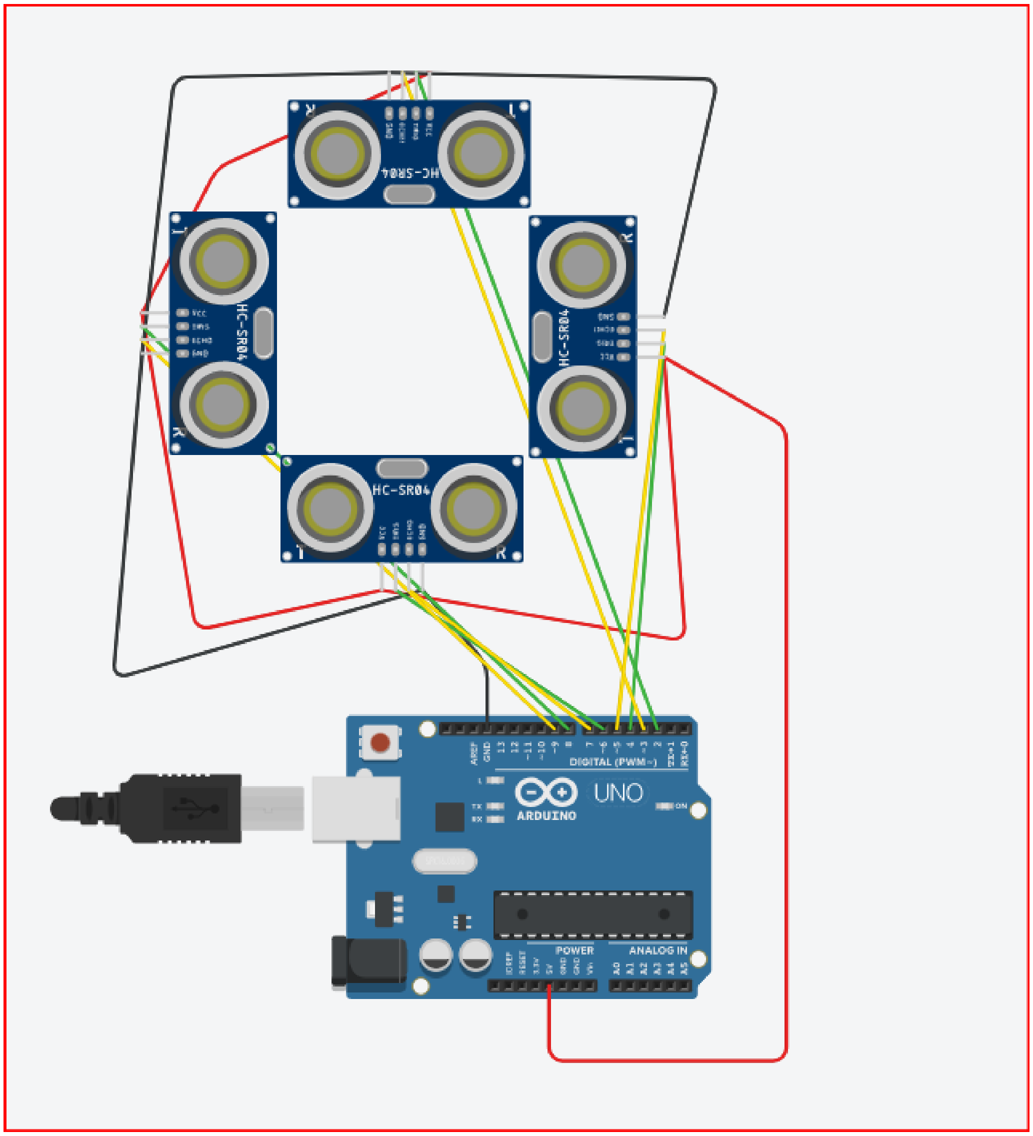
Code

```
#define TRIGGER_PIN_0 2
#define ECHO_PIN_0 3

#define TRIGGER_PIN_90 4
#define ECHO_PIN_90 5

#define TRIGGER_PIN_180 6
#define ECHO_PIN_180 7

#define TRIGGER_PIN_270 8
#define ECHO_PIN_270 9

#define CHAMBER_WIDTH 500
#define CHAMBER_HEIGHT 600

float x = 0;
float y = 0;

void setup() {
  Serial.begin(9600);
  pinMode(TRIGGER_PIN_0, OUTPUT);
  pinMode(ECHO_PIN_0, INPUT);
  pinMode(TRIGGER_PIN_90, OUTPUT);
  pinMode(ECHO_PIN_90, INPUT);
  pinMode(TRIGGER_PIN_180, OUTPUT);
  pinMode(ECHO_PIN_180, INPUT);
  pinMode(TRIGGER_PIN_270, OUTPUT);
  pinMode(ECHO_PIN_270, INPUT);
```

```
}

void loop() {
  int distance_0 = measureDistance(TRIGGER_PIN_0, ECHO_PIN_0);
  int distance_90 = measureDistance(TRIGGER_PIN_90, ECHO_PIN_90);
  int distance_180 = measureDistance(TRIGGER_PIN_180, ECHO_PIN_180);
  int distance_270 = measureDistance(TRIGGER_PIN_270, ECHO_PIN_270);

  x += distance_0 - distance_180;
  y += distance_90 - distance_270;

  if (x < 0) {
    x = 0;
  }
  if (x > CHAMBER_WIDTH) {
    x = CHAMBER_WIDTH;
  }
  if (y < 0) {
    y = 0;
  }
  if (y > CHAMBER_HEIGHT) {
    y = CHAMBER_HEIGHT;
  }

  Serial.print("X: ");
  Serial.print(x);
  Serial.print(" cm, Y: ");
  Serial.print(y);
  Serial.println(" cm");
```

```
  delay(1000);
}

int measureDistance(int triggerPin, int echoPin) {
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);

  long duration = pulseIn(echoPin, HIGH);

  int distance = duration * 0.034 / 2;

  return distance;
}
```

Schematic


HC-SR04
HC-SR04
HC-SR04
HC-SR04
DIGITAL (PWM~)
UNO
ARDUINO
POWER
ANALOG IN

Bonus

Here's an outline of what sensors may be needed and the algorithm to implement:

Sensors:

1. **Lidar or Laser Range Finder**: Lidar sensors emit laser beams and measure the time it takes for the beams to bounce back. This data is used to create a 2D or 3D map of the surroundings, including obstacles and features.
2. **IMU (Inertial Measurement Unit)**: IMU sensors provide information about the robot's acceleration, angular velocity, and orientation. This data helps in estimating the robot's movement.
3. **Wheel Encoders**: These sensors measure the rotation of the robot's wheels, which can be used to estimate the robot's movement over time.
4. **Camera**: Cameras can be used for visual SLAM, where the robot captures images of its environment and extracts features to create a map and localize itself.
5. **Ultrasonic Sensors**: Ultrasonic sensors, like the ones you mentioned earlier, can provide additional distance information for close-range obstacles.

Algorithm:

1. **Extended Kalman Filter (EKF) or Particle Filter**: These are common algorithms used for SLAM. They combine sensor measurements with motion models to estimate the robot's position and simultaneously update the map.
2. **Feature-Based SLAM**: This approach involves detecting and tracking distinct features in the environment, such as corners or edges, using sensors like Lidar or cameras. The robot then builds a map based on the observed features and estimates its position relative to them.
3. **Graph-Based SLAM**: In this method, the environment is represented as a graph, with nodes representing robot poses and edges representing measurements between poses. Optimization techniques like the Gauss-Newton method or Levenberg-Marquardt algorithm are used to find the best estimates of robot poses and map features that minimize measurement errors.
4. **Visual SLAM**: If you're using cameras, Visual SLAM algorithms like ORB-SLAM or PTAM (Parallel Tracking and Mapping) can be employed. These algorithms track visual features in real-time to estimate both the robot's pose and the 3D map.
5. **Monte Carlo Localization (Particle Filter)**: This is a popular approach for mobile robot localization. It involves maintaining a set of particles that represent possible positions of the robot. The particles are updated based on sensor measurements and motion models.
6. **Mapping Techniques**: For map creation, techniques such as occupancy grid mapping or grid-based mapping can be used to build a representation of the environment based on sensor data.