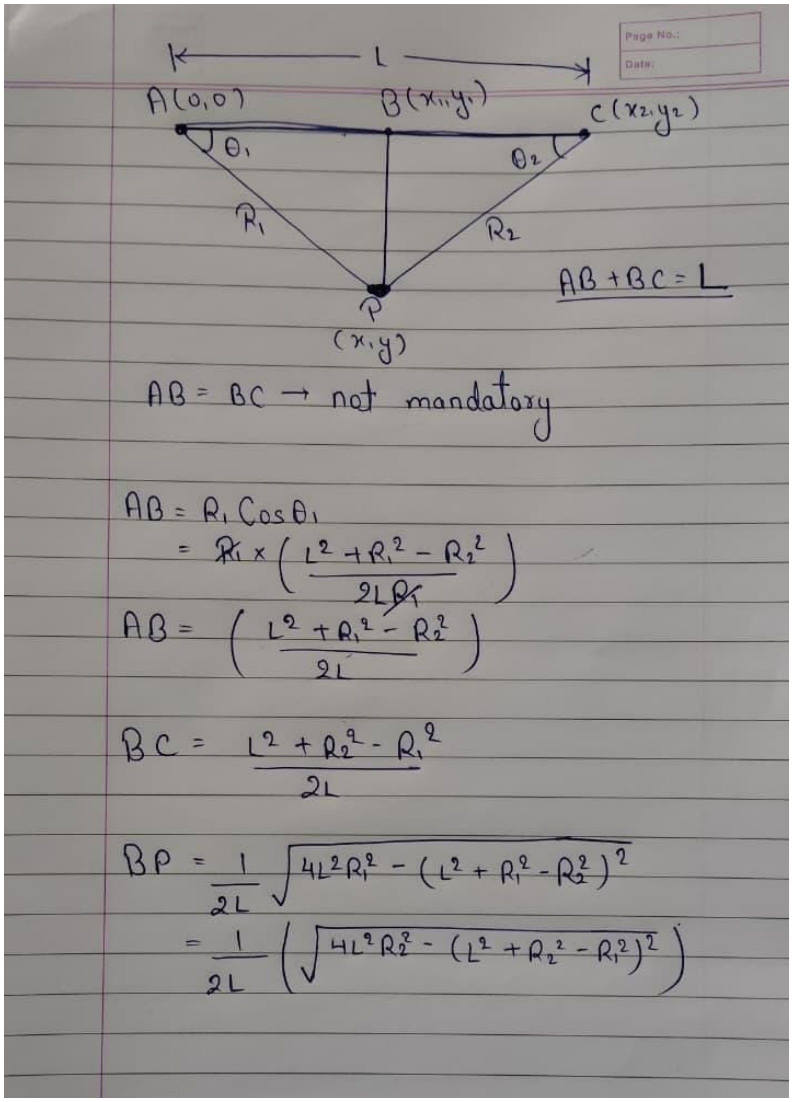
Air-Pen Geometry and Position Estimation

`![Air Pen Geometry](air_pen_geometry.png)`



1. System Overview

The Air-Pen system tracks the 2-D position of a pen tip (**point P**) using two distance measurements from fixed reference points.

- Anchor **A** = (0, 0)
- Anchor **C** = (L, 0)
- Fixed baseline distance **L = 14 cm**
- Pen tip **P = (x, y)** (unknown, time-varying)
- Sensor measurements:
  - **R1** = distance from A to P
  - **R2** = distance from C to P
- R1 and R2 are transmitted via ESP32 over serial

The objective is to compute (x, y) in real time and render pen motion on the screen.

---

2. Mathematical Model

Using Euclidean distance:

$$R1^2 = x^2 + y^2$$
$$R2^2 = (x - L)^2 + y^2$$

Subtracting the equations eliminates y^2:

$$R2^2 - R1^2 = L^2 - 2Lx$$

Solving for x:

$$x = (L^2 + R1^2 - R2^2) / (2L)$$

---

3. Computing y

Substitute x into the first equation:

$$y^2 = R1^2 - x^2$$

$$y = \pm\sqrt{R1^2 - x^2}$$

Sign Selection

Geometry yields two solutions. In practice:

- Pen always below baseline → use **negative y**
- Pen always above baseline → use **positive y**

Failing to enforce this will cause vertical flipping in animations.

---

4. Relation to Diagram Variables

- $AB = x = (L^2 + R1^2 - R2^2) / (2L)$
- $BP = |y| = \sqrt{R1^2 - AB^2}$

No symmetry assumption ($AB = BC$) is required.

---

5. Implementation Notes

Data Flow
1. ESP32 sends (R1, R2) over serial
2. Python reads values
3. Compute (x, y)
4. Scale to screen coordinates
5. Render animation frame

Scaling
$x_{px} = s \cdot x$
$y_{px} = s \cdot y$

Where s is pixels per centimeter.

---

6. Numerical Stability

If $R1^2 - x^2 < 0$:
- Sensor noise or invalid geometry
- Discard or clamp the frame
- Do NOT take sqrt blindly

---

7. Summary

- This is a 2-D trilateration problem
- x is uniquely determined
- y requires a sign convention
- Noise handling is mandatory
- Symmetry assumptions will break real-world behavior