Smart Clothes Drying System

Introduction

This project is based on the **ATmega328P Xplained Mini** microcontroller and transforms a conventional clothes drying stand into a smart system, capable of responding automatically to outdoor weather changes.

The idea originated from smart home awnings and automated pergola covers, adapting their weather-responsive behavior to a clothes drying stand.

The main goal of the project is to automate the clothes drying process, save time and make it more efficient and reliable, while still giving the user full control when needed.

This system is useful for anyone who dries clothes outdoors, especially people who are away from home during the day and cannot monitor weather changes.

General Description

The system is fully standalone and operates independently, requiring only a 5V power supply via a MicroUSB adapter.

The system starts in **Automatic Mode** by default. Once powered, it begins continuously reading data from the rain sensor and the LDR. The behavior of the system adapts based on the following scenarios:

- Optimal Conditions (Sunny and Dry): if there is enough light and no rain is detected, the microcontroller commands the servomotor to keep the drying stand in the extended position. The green LED turns on to signal normal operation. At the same time, the I2C LCD shows a message indicating the current favorable weather and the extended position of the rack.
- Unfavorable Conditions (Rain or Night): if the rain sensor detects precipitation, or if the LDR registers a drop in light levels below a specific threshold (indicating nighttime), the system reacts immediately. The servomotor automatically retracts the drying stand to protect the clothes. The red LED turns on to signal bad weather, and the LCD screen updates to reflect the status and the retracted position.
- **Manual mode**: at any point, the user can briefly press the push-button to switch the system into Manual Mode. In this mode, the yellow LED turns on to indicate manual control, and the user can long press the button to toggle the extension or retraction of the stand. Pressing the button again returns the system to the standard Automatic Mode.

The system logic is built around a state machine with two main states: **Automatic Mode** and

Manual Mode. These are the main modules:

- ATmega328P Xplained Mini - main controller, reads sensor data and controls all outputs
- Rain sensor — detects precipitation and provides analog readings via ADC
- LDR Light sensor — measures ambient light level and transmits analog data via ADC
- Push button — triggers mode switching or stand toggling via external interrupt
- Servomotor — controls the physical position of the drying stand via PWM signal
- LCD 16×2 Display — shows system status and weather conditions via I2C communication
- Status LEDs — provide visual feedback: green (normal), red (bad weather), yellow (manual mode)
- State machine — core software logic that processes sensor input and decides system behavior

Hardware Design

Component	Quantity	Role
ATmega328P XMini	1	Main controller
Rain sensor module	1	Detects precipitation
LDR Light sensor	1	Measures ambient light level
LCD 1602 16×2 with I2C	1	Displays system status
Servomotor SG90 180°	1	Controls drying stand position
LED 5mm Green	1	Signals normal conditions
LED 5mm Red	1	Signals bad weather
LED 5mm Yellow	1	Signals manual mode
Resistor 220Ω 0.25W	3	Current limiting for LEDs
Push button 6x6x5	1	Mode switching and stand toggling
Breadboard 830 points	1	Circuit prototyping
Jumper wires	as needed	Electrical connections
5V 2A MicroUSB Power Supply	1	Powers the system
MicroUSB breakout board	1	MicroUSB power input connector
Capacitor 1000uF 50V	1	Power supply filtering for servomotor

Electric Diagram

The ATmega328P Xplained Mini board was not available in the EasyEDA library, so an Arduino Nano symbol was used as a pin-compatible schematic representation. The following pin mapping was applied:

- D3 = PD3 → Push Button
- D4 = PD4 → Green LED
- D5 = PD5 → Red LED
- D6 = PD6 → Yellow LED
- D9 = PB1 → Servomotor PWM
- A0 = PC0 → Rain Sensor AO
- A1 = PC1 → LDR Light Sensor AO

- A4 = PC4 → LCD SDA
- A5 = PC5 → LCD SCL

Pin Mapping

Pin ATmega	Connected to	Function
PD3 (INT1)	Push Button	Mode switching via external interrupt
PD4 (GPIO)	Green LED	Signals normal conditions
PD5 (GPIO)	Red LED	Signals bad weather
PD6 (GPIO)	Yellow LED	Signals manual mode
PB1 (PWM)	Servomotor SG90 Signal	Controls drying stand position
PC0 (ADC0)	Rain Sensor AO	Reads analog rain level
PC1 (ADC1)	LDR Light Sensor AO	Reads analog ambient light level
PC4 (SDA)	LCD 16×2 I2C SDA	I2C data line for LCD
PC5 (SCL)	LCD 16×2 I2C SCL	I2C clock line for LCD
External 5V supply / VBUS	Servo VCC, LCD VCC, Rain Sensor VCC, LDR Sensor VCC	Provides 5V power for external modules
GND	All components GND	Common ground

I connected the **LEDs** to pins PD4, PD5, and PD6 because I only needed basic digital output pins for simple On/Off states. Each LED is connected through a 220Ω resistor in order to limit the current.

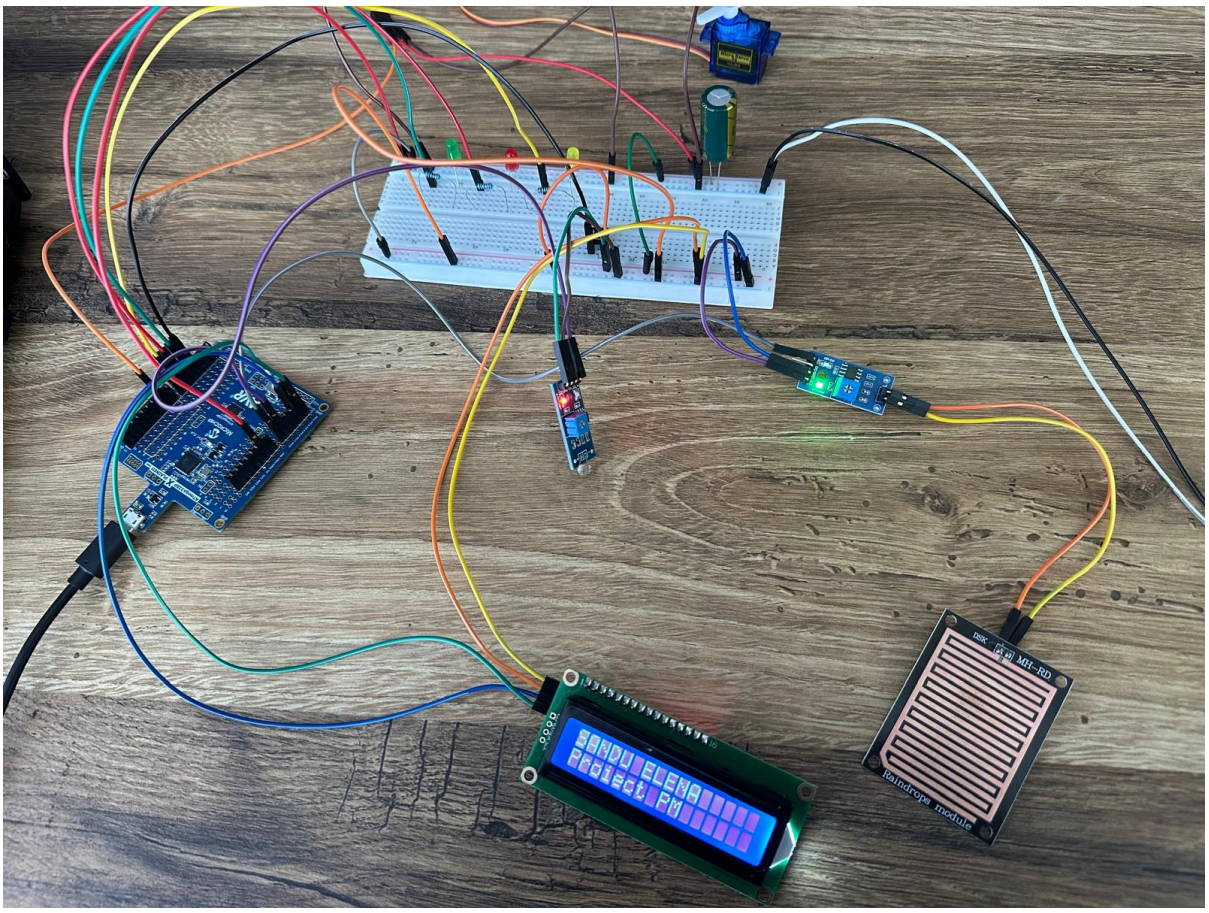
The push button is connected to PD3 because this pin also supports the INT1 external interrupt function. This makes it suitable for detecting when the button is pressed and switching between automatic and manual mode. In the current implementation, the button uses the internal pull-up resistor, so no external resistor is required.

The servomotor is connected to PB1 because this pin corresponds to OC1A and supports PWM signal generation. This is useful for controlling the position of the SG90 servomotor. I also added a 1000µF electrolytic capacitor in parallel with the servomotor power supply. It helps stabilize the 5V line when the servo moves and reduces voltage drops or noise that could affect the rest of the circuit.

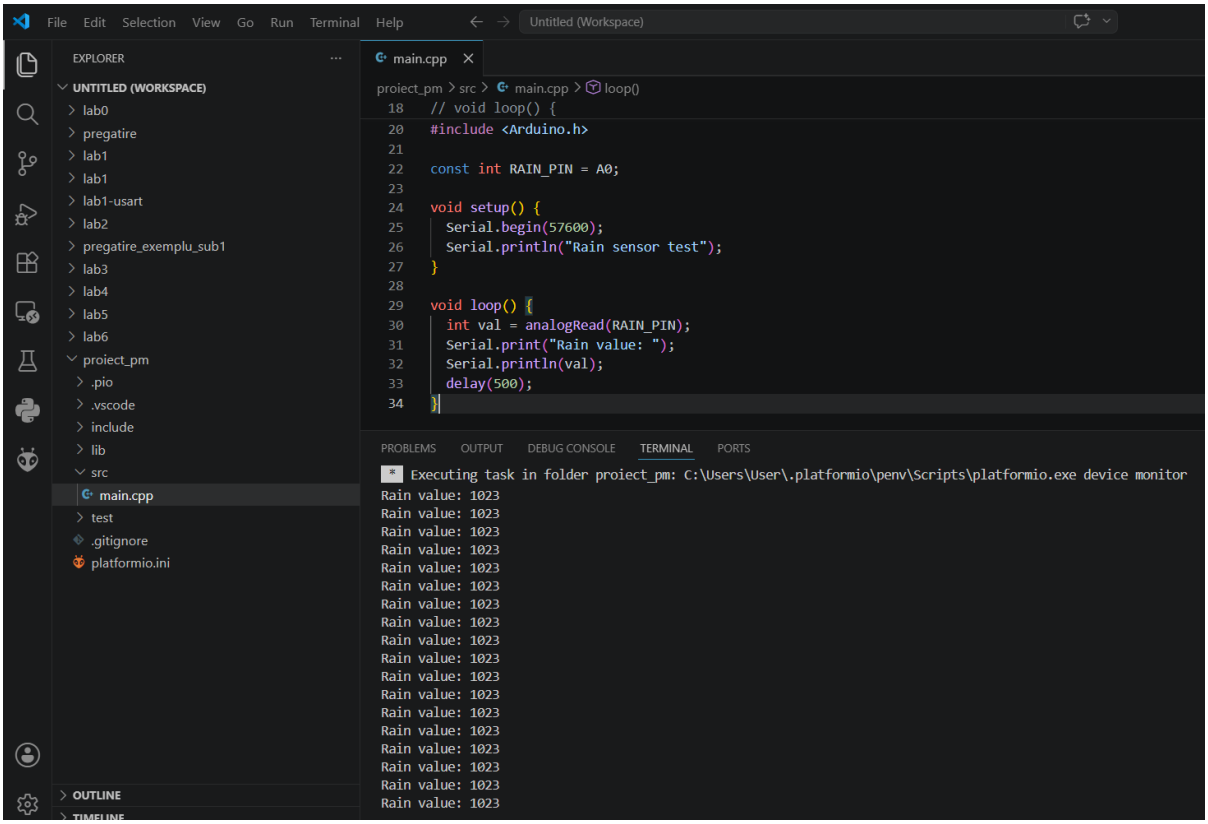
The rain sensor is connected to PC0 / ADC0 and the **LDR light sensor** is connected to PC1 / ADC1 because both sensors provide analog output values. Using ADC pins lets the software read the actual sensor level, so I can choose my own thresholds for rain and light detection.

The LCD 16×2 display uses the I2C interface, so it is connected to the dedicated I2C pins of the ATmega328P: PC4 / SDA for the data line and PC5 / SCL for the clock line.

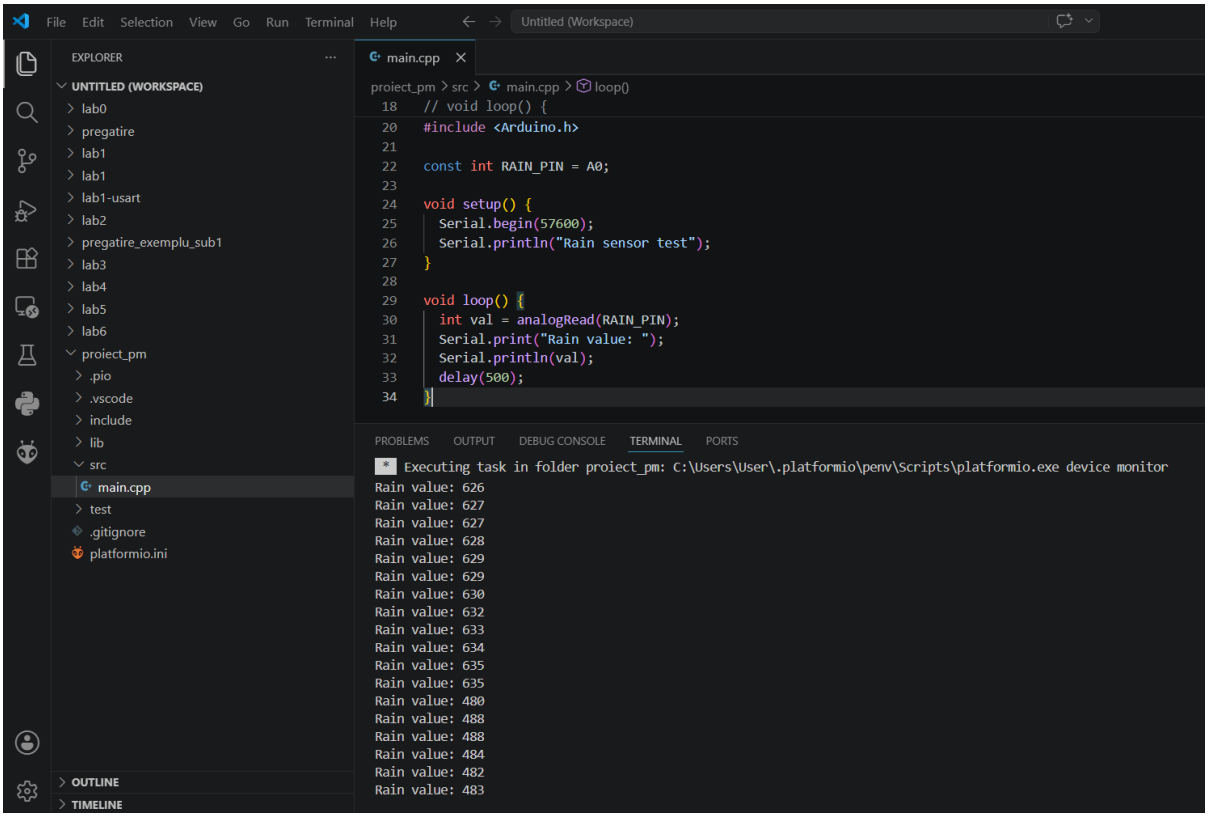
The external **5V supply** is used for the servo and the external modules. The GND of the external supply is connected to the **GND** of the ATmega board so that all signals have the same reference voltage.



As we can see in the picture, the LCD display is working, as it successfully shows a test message on the screen. This confirms that the display is powered correctly and that the I2C communication through SDA and SCL is functional. The rain sensor and LDR sensor modules are also active, which can be seen from their visible indicator LEDs. Their analog output values were tested separately using the Serial Monitor.



We can see the rain sensor in a dry state, where the analog value remains 1023. This indicates that the sensor output is at the maximum ADC value when no water is detected.



After adding water to the sensor surface, the values shown in the Serial Monitor significantly decreased. This confirms that the sensor reacts to the presence of water and that the analog output can be used to detect rain. Based on these measurements, a threshold can be chosen in software to distinguish between dry and wet conditions.

Software Design

Libraries Used

The software implementation for this project uses 4 libraries:

- **Arduino.h** - used as the base library of the project. It provides the standard functions needed for pin configuration, digital output, analog input, timing and interrupts. In this project, it is used for functions such as pinMode(), digitalWrite(), analogRead(), millis(), delay() and attachInterrupt().
- **Wire.h** - used for I2C communication. This is necessary because the LCD display communicates with the microcontroller through the SDA and SCL lines. Using I2C reduces the number of pins needed for the display.
- **LiquidCrystal_I2C.h** - used to control the 16×2 LCD display. It simplifies the process of initializing the display, turning on the backlight, positioning the cursor and printing messages on the screen. The LCD is used to show the current mode, weather status and rack position.
- **Servo.h** - used to control the SG90 servomotor. It allows the program to send position commands in degrees using the write() function. This is used to move the drying stand between the extended and retracted positions.

Novelty Element

The element of novelty in this project comes from the practical application and transposition of high-end smart home technologies into an accessible, everyday household utility. While automated systems in general are not new, this specific combination and implementation brings several notable elements:

- **Dual-mode control with real-time switching**: The system operates in two distinct modes — Automatic and Manual — that can be switched at any time via a single button. Most similar projects implement only one mode or require a separate switch for each function. Here, a single button handles two distinct actions through interrupt-driven logic: a short press switches between Auto and Manual mode, while a long press toggles the rack position when in Manual mode, all without blocking the main program flow.
- **Multi-sensor decision logic**: The retraction decision is not based on a single sensor but on the combined input of two independent sensors — a rain sensor and an LDR. The stand retracts if it rains or if it gets dark, meaning it protects the laundry both from bad weather and from overnight dew, which is an often overlooked real-world problem.
- **Real-time feedback**: The system provides simultaneous feedback through two channels — a tri-color LED indicator and a 16×2 LCD display — giving the user a clear picture of the current state at all times. The LED offers an instant at-a-glance status, while the LCD provides detailed information about the current operating mode and rack position.

Labs Used

- **GPIO**: The three status LEDs (green, red and yellow) are controlled through digital output pins using pinMode() and digitalWrite(). This functionality provides quick visual feedback for normal conditions, bad weather and Manual Mode.
- **Interrupts**: The push button is connected to pin D3 / PD3 and handled using an external interrupt with the CHANGE trigger mode. This allows the software to detect both press and release events. The interrupt records the button state and timing information, while the main loop uses this information to distinguish between short and long presses without constantly polling the button.
- **PWM**: The SG90 servomotor is controlled from pin D9 / PB1. The Servo.h library generates the required servo control signal, and the rack position is set by writing an angle value. This allows the system to move the drying stand between the extended and retracted positions.
- **ADC**:The ADC functionality is used on pins PC0 / A0 and PC1 / A1 to read the rain sensor and the LDR light sensor. The ADC converts the analog voltage into a 10-bit digital value between 0 and 1023. These values are compared with predefined thresholds, RAIN_THRESHOLD and LDR_THRESHOLD, so the software can decide whether the weather conditions are favorable or not.
- **I2C**: The 16×2 LCD display communicates with the microcontroller through the I2C protocol using the SDA and SCL pins. This is handled by the Wire.h and LiquidCrystal_I2C libraries. Using I2C reduces the number of required LCD connections from several parallel data lines to only two communication lines, leaving more pins available for sensors, LEDs and the servomotor.

Project Structure

The project is structured around a single continuous loop with four clearly separated sections:

- **Button processing** — reads and interprets user input via flags set by the ISR
- **Sensor reading** - reads the sensors and compares the values with the thresholds
- **LED indicators and servo positioning** - applies the current state to all physical outputs, setting the servo position and turning the appropriate LED on or off
- **LCD update** — changes the display only when the system state has changed

Above the loop() function, I have the **buttonISR()** interrupt routine. This routine is called every time the button changes state. A debounce check is implemented using millis(), so changes that happen too quickly are ignored. When the button is pressed, the program saves the press time and marks the button as currently pressed. When the button is released, the program calculates how long the button was held. If the press duration is between SHORT_MIN_MS and LONG_PRESS_MS, it is treated as a short press.

Interaction Between Functionalities

The system operates as a finite state machine, with two main operating modes dictating how information flows between components:

- **In Automatic Mode**: it continuouslly reads the analog values from the rain sensor (RAIN_PIN) and the light sensor (LDR_PIN). If both values are within normal parameters (sunny weather), the system

keeps the isExtended variable set to true, keeps the Green LED on, and holds the servomotor at 100 degrees. If it detects rain (value < 500) or darkness (value > 700), the logic overrides the variable to isExtended = false, turns on the Red LED, updates the LCD text (“Bad_W”), and commands the servomotor to retract to 0 degrees.

- **In Manual Mode**: Interaction with environmental sensors is suspended. Full control is taken over by the button's interrupt system. When the user short-presses the button, the isAutoMode variable toggles, switching the system into Manual mode (indicated by the Yellow LED). A long press (> 1000ms) in this mode is instantly detected in the main loop, inverting the physical state of the stand (isExtended = !isExtended) and commanding the servomotor, giving the user total control regardless of weather conditions.
- **LCD Display**: To prevent screen flickering, the project implements LCD state-tracking utilizing the currentDisplayState and lastDisplayState variables. The display is cleared (lcd.clear()) and updated strictly upon a state transition—meaning the system only rewrites the screen when comparing these two variables reveals that the combination of the current operating mode (Auto/Manual) and the stand's physical position (Extended/Retracted) has actually changed.

Validation

The project was validated step by step before full integration. The LEDs were tested first by manually calling digitalWrite() in setup() to confirm correct wiring and pin mapping, the button was tested using short and long press detection, the servomotor was tested with several angle values, and the LCD was tested by displaying a static message. The rain sensor and LDR values were were printed to Serial Monitor to verify the raw readings and determine appropriate threshold values for rain and darkness detection. After each component worked separately, they were integrated into the final program and tested together.

Demo Video

https://youtu.be/nS0Sqs8Wr7k?feature=shared

Sensors Calibration

Calibration was made using the Serial Monitor to observe the raw ADC values returned by each sensor under different real-world conditions.

The **Rain sensor** was tested in two states: dry and wet (by applying water directly to the sensor surface). In dry conditions, the readings were consistently above 800. When moisture was applied, the values dropped below 300. A threshold of 500 was chosen as a safe midpoint between the two ranges, ensuring reliable detection without false triggers from minor humidity variations.

The **LDR light sensor** was tested under normal indoor lighting and in complete darkness (by covering the sensor). Under normal light conditions, readings were below 400. In darkness, values

rose above 800. A threshold of 700 was chosen to reliably detect nighttime or low-light conditions while avoiding false triggers from brief shadows or minor light fluctuations.

The two **servo positions** were also determined through physical testing. The motor was manually commanded to various angles using standServo.write() in setup() to observe the mechanical behavior of the rack. An angle of 90 degrees was chosen for the extended position, as it corresponds to the physical position where the drying rack is fully open and stable. An angle of 0 degrees was selected for the retracted position, bringing the rack to its fully closed position. These values were saved as POS_EXTENDED = 90 and POS_RETRACTED = 0.

Optimizations

Several optimizations were made to improve stability and reliability across the entire system.

The LCD is updated only when the system state changes, tracked via the lastDisplayState variable, preventing flickering and reducing unnecessary I2C communication.

The user button is managed through an external hardware interrupt rather than continuous polling in the main loop. To ensure clean signal processing, a 50ms software debounce filter was integrated directly inside the ISR to ignore mechanical noise from the button contacts. Additionally, the long-press action is executed only once per press using the longPressHandled flag, preventing the rack from toggling repeatedly while the button is held down.

The software avoids long blocking delays and uses millis() to measure time when needed. This keeps the main loop responsive, so the system can continue reading the sensors, checking the button state and updating the actuators without waiting unnecessarily. As a result, the project can react faster to weather changes and user input.

Github Repository

https://github.com/elena-sandu/Smart-Clothes-Drying-System

References

ATmega328P Xplained Mini Datasheet:
https://ww1.microchip.com/downloads/en/DeviceDoc/50002659A.pdf

Arduino Uno Datasheet: https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf

Arduino Servomotor control: https://docs.arduino.cc/learn/electronics/servo-motors

Arduino Wire Comunication: https://docs.arduino.cc/learn/communication/wire

Export to PDF