Final Project Report

Smart Rain-Sensing System – *Auto Shade*

Session: 2023 – 2027

Submitted by:

Ahmad Bin Rashid	2023-CS-53
Abu Tayyab	2023-CS-54
Mian Saad Tahir	2023-CS-62
Muhammad Talha	2023-CS-70

Supervised by:

Sir Syed Tehseen-ul-Hasan Shah

Ma’am Aroosh Fatima

Course:

CSC-205L Computer Organization and Assembly Language Lab

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

Table of Contents

Table of Figures

1- Project Title:

Smart Rain-Sensing System – *Auto Shade*

2- Project Abstract:

The Smart Rain-Sensing System – Auto Shade is a practical project designed to automatically protect outdoor items from rain using a shade controlled by a servo motor. The system uses an ESP32 microcontroller to read data from a water sensor, which detects rain by measuring moisture levels. The ESP32 sends this information to an Arduino Uno through a UART connection. If the sensor detects rain (a value above 200), the Arduino displays "Rain Detected" on a 16x2 LCD screen and moves the servo to 0° to close the shade. If no rain is detected (a value below or equal to 200), it shows "Clear Weather" on the LCD and moves the servo to 90° to open the shade. The system includes a 5V power supply and a 3.9kΩ resistor for the LCD contrast, ensuring reliable operation. This setup provides an efficient, automatic solution to shield outdoor equipment from weather changes.

3- Introduction:

- The Smart Rain-Sensing System – Auto Shade is a project to automatically cover outdoor items during rain.
- It uses an ESP32 to check a water sensor that measures moisture to detect rain.
- The ESP32 sends the sensor data to an Arduino Uno using a UART connection, where the ESP32's TX2 pin connects to the Arduino's RX0 pin, and both their GND pins are connected together for a common ground.
- The Arduino Uno shows the weather status on a 16x2 LCD screen: "Rain Detected" if the sensor value is above 200, or "Clear Weather" if the value is 200 or below.
- The Arduino also controls an SG90 servo motor to move the shade: it sets the servo to 0° to close the shade during rain, and to 90° to open it when there's no rain.
- The system runs on a 5V power supply, and a 3.9kΩ resistor adjusts the LCD contrast for clear display.
- This project makes it easy to protect outdoor things from rain without manual work.

4- Components Used and Their Description:

Components	Quantity	Description
ESP32 Devkit V1	1	Reads water sensor data, sends status to Arduino via UART (TX2 to RX0), powered by laptop USB.
Arduino Uno R3	1	Controls LCD and servo based on ESP32 data, shows weather status, powered by laptop USB.
Water Sensor	1	Detects rain by measuring moisture, connected to ESP32 GPIO34.
16x2 LCD (LM016L)	1	Displays "Rain Detected" or "Clear Weather" using Arduino pins 4-9, 13.

SG90 Servo Motor	1	Moves shade to 0° (rain) or 90° (clear), controlled by Arduino pin 10.
Resistor (3.9kΩ)	1	Adjusts LCD contrast, connected to V0 pin of LCD.
Connecting Wires	As needed	Links all components, including UART (ESP32 TX2 to Arduino RX0) and GND.

5- Methodology:

I. Collecting All Components:

- All required components were gathered: ESP32 Devkit V1 (1), Arduino Uno R3 (1), water sensor (1), 16x2 LCD (LM016L) (1), SG90 servo motor (1), 3.9kΩ resistor (1), and connecting wires.
- It was ensured that the ESP32 and Arduino Uno could be powered separately using laptop USB connections.

II. Designing and Assembling the Circuit:

- The water sensor was connected as follows: its analog OUT pin to ESP32 GPIO34, VCC pin to ESP32 3V3 pin (powered via USB), and GND pin to ESP32 GND pin.
- The ESP32 was connected to the Arduino for UART communication: ESP32 TX2 (GPIO17) was linked to Arduino RX0 (pin 0), and their GND pins were joined for a common ground.
- The LCD was connected to the Arduino with these connections:
  - Pin 1 (VSS) and Pin 16 (BLK) to Arduino GND.
  - Pin 2 (VDD) to Arduino 5V (from USB power).
  - Pin 3 (V0) to Arduino GND with a 3.9kΩ resistor in between for contrast.
  - Pin 4 (RS) to Arduino Digital Pin 8 (PB0).
  - Pin 5 (RW) to Arduino GND (to keep it in write mode).
  - Pin 6 (EN) to Arduino Digital Pin 9 (PB1).
  - Pin 11 (D4) to Arduino Digital Pin 4 (PD4).
  - Pin 12 (D5) to Arduino Digital Pin 5 (PD5).
  - Pin 13 (D6) to Arduino Digital Pin 6 (PD6).
  - Pin 14 (D7) to Arduino Digital Pin 7 (PD7).
  - Pin 15 (BLA) to Arduino Digital Pin 13 (PB5) for backlight control.
- The SG90 servo was connected to the Arduino:
  - Red wire (VCC) to Arduino 5V (from USB power).
  - Brown wire (GND) to Arduino GND.
  - Orange wire (Signal) to Arduino Digital Pin 10.
- Connecting wires were used to join all parts, ensuring the ESP32 and Arduino were powered separately by laptop USB ports.

III. Programming the Microcontrollers:

- Code was written for the ESP32 to read the water sensor's analog value on GPIO34, determine the status ("1" for rain if value > 200, "0" for clear if ≤ 200), and send it to the Arduino via UART (TX2 to RX0).
- Code was written for the Arduino Uno to receive the status from the ESP32, display "Rain Detected" on the LCD and set the servo to 0° if the status was "1," or display "Clear Weather" and set the servo to 90° if the status was "0"; the LCD backlight was also controlled using Pin 13 (PB5).
- The code was uploaded to the ESP32 and Arduino Uno using Arduino IDE on the laptop through USB connections.

IV. Testing the System:

- The ESP32 and Arduino Uno were powered using separate USB ports on the laptop.
- The water sensor was tested by dipping it in water to simulate rain (value > 200); the LCD displayed "Rain Detected," the servo moved to 0° to close the shade, and the backlight turned on via Pin 13.
- The sensor was dried to simulate clear weather (value ≤ 200); the LCD displayed "Clear Weather," the servo moved to 90° to open the shade, and the backlight remained on.
- The UART communication was checked to ensure the ESP32 sent "1" or "0" correctly to the Arduino, and the LCD and servo responded as expected.

V. Making Adjustments and Documenting

- The 3.9kΩ resistor on the LCD's V0 pin was adjusted if the display contrast was unclear.
- All connections were verified to be secure, and the common ground between the ESP32 and Arduino was confirmed to be working.
- The circuit diagram, test results, and observations were documented, noting the servo angles (0° for rain, 90° for clear) and LCD messages for each condition.

6-Implementation (Coding Details):

I. Hardware Connections:

- **Water Sensor**: Analog output connects to ESP32 GPIO34, power (3V) to ESP32 3V3, and GND to ESP32 GND.
- **ESP32 to Arduino**: ESP32 TX2 (GPIO17) connects to Arduino RX0 (pin 0) for UART communication, with a common GND between both.
- **Power and Ground for LCD**:
  - LCD Pin 1 (GND) and Pin 16 (BLK) to Arduino GND.
  - LCD Pin 2 (VCC) to Arduino 5V.
  - LCD Pin 5 (RW) to Arduino GND for write mode.
- **LCD Contrast (V0)**: LCD Pin 3 (V0) connects to Arduino GND with a 3.9kΩ resistor.
- **LCD Control Pins**:
  - LCD Pin 4 (RS) to Arduino Digital 8 (PB0).
  - LCD Pin 6 (EN) to Arduino Digital 9 (PB1).
- **LCD Data Pins**:

    - LCD Pin 11 (D4) to Arduino Digital 4 (PD4).
    - LCD Pin 12 (D5) to Arduino Digital 5 (PD5).
    - LCD Pin 13 (D6) to Arduino Digital 6 (PD6).
    - LCD Pin 14 (D7) to Arduino Digital 7 (PD7).
- **LCD Backlight**:
    - LCD Pin 15 (BLA) to Arduino Digital 13 (PB5) for backlight control via code.
    - LCD Pin 16 (BLK) to Arduino GND (already connected).
- **SG90 Servo**:
    - Red wire (VCC) to Arduino 5V.
    - Brown wire (GND) to Arduino GND.
    - Orange wire (Signal) to Arduino Digital Pin 10.

II. ESP32 Functions:

```
void setup_wifi() {}

void callback(char* topic, byte* payload, unsigned int length) {}

void reconnect() {}

void setup() {}

void loop() {}
```

- **setup_wifi()**: Connects the ESP32 to the Wi-Fi network using the provided SSID and password, with a 10-second limit (20 attempts of 500ms) before falling back to sensor-only mode.
- **callback()**: Processes MQTT messages, extracts the payload, and sets the status (“0” or “1”) if valid.
- **reconnect()**: Attempts to reconnect to the MQTT broker if the connection is lost.
- **setup()**: Initializes serial communication, UART (Serial2) to Arduino, and MQTT setup.
- **loop()**: Reads the sensor, publishes data to MQTT if connected, waits 3 seconds for a response, and sends “1” (rain) or “0” (clear) to the Arduino via UART based on server data or local sensor value.

III. Arduino Code:

```
void setup() {}

void loop() {}
```

- **setup()**: Initializes serial communication at 9600 baud, sets up the LCD with 16 columns and 2 rows, displays an initial message, and attaches the servo to pin 10 with a default position of 0°.
- **loop()**: Checks for data from the ESP32, clears the LCD, and updates the display and servo: “Rain Detected” with 0° for rain, “Clear Weather” with 90° for clear, or “Invalid input” for other values, with a 2-second delay to show the message.

IV. Python Code (MQTT Server):

```
def on_connect(client, userdata, flags, rc):
def on_message(client, userdata, msg):
client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.connect(BROKER, PORT, 60)
```

client.loop_forever()

- **on_connect()**: Connects to the MQTT broker and subscribes to the sensor topic.
- **on_message()**: Reads the sensor value from the ESP32, decides the status ("1" for >200, "0" for ≤200), and publishes it to the status topic.
- **main setup**: Initializes the client, sets callbacks, connects to the broker, and runs the loop continuously.

7-Data Flow Diagram (DFD):

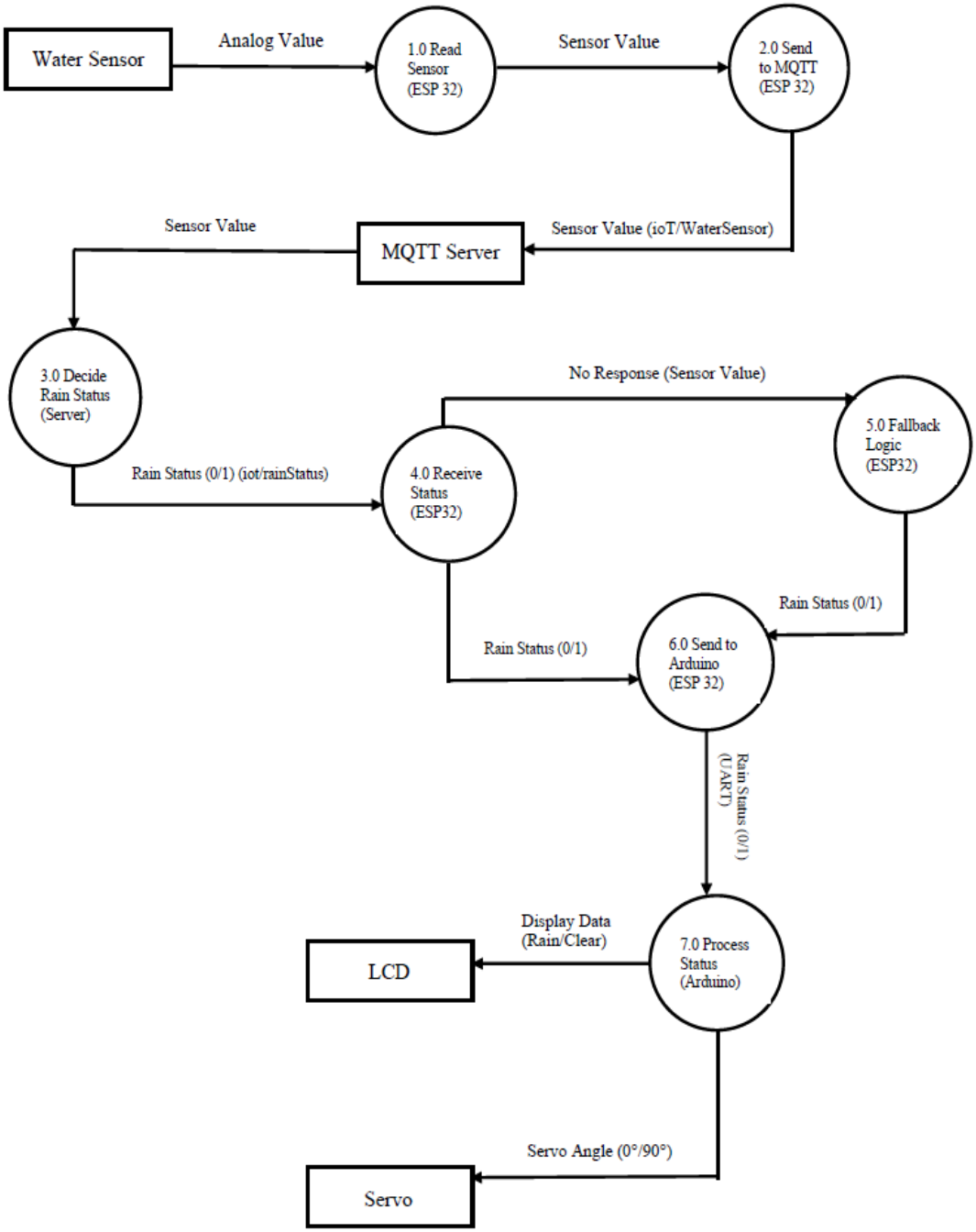

Figure 1-Data Flow Diagram

8- Flow Chart:

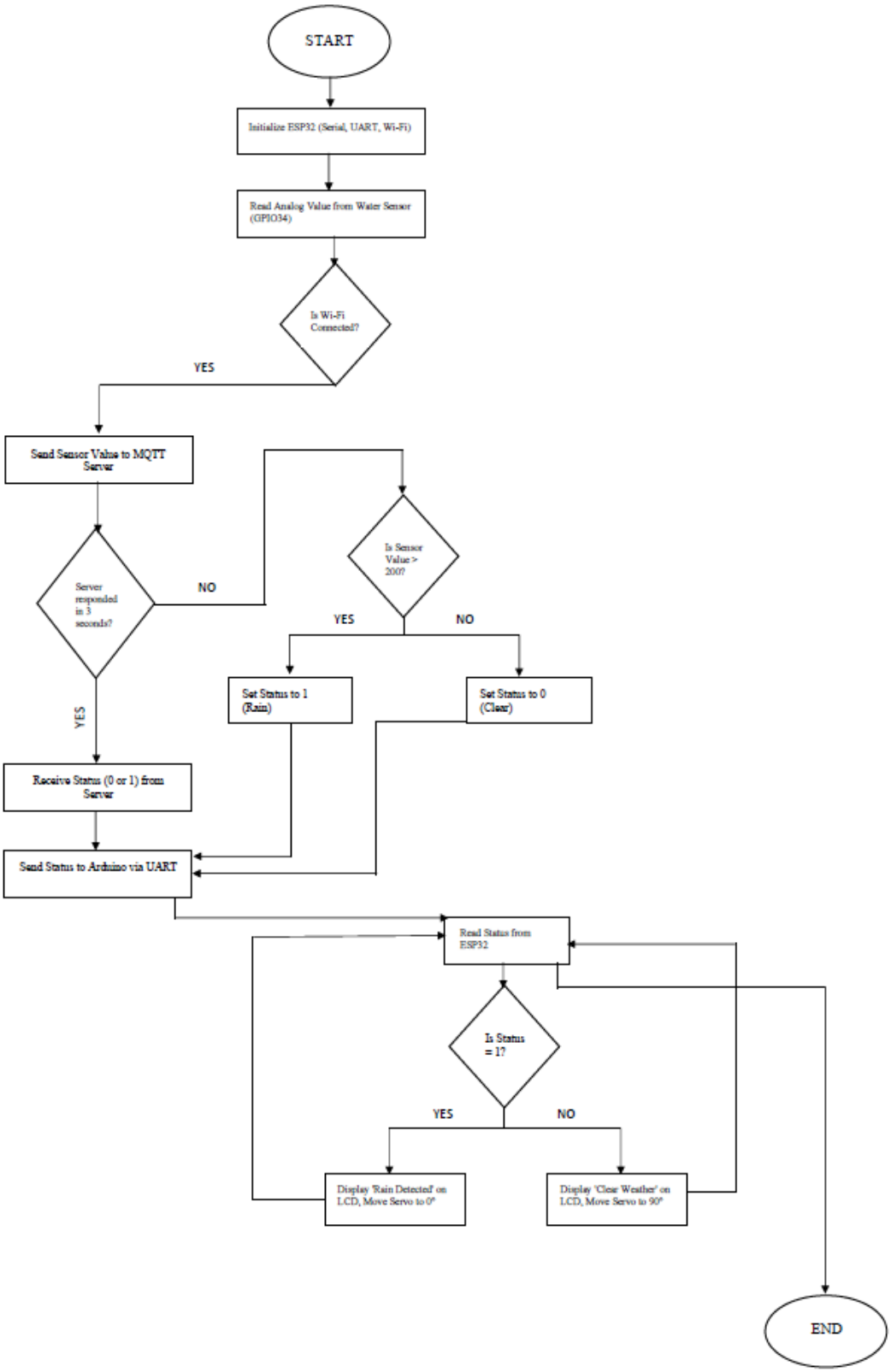

Figure 2-Flow Chart

9- Circuit Diagram:

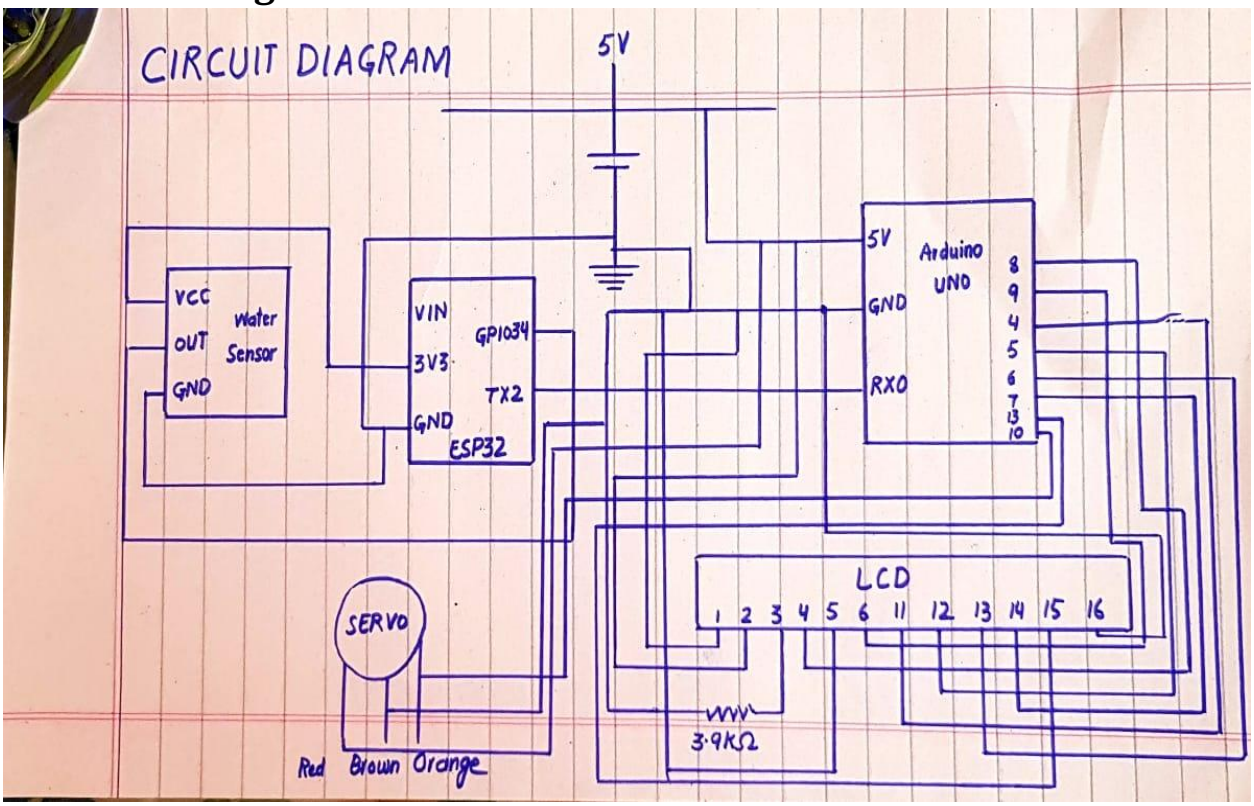

Figure 3-Circuit Diagram

10- Future Works:

- Add a buzzer to alert when rain is detected for extra notification.
- Include a mobile app to check weather status and control the shade remotely.
- Upgrade the water sensor with a more accurate model for better rain detection.
- Add a solar panel to power the system with renewable energy.
- Integrate a temperature sensor to adjust the shade based on weather conditions.

11-Project Links:

LinkedIn:

https://www.linkedin.com/posts/muhammad-talha-42a403298_iot-embeddedsystems-arduino-activity-7332324998474313728-xPY8?utm_source=share&utm_medium=member_desktop&rcm=ACoAAEf2hqIBaQ0yRSubu7bYVCSSx2_ySurRMLM

Individual Profile Links:

2023-CS-53: https://www.linkedin.com/in/ahmad-bin-rashid

2023-CS-54: https://www.linkedin.com/in/abu-tayyab-86129427b

2023-CS-62: https://www.linkedin.com/in/miansaadtahir

2023-CS-70: www.linkedin.com/in/muhammad-talha-42a403298

YouTube:
https://youtu.be/oumvFWZvYqQ?si=qy9CMbovo4g0K8F_

Git-hub:
https://github.com/ABUTAYYAB/IOT-based-AutoShade-Project

12-References:

Textbooks and Academic Books:

- "Microcontroller Programming: An Introduction" by Ted Van Sickle
- "Embedded Systems: Real-Time Interfacing to ARM Cortex-M Microcontrollers" by Jonathan W. Valvano

YouTube Videos:

- https://youtu.be/TGusjcKSNIU?si=l7JP_6K_EZdqC25a (ESP32 Basics and UART Communication)
- https://youtu.be/CvqHkXeXN3M?si=SuLVaZbKpJU_ZtBH (Arduino LCD Tutorial)
- https://youtu.be/C_pWNQ6H9EE?si=TYEdFddazeyQIWnC (Servo Motor Control with Arduino)

Websites and Online Resources:

- ESP32 Official Website. "ESP32 Getting Started Guide." Retrieved from https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/index.html
- Arduino Official Website. "Arduino Uno Pinout and Specifications." Retrieved from https://www.arduino.cc/en/Guide/ArduinoUno

Software Installation for Arduino:

The software setup was downloaded from the following link:
https://www.arduino.cc/en/software/

13-Advantages and Weaknesses:

Advantages:

- Automatically protects from rain.
- Clear LCD status display.
- Runs on easy USB power.
- Simple and affordable setup.

Weaknesses:

- **Slow Response Time:** The system checks every 5 seconds, so there's a delay in detecting sudden rain.
- **No Manual Override:** There's no way to manually control the shade if the sensor or code fails.
- **LCD Visibility Issue:** The LCD might be hard to read in bright sunlight, making it tough to see the status.