# ***Drone Ring Alignment Simulation using Arduino(Weekly Report)***

## ***1. Task***

***Simulation of Drone Ring Alignment Around a Coconut Tree Trunk using Arduino and Ultrasonic Sensors***

## ***2. Objective***

*The objective is to simulate the behavior of a drone aligning a ring around a vertical coconut tree trunk using an Arduino, ultrasonic sensors, and PID control logic. The alignment ensures the ring is perfectly centered around the trunk by minimizing horizontal and vertical position errors.*

## ***3. System Overview***

*A circular ring equipped with* ***four ultrasonic sensors*** *is designed to detect the position of the coconut tree trunk in four directions:*

* *Left (S1)*
* *Top (S2)*
* *Right (S3)*
* *Bottom (S4)*

*These sensors measure the distances:*

* *d1: Left*
* *d2: Top*
* *d3: Right*
* *d4: Bottom*

*To achieve proper alignment, the goal is to maintain:*

* *d1−d3=0 → horizontal centering*
* *d2−d4=0 → vertical centering*

*This mimics the control behavior of a drone that self-corrects to keep the ring centered around the tree trunk.*

## ***4. Hardware Components***

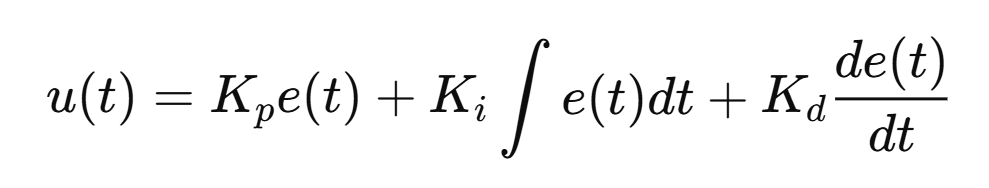
| ***Component*** | ***Quantity*** | ***Purpose*** |
| --- | --- | --- |
| *Arduino Uno/Nano* | *1* | *Main controller* |
| *HC-SR04 Ultrasonic Sensors* | *4* | *Distance measurement* |
| *Servo Motors* | *2* | *Simulate motor response for X and Y alignment* |
| *PVC Pipe or Cardboard Tube* | *1* | *Simulated coconut tree trunk* |
| *Breadboard + Jumper Wires* | *As needed* | *Wiring* |

## ***5. Working Principle***

1. *Arduino continuously reads distances from the four ultrasonic sensors.*
2. *It calculates the errors:*
   * *ex=d1−d3 (left-right)*
   * *ey=d2−d4 (top-bottom)*
3. *A* ***PID controller*** *calculates correction values:*
   * *ux controls servo for horizontal correction.*
   * *uy controls servo for vertical correction.*
4. *Servos simulate drone's positional adjustments.*

## ***6. Control Algorithm***

### ***PID Equation:***



*This helps achieve stable, smooth control without oscillations.*

## ***7. Pin Configuration***

| ***Sensor*** | ***Trigger*** | ***Echo*** |
| --- | --- | --- |
| *S1 (Left)* | *Pin 2* | *Pin 3* |
| *S2 (Top)* | *Pin 4* | *Pin 5* |
| *S3 (Right)* | *Pin 6* | *Pin 7* |
| *S4 (Bottom)* | *Pin 8* | *Pin 9* |

| ***Servo*** | ***Pin*** |
| --- | --- |
| *Servo X (horizontal)* | *10* |
| *Servo Y (vertical)* | *11* |

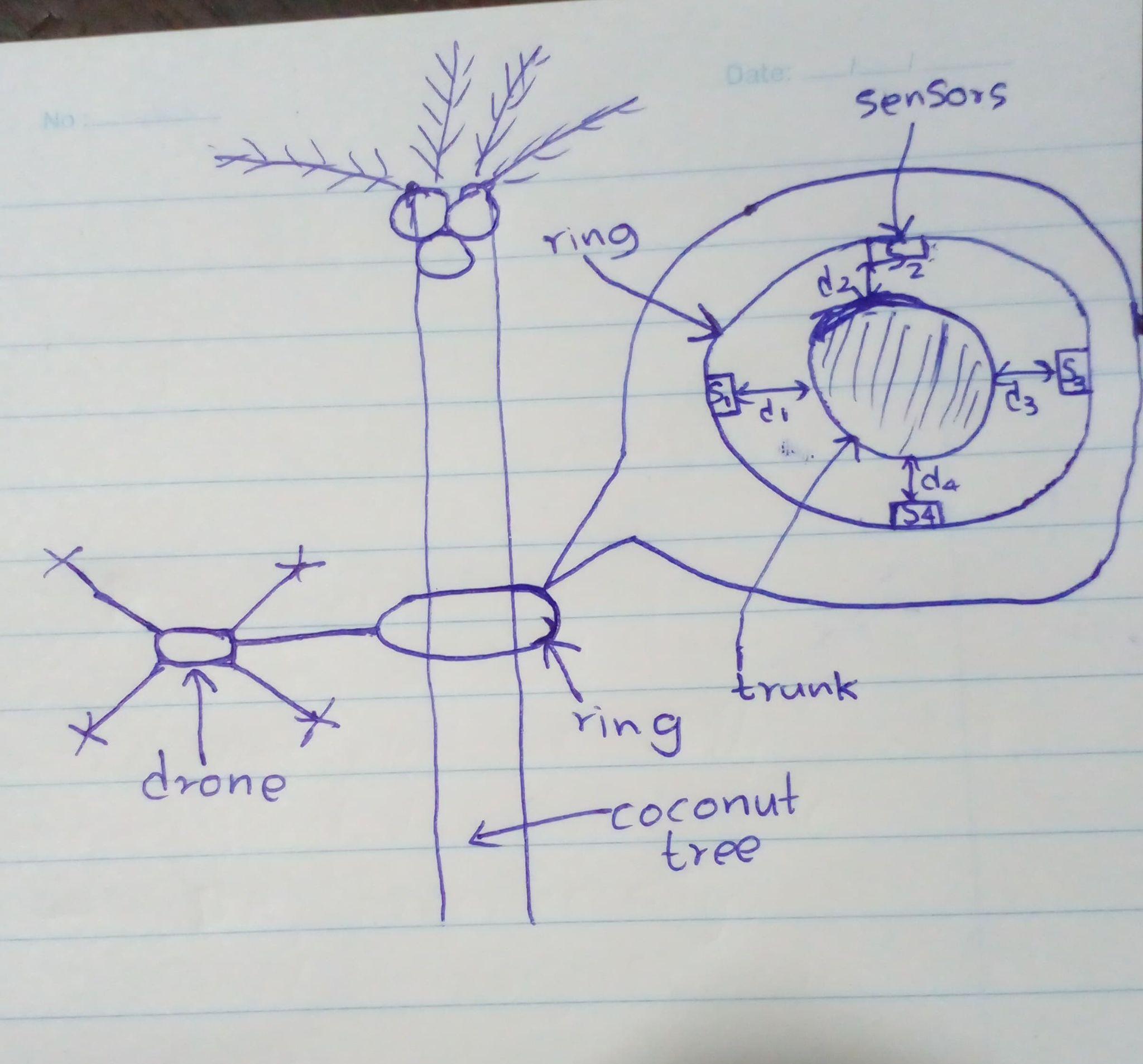
## ***8. Testing Setup***

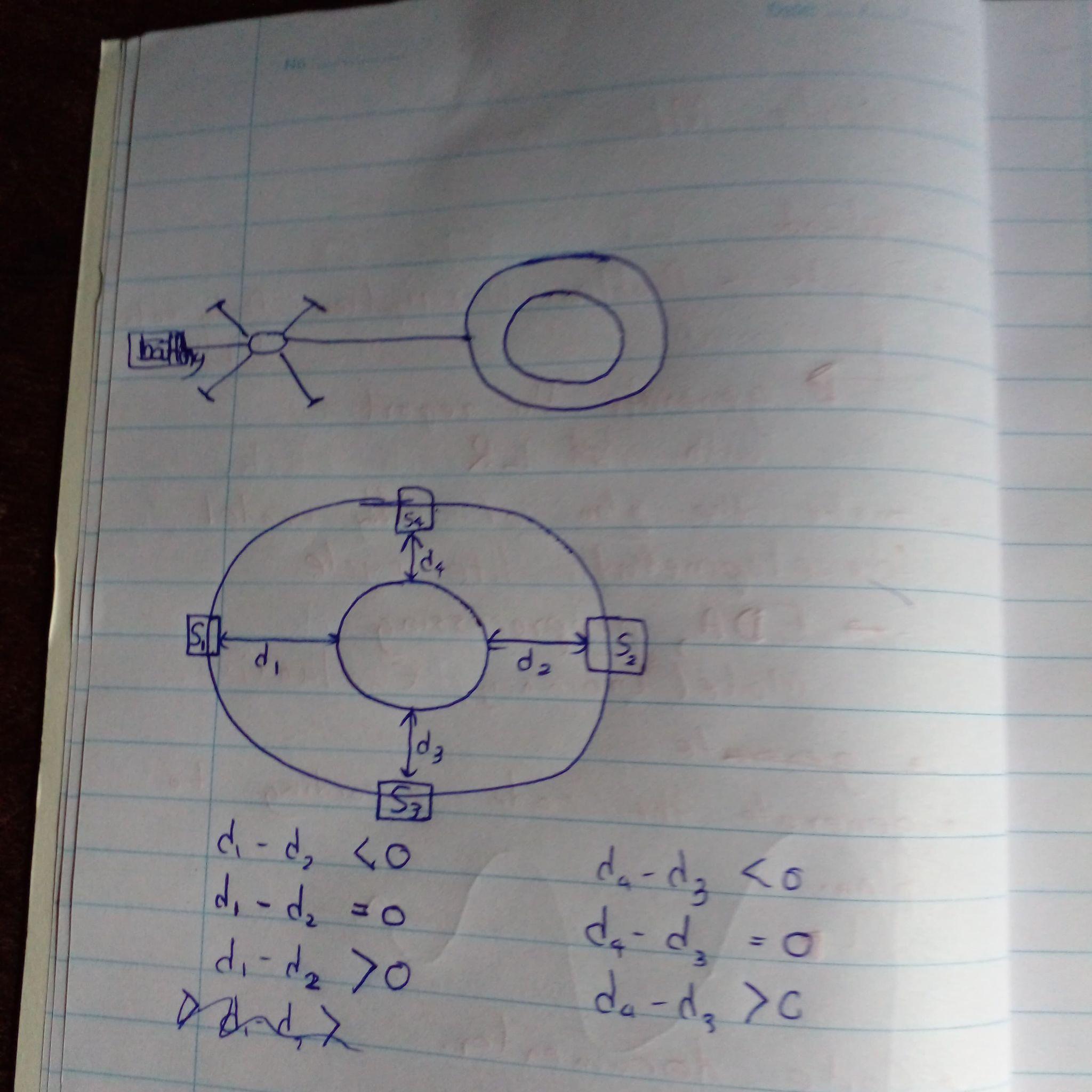
* *A small stand holds the ring.*
* *A vertical pipe simulates the coconut tree trunk.*
* *Ring is moved off-center manually.*
* *The Arduino detects offset and adjusts servos to simulate re-centering.*

## ***9. Sample Serial Output***

*// d1: 10.1 d2: 11.2 d3: 9.8 d4: 11.4 | ex: 0.3 ey: -0.2 | uX: 1.2 uY: -0.6*

## ***10. Sketch***





## ***11. Arduino Code***

#include <Servo.h>

// ----- Sensor Pins -----

const int trigPins[4] = {2, 4, 6, 8}; // S1, S2, S3, S4 (Trig)

const int echoPins[4] = {3, 5, 7, 9}; // S1, S2, S3, S4 (Echo)

// ----- Servo Pins -----

const int servoXPin = 10; // Horizontal (Roll)

const int servoYPin = 11; // Vertical (Pitch)

// ----- Servo Setup -----

Servo servoX;

Servo servoY;

// ----- PID Parameters -----

float Kp = 2.0;

float Ki = 0.0;

float Kd = 0.5;

float ex\_prev = 0, ey\_prev = 0;

float ix = 0, iy = 0;

unsigned long lastTime = 0;

void setup() {

Serial.begin(9600);

servoX.attach(servoXPin);

servoY.attach(servoYPin);

// Set servos to neutral

servoX.write(90);

servoY.write(90);

for (int i = 0; i < 4; i++) {

pinMode(trigPins[i], OUTPUT);

pinMode(echoPins[i], INPUT);

}

lastTime = millis();

}

void loop() {

// Read distances

float d[4];

for (int i = 0; i < 4; i++) {

d[i] = readUltrasonic(trigPins[i], echoPins[i]);

}

// Compute errors

float ex = d[0] - d[2]; // d1 - d3 (left - right)

float ey = d[1] - d[3]; // d2 - d4 (top - bottom)

// Compute time delta

unsigned long now = millis();

float dt = (now - lastTime) / 1000.0;

lastTime = now;

// PID for X-axis

ix += ex \* dt;

float dx = (ex - ex\_prev) / dt;

float ux = Kp \* ex + Ki \* ix + Kd \* dx;

ex\_prev = ex;

// PID for Y-axis

iy += ey \* dt;

float dy = (ey - ey\_prev) / dt;

float uy = Kp \* ey + Ki \* iy + Kd \* dy;

ey\_prev = ey;

// Map to servo (0-180) with 90 as center

int servoXVal = constrain(90 + ux, 0, 180);

int servoYVal = constrain(90 + uy, 0, 180);

servoX.write(servoXVal);

servoY.write(servoYVal);

// Debug output

Serial.print("d1: "); Serial.print(d[0]);

Serial.print(" d2: "); Serial.print(d[1]);

Serial.print(" d3: "); Serial.print(d[2]);

Serial.print(" d4: "); Serial.print(d[3]);

Serial.print(" | ex: "); Serial.print(ex);

Serial.print(" ey: "); Serial.print(ey);

Serial.print(" | uX: "); Serial.print(ux);

Serial.print(" uY: "); Serial.println(uy);

delay(100); // 10 Hz control loop

}

float readUltrasonic(int trigPin, int echoPin) {

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

long duration = pulseIn(echoPin, HIGH, 30000); // 30ms timeout

return duration \* 0.0343 / 2.0; // distance in cm

}