Spring 2025

Yeditepe University
Mechanical Engineering Department

ME 456

Mechatronics

Workshop #3

Triangulation with 2 Sensors

FINAL REPORT

Instructor

- Assoc. Dr. Koray Kadir Şafak

Lab Partners

- Abdullah Serdar Özdemir 20200705049
- Ömer Dolaş 20200705001
- Abdulkadir Salman 20200705027
- Furkan Yılmaztekin 20180705040

Due Date: 25/04/2025 Group Number: 5

Table Of Content

DISTANCE MEASUREMENT **3**

TARGET VELOCITY CALCULATION **4**

1. Distance Measurement

- **Objective:** In the 3rd Lab mission, it was planned to add a second ultrasonic sensor to the robot platform and measure two different distances of the target object through this sensor. In this context, the distance data obtained was processed with the following formulas and its position on the 2D coordinate plane was calculated.

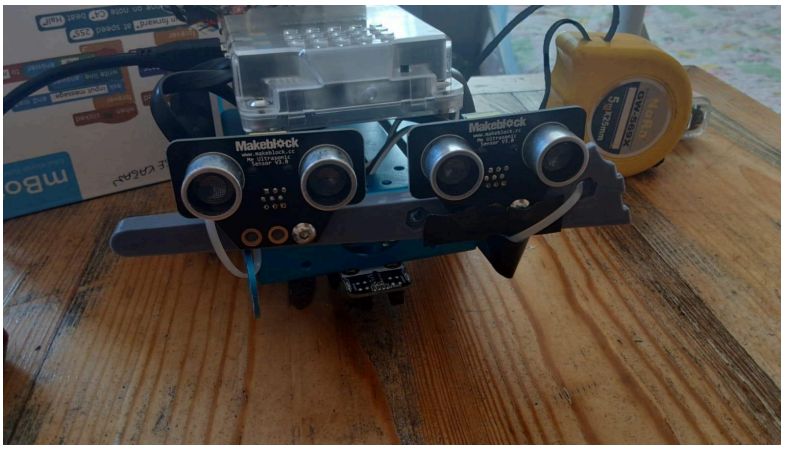


Figure-1: Twin Ultrasonic Sensor Integration

$$(d_1, d_2) \rightarrow (x, y)$$

$$y^2 + \left(x + \frac{L}{2}\right)^2 = d_2^2$$

$$y^2 + \left(x - \frac{L}{2}\right)^2 = d_1^2$$

- **Code and Explanation:** The following section contains the code and explanations prepared for the lab task.

```
#include <MeMCore.h>
```

⇑ *imports the mBot library to control sensors and components.*

```
MeUltrasonicSensor sensor1(PORT_3); // S1 - sol sensör
MeUltrasonicSensor sensor2(PORT_4); // S2 - sağ sensör
```

⇑ *defines two ultrasonic sensors connected to port 3 (left) and port 4 (right).*

```
float d1, d2;     // sensör ölçümleri
float x, y;       // target konum
float L = 7.5;   // iki sensörün arası
```

⇑ *declares variables for distance readings and target position. L is the baseline distance between sensors.*

```
void setup() {
  Serial.begin(9600);
}
```

⇑ *starts serial communication to send data to the computer at 9600 baud rate*

```
void loop() {
  // ölçümleri al
  d1 = sensor1.distanceCm();
  d2 = sensor2.distanceCm();
```

⇑ *reads distance values from the left and right ultrasonic sensors.*

```
  // x yeri
  x = (pow(d1, 2) - pow(d2, 2)) / (2 * L);
```

⇑ *calculates the x − position of the target using the triangulation formula.*

```
  // y yeri
  float temp = pow(d1, 2) - pow(x - (L / 2.0), 2);
```

⇑ *prepares the inside of the square root to compute the y − position geometrically.*

```
if (temp >= 0) {
    y = sqrt(temp); // reel sayı ver
  } else {
    y = -1; // Hata
  }
```

⇑ *prepares the inside of the square root to compute the y − position geometrically.*

```
  // serialden bas
  Serial.print("d1: "); Serial.print(d1); Serial.print(" cm\t");
  Serial.print("d2: "); Serial.print(d2); Serial.print(" cm\t");
  Serial.print("x: "); Serial.print(x); Serial.print(" cm\t");
  Serial.print("y: "); Serial.println(y); // cm
```

⇑ *prints all distance and position values to the serial monitor in a readable format.*

```
  delay(200);
}
```

⇑ *waits 200 milliseconds before repeating the loop (controls reading frequency).*

2. Target Velocity Calculation

- **Objective:** The objective of this exercise is to add two ultrasonic sensors with real-time velocity estimation of a moving target to the position tracking system. The velocity of the target in the x and y directions is calculated using a numerical differentiation method, based on the change in position over a defined time interval. The system keeps the current and previous position measurements and computes the velocities V_x and V_y, respectively. The objective is to extrapolate the position and velocity data continuously cm/s, while keeping uniform timing between measurements. This enhancement enables dynamic target tracking and improves the robot's capability to sense and respond to movement in its environment.

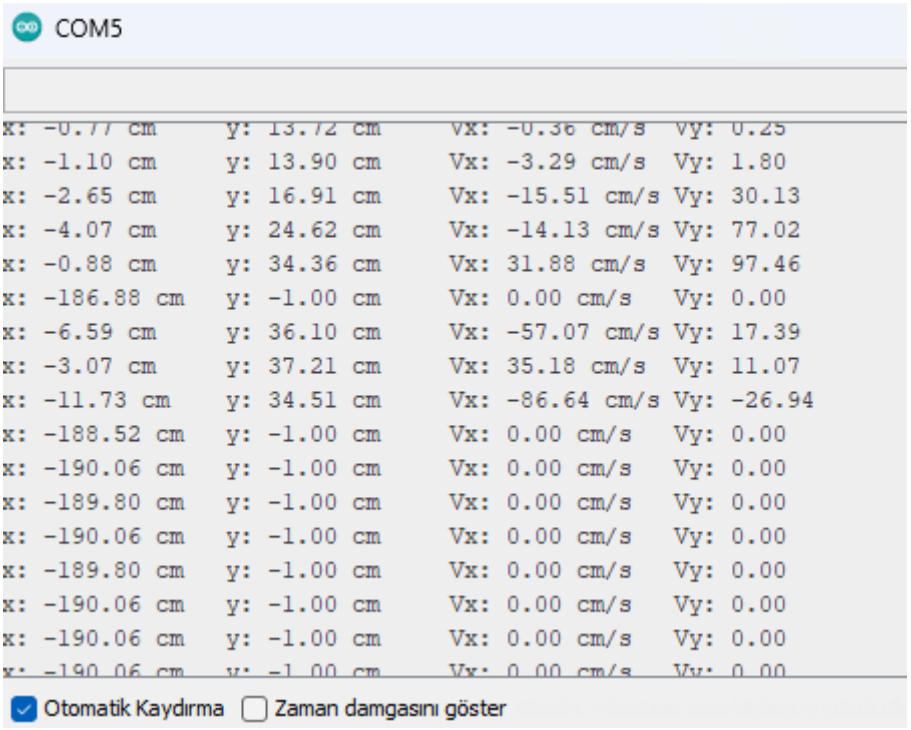
$$V_x(k) = \frac{x(k) - x(k-1)}{\Delta T}$$

$$V_y(k) = \frac{y(k) - y(k-1)}{\Delta T}$$

- **Code and Explanation:** The following section contains the code and explanations prepared for the lab task.

```
#include <MeMCore.h>
↑ includes the mBot library to control hardware components like sensors and motors

MeUltrasonicSensor sensor1(PORT_3); // sol
MeUltrasonicSensor sensor2(PORT_4); //sağ
↑ defines two ultrasonic sensors connected to PORT_3 and PORT_4

float d1, d2;
↑ distance readings from the sensors (in cm)
float x, y;
 ↑ calculated position of the target
float prev_x = 0, prev_y = 0;
↑ stores previous position values
float vx = 0, vy = 0;
↑ velocity components (x and y directions)
float L = 12.0;
↑ distance between the centers of the two sensors (in cm)

unsigned long prevMillis = 0;
↑ time of last measurement
const unsigned long interval = 100;
↑ measurement interval (ms)
float deltaT = interval / 1000.0;
↑ time step converted to seconds (e.g., 0.1s)

void setup() {
  Serial.begin(9600);
↑ starts serial communication for debugging/output
}

void loop() {
  unsigned long currentMillis = millis();
↑ get current time in milliseconds

  if (currentMillis - prevMillis >= interval) {
↑ only continue if interval has passed
    prevMillis = currentMillis;
↑ update previous time marker

    d1 = sensor1.distanceCm();
 ↑ read distance from left sensor
    delay(20);
↑ short delay to avoid interference (sometimes it go crazy with the results)
    d2 = sensor2.distanceCm();
↑ read distance from right sensor


    if (d1 > 65) d1 = 65;
↑ clamp d1 to max 65 cm
    if (d2 > 65) d2 = 65;
↑ clamp d2 to max 65 cm

    x = (pow(d1, 2) - pow(d2, 2)) / (2 * L);
```

```
  ↑ triangulation formula derived from circle intersection


    float temp = pow(d1, 2) - pow(x - (L / 2.0), 2);
↑ inner part of square root
    if (temp >= 0) {
      y = sqrt(temp);
↑ valid y value
    } else {
      y = -1;
↑ invalid reading, set error flag
    }

    if (y != -1) {
      vx = (x - prev_x) / deltaT;
 ↑ velocity in x (cm/s)
      vy = (y - prev_y) / deltaT;
 ↑ velocity in y (cm/s)

      prev_x = x;
↑ update previous x
      prev_y = y;
↑ update previous y
    } else {
       ↑ skip velocity update if y is invalid
    }

    Serial.print("x: "); Serial.print(x); Serial.print(" cm\t");
    Serial.print("y: "); Serial.print(y); Serial.print(" cm\t");
    Serial.print("Vx: "); Serial.print(vx); Serial.print(" cm/s\t");
    Serial.print("Vy: "); Serial.println(vy);
↑ end of output line
  }
}


      vx = 0;
      vy = 0;
    }

    Serial.print("x: "); Serial.print(x); Serial.print(" cm\t");
    Serial.print("y: "); Serial.print(y); Serial.print(" cm\t");
    Serial.print("Vx: "); Serial.print(vx); Serial.print(" cm/s\t");
    Serial.print("Vy: "); Serial.println(vy);
  }
}
```



- **Discussion:** It has been determined that some sensor data gives incorrect feedback while performing the task. In this context, a delay mechanism has been established in order for the two sensors to work synchronously and to prevent the signals they send from interfering (cross-talk). This system ensures that one sensor works and sends frequency while the other remains inactive. This system provides positive feedback for position calculation, but it can produce incorrect results in speed calculation. Because the data comes in pieces.

In addition to all these, in the tests conducted on the system, it was observed that more optimal position feedback was measured for objects with small surface areas when they were stationary, but errors occurred in speed calculations when moving. The opposite is the case for objects with large surface areas. Errors were occasionally observed in position feedback when stationary, but speed data in motion were mostly calculated correctly.