Summer 2025
IoT SW & HW Engineering - Field Training

Assignment 3

Deadline: 31th of July, 2025 at 11:59. All team members submit on teams.

Problem 1: Conceptual Questions

Answer the following questions with references and at least one figure per each question or one diagram.

- **Theoretical Assignment Questions**

1. IR Sensor Basics

What is the difference between an analog IR sensor and a digital IR sensor, and how does the ESP32 read signals from each type?

2. ADC in ESP32

Why does the ESP32 need an Analog-to-Digital Converter (ADC) when using an IR sensor, and what does the ADC actually do?

3. PWM for Servo Control

What is Pulse Width Modulation (PWM), and why is it used to control servo motor angles instead of sending a simple HIGH or LOW signal?

4. Duty Cycle Meaning

What does “duty cycle” mean in PWM, and how does changing it affect the position of a servo motor?

5. Analog vs. Digital Signals

What is the main difference between analog and digital signals, and why does an LDR or IR sensor often give analog output while a push button gives digital output?

6. LCD Communication

How does the ESP32 send data to an LCD (e.g., using I2C or parallel communication), and why is I2C often preferred?

7. Debouncing Concept

When using a push button to control an LED or servo, why is “debouncing” needed, and what problem does it solve?

🔹 ESP32 Project – Problem 1: Smart Door System

📌Description

Design an ESP32-based Smart Door system using a servo motor and an analog IR sensor. The door should open automatically when a person is detected and close after a specified time or when no presence is detected.

📌Requirements

- Detects a person using an analog IR sensor.
- Control a servo motor: rotate from 0° → 180° (open), then back to 0° (close).
- Implement smooth automatic transitions.
- Provide a schematic using **Fritzing**.
- Validate the design using **Wokwi simulation** (instead of Tinkercad).

📌Deliverables

1. ESP32 source code.
2. Fritzing schematic file.
3. Wokwi simulation link or screenshots.
4. Documentation explaining the wiring, code, and operation in pdf format.
5. Video documentation in mp4.

🔹 ESP32 Project – Problem 2: Keypad-Based Servo Controller

📌Description

Develop an ESP32 system to control an SG90 servo motor using a keypad interface, allowing selection of predefined angles.

📌Requirements

- Interface a keypad with ESP32.
- Map keys to specific servo angles (30°, 45°, 60°, 90°, 120°, 180°).
- Implement a Stop button (freeze servo position).
- Include a Reset button (return servo to 0°).
- Provide schematic and **Wokwi simulation**.
- If you don't have a keypad, map the IR analog reading to the servo angle.

📌Deliverables

1. Fully functional ESP32 code.
2. Circuit schematic (Fritzing).
3. Wokwi simulation link/screenshots.
4. Documentation explaining the wiring, code, and operation in pdf format.
5. Video documentation in mp4.

🔹 ESP32 Project – Problem 3: IR Sensor Real-Time LCD Display

📌**Description**

Build an ESP32 project that reads analog values from an IR sensor, calculates distance, and displays both on an LCD.

📌**Requirements**

- Continuously read analog IR sensor values.
- Display readings and calculated distance on an LCD.
- Detect missing/poor sensor connection and display a warning:
  "No input detected, please connect the sensor."
- Provide Fritzing schematic and Wokwi simulation.

📌**Deliverables**

1. ESP32 code with real-time display and error handling.
2. Circuit schematic.
3. Wokwi simulation link/screenshots.
4. Documentation explaining the wiring, code, and operation in pdf format.
5. Video documentation in mp4.

🔹 ESP32 Project – Problem 4: Light-Sensitive Alarm and Door Control

📌Description

Create an ESP32 system combining an **LDR, buzzer, LED, push button, and servo motor**. The system acts as a light-sensitive security door:

📌Requirements

- Use LDR to detect light levels.
- If light < threshold → Activate buzzer & LED, and rotate servo to 180° (lock mode).
- If light ≥ threshold → Stop buzzer, turn off LED, and return servo to 0°.
- Add a **manual override push button** that resets servo to 0° and silences alarms regardless of light level.
- Provide Fritzing schematic and **Wokwi simulation**.

📌Deliverables

1. ESP32 code for full functionality.
2. Circuit schematic.
3. Wokwi simulation link/screenshots.
4. Documentation explaining the wiring, code, and operation in pdf format.
5. Video documentation in mp4.

🔹 ESP32 Project – Problem 5: Intelligent Multi-Sensor Security and Environment Control System

📌 Description

Develop an ESP32-based system that integrates **LDR, IR sensor, LCD, servo motor, buzzer, LED, and a push button** to manage environmental monitoring and automated security actions. This project combines light-based automation, intrusion detection, and manual override control.

📌 Requirements

1. **Ambient Light Monitoring (LDR):**
    - Continuously measure ambient light intensity.
    - If light < a threshold → Enter "Night Security Mode":
        - Close a servo-controlled door/window to **180° (locked position)**.
        - Turn **ON LED** as a visual indicator.
        - Display "Night Mode: Door Locked" on the LCD.
    - If light ≥ threshold → Return servo to **0° (open)**, turn **OFF LED**, and update LCD.
2. **Intrusion Detection (IR Sensor):**
    - Continuously read IR sensor data.
    - If an object/person is detected during Night Mode →
        - Trigger **buzzer alarm** for 5 seconds.
        - Display "Alert: Motion Detected!" on LCD.

3. **Manual Override (Push Button):**
    - A push button should allow the user to **immediately unlock (servo to 0°)** and **silence the buzzer/LED** regardless of sensor readings.
    - LCD should display "Manual Override Activated" when pressed.
4. **Error Handling:**
    - If any sensor returns invalid readings (e.g., disconnected), display: "Sensor Error - Check Wiring".
5. **Simulation and Documentation:**
    - Provide a **Wokwi simulation** demonstrating full functionality.
    - Provide a **Fritzing schematic** showing all component connections.

📌 **Deliverables**

1. ESP32 source code implementing all described logic.
2. Circuit schematic (Fritzing).
3. Wokwi simulation link or screenshots.
4. Documentation explaining the wiring, code, and operation in pdf format.
5. Video documentation in mp4.