BAU

BEIRUT ARAB UNIVERSITY

Faculty of Engineering

COMP325 - CPU DESIGN

Smart Home Automation and Security System Using PIC16F877A and Raspberry Pi Pico W

Prepared by:
Ali Naser - 202501429
Arwa Saad - 202502230
Omar Kilani - 202500832

Submitted to: Dr. Joseph Massoud

Abstract

This report presents the design and implementation of a smart home automation and security prototype developed for COMP325 - CPU Design. The system uses a PIC16F877A microcontroller as the main hardware controller and a Raspberry Pi Pico W as the communication and monitoring controller. The final version integrates gas detection, water detection, keypad password access, gate servo control, automatic gate closing using IR detection, a second servo actuator, buzzer alarm, fan relay control, heater indicator, LCD user interface, UART communication, a local web dashboard, Telegram alerts, and Thonny-based testing commands.

The PIC16F877A reads the sensors and keypad, controls the servos and actuators, and sends one-character status codes to the Pico W through UART. The Pico W receives these codes, updates a web dashboard, sends Telegram notifications, and allows the user to send commands back to the PIC. The final prototype demonstrates how a traditional microcontroller can be combined with an IoT-capable board to provide both local embedded control and remote monitoring.

Keywords - PIC16F877A, Raspberry Pi Pico W, smart home, UART, IoT, keypad access, gas detection, water detection, servo motor, Telegram bot, web dashboard.

I. Phase 0: Introduction, Objective, Purpose and Problem Solving

Smart home automation systems are designed to improve safety, comfort, and monitoring inside residential buildings. A smart system can automatically detect dangerous environmental conditions, control physical devices, and notify the user when an event occurs. This project implements a practical smart home model using components that are available in a microcontroller laboratory. The project focuses on combining multiple subsystems into a single working embedded system.

The main purpose of this project is to demonstrate how a CPU-based embedded controller can manage a set of sensors and actuators while communicating with a separate wireless controller. The PIC16F877A handles the real-time hardware side. The Raspberry Pi Pico W handles the computer and IoT side by providing WiFi, Telegram notifications, a local webpage, and Thonny testing commands.

The problem solved by this project is the need for a low-cost smart home prototype capable of detecting gas leakage, detecting water leakage, controlling a gate, monitoring access attempts, and alerting the user locally and remotely. Instead of using one large high-level computer to do all tasks, the design separates the system into a hardware controller and a communication controller. This makes the design easier to debug and more reliable.

Project objectives

- Design a PIC16F877A-based smart home controller for sensors and actuators.
- Read an MQ135 gas sensor through analog channel AN0 and control a fan relay according to a threshold.
- Read a water sensor through analog channel AN2 and move servo2 when water is detected.
- Use a 4x4 keypad with a 74C922 encoder to enter the access password.
- Open the gate servo when the correct password 2006 is entered.
- Use the IR sensor to close the gate after an object passes through the gate area.
- Trigger a security alarm if IR detects movement after a wrong password attempt.
- Display system states on the LCD and activate the buzzer during alerts.
- Use Raspberry Pi Pico W for WiFi, web dashboard, Telegram alerts, and UART commands.
- Provide manual Thonny commands for testing open, close, acknowledge, denied access, and Telegram test functions.

II. Final System Overview

The final system is divided into two main parts. The first part is the embedded hardware section controlled by the PIC16F877A. This section includes all sensors, actuators, the LCD, keypad, buzzer, relay, and servo motors. The second part is the computer/IoT section controlled by the Raspberry Pi Pico W. This section receives status from the PIC through UART, shows it on a webpage, and sends Telegram messages to the user.

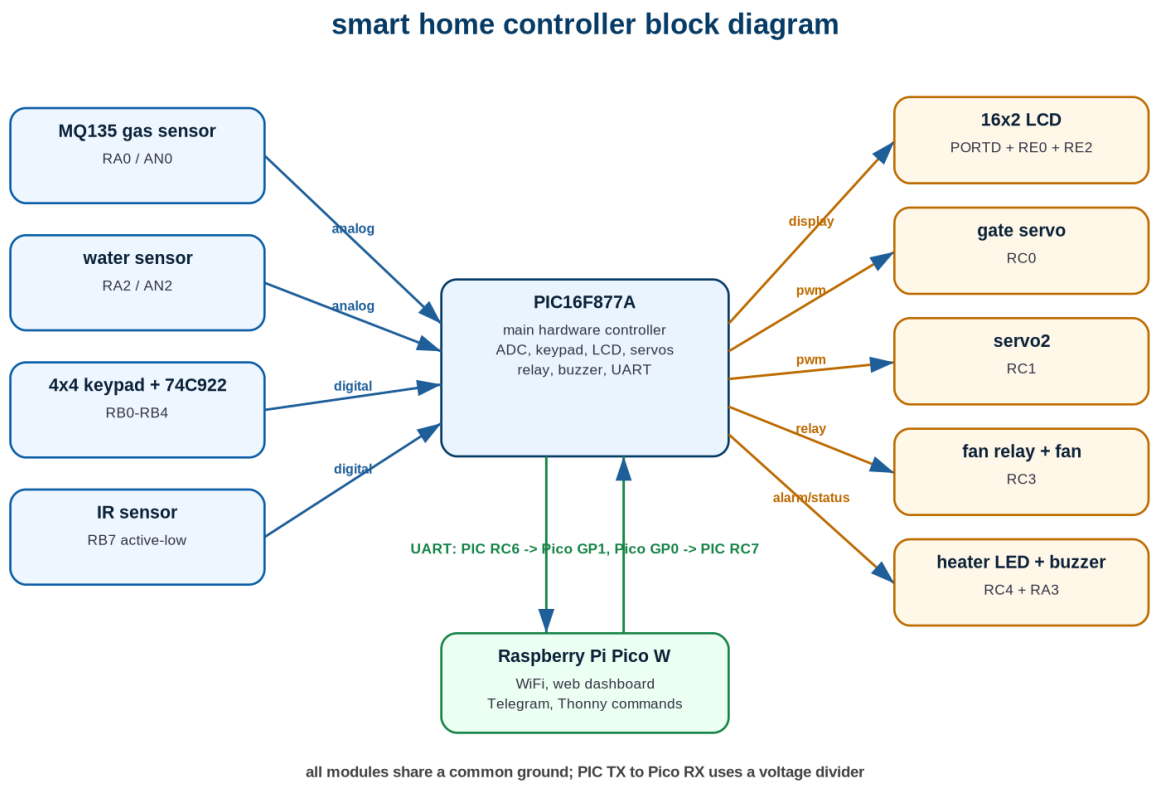

Fig. 1. Block controller diagram showing sensors, actuators, PIC16F877A and Raspberry Pi Pico W.

Explanation: This figure shows the final controller structure. The PIC16F877A is the main controller that reads the MQ135 gas sensor, water sensor, keypad encoder, and IR sensor. It drives the LCD, gate servo, servo2, fan relay, heater LED, and buzzer. The Raspberry Pi Pico W is connected through UART and is responsible for WiFi, the dashboard, Telegram alerts, and test commands.

III. Hardware Components

Component	Purpose in the Project
PIC16F877A	Main embedded controller responsible for sensors, keypad, LCD, servos, buzzer and relay control.
Raspberry Pi Pico W	IoT and computer-side controller for WiFi, web dashboard, Telegram and Thonny testing.
16x2 LCD	Displays system messages such as ENTER PASS, GAS ALERT, WATER ALERT and SECURITY.
MQ135 gas sensor	Detects gas level through analog output connected to AN0.
Water sensor	Detects water leakage through analog output connected to AN2.
4x4 keypad	Allows the user to enter password 2006 and manual commands A, B, C and D.
74C922 keypad encoder	Converts keypad press into a 4-bit code and data available signal.
IR sensor	Detects object passing through the gate and detects intrusion after a wrong password.
Gate servo	Opens and closes the gate entry.

Servo2	Acts as the second actuator, used with the water sensor condition.
Fan relay and DC fan	Fan turns on when the gas threshold is reached.
Heater LED	Represents the heater status and turns off when the fan is on.
Buzzer	Provides local alarm output during security and alert states.

IV. Pin Connections

Module / Signal	PIC16F877A Pin	Description
LCD D0-D7	RD0-RD7	LCD 8-bit data bus.
LCD Enable	RE0	LCD enable control.
LCD RS	RE2	LCD register select.
LCD RW	GND	LCD write mode only.
MQ135 AO	RA0 / AN0	Gas sensor analog input.
Water sensor AO	RA2 / AN2	Water sensor analog input.
74C922 D0-D3	RB0-RB3	Keypad encoded data.
74C922 DA	RB4	Data available, active-high.
IR sensor OUT	RB7	Active-low detection input.
Gate servo signal	RC0	PWM-like servo signal.
Servo2 signal	RC1	PWM-like servo signal.
Fan relay IN	RC3	Relay control for the fan.
Heater LED	RC4	Heater indicator output.
Buzzer	RA3	Alarm output.
PIC UART TX	RC6	Sends status to Pico GP1 through voltage divider.
PIC UART RX	RC7	Receives commands from Pico GP0.

Pico W Pin	Connected To	Description
GP0 TX	PIC RC7 RX	Pico sends commands to the PIC.
GP1 RX	PIC RC6 TX through voltage divider	Pico receives status messages from the PIC.
GND	PIC GND and external supply GND	Common reference ground.
VBUS / USB	5 V supply or USB	Pico power input.

V. Electrical Diagram

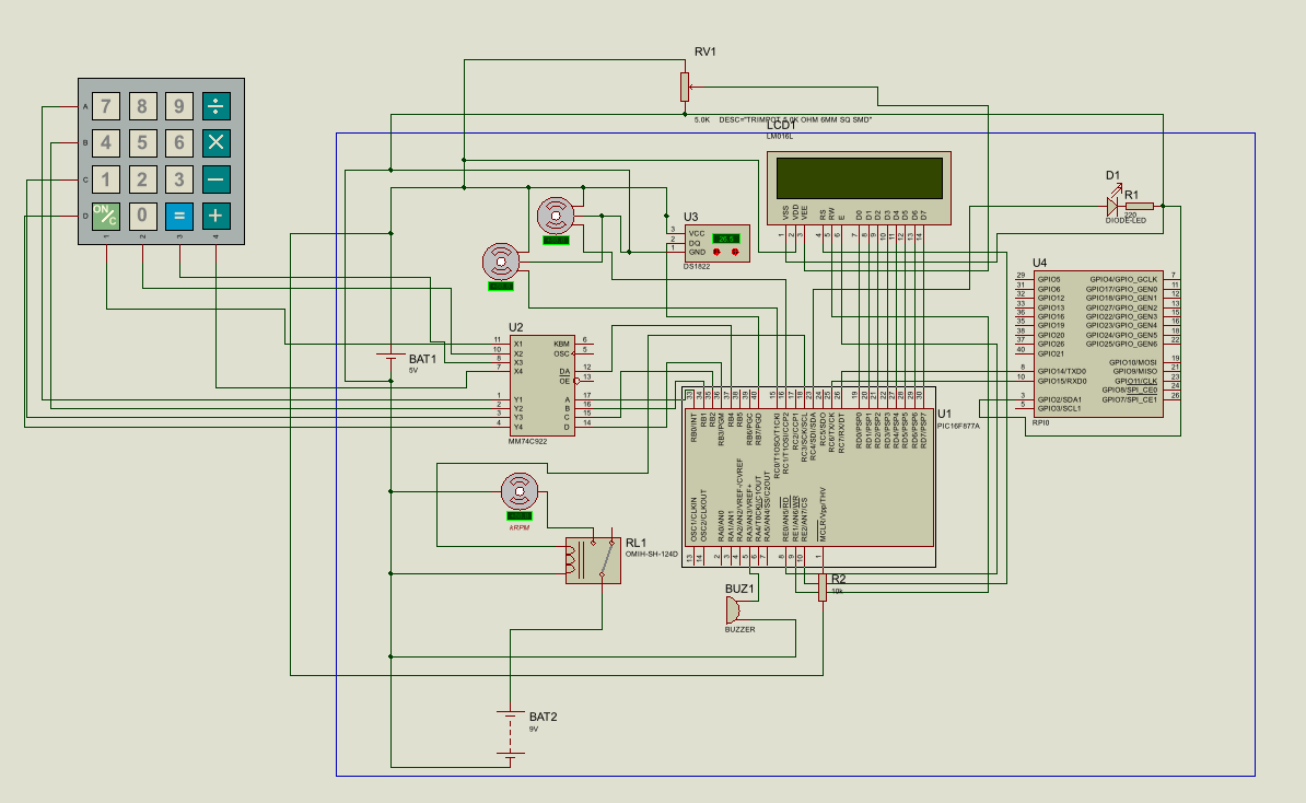

Fig. 2. Proteus simulation schematic of the smart home control system.

Explanation: This figure shows the Proteus simulation schematic used to represent the main controller connections. It includes the PIC16F877A, LCD, keypad and 74C922 encoder, Raspberry Pi Pico W interface, buzzer, relay output, servo outputs, and heater indicator. Some physical sensor modules used in the hardware prototype are not directly available in Proteus, so equivalent input connections and available simulation components were used to represent their signals. The final physical wiring still follows the connection table in this report.

VI. Operation Description

When the project is powered, the PIC initializes its ports, ADC, UART, LCD and variables. The LCD displays a ready message and then shows the password screen. The Pico W connects to WiFi, starts the web server and sends a Telegram startup message if the network allows it. After initialization, both controllers continue running in loop mode.

During normal operation, the PIC continuously reads the gas and water sensors. If gas rises above the threshold, the fan relay turns on and the heater LED turns off. If water is detected, servo2 opens. The LCD displays the most important active state using the alert priority system. Security has the highest priority, followed by water, gas and normal operation.

For access control, the user enters the password using the keypad. The password is 2006. When the password is correct, the gate servo opens and the PIC sends the code O to the Pico W. The system then waits for the IR sensor to become low, meaning the object passed through the gate area. After a delay, the gate closes automatically. If the password is wrong, the LCD displays DENIED / WATCH IR and the system monitors the IR sensor. If IR becomes low after a wrong password, the security alarm is activated.

VII. Technical Explanation of Each Step

A. Startup step

The PIC enters the CONFIGURATING label after reset. It sets TRISA, TRISB, TRISC, TRISD and TRISE to define the direction of the pins. UART is configured at 9600 baud by setting SPBRG and enabling transmission and reception through TXSTA and RCSTA. The LCD is initialized in 8-bit mode and the buzzer is tested briefly.

B. Sensor reading step

The MQ135 and water sensor are analog sensors. The PIC temporarily sets ADCON1 to analog mode, selects the required channel in ADCON0, starts the conversion using the GO bit, waits for the conversion to finish and then stores ADRESH as an 8-bit value. After each analog read, ADCON1 is restored to digital mode so that RA3 remains usable as a buzzer output.

C. Keypad step

The 74C922 encoder sends a 4-bit key value to RB0-RB3 and raises RB4 when a key is available. The PIC waits for RB4 to become high, waits for debounce, reads the value, waits again and reads it a second time. The key is accepted only if both readings are equal. This avoids unstable key detection.

D. Password and gate step

The PIC stores four entered digits and compares them with 2006. If the password is correct, the gate servo opens and AUTO_CLOSE_FLAG is set. The main loop continues running while the system waits for RB7 to become low. This prevents the controller from freezing while waiting for IR. Once RB7 becomes low, the PIC waits for a delay, closes the gate and returns to the normal screen.

E. Security step

After a wrong password, FAILED_FLAG is set. The IR sensor is then checked as a security sensor. If RB7 becomes low after failed access, SEC_FLAG is set, the buzzer starts beeping, the LCD shows SECURITY / ACK, and the Pico receives the S code. The user can press ACK from the webpage or Thonny to send K to the PIC and silence the alarm.

F. Pico W communication step

The Pico W receives UART codes from the PIC and converts them into readable messages. It updates the web dashboard, sends Telegram messages and allows the user to send commands back to the PIC. The webpage buttons send O for open gate, C for close gate, K for ACK and D for denied access test. The Pico code also accepts Thonny keyboard commands for testing.

VIII. Flowcharts

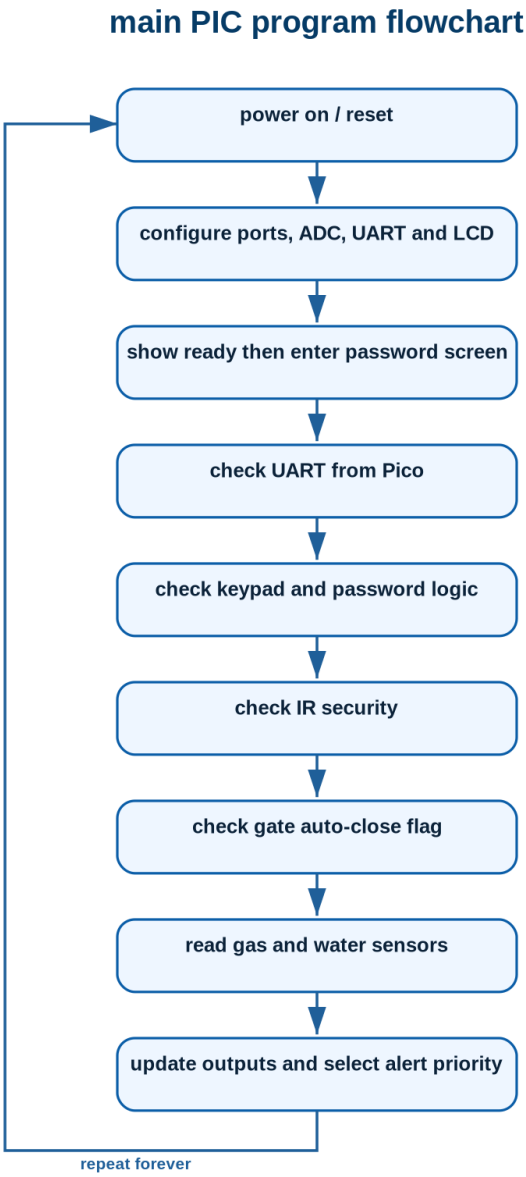

Fig. 3. Main PIC system flowchart.

Explanation: This flowchart summarizes the continuous loop of the PIC program. The PIC checks UART, keypad, IR, gate auto-close and sensors repeatedly so that the system remains responsive.

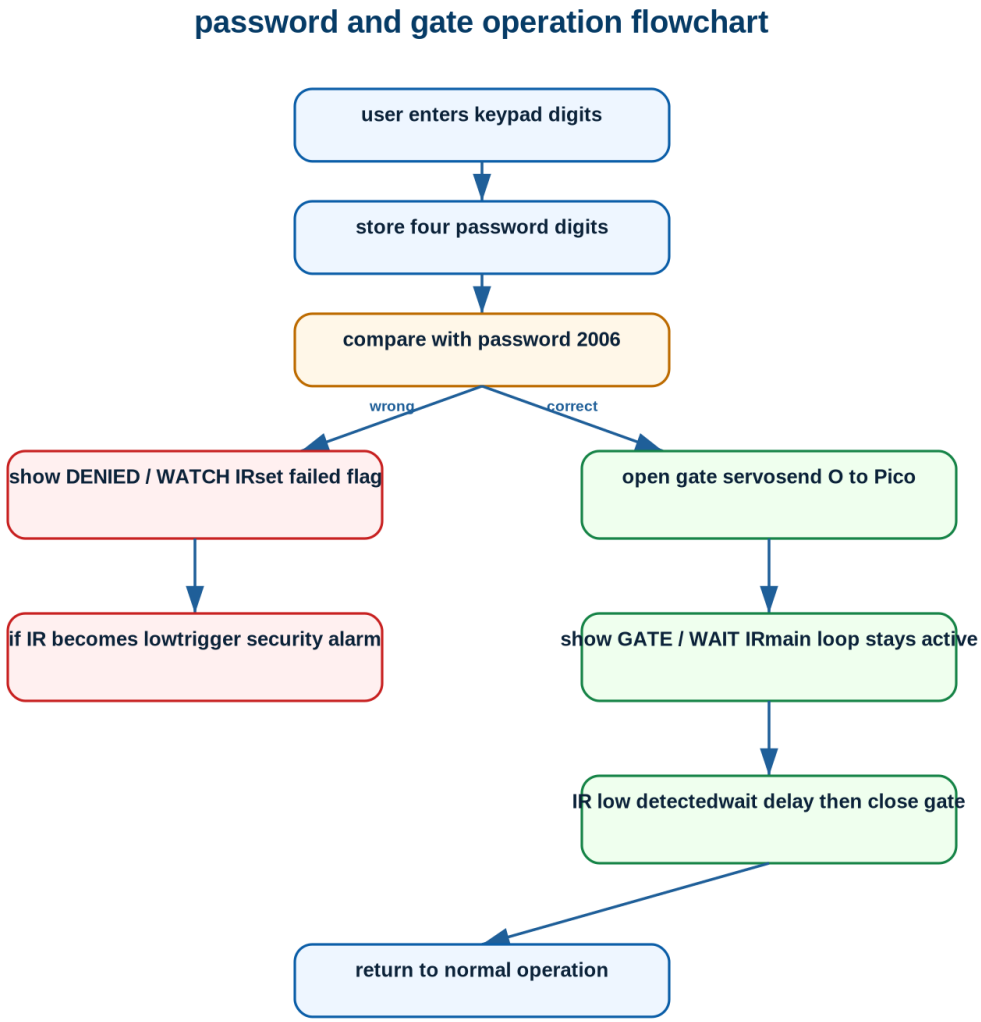

Fig. 4. Password and gate operation flowchart.

Explanation: This flowchart shows how the password system works. Correct password opens the gate and waits for the IR sensor before closing. Wrong password activates the IR security watching mode.

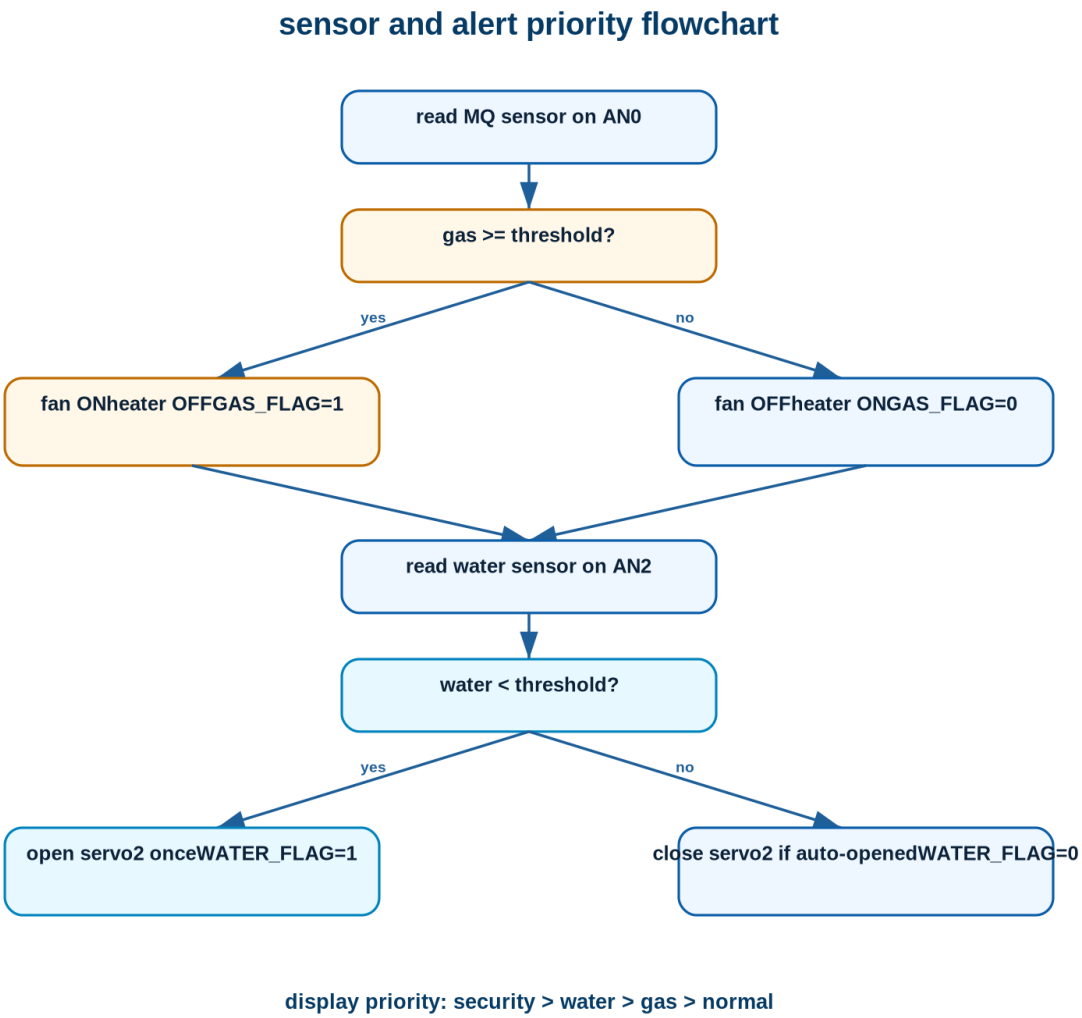

Fig. 5. Sensor and alert priority flowchart.

Explanation: This flowchart shows how gas and water sensor readings are processed and how the final LCD state is selected according to priority.

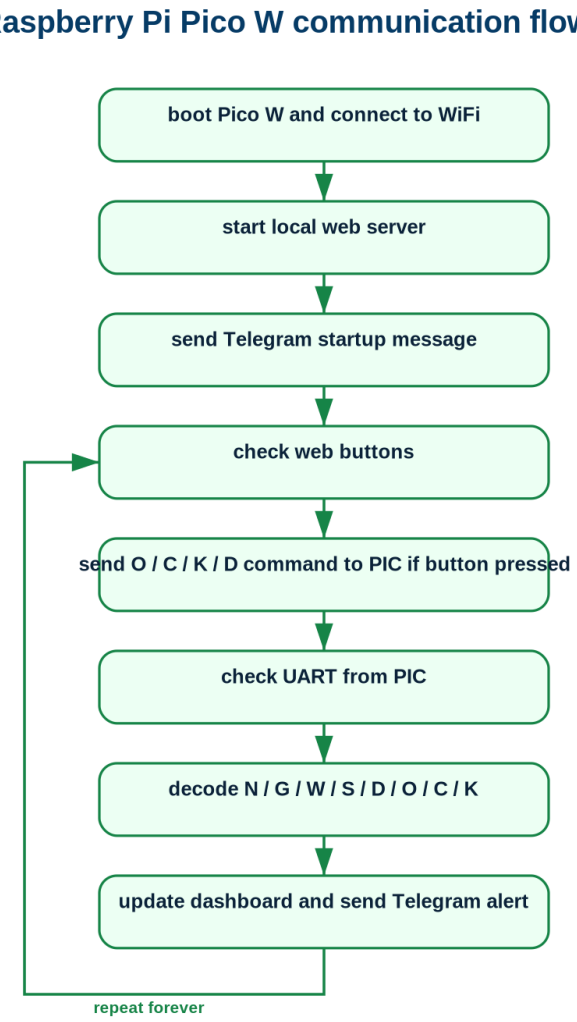

Fig. 6. Raspberry Pi Pico W communication flowchart.

Explanation: This flowchart shows the computer-side part of the project. The Pico connects to WiFi, serves the webpage, handles Telegram and communicates with the PIC through UART.

IX. Phase 3: Computer Part

The computer part of the project is implemented using the Raspberry Pi Pico W. Although the PIC16F877A controls the hardware, it does not have built-in WiFi. The Pico W solves this problem by receiving status messages from the PIC and making them available through a web dashboard and Telegram messages. This separation makes the system easier to debug and allows the PIC to remain dedicated to real-time control.

The Pico W webpage shows the current PIC status, PIC code, raw UART bytes, IP address, WiFi state and Telegram state. The webpage also includes buttons for ACK, open gate, close gate, simulate failed access and Telegram test. These buttons send UART commands back to the PIC. This turns the phone into a local control panel for the smart home prototype.

Fig. 7. Web dashboard status section.

Explanation: This screenshot shows the web dashboard running on the Pico W. It displays the current PIC status as SYSTEM NORMAL, the received code N and the raw UART bytes. The IP address 172.20.10.3 is the local address used to open the dashboard from the phone.

Fig. 8. Web dashboard control buttons.

Explanation: This screenshot shows the phone control section of the dashboard. The buttons allow the user to acknowledge alarms, open the gate, close the gate, simulate failed access and send a Telegram test message.

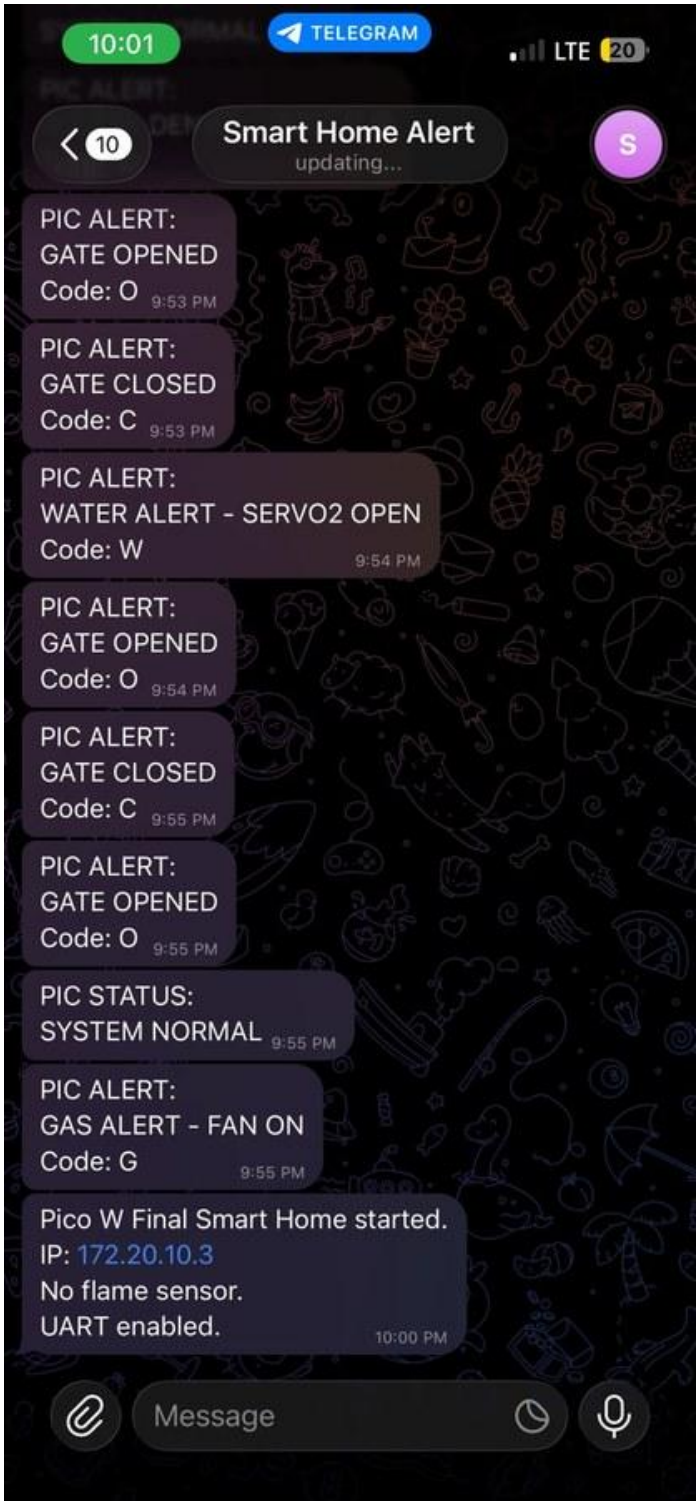

Fig. 9. Telegram alert messages.

Explanation: This screenshot shows Telegram messages received from the Pico W. The messages confirm that the Pico received PIC codes such as O, C, W, N and G and converted them into readable alerts.

X. Physical Prototype and Project Pictures

The project was built as a physical smart home model using a cardboard house structure, breadboard wiring, the PIC16F877A board area, the Pico W, sensors, servos, fan, relay and LCD. The following figures show the final prototype and the role of each visible part.

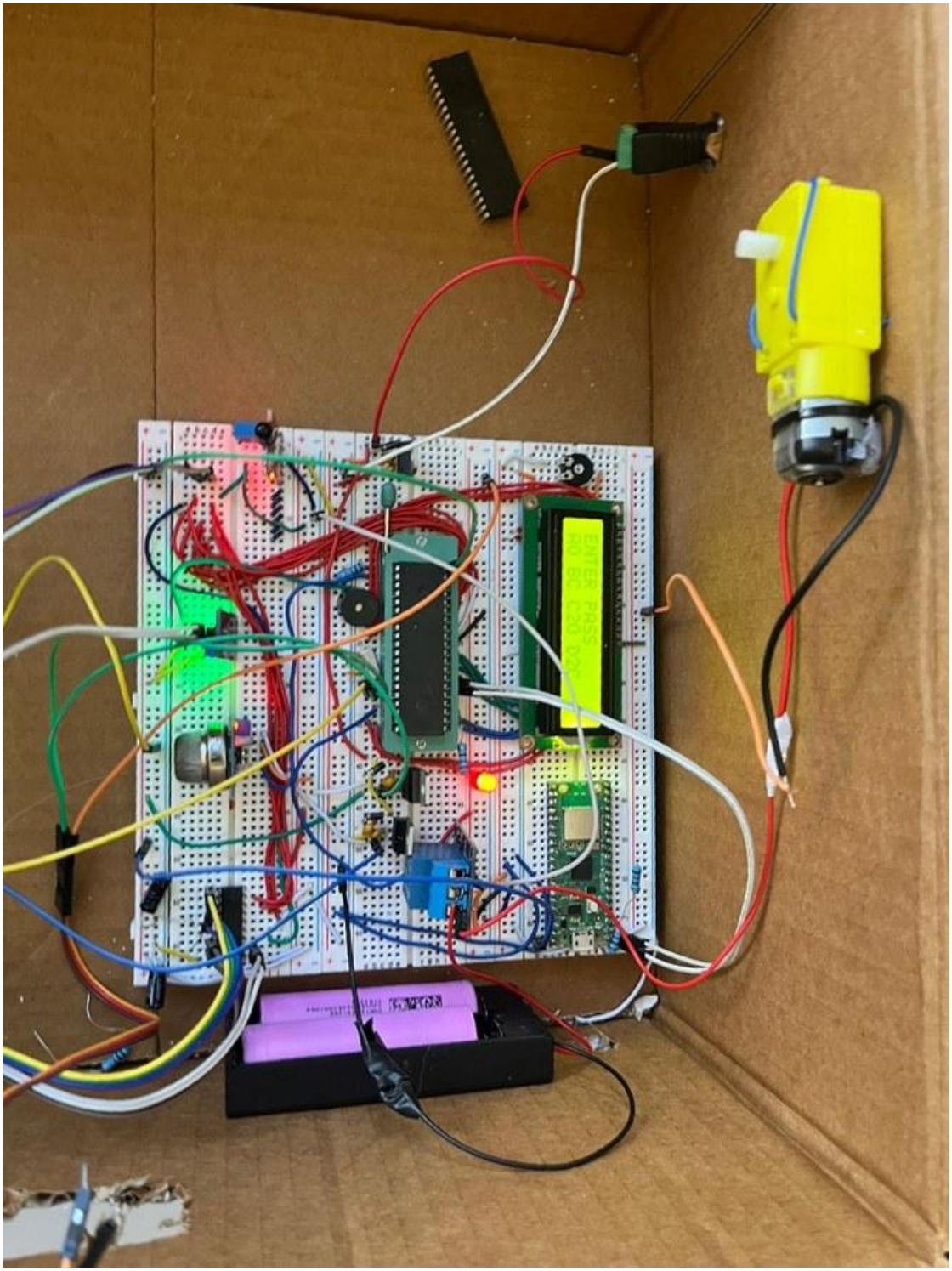

Fig. 10. Internal prototype wiring and components.

Explanation: This picture shows the main hardware platform mounted inside the cardboard model. The PIC16F877A, LCD, Pico W, sensors, buzzer, relay module and wiring are visible. The LCD shows the password screen, which means the controller is running and waiting for keypad input.

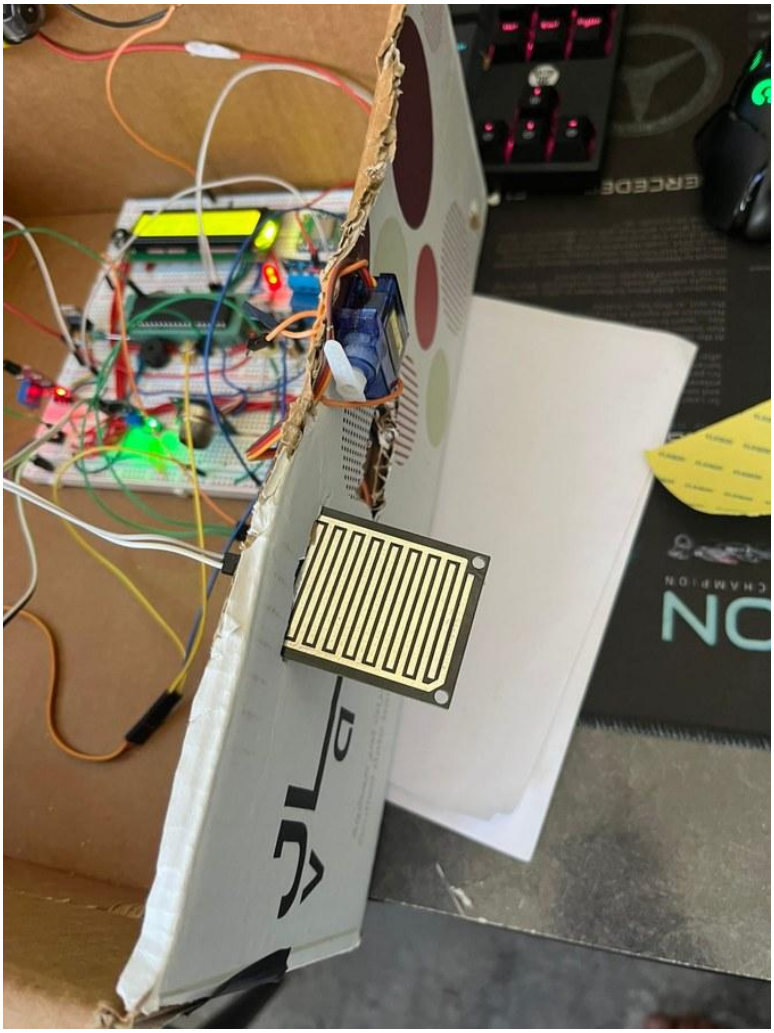

Fig. 11. Water sensor and servo2 area.

Explanation: This picture shows the water sensor mounted on the model and a servo actuator installed near it. When the water sensor detects water, the PIC opens servo2 and sends a water alert to the Pico W.

Fig. 12. Gate servo entry and keypad.

Explanation: This picture shows the front gate area of the prototype. The keypad is used to enter the password 2006, and the gate servo controls the opening and closing of the entry. The gate closes automatically after the IR condition is detected.

XI. Testing and Results

The system was tested in stages. Individual modules were tested first, including the LCD, keypad, servos, buzzer, gas sensor, water sensor, IR sensor, UART communication, web dashboard and Telegram messaging. After confirming that the modules worked individually, the tested subsystems were integrated into the final system.

Test	Expected Result	Final Result
Power on system	LCD shows ready then password screen.	Passed
Enter 2006	Gate opens and sends O to Pico.	Passed
IR low after gate opens	Gate closes after delay.	Passed
Wrong password	LCD shows DENIED / WATCH IR.	Passed
Wrong password then IR low	Security alert and buzzer activate.	Passed
Web ACK button	Pico sends K and PIC clears alarm.	Passed
Gas threshold reached	Fan turns on and heater LED turns off.	Passed
Water detected	Servo2 opens and water alert appears.	Passed
Web dashboard	Dashboard displays current PIC state.	Passed
Telegram	Pico sends status and alert messages.	Passed when internet/DNS is available
Thonny commands	o/c/k/d/t commands send UART actions.	Passed

XII. Challenges and Solutions

Challenge	Solution Applied
LCD flickering and repeated updates	LAST_STATE and NEW_STATE were used to update the LCD only when the system state changes.
Keypad unstable readings	The keypad value is sampled twice after debounce and accepted

	only if both readings match.
Gate auto-close freezing the system	A non-blocking AUTO_CLOSE_FLAG routine was used instead of a blocking IR waiting loop.
PIC to Pico voltage mismatch	A voltage divider is used from PIC RC6 TX to Pico GP1 RX.
Telegram DNS failures	The Pico code includes retry/fallback logic and the webpage remains functional even if Telegram fails.
Subsystem integration complexity	Each module was tested separately before integration into the final code.

XIII. Limitations

- The password is fixed as 2006 and is not stored in EEPROM.
- Telegram depends on internet access and DNS availability.
- Proteus simulation uses equivalent input sources for physical sensors that are not directly available as ready-made modules.
- Sensor thresholds are fixed in the code and may require calibration in different environments.
- Servo timing is blocking during the actual movement period, although the IR waiting phase is non-blocking.

XIV. Future Improvements

- Add EEPROM password storage and allow password change from the keypad or webpage.
- Add live sensor values to the web dashboard instead of only alert states.
- Add live calibration controls for sensor thresholds from the web dashboard.
- Add more detailed sensor logging and automatic report generation for test results.
- Use interrupt-based UART handling for faster communication response.
- Create a mobile application or cloud dashboard for remote access outside the local network.
- Improve the physical model using a printed enclosure and cleaner wiring harness.

XV. Conclusion

The final smart home system successfully combines embedded hardware control and IoT monitoring. The PIC16F877A controls the sensors, actuators, keypad, LCD and buzzer, while the Raspberry Pi Pico W provides WiFi, Telegram notifications, web dashboard control and Thonny testing commands. The project demonstrates the practical use of CPU design and embedded programming concepts in a real smart home prototype.

The final tested system detects gas, detects water, controls a fan relay, controls two servos, checks keypad password access, uses an IR sensor for gate auto-close and security detection, displays system states on the LCD and communicates with a phone through the Pico W. The design is modular, testable and suitable for demonstration in COMP325 - CPU Design.

References

[1] Microchip Technology Inc., PIC16F87XA Data Sheet.

[2] Raspberry Pi Ltd., Raspberry Pi Pico W Documentation.

[3] Hitachi, HD44780U LCD Controller/Driver Data Sheet.

[4] Telegram, Telegram Bot API Documentation.

[5] 74C922 Keypad Encoder Data Sheet.

[6] MQ Gas Sensor Module Technical Documentation.