Department of Electrical and Computer Engineering

North South University

SENTINEL

(Smart Home Management and Security System using Raspberry Pi Pico W, Firebase, and Bluetooth Voice Control)

Course Code: CSE331

Group Members:

Durjoy Barua ID# 2131766642

Nafees Mahmud ID# 2022396642

Shariar Ratul ID# 2111514642

Rapa ID# 2121438042

Section: 1

Faculty Advisor:

Dr. Mohammad Abdul Qayum

Professor

ECE Department

Summer, 2025

ACKNOWLEDGEMENTS

The authors would like to express their heartfelt gratitude to their project supervisor, Dr. Mohammad Abdul Qayum, Professor, ECE Department, North South University, for his timely guidance and constructive feedback throughout the project. The authors are grateful to the ECE laboratory staff for providing access to test equipment and safety resources. We also thank our peers for their feedback during design reviews and our families for their constant encouragement.

ABSTRACT

This project presents a low-cost, Wi-Fi-enabled smart home management and security system built on Raspberry Pi Pico W. The system integrates safety sensing (MQ-2 gas leak, temperature and humidity via DHT11), intrusion monitoring (PIR/IR and ultrasonic-based door proximity), door access control (4×4 keypad and password verification) with a servo-actuated door lock, and utility control (fan and light represented by LEDs) with optional smartphone voice commands over a Bluetooth H5 module using the Arduino Bluetooth app. Real-time status, event logs, and alerts are synchronized with a Firebase Realtime Database and consumed by a companion mobile app for notifications such as gas leak alarms and door status. The prototype was initially attempted on STM32; however, we shifted to Raspberry Pi Pico W due to reliable Wi-Fi connectivity and readily available libraries for OLED displays and Firebase integration. Experiments confirm prompt gas-leak detection, correct password handling with a 3-attempt lockout and continuous buzzer alert, and sub-second dashboard updates. The proposed design demonstrates an affordable and extensible approach to home safety and automation suitable for student projects.

TABLE OF CONTENTS

Chapter 1 Introduction

1.1 Background and Motivation

Residential safety and convenience are increasingly being enhanced by low-cost IoT technologies. Everyday risks such as gas leaks, which can escalate into fire or health hazards if undetected, or unauthorized access that threatens security, highlight the need for smarter solutions at home. At the same time, even simple utilities like fans and lights benefit from automation and remote monitoring, improving comfort and efficiency.

Smart home technologies are emerging as a practical answer to these needs, combining safety, security, and convenience into one platform. Earlier home systems were often limited to standalone alarms or manual supervision, which lacked flexibility and real-time connectivity. In contrast, today's Wi-Fi enabled microcontrollers, such as the Raspberry Pi Pico W, paired with cloud platforms like Firebase, allow real-time sensing, instant notifications, and remote control at very low cost.

By unifying hazard detection, secure access, and utility management under a single affordable framework, systems like this not only enhance household safety but also make modern automation accessible to students, hobbyists, and homeowners alike.

1.2 Purpose and Goal of the Project

Our goal is to design and implement a complete smart home management system integrating:

1. Gas leak and fire indicative sensing via MQ-2 and temperature & humidity via DHT11,
2. Door access with 4×4 keypad and servo-actuated lock, with three password attempts(also make key sound) and buzzer lockout on failure,
3. Intrusion detection using an IR sensor and visitor proximity using an ultrasonic module as an intruder alert in the door or as a calling bell indicator,
4. Bluetooth voice control for fan and light,
5. An OLED display for local status plus Firebase cloud logging for remote monitoring and notifications.
6. A fluter app named "SENTINEL" take data from the realtime database(firebase)
   We first attempted STM32, then migrated to Raspberry Pi Pico W for improved connectivity and library support.

1.3 Organization of the Report

Chapter 2 reviews related IoT home security works and limitations. Chapter 3 details the system design, components, and implementation. Chapter 4 presents experiments and results. Chapter 5 discusses societal and environmental impacts. Chapter 6 provides planning and budget. Chapter 7 maps the work to Complex Engineering Problems and Activities. Chapter 8 concludes with limitations and future work.

Chapter 2 Research Literature Review

2.1 Existing Research

1) Survey: Smart-home security risks and mitigations

A. What they studied. A 2024 survey maps security threats in smart homes (unauthorized access, data leaks, device tampering) and reviews mitigations like authentication, encryption, and intrusion detection.
B. Methodology. Systematic review across common device classes (locks, sensors, hubs) and protocols; compares defensive mechanisms and gaps.
C. Results. Highlights that end-to-end auth, secure storage of secrets, and hardening of companion apps are often weak links; recommends layered security (device + app + cloud). This directly supports adding lockout policies and secure cloud rules in systems like yours. [1]

2) Home security + fire detection with PIR & MQ-2 (Arduino/Blynk)

A. What they built. Low-cost security/ safety node with PIR (HC-SR501) and MQ-2 gas sensor, clouded to a phone dashboard.
B. Methodology. Arduino/ATmega MCU; Blynk IoT app for notifications; threshold-based alarms for motion and gas.
C. Results. Demonstrated remote alerts and live status on mobile; authors note affordability and responsiveness, but dependence on stable connectivity. This mirrors your intrusion + gas pipeline and motivates your Firebase choice. [2]

3) Android-based home security with Firebase alerts

A. What they built. Android app + IoT node (PIR, flame/smoke sensors) pushing events to Firebase, which triggers user notifications.
B. Methodology. Real-time database paths mapped to app listeners; simple sensor thresholds.
C. Results. Achieved timely alerts and centralized logging; paper flags network dependence and urges stronger data security useful for shaping your Firebase rules and minimal data storage approach. [3]

4) Firebase smart-home control via NodeMCU/ESP8266

A. What they built. Appliance control and monitoring using ESP8266 + Firebase Realtime Database with a mobile UI.
B. Methodology. REST/SDK updates to Firebase for state sync; command topics mirrored to actuators.
C. Results. Validated reliable two-way sync and low cost; demonstrates the same cloud pattern you adopted (state mirroring + notifications), supporting your design choice. [4]

5) IoT gas-leakage detection with ESP32 + MQ-2

A. What they built. Real-time gas monitoring using ESP32 and MQ-2 with IoT alerts (visual, audible, and remote).
B. Methodology. Sensor warm-up and thresholding on MQ-2; event publishing to cloud/UX.
C. Results. Reports effective early warning at low cost; stresses calibration and environment effects precisely the limitations you noted for MQ-2. [5]

6) Gas leakage & fire detector with cloud (ThingSpeak)

A. What they built. Two-part system: MQ-2-based gas detection + fire alarm, with continuous/intermittent buzzer patterns and ThingSpeak cloud monitoring.
B. Methodology. ESP32 + cloud channel logging; rule-based alerts.
C. Results. Demonstrated remote dashboards and differentiated alarm behavior; supports your buzzer-lockout UX and cloud logging concept.[6]

7) Bluetooth voice-controlled home automation (Arduino + HC-05)

A. What they built. Voice commands from a smartphone over classic Bluetooth to toggle loads via Arduino/relays.
B. Methodology. On-device command parser mapping recognized words to actions; Android voice input → serial Bluetooth frames.
C. Results. Shows robust short-range control without Wi-Fi; underlines dependence on phone speech recognition quality—exactly the caveat you observed.[7] [8]

8) Ultrasonic-triggered door/doorbell automation

A. What they built. Door/doorbell systems that sense proximity with ultrasonic sensors (and sometimes IR), driving a servo or notifications.
B. Methodology. HC-SR04 ranging with distance thresholds; PWM servo actuation; some versions pair with cloud or app UIs.
C. Results. Validates proximity-based calling-bell/door actuation and typical latencies; aligns with your "visitor proximity as bell indicator" and servo lock.[9] [10]

9) Pico W as a Wi-Fi IoT platform

A. What it is. Official announcement/tutorials on Raspberry Pi Pico W: on-board 802.11n Wi-Fi with MicroPython/C SDKs for fast IoT prototyping.
B. Relevance. Corroborates your platform shift from STM32 to Pico W for Wi-Fi stability and library support (OLED, cloud clients, MQTT/HTTP).[11] [12]

10) Companion-app security for smart devices

A. What they studied. Static and manual analysis of 54 Android companion apps for smart security devices.
B. Methodology. SAST + manual testing in a SecDevOps style.
C. Results. Common vulnerabilities (insecure storage, weak auth) are widespread; reinforces your decision to store minimal data and to implement strict Firebase rules and lockout.[13]

2.2 Limitation of those works

1) Survey on Smart-Home Security Risks (MDPI, 2024)

1. It was a review only, not an implementation.
2. Did not propose a concrete prototype.
3. Highlighted that most systems suffer from weak authentication, insecure app storage, and lack of multi-layer protection.

2) PIR + MQ-2 with Blynk (GCISTEM Proceedings, 2022)

1. Depended on Blynk; reliability falls with unstable Wi-Fi or server downtime.
2. Very limited scope: only motion + gas detection, no secure access or multiple features.

3) Android + Firebase Alerts (ResearchGate, 2021)

1. Limited security; only used Firebase listeners without authentication rules.
2. Single-app dependency, less hardware integration (mostly PIR + fire sensors).
3. Network dependency not addressed (fail-safes absent).

4) NodeMCU + Firebase Smart-Home Control

1. Focused only on appliance control (lights, fans).
2. Lacked security measures (no door access, no lockout policy).

5) ESP32 + MQ-2 Gas Leakage Detection (IJCRT, 2024)

1. Reliant solely on MQ-2, which is known for poor selectivity and environmental sensitivity.
2. No access control or other security features.

6) ThingSpeak Gas & Fire Detector

1. Depended on a public IoT platform (ThingSpeak), which has slower update cycles than Firebase.
   Did not integrate local alarm prioritization (cloud-first approach).

7) Bluetooth Voice-Controlled Automation (Arduino + HC-05)

1. Relies fully on smartphone speech recognition (voice quality dependent).
2. No security or fallback if Bluetooth is not available.

8) Ultrasonic Door Automation

1. Ultrasonic sensors struggle with soft/absorptive materials (clothing).
2. Standalone door control—no integration with alarms/cloud.

9) Raspberry Pi Pico W Tutorials

1. Tutorials focus only on connectivity examples, not full systems.
2. No security or multi-sensor integration.

10) Companion App Security Evaluation (MDPI, 2024)

1. Found common vulnerabilities in Android IoT apps (hardcoded keys, poor encryption).
2. No prototype was built, only an analysis.

2.3 How these works align with our system

1) Survey on Smart-Home Security Risks (MDPI, 2024)

1. We addressed this by enforcing a 3-attempt lockout with buzzer alarm for keypad access.
2. We restricted Firebase to minimal data logging (no video/audio), reducing data exposure.
3. This directly strengthens the weak points identified in the survey.

2) PIR + MQ-2 with Blynk (GCISTEM Proceedings, 2022)

1. We expanded scope by including door access control, ultrasonic visitor sensing, OLED display, and Bluetooth voice control.
2. We shifted to Firebase, which provides real-time sync with fine-grained rules, making it more robust than Blynk.

3) Android + Firebase Alerts (ResearchGate, 2021)

1. We used Firebase but implemented a state machine with a local OLED + buzzer to ensure alerts work even without the internet.
2. Multiple sensor fusion (PIR + ultrasonic + IR) reduced false positives compared to single PIR sensors.

4) NodeMCU + Firebase Smart-Home Control

1. We included secure keypad entry and intrusion monitoring in addition to utilities.
2. We proved Firebase can handle both safety-critical alerts (gas, intrusion) and convenience controls (fan/light).

5) ESP32 + MQ-2 Gas Leakage Detection (IJCRT, 2024)

1. We acknowledged MQ-2's limitations but compensated with a threshold + calibration routine.
2. We supplemented safety with intrusion + access security features beyond just gas sensing.

6) ThingSpeak Gas & Fire Detector

1. We prioritized local alarms via OLED + buzzer with sub-second updates, while still logging to Firebase.
2. This makes our system more reliable under poor connectivity.

7) Bluetooth Voice-Controlled Automation (Arduino + HC-05)

1. We kept Bluetooth voice control for convenience only, not for security-critical functions.
2. Primary security uses keypad + Firebase logging, so even if Bluetooth fails, the system remains secure.

8) Ultrasonic Door Automation

1. We fused PIR + ultrasonic to reduce false triggers.
2. Integrated with door lock + buzzer + Firebase, creating a more complete solution.

9) Raspberry Pi Pico W Tutorials

1. We extended Pico W's capabilities into a full-fledged multi-sensor + Firebase + Bluetooth system, showing its feasibility for real deployments.

10) Companion App Security Evaluation (MDPI, 2024)

1. We avoided building a complex app with sensitive data; instead, we used Firebase minimal logs + notifications.
2. Stored no passwords/tokens in the app, reducing risk.

Chapter 3 Methodology

3.1 Design methodology

Figure 1 illustrates the system block diagram: Sensors (MQ-2, DHT11, PIR/IR, ultrasonic) feed the Raspberry Pi Pico W. The 4×4 keypad provides password input; a micro servo drives the door lock. A buzzer signals alarms and lockouts. User utilities (fan and light represented by LEDs via relays/transistors) can be toggled through Bluetooth voice commands received from a smartphone app. An SSD1306-based OLED (I²C) shows status and alerts locally. The Pico W publishes structured data to Firebase (door state, intrusion flag, gas level, temperature, humidity, fan/light state, attempts remaining, timestamps). A companion mobile app subscribes to Firebase for notifications and dashboards.

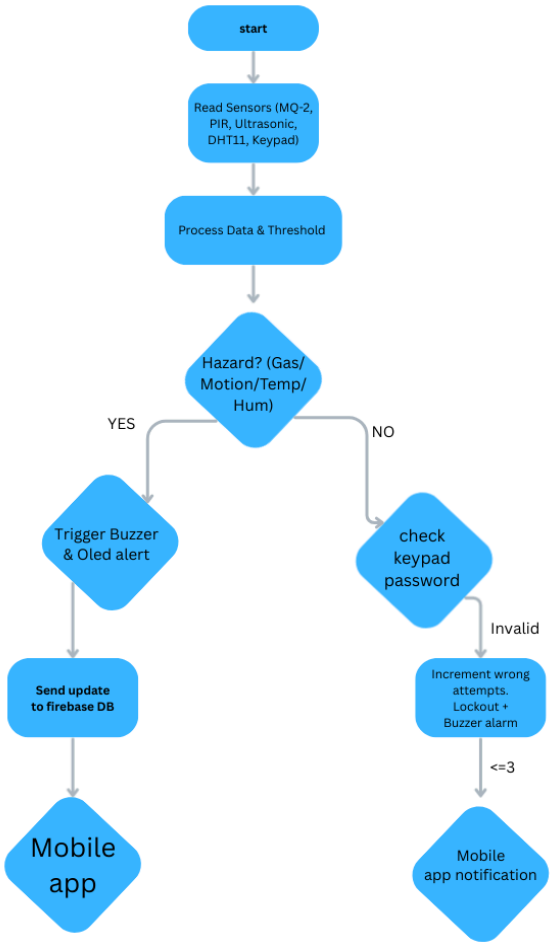

Figure 1:flowchart

The diagram shows the central SmartHomeController on the Raspberry Pi Pico W, which interacts with sensor modules (MQ-2, DHT11, PIR/Ultrasonic, Keypad), actuator modules (Servo, Buzzer, LEDs), and service modules (OLED Display, Bluetooth, Firebase Client). External actors such as the user, visitor, and mobile app connect through keypad input, Bluetooth commands, and Firebase synchronization, illustrating how sensing, control, and cloud communication are integrated into one system.

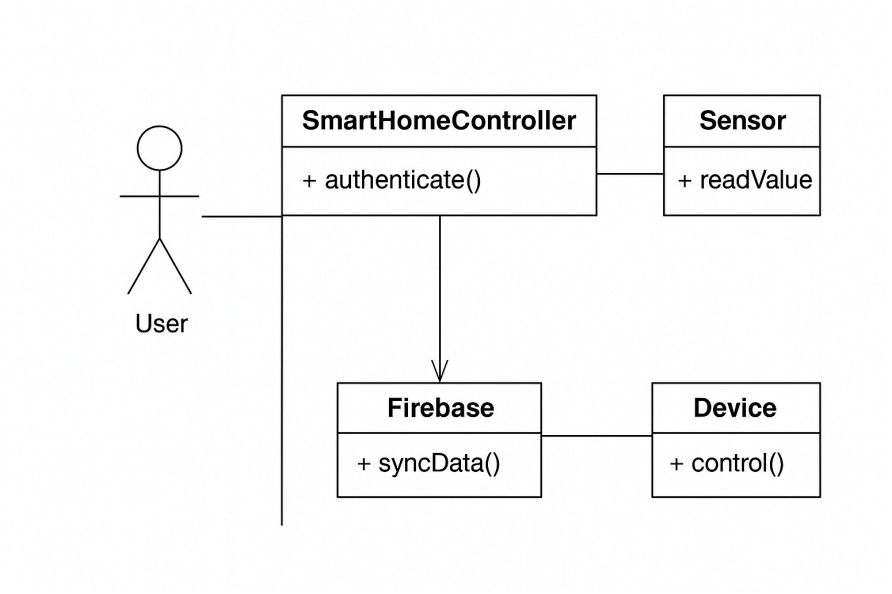

Figure 2:Class diagram

3.2 Hardware and/or Software Components

Tool	Functions	Why selected
Raspberry Pi Pico W	Core MCU with Wi-Fi; GPIO for sensors/actuators; MicroPython/C SDK	Low cost; stable Wi-Fi; strong library ecosystem
MQ-2 Gas Sensor	Detects LPG/propane/smoke concentration	Widely available; fast response for leak alarms
DHT11	Temperature & humidity measurement	Simple digital interface; sufficient for thresholding
IR Sensor	Detect human motion for intrusion	Low power; common in security use
Ultrasonic (HC-SR04)	Visitor distance for calling bell trigger	Reliable short-range ranging

4×4 Keypad	Password input for door access	Simple matrix scanning; robust
Servo Motor (SG90)	Door lock actuation	Low torque needs; easy PWM control
Buzzer	Audible alerts and lockout alarm	Immediate attention cue
OLED 0.96" SSD1306 (I²C)	Local status display	Compact; crisp text/icons
Bluetooth H5 module	Serial Bluetooth link to phone	Works with Arduino Bluetooth app for voice/text commands
Firebase Realtime DB	Cloud data store and notifications	Real-time sync; cross-platform SDKs
Companion Mobile App	User dashboard & notifications	Shows door, gas, intrusion, fan/light status

3.3 Hardware and/or Software Implementation

Hardware: The Pico W interfaces MQ-2 via an ADC channel with a voltage divider; DHT11 via a single data pin; PIR/IR and ultrasonic via digital I/O and trigger/echo pins (with echo via level-safe input). The 4×4 keypad uses matrix scanning over eight GPIOs with pull-ups. The servo is driven by PWM (50 Hz), powered from a dedicated 5 V rail with a common ground and decoupling. The buzzer is a digital output (through a transistor if active current is higher). The OLED connects over I²C (SDA, SCL). Proper series resistors and transistor drivers are used for LEDs/relays.

Software: Firmware is organized into modules: sensor drivers, keypad/password manager with attempt counter, actuator control (servo, buzzer, LEDs/relays), display service, Bluetooth command parser, Firebase client, and state machine. Password verification allows three attempts; on failure a continuous buzzer is latched until a master reset. Events (gas leak, intrusion, door open/closed, fan/light toggles) are pushed to Firebase with timestamps. The companion mobile app subscribes to key paths for notifications. We first attempted an STM32-based build (HAL drivers + UART/Wi-Fi shield), but unstable Wi-Fi stack integration and lack of Firebase libraries impeded progress. Migrating to Pico W simplified Wi-Fi and accelerated development.

3.4 Testing methodology

1. Test Objectives

1. Verify each hardware module and driver works to spec (functional correctness).
2. Validate integrated behaviors: hazard detection, access control, utility control, display, and cloud sync.
3. Quantify performance: detection latency, cloud round-trip time, servo actuation time, keypad debouncing accuracy, and Bluetooth command response.
4. Assess robustness under edge cases: unstable Wi-Fi, incorrect passwords, noisy sensor readings, and power disturbances.
5. Confirm security controls: 3-attempt lockout with continuous buzzer, limited data exposure to Firebase.

2. Test Environment & Instrumentation

1. Hardware under test: Raspberry Pi Pico W, MQ-2, DHT11, PIR/IR, HC-SR04, SG90 servo, 4×4 keypad, buzzer, SSD1306 OLED, HC-05 Bluetooth, LEDs/relays for fan/light.

2. Firmware: hardware_code(rasberry_pie).py with constants ULTRA_THRESHOLD_CM=10.0, TEMP_THRESHOLD=40, HUM_THRESHOLD=120, GAS_THRESHOLD=10000, update intervals: DHT=3000 ms, Firebase=2000 ms, OLED=200 ms.

3. Network: Campus Wi-Fi and a mobile hotspot (for variation), Firebase Realtime Database endpoint used in code.

4. Tools: Stopwatch (or phone timer), reference thermometer/hygrometer, ruler/tape (distance checks), multimeter (supply and ground noise), Android phone with Bluetooth terminal/voice app, Firebase console for timestamps/logs.

5. Safety: Gas tests conducted with small, controlled vapors (e.g., isopropyl alcohol at a distance) in a ventilated lab; no open flame near the sensor. Eye/hand protection during wiring and servo tests.

3. Unit Tests (Module-Level)

3.1 MQ-2 Gas Sensor

1. Calibration: Warm up for ≥2–3 min. Record ambient gas_value baseline (n=30 samples).
2. Trigger test: Briefly expose to a safe vapor source; verify gas_value > GAS_THRESHOLD (10000) toggles gas_detect=True and buzzer on.

3. Acceptance: Alarm asserted within ≤2 s from threshold crossing; value returns below threshold after dispersion.

3.2 DHT11 (Temperature/Humidity)

1. Stability: Sample every 3 s (per code) for 5 min; compute mean/variance.
2. Cross-check: Compare temperature with a reference meter ($\Delta T \leq \pm 2$ °C is acceptable for DHT11).
3. Threshold tes**t:** Simulate TEMP > 40 °C (e.g., by warming sensor area carefully) to confirm alarm path.

3.3 PIR/IR Motion

1. False-idle: No motion condition for 2 min → ir_detect=False, buzzer off.
2. Motion pass: Walk across sensor FOV at 2–3 m → ir_detect=True and alarm active.
3. Repeatability**:** 10 trials; expect ≥9/10 detections.

3.4 Ultrasonic (HC-SR04)

1. Distance sweep: Place a flat target at 5, 10, 15, 20 cm; record distance_cm.
2. Threshold trip: At <10 cm (ULTRA_THRESHOLD_CM), alarm must assert.
3. Edge case: Soft/absorptive cloth target → note any missed echoes; confirm PIR compensates at system level.

3.5 Keypad & Lockout

1. Debounce: Rapid single key presses (n=20) should register once each.
2. Password flow: Enter PASSWORD_CODE=['1','1','1','1'] + D → ACCESS GRANTED, servo opens.
3. Lockout: Enter 3 wrong codes → password_alarm=True, continuous buzzer; **C** clears alarm (per code).

3.6 Servo (SG90)

1. Actuation: Measure time from command to stable angle (0°↔90°). Expect ≈0.4–0.6 s.
2. Jitter/noise**:** Observe 5 V rail with multimeter during moves; verify no brownouts/reset.

3.7 OLED (SSD1306)

1. Update cadence: Every ≈200 ms show fresh state (distance/IR/gas/temp/hum/door/alarm/tries/pass buffer).
2. Priority: When the alarm is true, the alert line must be visible on the first screen without scrolling.

3.8 Bluetooth (HC-05, UART0@9600)

1. Command set:
   turn on led → LED ON; turn off led → LED OFF
   open door/close door → servo target updates
   alarm on/alarm off → alarm latch
2. Latency: Measure time between send and action ≤500 ms.

4. Integration Tests

4.1 Intrusion + Access Fusion

1. Scenario A: Visitor approaches to <10 cm → ultrasonic alarm; then valid password opens door and silences alarm.
2. Scenario B: Motion only (PIR) without proximity → alarm still asserts; confirm fusion logic works (OR condition in code).

4.2 Gas + Intrusion Race

1. Trigger gas alarm and approach door simultaneously; confirm buzzer remains active (gas has equal priority), OLED shows alarm, Firebase logs both events.

4.3 Keypad Abuse + Lockout

1. Enter 3 wrong codes while motion is detected → ensure password_alarm dominates (buzzer continuous) until **C** reset; door cannot be opened via **A** until alarm cleared.

4.4 Utility + Security Separation

1. While the alarm is on, send Bluetooth turn on led / turn off led and observe: convenience actions still function but do not suppress alarms.

5. Cloud & Connectivity Tests (Firebase)

5.1 Write Frequency & Integrity

1. With FIREBASE_UPDATE_MS=2000, verify updates every ≈2 s. Cross-check payload fields (distance_cm, ir_detect, gas_value, door_open, wrong_attempts, status, timestamp) on Firebase console.

5.2 End-to-End Latency

1. Record local event timestamp (from OLED observation or a debug print) and the Firebase timestamp arrival to the mobile app.

2. Acceptance: Typical campus Wi-Fi <500 ms average; hotspots may be higher—log mean and 95th percentile over 50 events.

5.3 Network Loss & Recovery

1. Disable Wi-Fi AP for 2–3 min: local alarms must continue (buzzer/OLED).
2. Re-enable AP: confirm automatic reconnection and resumed PATCH updates; no firmware crash.

6. Performance & Soak Tests

6.1 12-Hour Soak

1. Run system overnight: log count of Firebase updates (~6/hr × 12 h ≈ 360 writes), number of alarms (if any), servo actuations.
2. Pa**ss:** No lockups; memory stable; no unintended resets.

6.2 Power-Noise Resilience

1. Toggle servo repeatedly (100 cycles open/close) while reading sensors; verify no OLED freezes, no Wi-Fi drops.
2. If issues occur, note whether isolating servo supply or adding decoupling mitigates (already suggested in design).

6.3 Threshold Margins

1. Sweep ultrasonic target slowly through 12→8 cm and back to check chatter around 10 cm; confirm minimal flicker due to 200 ms OLED cadence and alarm latching.

7. Data Quality & Logging

1. Sampling checks: Ensure DHT readings update every ≈3 s; reject None values in code path (already handled in read_dht_safe()).
2. Rounding: Verify distance_cm rounding to 0.1 cm matches observed changes; "---" shown when invalid (timeout).
3. Attempts counter: Confirm wrong_attempts increments only on D (submit) and resets on success or C.

8. Usability Tests

1. Readability: From 1 m away, confirm OLED text is legible in typical room lighting.
2. Keypad feel: Users enter the 4-digit code consistently without missed keys (≥95% success across 20 trials).

3. App side: Mobile notifications arrive promptly; labels match OLED states.

9. Acceptance Criteria (Summary)

Area	Metric	Target
Gas detection	Time to alarm after threshold	≤ 2 s
Motion/proximity	Detection rate	≥ 90% over 10 trials
Servo	Open/close time	≤ 0.6 s
Keypad	Lockout after wrong attempts	Exactly 3 attempts
Cloud	Avg E2E latency (Wi-Fi)	≤ 500 ms
OLED	Update cadence	≈200 ms
Soak	Uptime over 12 h	0 crashes/lockups

10. Test Cases

1. TC-KPAD-01: Enter 1111 + D → Door opens, status='ACCESS GRANTED', Firebase logs door_open=True.

2. TC-KPAD-02: Enter 3 wrong codes → password_alarm=True, buzzer continuous until C pressed.

3. TC-GAS-01: Expose MQ-2 to safe vapor → gas_detect=True, buzzer on, Firebase field changes within 2 s.

4. TC-ULTRA-01: Place target at 8 cm → alarm on; move to 15 cm → alarm off (assuming no PIR).

5. TC-BT-01: Send open door via Bluetooth → Servo opens; Door OPENED echoed.

6. TC-CLOUD-01: Trigger any event → Firebase update observed within ≤500 ms on app.

7. TC-NET-01: Disconnect Wi-Fi → local alarms persist; reconnect → updates resume.

8. TC-SOAK-01: 12-hour run with periodic alarms/commands → no resets; all modules responsive.

Chapter 4 Experiment, Result, Analysis

We validated each subsystem and the integrated system. MQ-2 was preheated and calibrated; we defined a leak threshold corresponding to a stable analog value after ambient baseline averaging. DHT11 readings were sampled every 2 s to compute moving averages. Intrusion detection fused PIR motion with ultrasonic distance to reduce false positives; a short dwell (e.g., 300 ms) prevented spurious triggers. The keypad module debounced rows/columns and enforced a three-attempt counter. The OLED displayed the highest-priority alert (gas, intrusion, lockout) with scrolling for secondary info. Firebase update latency was measured by timestamping writes and observing them on the app, averaging < 500 ms on campus Wi-Fi.

Key results: (i) gas leak alarms triggered within ~2 s of threshold crossing;

(ii) password lockout engaged consistently after three wrong entries, with continuous buzzer until reset;

(iii) servo opened/closed the door in ~0.4 s travel;

(iv) Bluetooth voice commands reliably toggled fan/light by mapping recognized words to single-character commands; and

(v) OLED legibly displayed multi-line status. Discussion: The Pico W provided adequate compute and memory headroom; Firebase greatly simplified multi-client synchronization. Proper grounding and a separate 5 V rail avoided servo-induced resets. Ultrasonic readings degraded on soft clothing; fusing with PIR mitigated this.

Chapter 5 Impacts of the Project

5.1 Impact on societal, health, safety, legal and cultural issues

The system improves household safety by detecting gas leaks early and notifying occupants. Intrusion detection and access control reduce security risks. Legal and ethical handling of data is respected by storing only necessary states (no audio/video), minimizing privacy concerns. The low cost broadens access to safety technology.

5.2 Impact on environment and sustainability

Automated control can reduce energy use by switching off utilities when not needed. The design uses common, repairable components and encourages modular upgrades, extending device lifetime. Cloud logging can inform preventive maintenance rather than reactive replacement.

Chapter 6 Project Planning and Budget

Weeks 1–2: requirements and STM32 prototype; Weeks 3–4: migration to Pico W and driver bring-up; Weeks 5–6: cloud and mobile app integration; Weeks 7–8: enclosure and wiring; Weeks 9–10: testing, calibration, and documentation.

Table II the budget.

Item	Qty	Unit Cost	Subtotal
Raspberry Pi Pico W	1	1550 BDT	1550 BDT
MQ-2 Gas Sensor	1	290 BDT	290 BDT
DHT11 Sensor	1	150 BDT	150 BDT
IR Motion Sensor	1	50 BDT	50 BDT
Ultrasonic HC-SR04	1	100 BDT	100 BDT
4×4 Keypad	1	90 BDT	90 BDT
SG90 Servo	1	135 BDT	135 BDT
OLED 0.96" SSD1306	1	290 BDT	290 BDT
Bluetooth H5 Module	1	400 BDT	400 BDT
Buzzer + Transistor + Resistors	1	120 BDT	120 BDT
wiring	1	200 BDT	200 BDT
Total			3375 BDT

Chapter 7 Complex Engineering Problems and Activities

7.1 Complex Engineering Problems (CEP)

Attribute	Addressing the complex engineering problems (P) in the project
P1: Depth of knowledge required (K3–K8)	Circuits & Electronics (K3), Embedded & Wireless (K4), Design & Simulation (K5), Engineering tools & mobile apps (K6), Environmental/safety effects (K7), literature survey (WK8).
P2: Range of conflicting requirements	Balancing responsiveness vs. false alarms; low cost vs. reliability; security vs. usability (lockout vs. convenience).
P3: Depth of analysis required	Sensor fusion thresholds; debounce and attempt counters; Wi-Fi vs. Bluetooth trade-offs; platform migration analysis (STM32 → Pico W).
P4: Familiarity of issues	Commodity sensors (MQ-2, DHT11, PIR), microcontrollers (STM32, Pico W), cloud services (Firebase).
P5: Extent of applicable codes	No formal standards; adopts best practices for electrical safety and data privacy.
P6: Stakeholder involvement	Homeowner, household members, visitors, maintenance; app users receiving notifications.
P7: Interdependence	Multiple subsystems: sensing, access, actuation, display, communication, cloud database, and mobile app.

7.2 Complex Engineering Activities (CEA)

Attribute	Addressing the complex engineering activities (A) in the project
A1: Range of resources	Human resources, budget, tools (IDE, mobile app builder), hardware lab equipment.
A2: Level of interactions	Team collaboration; stakeholder feedback; interactions between embedded firmware, cloud database, and mobile app.
A3: Innovation	Combining keypad-based access, servo door control, Bluetooth voice utility control, and Firebase logging on Pico W.
A4: Consequences to society/environment	Enhanced safety from gas/intrusion; potential energy savings via utility control.
A5: Familiarity & UN SDGs	Familiarity with sensors/MCUs/cloud; aligns with SDG #9 (Industry, Innovation), #11 (Sustainable Cities).

7.2 Contribution table

Member	Contributions
Durjoy barua(Project lead)	Report, Keypad, Servo Motor, Ultrasonic,display
Nafees Mahmud	Voice Control LED, Fan, IR Sensor,Buzzer
Shariar Ratul	App, Database Handle, Bluetooth sensor, Connecting wifi
Rapa	DHT11, MQ-2, Report

Chapter 8 Conclusions

8.1 Summary

We built a smart home management and security prototype on Raspberry Pi Pico W that integrates safety sensing, access control, voice-assisted utility toggling, local OLED status, and Firebase cloud synchronization with a companion app for notifications. The system enforces a 3-attempt lockout policy and continuous buzzer alarm for security.

8.2 Limitations

MQ-2 selectivity varies with gas type and environmental conditions; DHT11 accuracy is modest. Bluetooth voice control depends on smartphone speech recognition quality. Ultrasonic ranging may be inconsistent on absorptive clothing. The prototype uses a simple password scheme rather than cryptographic authentication.

8.3 Future Improvement

Upgrade to DHT22 and a calibrated gas sensor (e.g., MQ-4 for methane) or an NDIR module; add camera-based verification; adopt secure tokens for app access; integrate OTA updates; design a custom PCB and enclosure for robustness; migrate mobile app to Flutter for cross-platform UI.

References

1. Vardakis, G., et al. “Review of Smart-Home Security Using the Internet of Things.” Electronics, 2024. MDPI

2. Rhizma, M.G.A., et al. “Home Security and Fire Detection System Design Using Arduino with PIR HC-SR501 and MQ-2 (Blynk Integration).” GCISTEM Proceedings, 2022. GCISTEM Proceeding

3. “Android-Based Home Security Systems Using IoT and Firebase.”. ResearchGate

4. “A Firebase-Based Smart Home Automation System Using IoT (NodeMCU/ESP8266).” (Project/paper page). ResearchGate

5. “IoT-Based Gas Leakage Detection System (ESP32 + MQ-2).” IJCRT, Dec 2024. IJCRT

6. “Gas Leakage and Fire Detector Based on IoT Network (ESP32 + ThingSpeak).” ResearchGateGlobal Research Network Journals

7. “Voice-Control Home Automation System (Arduino + Bluetooth).” TIJER, 2023. Tijer

8. “Home Automation Using Bluetooth Voice Control.” IRJMETS, 2024. IRJMETS

9. “Automatic Smart Doorbell Using Ultrasonic Sensor.” IRJMETS, 2024. IRJMETS

10. “Arduino-Based Door Automation Using Ultrasonic Sensor and Servo Motor.”. ResearchGate

11. Raspberry Pi. “Pico W: Your $6 IoT Platform.” 2022. Raspberry Pi

12. HiveMQ. “Reading Sensor Data with Raspberry Pi Pico W (MicroPython, MQTT, Node-RED).” 2025. HiveMQ

13. Allen, A., et al. “Security Evaluation of Companion Android Applications in Smart Security Devices.” Sensors, 2024. MDPI

CODE LINK:

(app+harware code)
Durjoy01/Smart_home_management_system