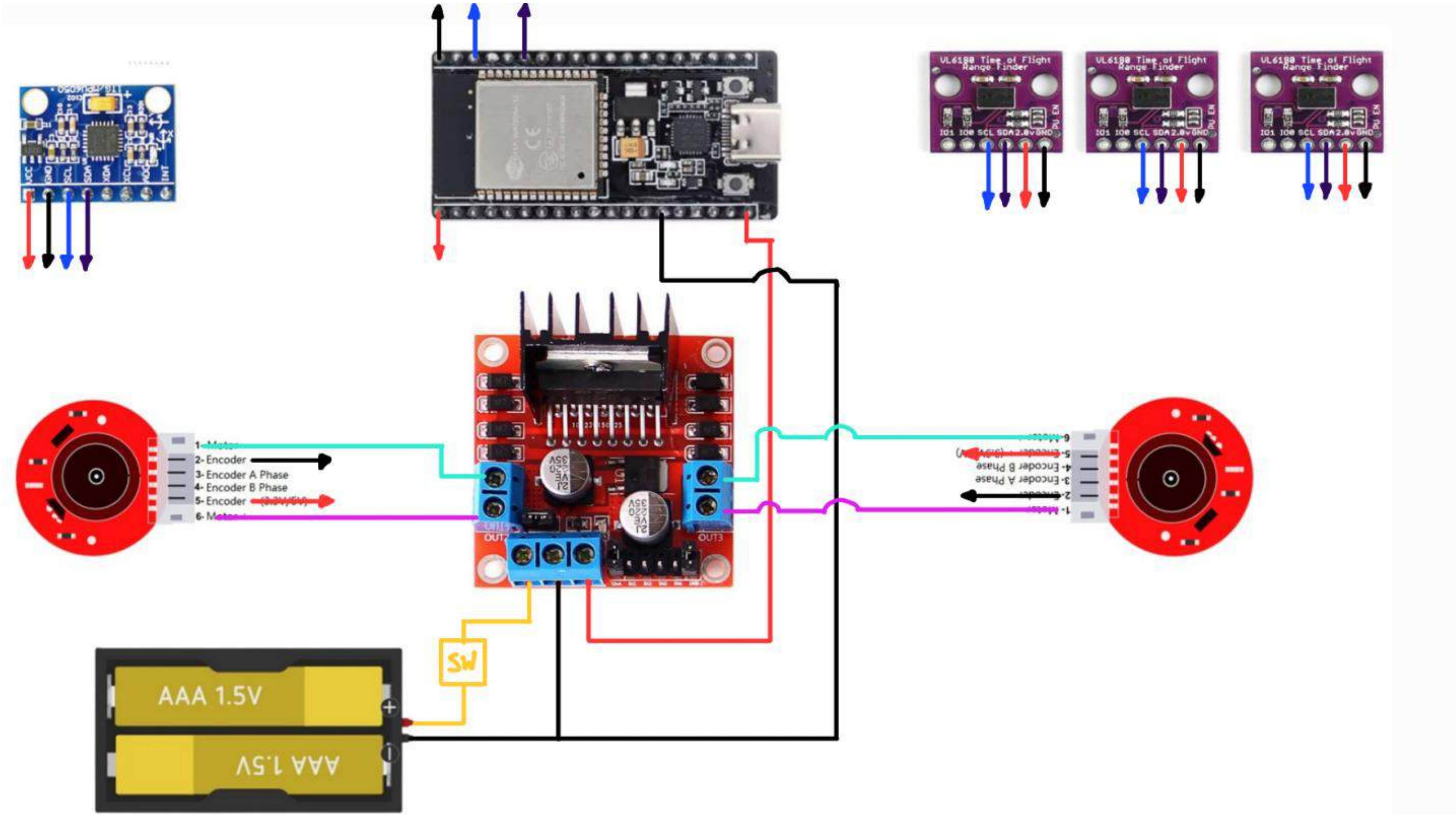
مش احسن خيار بس ممكن يريحكم اذا بلشتوا فيه ويوفر عليكم حرق قطع

Battery (Lithium Ion Battery – 18650 Cell (5000mAh))

Chassis

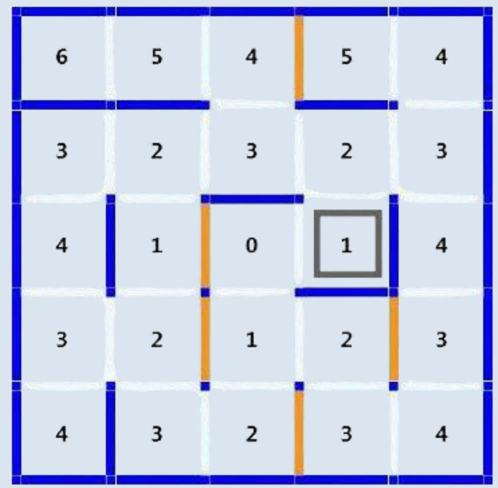


EXAMPLE



VL6180 Time of Flight Range Finder
IO1 IO0 SCL SDA 2.8v GND
VL6180 Time of Flight Range Finder
IO1 IO0 SCL SDA 2.8v GND
VL6180 Time of Flight Range Finder
IO1 IO0 SCL SDA 2.8v GND
VCC GND SCL SDA XDA XCL AD0 INT
1- Motor
2- Encoder
3- Encoder A Phase
4- Encoder B Phase
5- Encoder (3.3V/5V)
6- Motor
OUT2
OUT3
SW
AAA 1.5V
AAA 1.5V

Robot Software



IEEE Robotics & Automation Society

What is required ?

The primary objective is for the robot to navigate through the maze and reach its center.

Once the robot has successfully explored the maze on its first attempt, its secondary goal is directly reaching the center on subsequent attempts without relying on its sensors to detect the maze's boundaries.

6	5	4	3	3	4	5	6
5	4	3	2	2	3	4	5
4	3	2	1	1	2	3	4
3	2	1	0	0	1	2	3
3	2	1	0	0	1	2	3
4	3	2	1	1	2	3	4
5	4	3	2	2	3	4	5
6	5	4	3	3	4	5	6

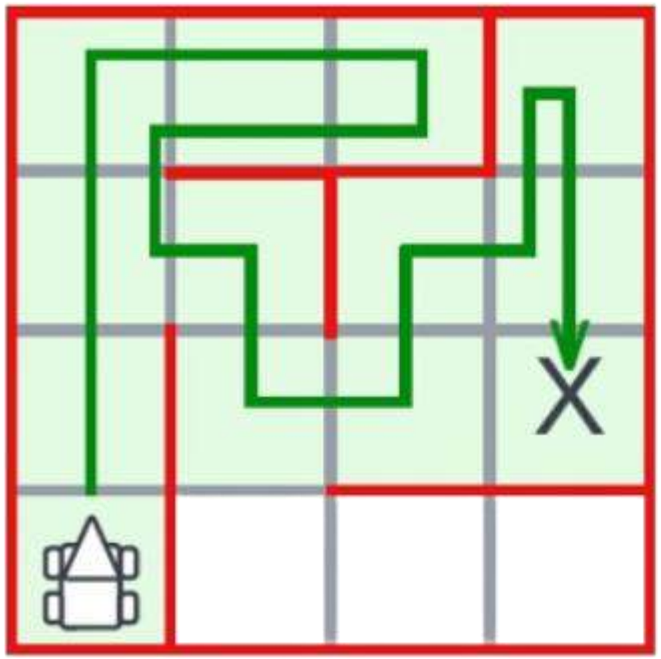
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	0	0	1	1	1
1	1	1	0	0	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

How to write code ?

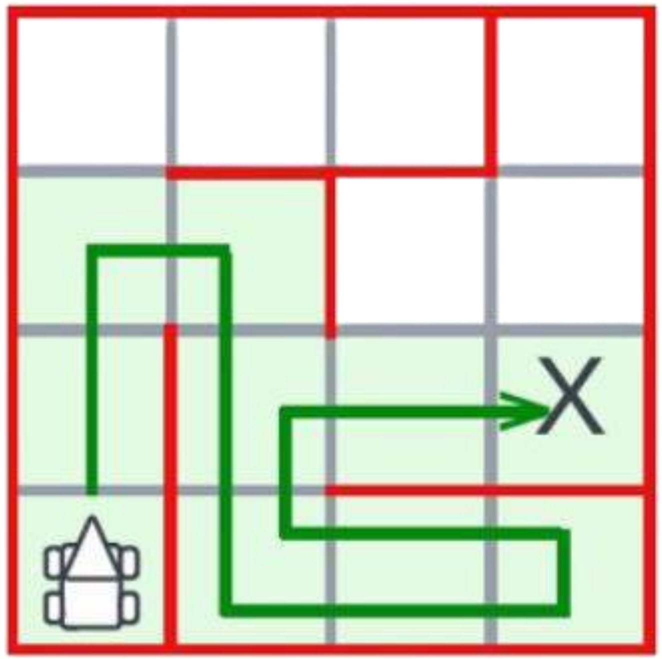
```
typedef struct {
    int row;
    int col;
} Position;

Position currentPosition = {0, 0};

int maze1Values[8][8] = {
        {6, 5, 4, 3, 3, 4, 5, 6},
        {5, 4, 3, 2, 2, 3, 4, 5},
        {4, 3, 2, 1, 1, 2, 3, 4},
        {3, 2, 1, 0, 0, 1, 2, 3},
        {3, 2, 1, 0, 0, 1, 2, 3},
        {4, 3, 2, 1, 1, 2, 3, 4},
        {5, 4, 3, 2, 2, 3, 4, 5},
        {6, 5, 4, 3, 3, 4, 5, 6}
};
```

```
void loop (){

   If (mazeValues [currentPosition.y][currentPosition.x]==0)
      return ;


   readSensors();

   makeDecision();

   moveMotor();

}
```

Left Hand Rule

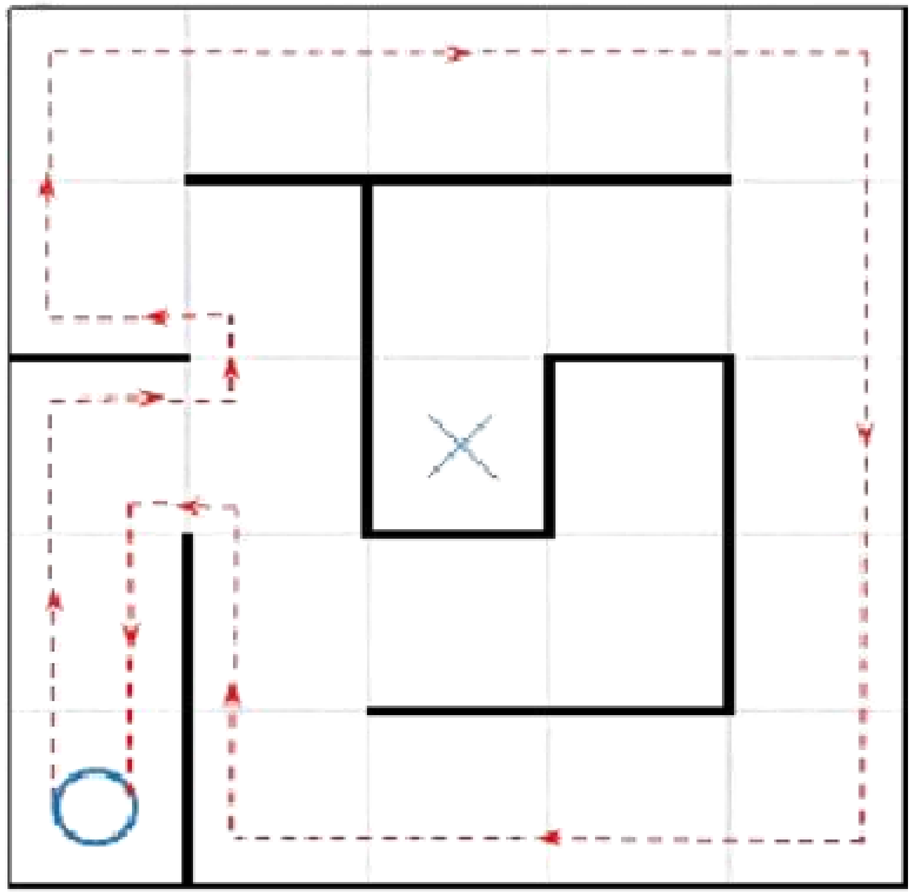


Right Hand Rule



Left or Right Hand Rule

In some cases, using the Left-Hand Rule or Right-Hand Rule alone may not lead to a solution when navigating a maze. To overcome this limitation and ensure the algorithm finds a path, we introduce a random choice between turning left or right at certain decision points



Flood-Fill

Flood-Fill Algorithm is a critical component for dynamically updating the maze's distance matrix when an inconsistency occurs during navigation.

It ensures that all cells in the maze are correctly labeled with their distances to the target cell.

Flood-Fill similar with A* at some point:

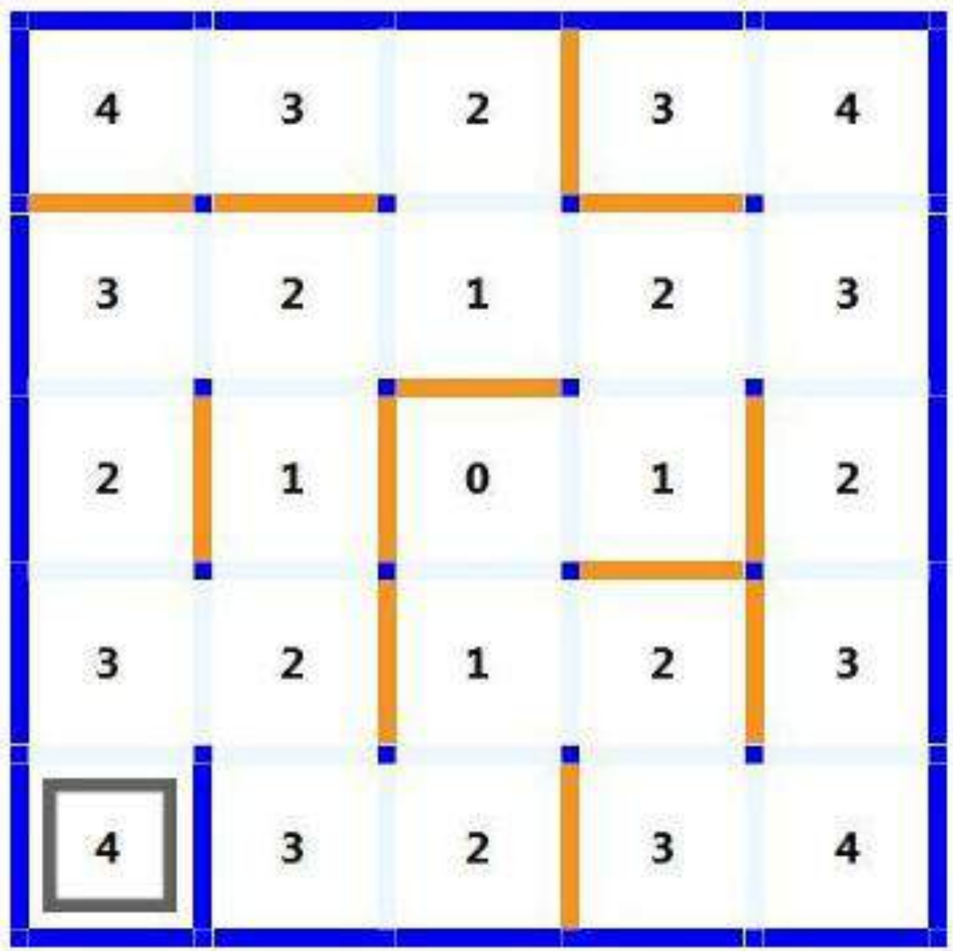
1- Node Representation:
- Each cell in the maze is a node.
- A node is represented by its coordinates (x, y).

2- Costs:
- h(n): Heuristic estimating the distance from the current cell to the goal cell.
- Use Manhattan distance for simplicity

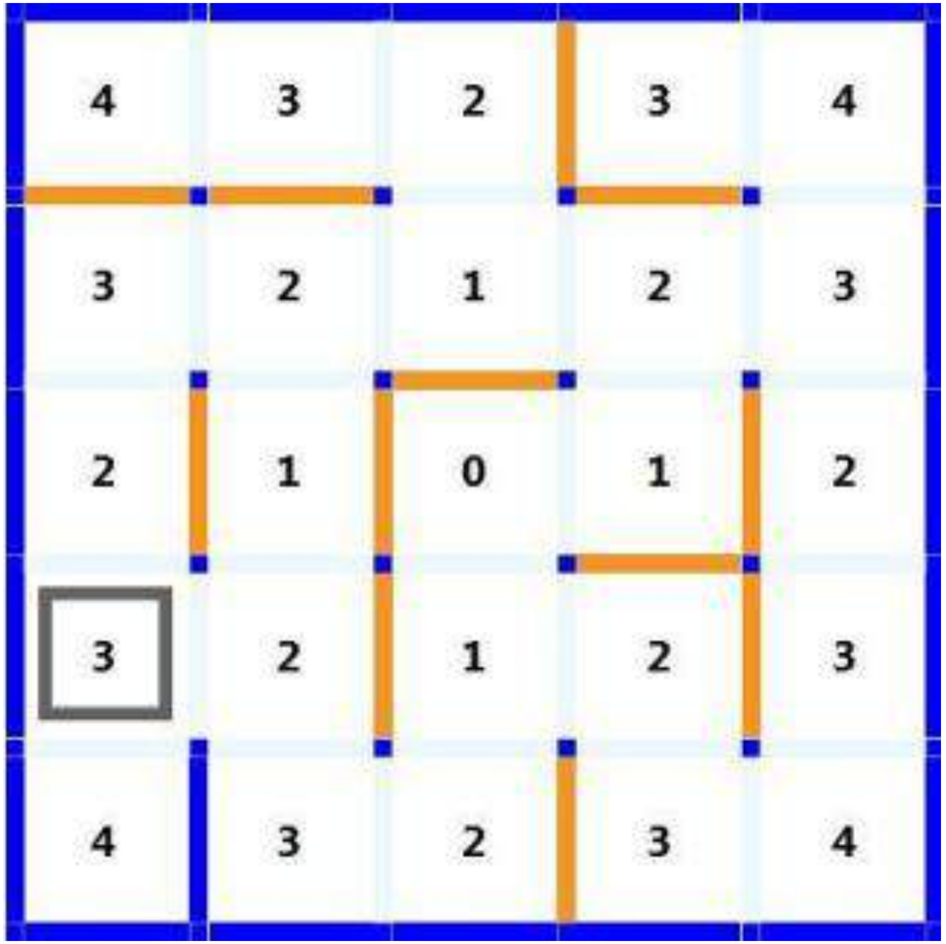
3- Update:
- Updates distances globally and works well for dynamic updates when new walls are detected

6	5	4	3	3	4	5	6
5	4	3	2	2	3	4	5
4	3	2	1	1	2	3	4
3	2	1	0	0	1	2	3
3	2	1	0	0	1	2	3
4	3	2	1	1	2	3	4
5	4	3	2	2	3	4	5
6	5	4	3	3	4	5	6

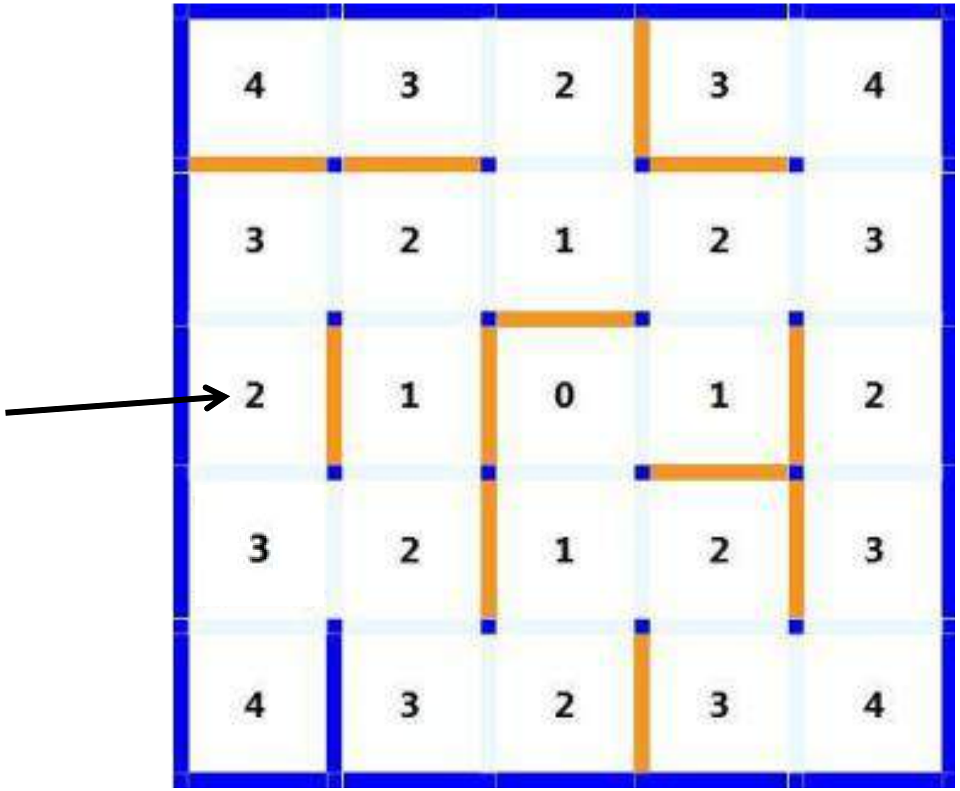
Flood-Fill



Flood-Fill



Flood-Fill



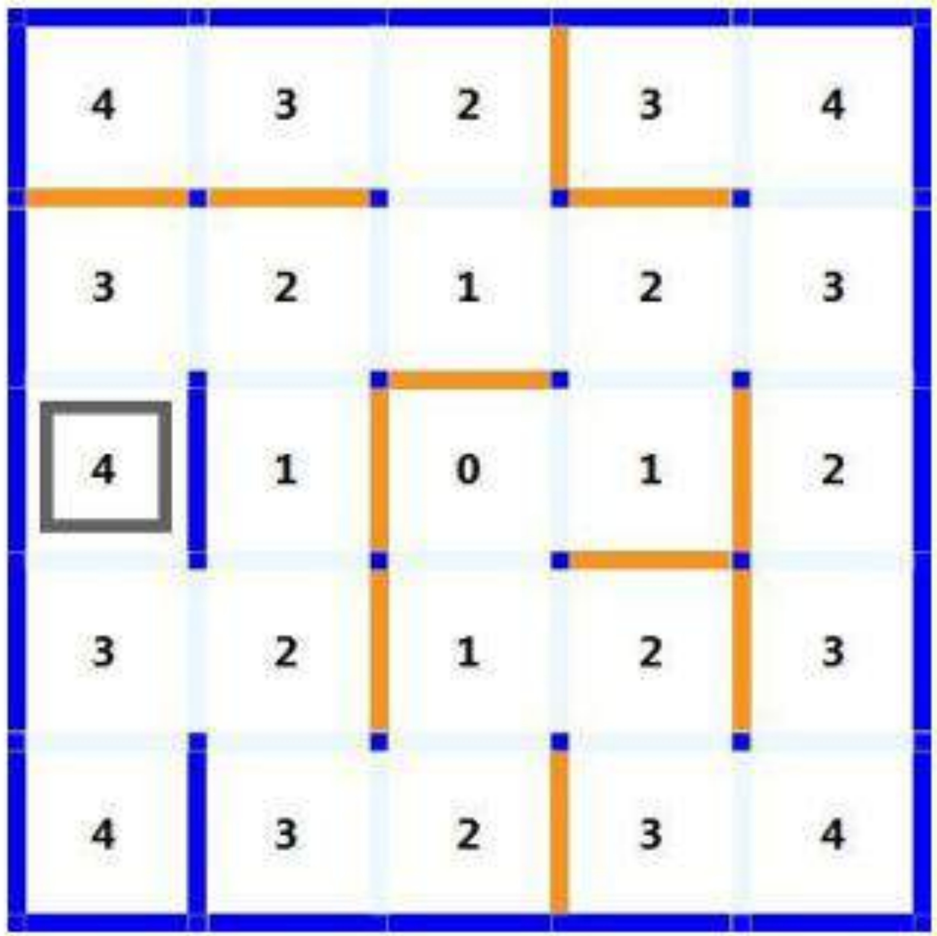
1- The current cell must have a weight greater than at least one neighbor

2- Hence when all neighbors have values greater than the current cell

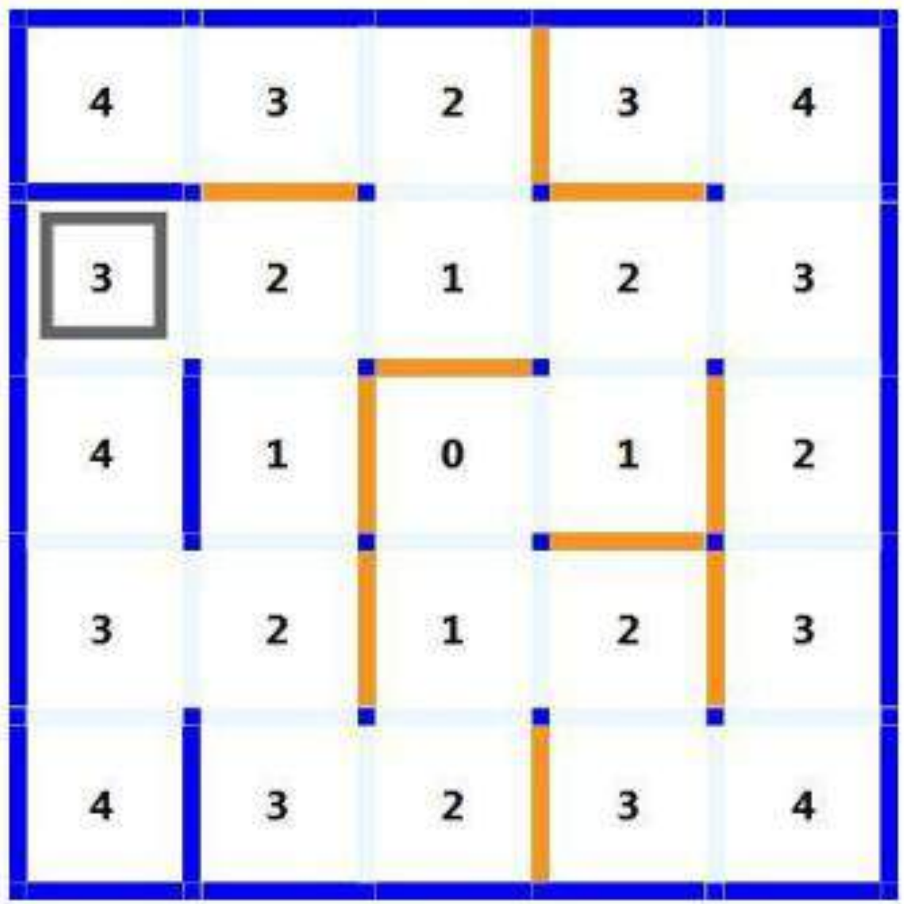
We need to edit the weights to achieve point 1

```
if (dp[current_cell.x][current_cell.y] - 1 != dp[min_cell.x][min_cell.y]) {

    //  edit weights
    //  add all neighbors to the qeueu

}
```

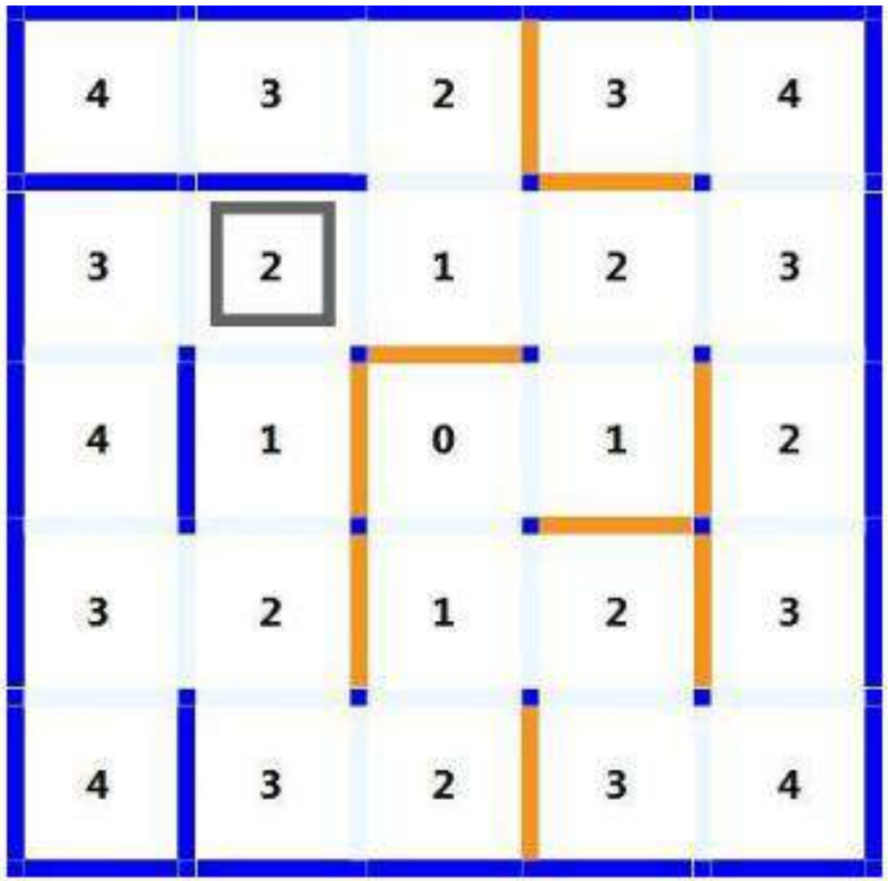
Flood-Fill



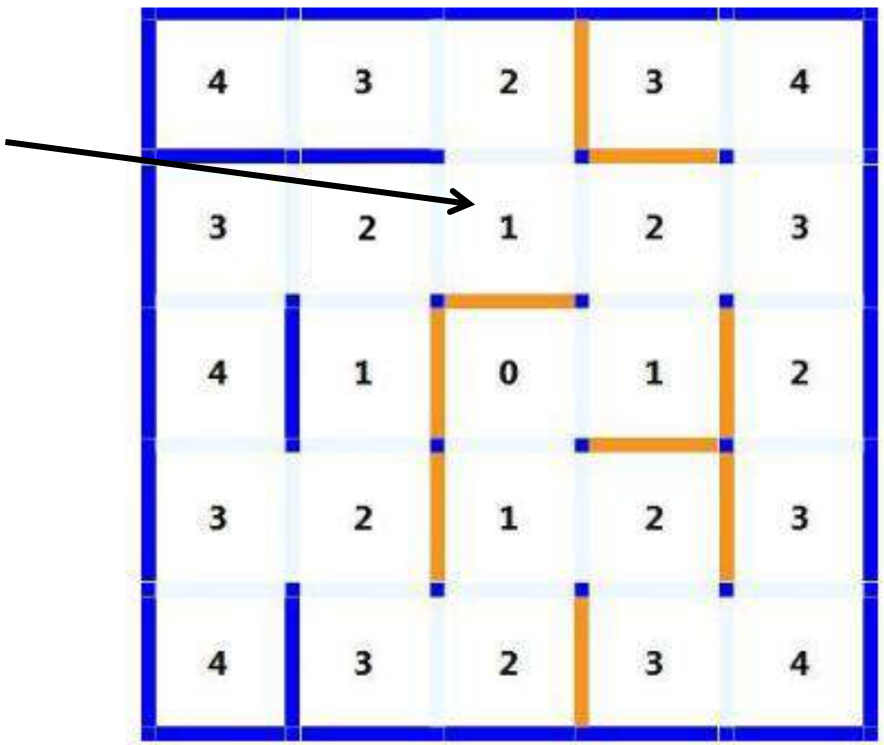
Flood-Fill



Flood-Fill



Flood-Fill



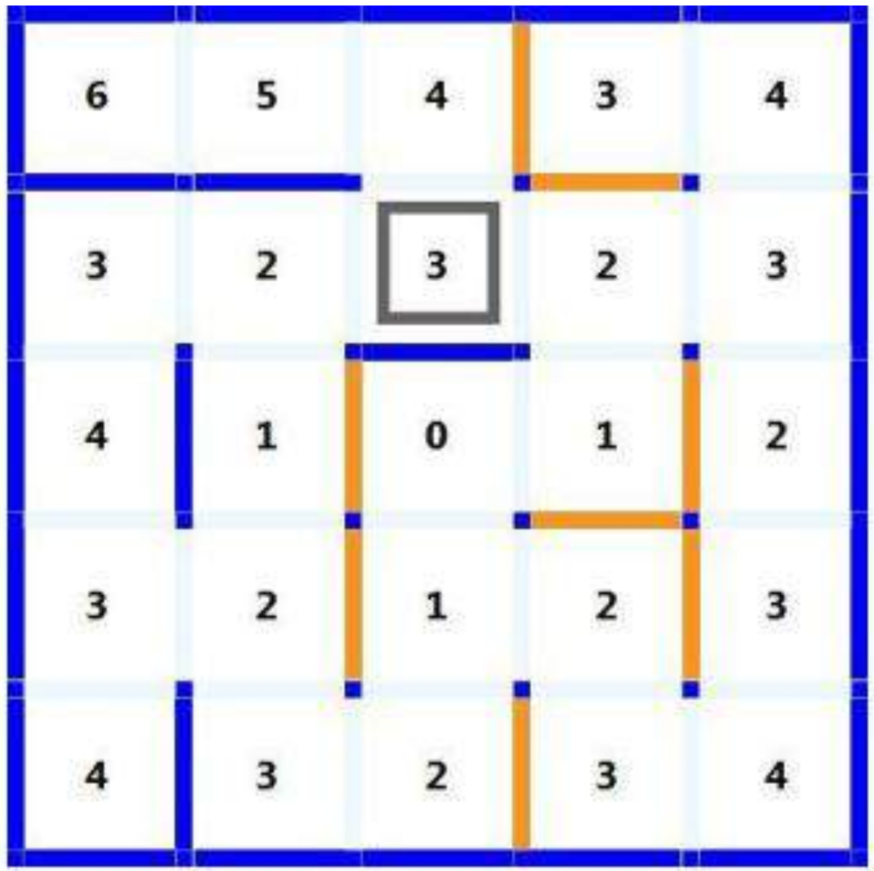
1- The current cell must have a weight greater than at least one neighbor

2- Hence when all neighbors have values greater than the current cell

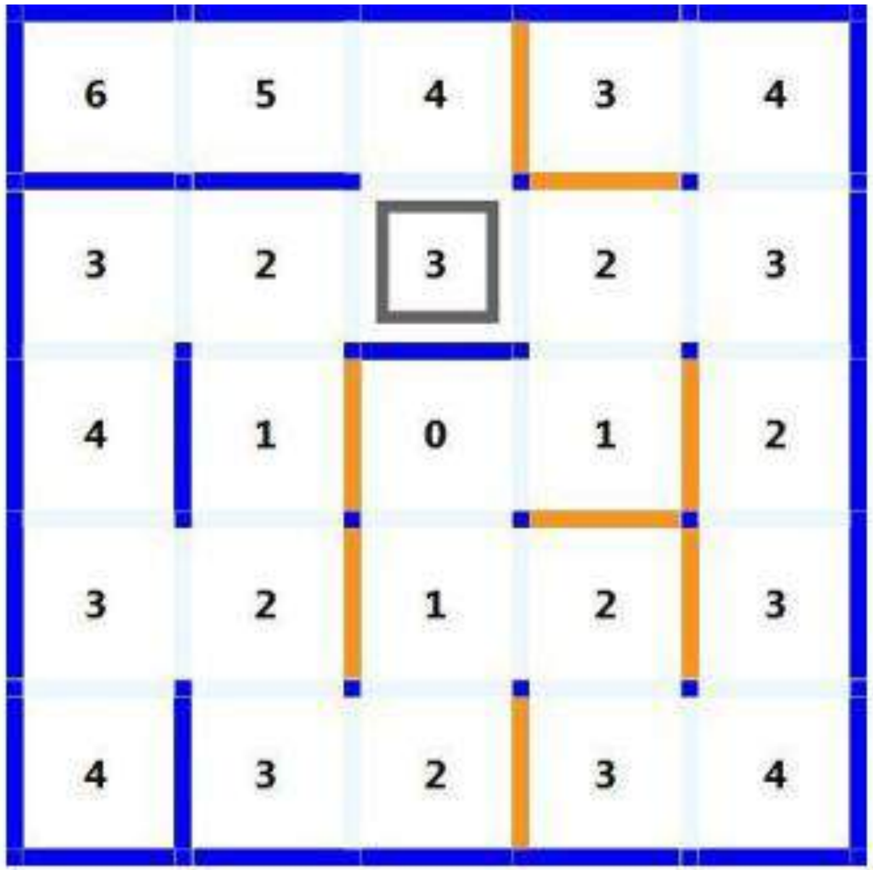
We need to edit the weights to achieve point 1

```
if (dp[current_cell.x][current_cell.y] - 1 != dp[min_cell.x][min_cell.y]) {

   // edit weights
   // add all neighbors to the qeueu

}
```

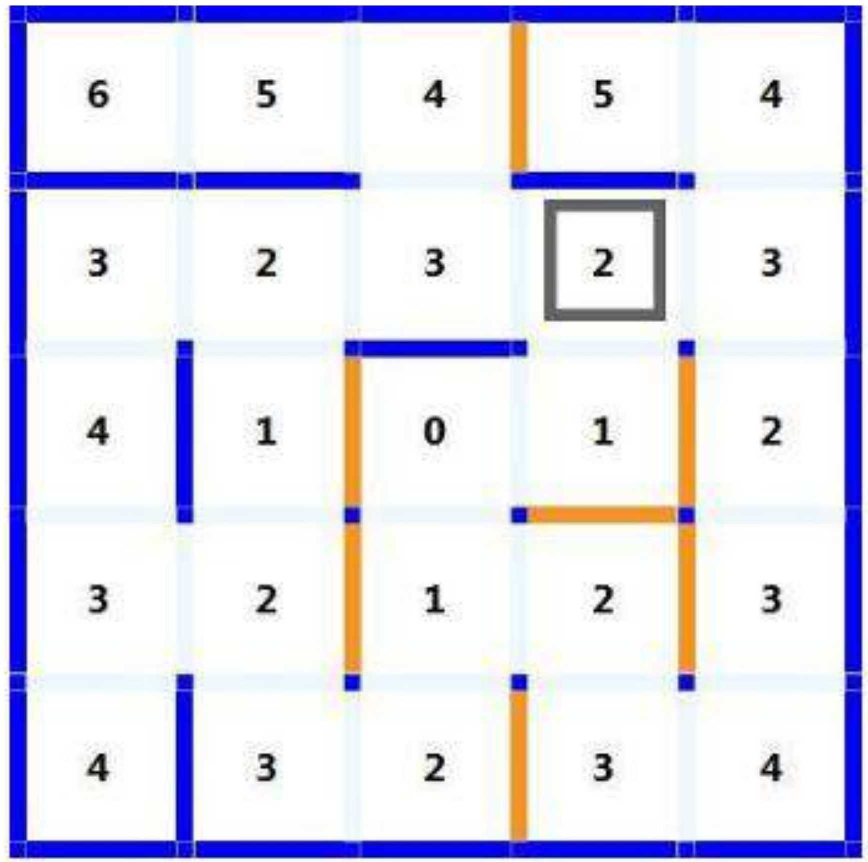
Flood-Fill



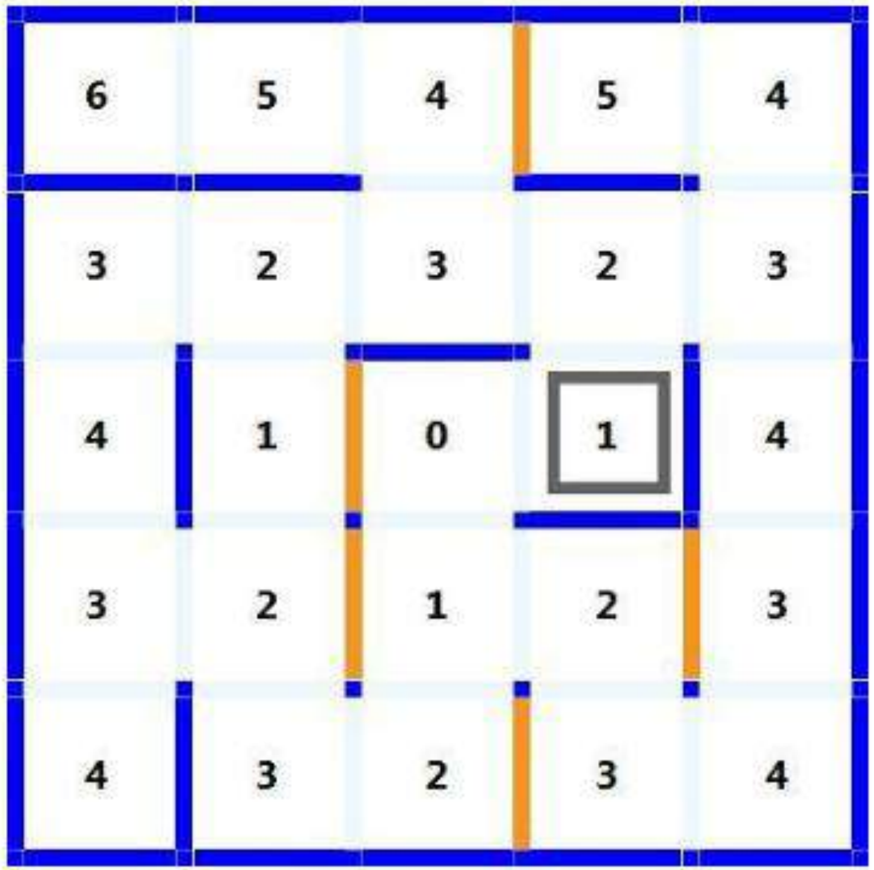
Flood-Fill



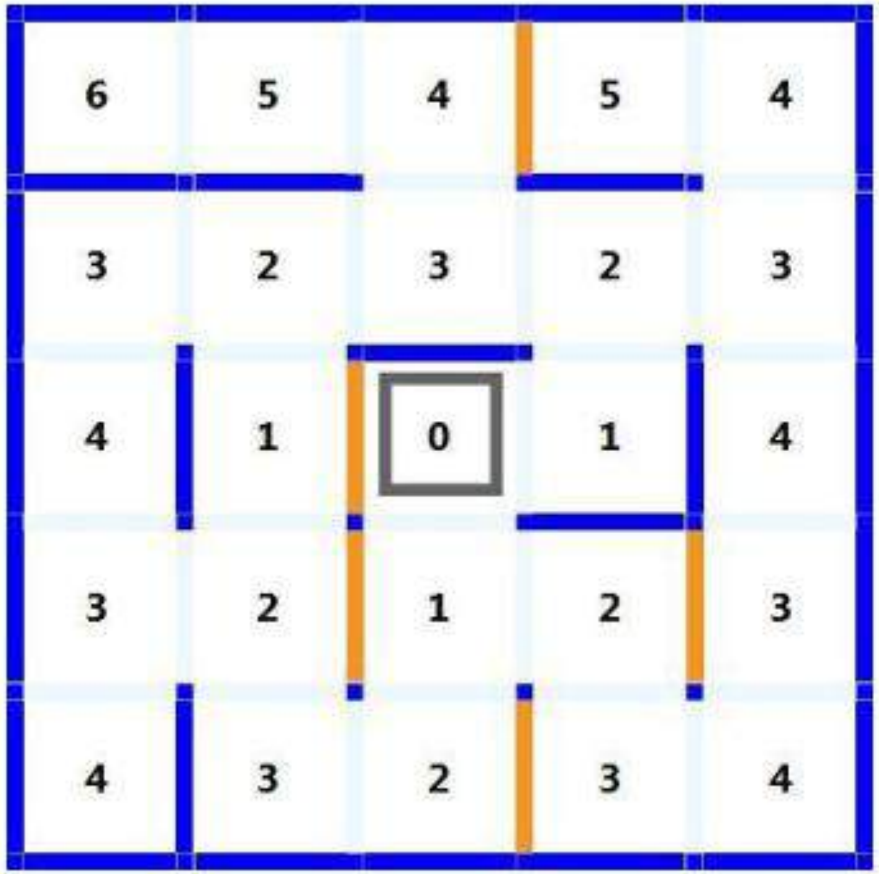
Flood-Fill



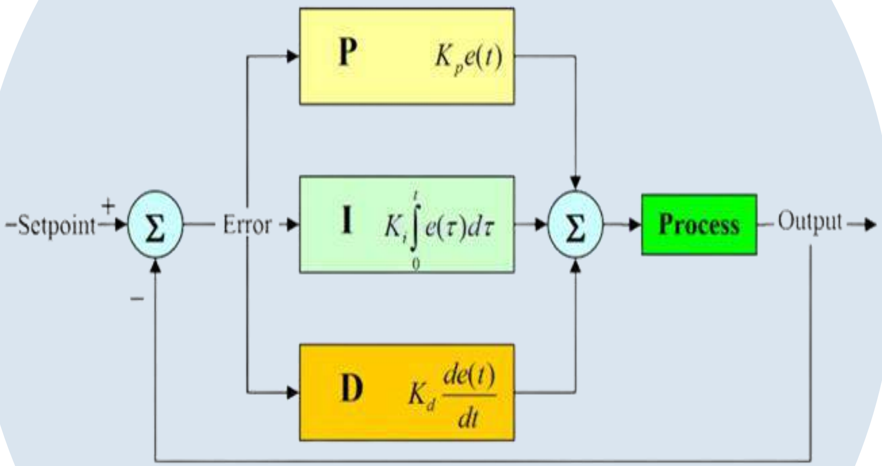
Flood-Fill



Flood-Fill



Motor Controller



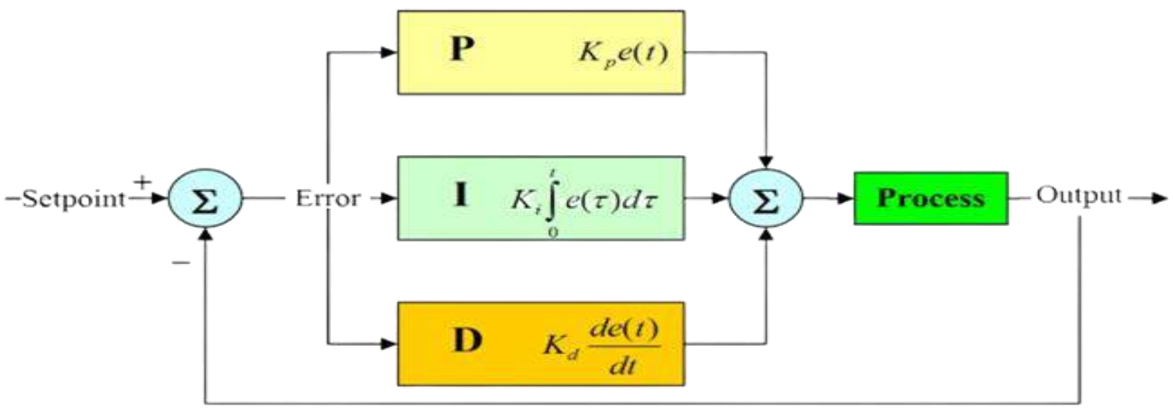
What is PID Control?

PID (Proportional-Integral-Derivative):

- A feedback control algorithm.
- Ensures precise and stable control of systems.

Purpose in Robotics:

- Maintain a straight path by balancing motor speeds.



$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{d}{dt} e(t)$$

Responsibilities of the PID control components (Kp, Ki, Kd)

Proportional Gain (Kp)

- Responsibility: Reacts to the current error.
- Effect: The larger the error, the stronger the correction.
- Drives the system to reduce the error quickly. (may cause overshooting)

Integral Gain (Ki)

- Responsibility: Reacts to the accumulation of past errors.
- Effect: remove drift in the system (removes steady-state error).
- Eliminates steady-state errors over time.

Derivative Gain (Kd)

- Responsibility: Reacts to the rate of change of the error.
- Effect: Predicts and dampens future errors based on the current trend.
- Smoothens response and reduces overshooting.

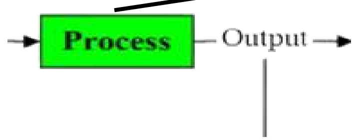
Method 1 - Using Encoders (Apply PID to Reach Reference Speed)

Purpose:

- Ensure each motor reaches and maintains a target speed.

Steps:

- Set Target Speed: Desired speed for each motor.
- Measure Actual Speed: Use encoder data to calculate the current speed of each motor.
- Calculate Speed Error
- Apply PID Algorithm:
    - Compute the correction for each motor using PID.
    - Adjust motor speeds via PWM signals.



Adjust motor speeds dynamically using PID output: decrease the speed of the motor with a higher encoder count and increase the speed of the motor with a lower encoder count. This ensures synchronized rotations and straight-line movement.

Method 2 - Using Encoders (Minimize Difference in Encoder Counts)

- Purpose:
  - Ensure the robot moves straight by synchronizing motor rotations.
- Steps:
  - Measure Encoder Counts: Track the number of rotations for each motor.
  - Calculate Difference in Counts
  - Calculate Speed Error
  - Apply PID Algorithm:
    - Adjust motor speeds to reduce the difference to zero.
    - Synchronize the motors dynamically.

Method 3 - Using MPU6050 (Apply PID for Robot Orientation)

Purpose:

- Maintain the robot's position and heading.

Steps:

- Set Target Orientation: Desired straight heading (yaw angle = 0°).
- Measure Actual Orientation: Use MPU6050 to read the robot's yaw angle.
- Calculate Yaw Error
- Apply PID Algorithm:
  - Adjust motor speeds to correct orientation.
  - Ensure the robot stays aligned with the target path.

فإن وُفِّقتُ فمن فضل الله

The floor is open for your questions and inquiries now only