Mechatronics System Integration (MCTA3203)

Week **12**: Distance-Triggered Door Automation Using TF-Luna ToF LiDAR Sensor (ver. **1**)

Automatic Door System Using TF-Luna ToF LiDAR and ESP32

Objectives:

- Interface the TF-Luna ToF LiDAR sensor with ESP32
- Read and interpret distance data via serial monitor
- Control a 12V linear actuator via relay based on sensor input
- Implement a basic threshold logic for distance-based automation
- Understand and apply actuator safety using timers or reset conditions

Required Hardware:

- NodeMCU ESP32 (or Robo ESP32)
- TF-Luna ToF LiDAR Sensor
- 12V linear actuator (100mm)
- Relay Module (1-Channel)
- 12V Power Supply
- 2x Limit Switches (optional)

Pre-lab Preparation

Students must:

- Review ESP32 pinout
- Understand basic UART/I2C communication
- Read TF-Luna datasheet section on communication protocol
- Revise how a relay works and how to safely switch 12V actuator

⚠ **Safety & Tips**

- Ensure power supply voltage matches components.
- Avoid forcing servo/actuator beyond physical limits.
- Keep wiring neat to avoid shorts.
- Respect time limits; focus on reliable basic operation.

Part 1 – Hardware Setup (20 mins)

- Connect TF-Luna sensor to ESP32 using UART (TX/RX).
- Connect relay module to ESP32 digital output (e.g., GPIO 5).
- Connect linear actuator to relay output.
- Set up 12V power to actuator (separate supply, common GND with ESP32).
- Optional: Add *limit switches* on both ends of actuator stroke for safety.

Part 2 – Basic Code and Distance Reading (20 mins)

Upload sketch to ESP32 to:

- Read distance from TF-Luna via UART
- Display real-time values on *Serial Monitor*

```
#include <HardwareSerial.h>

HardwareSerial mySerial(1); // Use UART1

void setup() {
  Serial.begin(115200);
  mySerial.begin(115200, SERIAL_8N1, 16, 17); // RX = GPIO16, TX = GPIO17
}

void loop() {
  while (mySerial.available()) {
    int dist = mySerial.read();
    Serial.println(dist);
  }
}
```

Note: Students may need to parse multi-byte frames per TF-Luna protocol — simplified here for learning pace.

Part 3 – Automation Logic (30 mins)

Modify code to:

- If distance > 200 cm → deactivate relay → close actuator
- If distance < 150 cm → activate relay → open actuator

```
#define RELAY_PIN 5

void setup() {
  Serial.begin(115200);
  mySerial.begin(115200, SERIAL_8N1, 16, 17);
  pinMode(RELAY_PIN, OUTPUT);
  digitalWrite(RELAY_PIN, LOW);
}

void loop() {
  static uint16_t distance = 999;

  if (mySerial.available() >= 9) {
    uint8_t buffer[9];
    mySerial.readBytes(buffer, 9);
    if (buffer[0] == 0x59 && buffer[1] == 0x59) {
      distance = buffer[2] + buffer[3] * 256;
      Serial.print("Distance: ");
      Serial.print(distance);
      Serial.println(" mm");

      if (distance < 1500) {
        digitalWrite(RELAY_PIN, HIGH); // Open actuator
      } else if (distance > 2000) {
        digitalWrite(RELAY_PIN, LOW); // Close actuator
      }
    }
  }
}
```

Additional Tasks (To complete in the lab)

Students must complete and demonstrate the following before session ends:

- Demonstrate actuator opening and closing based on distance threshold (10 mins)
- Modify the code to include a delay to keep door open for 5 seconds (5 mins)
- Add an LED indicator that lights up when door is open (5 mins)
- Discuss and document how to handle actuator safety using limit switches (10 mins)

Post-Lab Report (To be submitted next session)

Students must include:

- Final working code
- Schematic diagram (hand-drawn or software)
- Screenshot or photo of working setup
- Short reflection: “What could go wrong in real-world use?”

Evaluation Criteria

Criteria	Marks
Functional Hardware Setup	20
Code Accuracy & Threshold Logic	25
Successful Completion of Additional Tasks	25
Report Clarity and Reflection	20
Creativity / Improvements Suggested	10
Total	**100**