## **Home Automation**

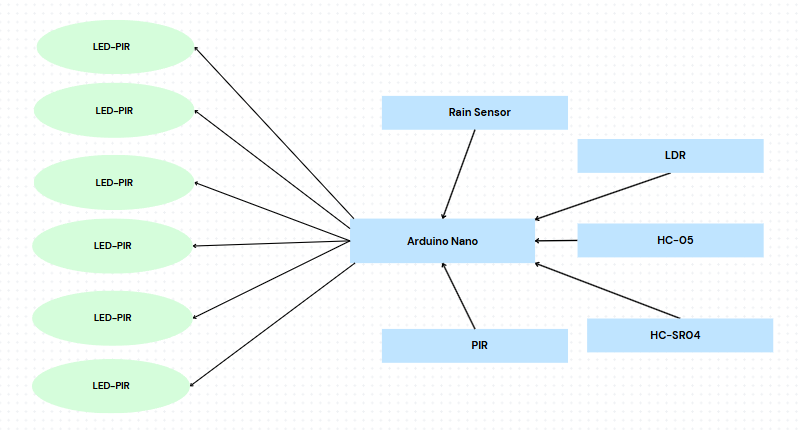
## Description:

Home automation involves using technology to intelligently control and monitor household systems such as lighting, security, and appliances. This project is a practical implementation of a basic home automation system using an Arduino microcontroller. It integrates multiple sensors—including a rain sensor, LDR (light sensor), PIR (motion sensor), and ultrasonic sensor—to automatically control devices like LEDs and a servo motor-operated window. Through serial communication, users can also manually override or trigger specific functions, such as opening or closing the window or turning devices on or off. When set to sensor mode, the system can detect rain and automatically close the window, respond to motion by turning on lights, or adjust lighting based on ambient brightness. This project demonstrates how low-cost hardware and simple programming can create an intelligent environment that improves comfort, energy efficiency, and safety in the home—paving the way toward more advanced smart home systems.

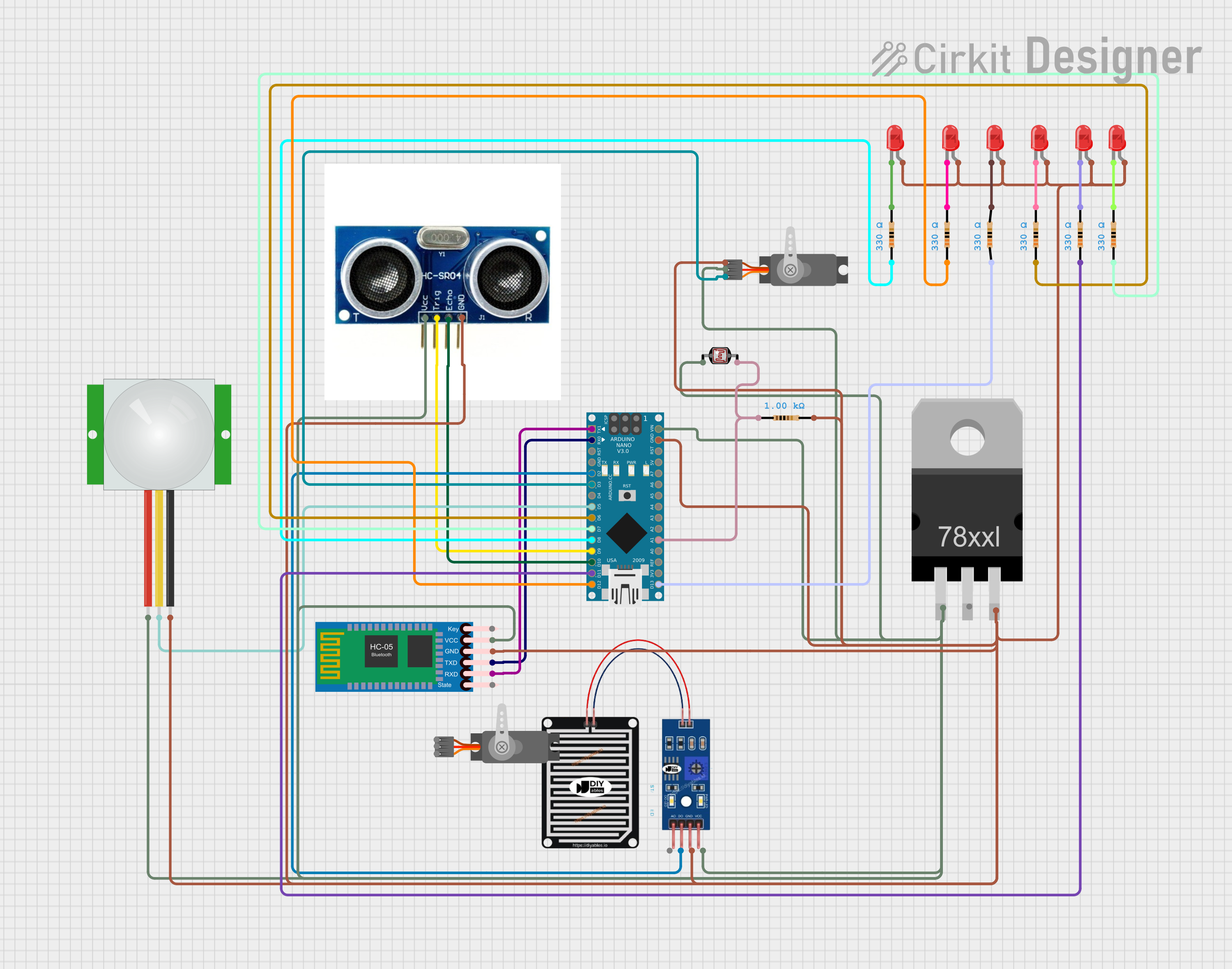
### Materials Used:

|  |  |  |
| --- | --- | --- |
| **S. No** | **Component** | **Description** |
| 1 | Arduino Nano | The microcontroller that processes all the sensors output and then controls the rest |
| 2 | Servo Motor | A motor used to open or close window or door |
| 3 | Rain sensor | A sensor that detects the presence of rain and triggers actions (servo movements and led). |
| 4 | Rocker Switch | A simple switch to turn ON or OFF the sysytem |
| 5 | Power Sorce | A power supply to provide the necessary voltage to the system. |
| 6 | HC-05 | Bluetooth module used to send messages to arduino via Bluetooth. |
| 7 | Ultrasonic Sensor | Used to check if a object is in a position ( eg check if car is in car poarch) and trigger a LED |
| 8 | LDR | Used to check the intensity of light and trigger a led if it is too low |
| 9 | PIR | To check if anything is in its proximity and trigger an LED if anything is in proximity |

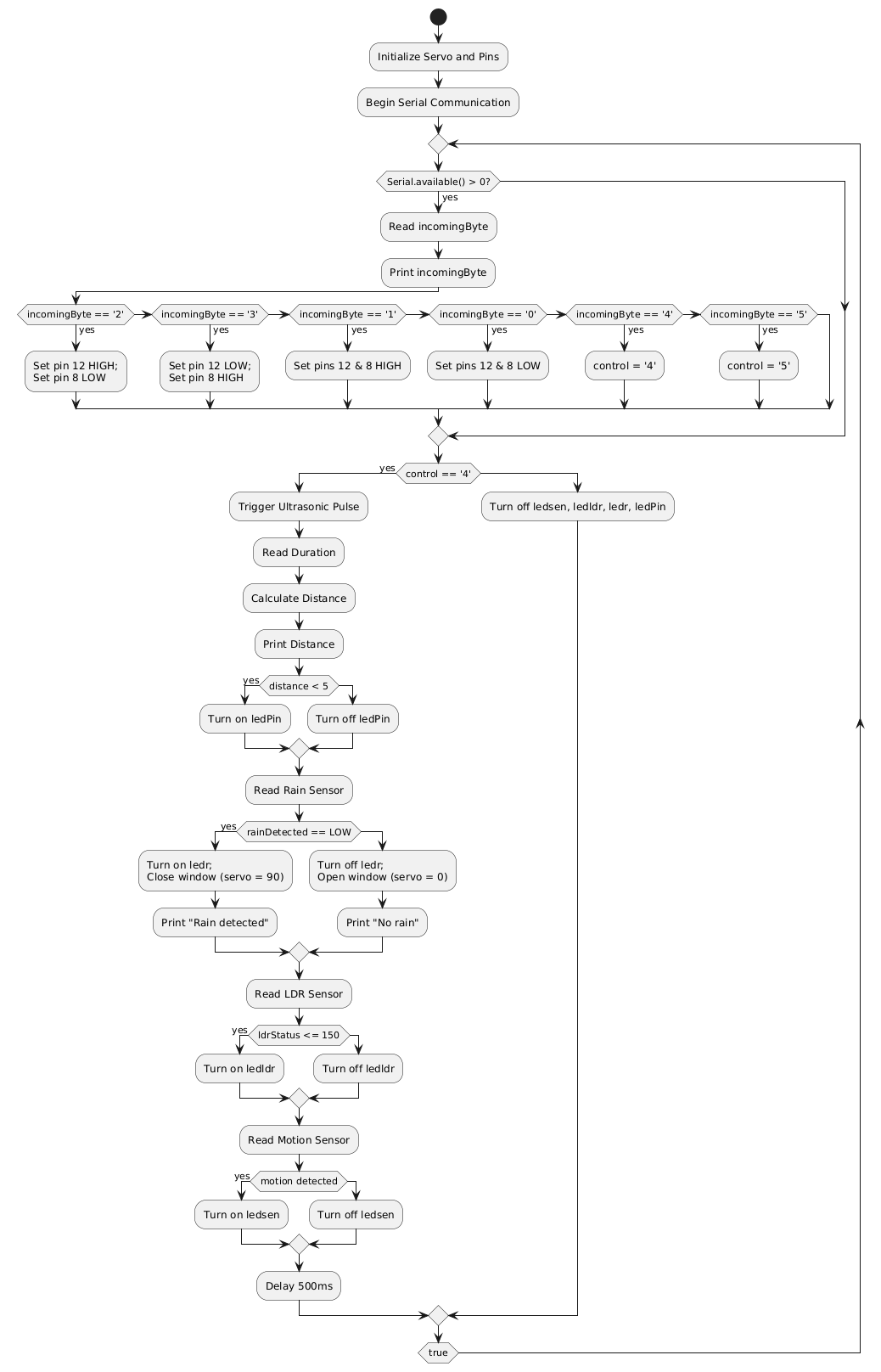
### **Block Diagram:**



### **Circuit Diagram:**



**Flow Chart:**



### **How to Explain the Project:**

### **Introduction to the Project:**

The Home Automation System using Arduino is designed to automate basic household functions such as window control, lighting, and motion-based responses using environmental sensors. The system reads data from sensors like rain, light (LDR), motion (PIR), and distance (ultrasonic), and takes appropriate action such as closing a window, turning on lights, or activating devices. It can also be manually controlled via serial commands, simulating remote or app-based home control.

### **Rain Sensor Role:**

The rain sensor detects rainfall. When rain is sensed, it sends a signal to the Arduino, which responds by closing the window using a servo motor. An LED is also turned on to indicate rain. This helps prevent rain from entering the room, especially when the user is away.

### **Arduino’s Role:**

The Arduino Uno acts as the central controller. It reads input from all sensors and sends commands to output devices like LEDs and a servo motor. It also processes serial input to allow manual device control. It switches between manual and automatic (sensor-based) modes based on commands.

### **Servo Movement for Window Control:**

When rain is detected, the Arduino activates the servo to close the window by rotating it to 90 degrees. When there is no rain, the servo rotates back to 0 degrees to open the window. This simulates smart window automation based on weather.

### **Role of LDR and PIR**

The system uses an LDR sensor to monitor ambient light. If it is too dark (e.g., at night), an LED (representing a light) is turned on. When there is sufficient light, the LED is turned off. Similarly, a PIR sensor detects motion and activates another LED to simulate automatic lighting when someone enters the room.

### **Ultrasonic Sensor for Object Detection:**

The ultrasonic sensor measures the distance of objects in front of it. If an object (e.g., person) comes closer than a certain range, an LED is turned on to indicate presence or for possible security alert functionality.

### **Serial Command / Bluetooth Command:**

Users can send commands (via serial monitor or Bluetooth) to manually turn devices on or off, or open and close the window. This allows flexible operation—automated when needed, and manual when desired.

### **Conclusion:**

This project automates several home functions based on real-time environmental data. It improves energy efficiency, comfort, and safety by reacting to light, rain, motion, and presence automatically, while still allowing manual override when needed. It demonstrates a practical and affordable smart home setup using Arduino.

### **Code:**

#include <Servo.h>

// Servo setup

Servo windowServo;

const int servoPin = 3; // Signal pin for servo

// Existing pins

const int trigPin = 9;

const int echoPin = 10;

const int ledPin = 11;

const int rainSensorPin = 2;

const int ledr = 13;

const int ledldr = 7;

const int ldrPin = A0;

const int sensor = 5;

const int ledsen = 6;

int state = LOW;

int val = 0;

long duration;

int distance;

char control;

void setup() {

windowServo.attach(servoPin); // Attach servo to pin 3

windowServo.write(90); // Start with window closed

pinMode(sensor, INPUT);

pinMode(ledsen, OUTPUT);

pinMode(rainSensorPin, INPUT);

pinMode(ledr, OUTPUT);

pinMode(ledldr, OUTPUT);

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

pinMode(ledPin, OUTPUT);

pinMode(12, OUTPUT);

pinMode(8, OUTPUT);

pinMode(ldrPin, INPUT);

Serial.begin(9600);

}

void loop() {

if (Serial.available() > 0) {

char incomingByte = Serial.read();

Serial.print("Received: ");

Serial.println(incomingByte);

if (incomingByte == '2') {

digitalWrite(12, HIGH);

digitalWrite(8, LOW);

} else if (incomingByte == '3') {

digitalWrite(12, LOW);

digitalWrite(8, HIGH);

} else if (incomingByte == '1') {

digitalWrite(12, HIGH);

digitalWrite(8, HIGH);

} else if (incomingByte == '0') {

digitalWrite(12, LOW);

digitalWrite(8, LOW);

} else if (incomingByte == '4') {

control = '4';

} else if (incomingByte == '5') {

control = '5';

}

}

if (control == '4') {

// ----- Ultrasonic Sensor -----

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

distance = duration \* 0.034 / 2;

Serial.print("Distance: ");

Serial.print(distance);

Serial.println(" cm");

if (distance < 5) {

digitalWrite(ledPin, HIGH);

} else {

digitalWrite(ledPin, LOW);

}

// ----- Rain Sensor -----

int rainDetected = digitalRead(rainSensorPin);

if (rainDetected == LOW) {

// Raining → Close window

digitalWrite(ledr, HIGH);

windowServo.write(90); // Close position

Serial.println("Rain detected! Window closed.");

} else {

// Not raining → Open window

digitalWrite(ledr, LOW);

windowServo.write(0); // Open position

Serial.println("No rain. Window open.");

}

// ----- LDR Sensor -----

int ldrStatus = analogRead(ldrPin);

Serial.println(ldrStatus);

if (ldrStatus <= 150) {

digitalWrite(ledldr, HIGH);

} else {

digitalWrite(ledldr, LOW);

}

// ----- Motion Sensor -----

val = digitalRead(sensor);

if (val == HIGH) {

digitalWrite(ledsen, HIGH);

} else {

digitalWrite(ledsen, LOW);

}

delay(500);

} else {

// Not in control mode → turn off outputs

digitalWrite(ledsen, LOW);

digitalWrite(ledldr, LOW);

digitalWrite(ledr, LOW);

digitalWrite(ledPin, LOW);

// You can keep the servo in last position or set to a default

}

}

### **Future Scope:**

### **1. Automatic Mode with Sensor Fusion**

Add a fully **autonomous mode** where the system uses a combination of sensors (light, rain, distance, motion) to make decisions **without serial input**.  
 For example:

* Close window if it rains **or** it’s nighttime.
* Open window if no rain and it's daytime.

### **2. Add Temperature and Humidity Sensor**

Use **DHT11/DHT22** or **BME280** to monitor indoor conditions:

* Control fan or window based on temperature.
* Display values on serial or OLED.

### **3. IoT Capabilities (ESP32/ESP8266)**

Switch from standard Arduino to **ESP32** (which you’re already using in another project):

* **Send data to cloud** (e.g., Blynk, Firebase, or Thingspeak).
* Control window/LEDs via smartphone app or web dashboard.
* Send alerts (rain, motion) via email/notification.

### **4. Add OLED or LCD Display**

Show live readings directly on the device:

* Distance
* Light intensity
* Rain status
* Motion detected

**5. Low Power and Power Supply Monitoring**

* Add deep sleep mode (for ESP32).
* Add a battery and monitor voltage.
* Alert on low power.