Helwan National University
Faculty of Engineering

Automated Washing Machine

Ahmed Mohammed Salah

Eslam Farag Mohammed

Mohammed Nabeel Mohammed

Mostafa Mahmoud Refaat

Mohammed Hesham Mohammed

Mohammed Essam Farouk

Mostafa Ibrahim Mostafa

Omar Amro Ahmed

Filobateer Samir

Under the supervision of Dr. Hossam

Abstract

This project presents the design and implementation of an automated washing machine controller using an ATmega32 microcontroller. The system handles water inlet and drainage, washing motor control, and user interface through an LCD and a 4x4 matrix keypad. Safety and flexibility were ensured by integrating water level sensors and relay-based switching for high-power components. Extensive testing and simulation were performed in Proteus to validate functionality before hardware implementation.

Helwan National University
Faculty of Engineering

Introduction

Washing machines are integral to modern households and automating their control has practical benefits in energy efficiency and user convenience. This project implements a microcontroller-based washing machine controller using ATmega32. The system supports multiple wash modes, safe water level management, and provides feedback to the user via an LCD and buzzer.

The aim was to create a low-cost, flexible system that mimics the operation of a commercial automatic washing machine while offering the learning experience of interfacing sensors, relays, and user I/O devices with a microcontroller.

System Overview

The washing machine controller can:

- Display status and prompts on a 16x2 LCD
- Accept mode selection from a 4x4 matrix keypad
- Control inlet and drain water pumps via relays
- Control a drum motor (simulated with DC motor or fan)
- Detect tank full and empty conditions using water level sensors
- Be paused and resumed at any time during operation
- Operate using a low-power ATmega32 microcontroller

Hardware Components

- ATmega32 microcontroller (40-pin DIP)
- 16x2 character LCD
- 4x4 matrix keypad
- 3 Relay modules (IN, COM, NO terminals)
- Two 220V water pumps (Inlet and Drain)

- 12V or 220V DC Motor
- 2 Water level sensors (digital type)
- Buzzer
- LEDs (3 for Fill, Wash, Drain)
- 10kΩ potentiometer (LCD contrast)
- Power supply: 5V for logic, 12V/220V for pumps and motor

LCD Connections

LCD Pin	Function	ATmega32 Pin	Description
1	GND	GND	Ground
2	VCC	+5V	Power Supply
3	V0	Potentiometer Center	Contrast Control
4	RS	PC0 (Pin 22)	Register Select
5	RW	GND	Write Mode
6	EN	PD3 (Pin 17)	Enable Signal
11–14	D4–D7	PC2–PC5 (Pins 24–27)	Data Bits
15–16	Backlight	220Ω to VCC and GND	Backlight Power

Relay and Pump Connections

Function	Relay Control Pin	Relay Terminal Connections
Motor Relay	PB0 (Pin 1)	COM → AC Live, NO → Motor

Inlet Relay	PB1 (Pin 2)	COM → AC Live, NO → Inlet Pump
Drain Relay	PB2 (Pin 3)	COM → AC Live, NO → Drain Pump

Relay VCC → +5V, GND → GND. IN pin connects to corresponding PORTB pin.

Water Level Sensor Connections

Sensor Type	Signal Pin	ATmega32 Pin	Function
Full Sensor	S	PD6 (Pin 34)	Tank Full Detection
Empty Sensor	S	PD7 (Pin 35)	Tank Empty Detection

Each sensor's + connects to +5V, - to GND.

Keypad

Matrix style keypad:

- Columns: PA0–PA3
- Rows: PA4–PA7

Code Overview and Function Explanations

lcd_send_nibble(uint8_t nibble)

Sends a 4-bit data nibble to the LCD, aligning bits to PC2–PC5.

lcd_cmd(uint8_t cmd)

Sends a command byte split into nibbles using lcd_send_nibble.

lcd_data(uint8_t data)

Displays a character on the LCD.

lcd_init()

Initializes the LCD in 4-bit mode and sends necessary config commands.

lcd_clear()

Clears the LCD display.

lcd_set_cursor(uint8_t row, uint8_t col)

Moves the LCD cursor to the given row and column.

lcd_print(char *str)

Prints a string to the LCD.

keypad_getkey()

Scans the keypad and returns the key character pressed.

wait_key_release()

Waits until no key is pressed to debounce input.

check_pause()

Allows user to pause/resume the washing cycle using the 'B' key.

wait_ms(uint16_t ms)

Custom delay with pause checking inside the loop.

beep(uint8_t times)

Beeps the buzzer a specified number of times.

water_full(), water_empty()

Checks the status of full and empty water level sensors.

run_cycle(uint8_t mode)

Runs the selected wash mode by performing filling, washing, and draining actions.

show_menu()

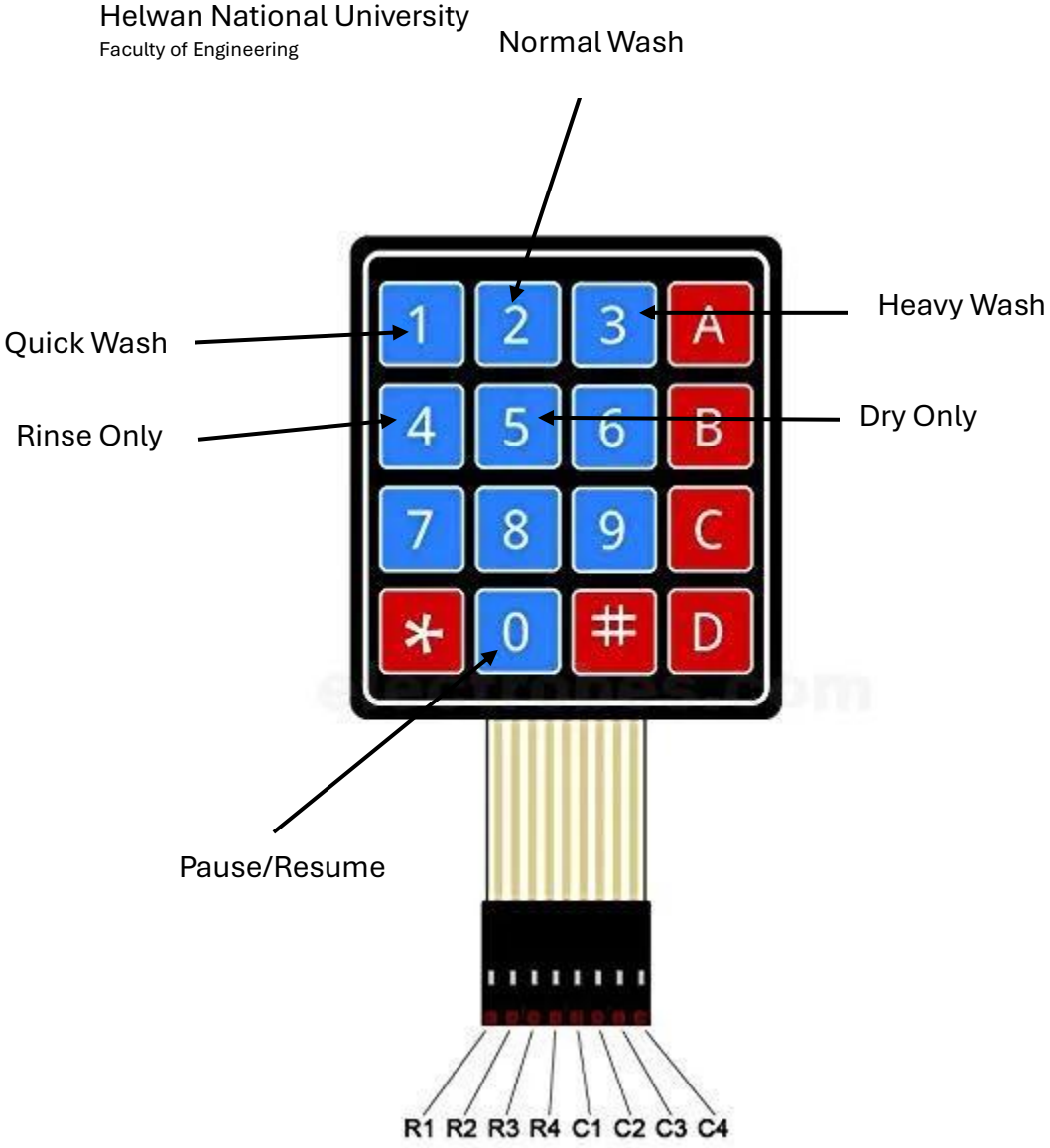
Displays available modes.

main()

Initial setup and loop to accept keypad inputs, execute selected wash mode.

Here’s what each mode does in our code:

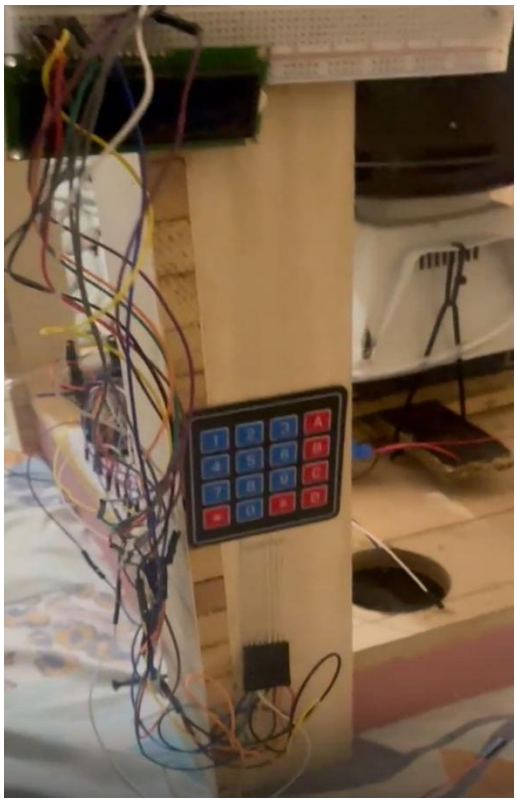
- **Mode 1 ("1") – Quick Wash:**
    - Fills the machine with water.
    - Washes for **60 seconds**.
    - Drains the water.
    - Spins briefly.
- **Mode 2 ("2") – Normal Wash:**
    - Fills the machine with water.
    - Washes for **90 seconds**.
    - Drains the water.
    - Spins briefly.
- **Mode 3 ("3") – Heavy Wash:**
    - Fills the machine with water.
    - Washes for **120 seconds**.
    - Drains the water.
    - Spins briefly.
- **Mode 4 ("4") – Rinse Only:**
    - Fills the machine with water.
    - Rinses only (same duration as wash).
    - Drains the water.
    - **No spin** cycle.
- **Mode 5 ("5") – Dry Only:**
    - Does **not fill** with water.
    - Only spins the drum for **7 seconds** to dry clothes.
- **Pause ("0"):**
    - Pauses the current cycle. Pressing '0' again resumes it.


Normal Wash
Quick Wash
Heavy Wash
Rinse Only
Dry Only
Pause/Resume
1
2
3
A
4
5
6
B
7
8
9
C
*
0
#
D
R1 R2 R3 R4 C1 C2 C3 C4

Breadboard Circuit Explanation

The system was first assembled on a breadboard for development and testing. Key points:

- ATmega32 was inserted in a 40-pin socket.
- All PORTC lines (PC0–PC5) connected to LCD (RS, D4–D7)
- PD3 was used for LCD EN
- Buzzer and LEDs were connected to PB3–PB6
- Relays were connected via IN pins to PB0–PB2
- Sensor inputs to PD6 and PD7
- A potentiometer adjusted LCD contrast between VCC and GND



Proteus Circuit Explanation

The project was simulated in Proteus before hardware implementation.

Components Used

- ATmega32 (loaded with compiled HEX)
- 16x2 LCD (connected in 4-bit mode)
- 4x4 Keypad (scanned with PA0–PA7)

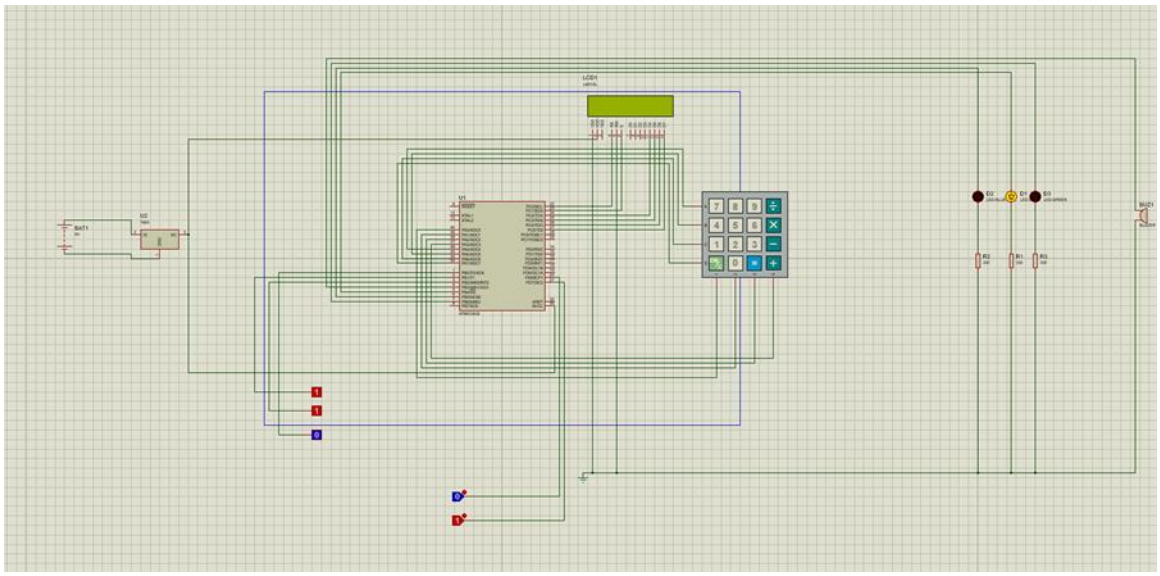
- 3 Relays (to control motor, inlet, and drain pumps)
- Switches (simulate water full and empty sensor inputs)
- LEDs and buzzer for effects
- Virtual terminal (optional for debug output)

Wiring Breakdown

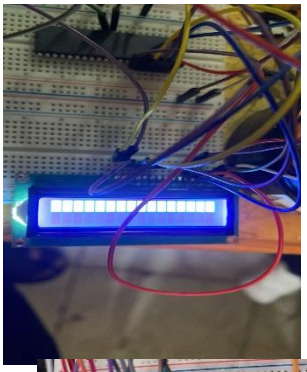
- Relays: connected to PORTB pins with pull-up resistors in Proteus
- Water sensors: simulated using logic switches to toggle high/low
- LCD: connected to PORTC and PD3 (RS and EN)
- Keypad: direct wire connections to PORTA
- Power: ATmega32 powered with 5V VCC and GND

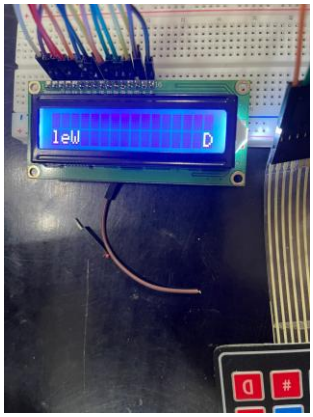
How It Works in Simulation

1. The LCD initializes and shows "WashMachine"
2. User selects mode using keypad (1–5)
3. Filling phase begins — inlet pump turns ON until the water full switch is closed
4. Washing starts — motor relay is activated and countdown timer is shown on LCD
5. Draining starts — drain pump is activated until water empty switch is toggled
6. LEDs light up during each phase
7. Buzzer beeps when cycle completes



Obstacles Met





- **LCD showing only blocks**: Fixed by adjusting contrast and switching EN pin
- **Incorrect display**: Some words were trimmed due to timing and wiring bugs; resolved by fixing delays and simplifying messages
- **Keypad unresponsive**: Solved with correct DDR/PORT setup and debounce logic
- **Motor running at wrong times**: Logic corrected to only run after water full detection
- **Relay pin confusion**: IN/NO/COM labeled in Chinese, verified with multimeter and standard mapping
- **USBASP undervoltage**: Could not power entire board; switched to external 5V power
- **Burned/faulty ATmega pins**: Pins 28 and 40 were unresponsive, reassigned EN pin to PD3
- **PCB troubleshooting**: Solder bridges and open joints caused malfunctions; reverted to breadboard for final testing

Conclusion

The washing machine controller system based on ATmega32 successfully demonstrated automated appliance control with safety and usability features. Through breadboarding, simulation, and real-world testing, the project handled sensor input, actuator control, and user interaction. Key learning areas included relay control, sensor interfacing, LCD debugging, and Proteus simulation.