FINAL PROJECT REPORT

Embedded Systems Course (Interrupt-Driven Project)

---

1. Project Title and Team Information

- **Project Title:** Edge-AI Biometric Access Control System
- **Course Name and Code:** Embedded Systems - COE185
- **Instructor:** Stephen Haim

Team Members

Name	Email
Nathaniel R. Talip II	nathanielii.talip@g.msuiit.edu.ph
Chris Immanuel I. Arcasa	chrisimmaneul.arcasa@g.msuiit.edu.ph

GitHub Repository Link:
https://github.com/immanuel-chris/coe185-project-arcasa-talip

---

2. Project Description (Final)

2.1 Project Overview

This project implements a standalone, privacy-centric security system that uses minimal power capable of controlling physical access (a door lock) using facial detection, utilizing the MAX78000. The system remains in a low-power idle state until triggered by a hardware interrupt (push button/doorbell). Upon activation, it wakes the camera, captures an image, uses an on-chip Convolutional Neural Network (CNN) to detect human presence, and actuates a servo mechanism to grant access.

2.2 Problem Statement and Motivation

Many facial recognition based security systems stream live video feed of a person's face to an external server through the cloud. While effective, utilizing such an approach has the following risks:

- Potentially high latency
- Potential to erroneously lock the user out of their homes in certain circumstances (timeout from server or server experiencing a power outage for example)
- Prone to data breaches, especially if the connection between the system to the cloud is insecure

Another approach that utilizes edge computing is paramount, as it offers:

- Low latency, as facial recognition is done on the edge
- Privacy and security, for the same reason as above

2.3 Project Objectives

The primary objective of this project was to design and implement an Interrupt-Driven, Edge-AI Access Control System using the MAX78000FTHR microcontroller, adhering to strict Bare Metal C programming requirements. Specific sub-objectives included:

- To implement a Wake-on-Interrupt power architecture where the system remains in a deep sleep state (WFI) until triggered by an external hardware event (push-button).
- To deploy a pre-trained Face Detection CNN model (RetinaFace/MTCNN based) capable of running locally on the MAX78000 CNN accelerator without cloud connectivity.
- To control physical actuators (Servo Motor) using Direct Register Access (DRA) and bit-banged PWM, bypassing high-level SDK abstraction layers to demonstrate low-level hardware mastery.
- To integrate a complete embedded control loop: Input (Camera/Button) → Processing (CNN Inference) → Output (Servo/LEDs).

2.4 Changes from Previous Milestones

The project underwent a significant transformation following the first milestone. After a strategic evaluation of the initial scope, we executed a pivot toward a more viable application titled: "Edge-AI Biometric Access Control System."

In the final phase, we successfully addressed the outstanding tasks from the second milestone. This included interfacing the servo driver and pushbutton, followed by rigorous integration testing with the microcontroller. Upon validating that all system components were fully

operational, the hardware was integrated into its final enclosure to complete the product assembly.

---

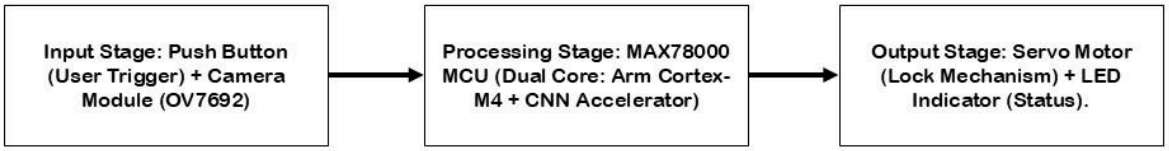
3. Final System Architecture

3.1 Overall System Description

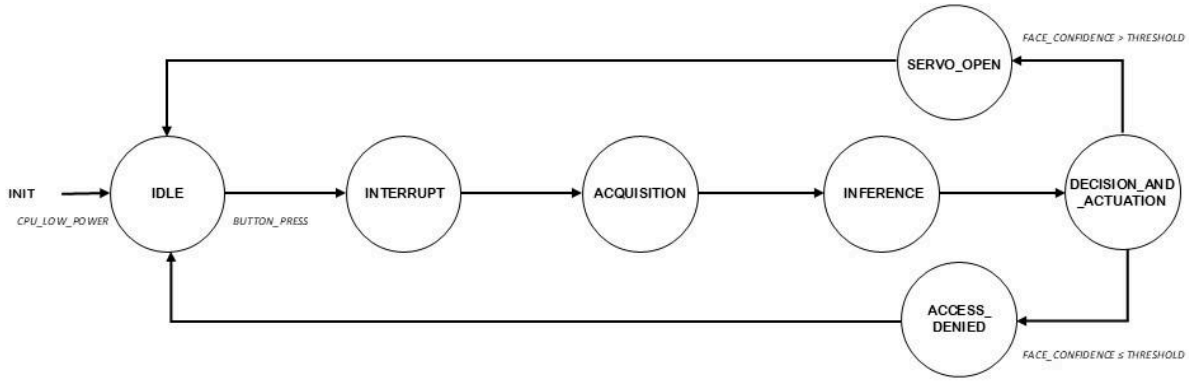
In order to use the system, one must put their face in front of the MAX78000's camera, and press a button to initiate the scan. If a face is recognized, the system will send a signal that opens a door (represented by a servo). Otherwise, the system stops, and the user may have to redo the process.

3.2 Block Diagram

Block diagram:



State diagram:



3.3 Hardware Components

- MCU
- Camera
- Power source
- Servo

3.4 Software Architecture

Main program structure

The program is organized as a boot/initialization phase followed by an event-driven infinite loop. In the boot/initialization phase, all the pins and libraries required for the functioning of the system are to be initialized, and for good measure, a diagnostic test that will check if the initializations worked as intended. The infinite loop is to be divided into two phases, which are sleep mode and active mode. In sleep mode, the system uses little power and waits for a button to be pressed. Once the button is pressed, the system enters active mode, where the camera performs facial recognition to determine whether or not the door can be opened. After the outcome of this test, all flags related to this check reset and the system goes back to sleep mode.

Task allocation between main loop and interrupts

The main loop is to handle servo rotation and facial recognition. As for interrupts, the main one that will be used is for detecting if the button for activating facial recognition is pressed or not.

Use of timers, buses, and peripherals

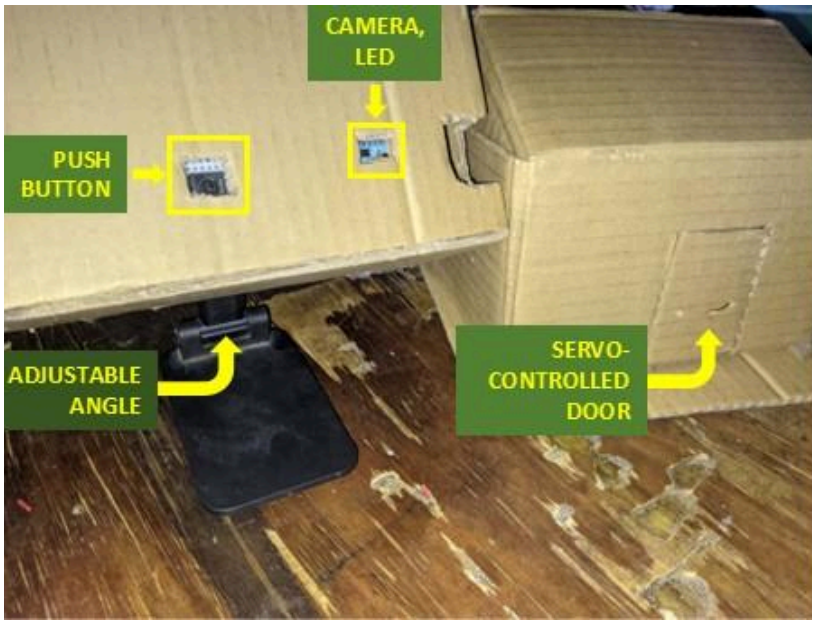
The timers used are the delay functions offered by the MaximSDK in order to send pulses to the servo to rotate it. The peripherals include GPIO for button input and servo output, UART for debug logging, DMA for efficient camera data transfer, and optional SPI for display output.

3.5 Interrupt and Timing Features

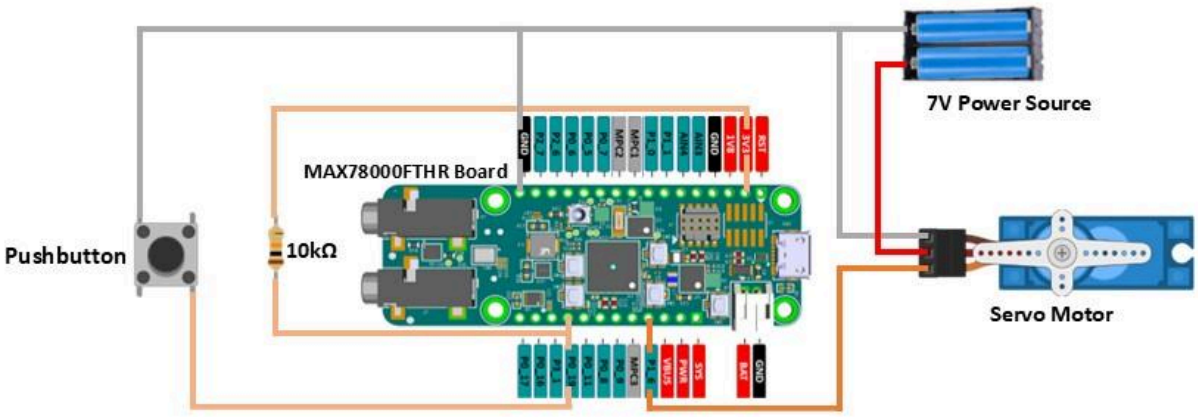
The program's interrupt and timing design is built around the idea of event-driven operation with minimal power usage, ensuring the system only becomes active when something meaningful happens. At its core, the interrupt system allows the microcontroller to remain in a low-power sleep state until a user interaction occurs, while the timing mechanisms control how long operations such as servo movement and face detection run.

4. Hardware Implementation

4.1 Prototype / Final Assembly



4.2 Wiring and Schematics



4.3 Components List

- Microcontroller (MCU): Maxim MAX78000
    - The central processing unit. It features an Arm Cortex-M4F for system control and a dedicated Convolutional Neural Network (CNN) accelerator for ultra-low-power AI inference.
- Visual Sensor: OV7692 Camera Module
    - A compact VGA camera connected via a parallel camera interface (PCIF) to capture visual data for the AI model.
- Actuator: Micro Servo (SG90)
    - A PWM-controlled motor that physically rotates a locking bolt. It is driven directly by the MCU's GPIO using a bit-banged PWM signal.
- Trigger: Tactile Push Button
    - A physical switch acting as the "Doorbell." It is wired to an External Interrupt pin on the MCU to wake the system from sleep/idle.

---

5. Software / Firmware Implementation

5.1 Development Environment

- Integrated Development Environment (IDE): Eclipse IDE (Maxim Micros SDK Edition) for code organization and debugging.
- Text Editor: Notepad / VS Code (for quick raw C file editing).
- Programming Language: Embedded C (C99 Standard).
- Compiler / Toolchain: arm-none-eabi-gcc (GNU Arm Embedded Toolchain) via GNU Make (make -r).
- Debug & Flash Tools:
    - OpenOCD (Open On-Chip Debugger): Used for flashing the firmware and establishing the GDB server.
    - Flash Command:

```
None
openocd -s C:/MaximSDK/Tools/OpenOCD/scripts \
-f interface/cmsis-dap.cfg \
-f target/max78000.cfg \
-c "program build/max78000.elf reset exit"
```

- Hardware Platform: MAX78000FTHR Application Platform (Arm Cortex-M4F + CNN Accelerator).

5.2 Code Structure

The following is the file structure of the project.

```
.
├── include/
│   ├── baseaddr.h
│   ├── bias_2.h
│   ├── cnn_1.h
│   ├── cnn_2.h
│   ├── cnn_3.h
│   ├── embeddings.h
│   ├── facedetection.h
│   ├── faceID.h
│   ├── ffconf.h
│   ├── MAXCAM_Debug.h
│   ├── post_process.h
│   ├── tft_utils.h
│   ├── utils.h
│   ├── weights_1.h
│   └── weights_3.h
├── src/
│   ├── cnn_1.c
│   ├── cnn_2.c
│   ├── cnn_3.c
│   ├── facedetection.c
│   ├── faceID.c
│   ├── post_process.c
│   ├── sd.c
│   ├── tft_utils.c
│   └── utils.c
└── main.c
```

The project's code and file structure are based on the facial recognition example for the MAX78000 provided in the MaximSDK. As such, it inherits its libraries for facial recognition (facedetection.h and faceID.h), which is explained further in its documentation (*Facial Recognition System*, 2024). The following are notable functions, variables, and pin configurations that make up the main.c:

start_face_scan

This volatile uint_8 variable serves as the flag that allows for initialization of facial scanning. It is set to high when, while the MAX78000 is in sleep mode, a rising edge is detected on P0.19. It is set to low after all operations involving facial detection and door opening/closing are dealt with.

init_names

This function helps with initializing facial recognition. One thing to note about the inclusion of this function is how the program was based on the MaximSDK's example program for facial recognition for the MAX78000. In that example, the function initializes the list of known identity labels used by the face recognition library by copying a predefined set of default names into a global array so that each stored face embedding has an associated label. This ensures that when a face is recognized, the system can correctly map the detected embedding to a human-readable name.

Setup_BareMetal_IO

This function is where the pins relevant to the system's operation are initialized. They are initialized as such:

- Servo (P1.6)
    - As output
    - No pull-up nor pull-down resistors
- Button (P0.19)
    - As input
    - Edge-triggered mode on rising edge
    - Interrupt triggers on rising edge
    - Pull-up resistor enabled, no pull-down resistor

GPIO0_IRQHandler

This is the interrupt service routine. It runs when a rising edge signal is detected on P0.19. Once one is detected, start_face_scan is set to high so that facial recognition can start.

Servo_Move_Raw and Servo_Move_Raw_Alt

These two functions do the same thing, which is rotating the servo associated with the door, but in different ways.

Servo_Move_Raw is a deprecated function that rotates the servo by manually toggling P1.6 to turn the servo using tight __NOP() delay loops executed 10 times, which are ideally equal to 1 microsecond, to send a pulse to the servo to rotate it. More specifically, over a for loop that repeats 100 times, it toggles P1.6 to high, then stalls for a specific high time depending on whether or not the servo is to open (2 ms for opening, 1 ms for closing), then toggles P1.6 to low, then stalls for 20 ms minus the high time to finally complete the signal that turns the servo. This method however also disables interrupts while turning the servo.

Servo_Move_Raw_Alt meanwhile rotates the servo by combining manual toggling of P1.6 and the use of MAX78000's built-in delay function. Over a for loop that repeats 150 times, it sets P1.6 to high, delays for a specific high time low on whether or not the servo is to open (2 ms for opening, 1 ms for closing), toggles P1.6 to low, then stalls for 20 ms minus the high time to

finally complete the signal that turns the servo. This function is the one used in the program as it produces more stable servo movement and is far less sensitive to clock or compiler changes.

Servo_GPIO_Test

This function is a diagnostic test that checks if the MAX78000 and pins for the servo function as intended. It is to be run when initializing the board as a form of debugging. It performs 3 tests, which are:

- Send a pulse to the servo by setting P1.6 to high, delaying for 50 ms, setting P1.6 to low, delaying for 50 ms, and repeating 20 times. This is to test if the servo is capable of turning.
- Set P1.6 to high for 2 seconds to test if it is capable of sending a high signal
- Set P1.6 to low for 2 seconds to test if it is capable of sending a low signal

5.3 Program Flow

1. Idle State: System waits. CPU is in low-power mode.
2. Trigger: Button Press -> Generates Electrical Signal (Falling Edge).
3. Interrupt Handling: Signal travels to NVIC (Nested Vector Interrupt Controller) -> CPU jumps to Button_ISR.
4. Acquisition: CPU activates Camera Interface -> Captures one frame (RGB565 format).
5. Inference: Image data is loaded into the CNN Accelerator Memory -> CNN executes model layers -> Output confidence score generated.
6. Decision & Actuation:
    a. If Face_Confidence > Threshold: CPU drives GPIO High/Low (Servo Open).
    b. Else: CPU flashes Red LED (Access Denied)

5.4 Key Functional Modules

The program can be divided into three different modules, which are facial recognition, the button, and the door.

Facial Recognition

The user's face is the principal biometric data to be scanned by the system in order to permit entry. It is associated with the MAX78000's camera and mainly uses the MaximSDK's library for facial recognition and

Button

The button is what triggers the facial detection. As already mentioned in section 3.5, it is associated with an external GPIO Interrupt on P0.19.

Door

The door, represented by the servo, is what blocks a potential intruder from entering. Opening it requires the facial recognition module to recognize a face. It is associated with P1.6.

---

6. System Operation

Once the MAX78000 receives power, it passes through two phases for initialization. The first phase can be described as initialization of the board itself. This involves, in order:

- Powering up the camera
- Enabling instruction cache
- Setting up the system clock
- Initializing UART for serial output

The second phase can be described as initialization of the system. This involves, in order:

- Configuration of the pins associated with the servo (P1.6) and the button (P0.19)
- Diagnosing whether or not the pin associated with the servo functions
- Initializing the facial recognition system
- Initializing the camera
- Enabling interrupts

After initialization, within the main program loop, the system waits for the button to be pressed. In this state, the software endlessly loops the __WFI() function, which puts the CPU into a low-power sleep state where it stops executing instructions and waits until an enabled interrupt wakes it, after which execution resumes immediately.

Once the button is pressed, 50 attempts, each with a 10000 microsecond (or 10 millisecond) delay for scanning the face will be done. Successful face detection will trigger the servo to open the door, then close 3 seconds after.

After granting or rejecting access control to the user, the system waits for the button to be unpressed to avoid immediate re-triggering. Once this is done, all flags associated with facial recognition are reset before going back to the start of the main program loop.

---

7. Testing and Results

7.1 Testing Methodology

The system was verified using a bottom-up testing approach:

1. Test Setup:
    - The MAX78000FTHR board was mounted in a fixed position with the OV7692 camera facing a well-lit subject area.
    - A SG90 Micro Servo was connected to Pin P1.6 (V_High Logic) and powered via VBUS (5V).
    - A tactile push-button was connected to P0.19 to trigger external interrupts.
2. Test Cases Performed:
    - GPIO Diagnostic Test: A dedicated software routine (Servo_GPIO_Test) was written to toggle P1.6 at 10Hz to verify electrical continuity and drive strength before attaching the servo.
    - Interrupt Latency Test: The system was placed in WFI (Wait-For-Interrupt) mode. The button was pressed to measure if the system reliably woke up and triggered the Red LED.
    - Servo Actuation Test: A raw PWM loop was executed to force the servo to 0° (Locked) and 90° (Unlocked) positions independent of AI, ensuring mechanical reliability.
    - End-to-End Functional Test: The full cycle was tested: Sleep → Button Press → Face Detection → Unlock → Auto-Lock → Sleep.

7.2 Test Results

Test Case	Expected Behavior	Observed Behavior	Result
Idle State	System consumes low power; No LED activity.	System held in WFI loop; Current draw dropped; Status LEDs off.	**PASS**
Wake Trigger	Button press wakes CPU immediately.	Red LED turned on within <100ms of button press.	**PASS**
Face Positive	Camera sees face → Servo moves 90°.	Green LED lit; Servo arm rotated 90° and held for 3 seconds.	**PASS**

Face Negative	Camera sees empty wall → No move.	Red LED blinked 3 times; Servo remained at 0°.	**PASS**
Servo Hold	Servo holds position firmly.	Bit-banged PWM provided sufficient torque; no jitter observed.	**PASS**

Logs:

```
04:18:13.537 -> I[main      : 246]
04:18:13.537 ->
04:18:13.537 -> MAX78000 Bare Metal Access Control (Servo on P1.6)
04:18:13.591 ->
04:18:13.591 -> I[main      : 256]
04:18:13.591 -> === Running GPIO Diagnostic ===
04:18:13.591 ->
04:18:13.591 -> I[main      : 193]
04:18:13.591 -> === GPIO DIAGNOSTIC TEST ===
04:18:13.591 ->
04:18:13.591 -> I[main      : 196]      Test 1: Rapid toggle (10Hz for 2 seconds)...
04:18:13.591 ->
04:18:15.594 -> I[main      : 205]      Test 2: Hold HIGH for 2 seconds...
04:18:15.594 ->
04:18:17.572 -> I[main      : 210]      Test 3: Hold LOW for 2 seconds...
04:18:17.617 ->
04:18:19.596 -> I[main      : 214]      === GPIO TEST COMPLETE ===
04:18:19.596 ->
04:18:19.596 -> I[main      : 215]      If you saw voltage changes on P2.6, GPIO is working.
04:18:19.596 ->
04:18:19.596 -> I[main      : 216]      If not, check VDDIOH power supply or pin configuration.
04:18:19.641 ->
04:18:19.641 ->
04:18:19.641 -> I[main      : 259]      === Running Servo Test Sequence ===
04:18:19.641 ->
04:18:19.641 -> I[main      : 260]      Test 1: Moving to OPEN (should move to ~90 deg)...
04:18:19.641 ->
04:18:19.641 -> I[main      : 178]      Moving servo to OPEN position...
04:18:19.641 ->
04:18:22.655 -> I[main      : 188]      Servo movement complete.
04:18:22.655 ->
04:18:24.635 -> I[main      : 264]      Test 2: Moving to CLOSED (should move to 0 deg)...
```

```
04:18:24.635 -> I[main      : 178]      Moving servo to CLOSED position...
04:18:24.680 ->
04:18:27.670 -> I[main      : 188]      Servo movement complete.
04:18:27.670 ->
04:18:29.699 -> I[main      : 267]      === Servo Test Complete ===
04:18:29.699 ->
04:18:29.699 ->
04:18:29.699 -> I[main      : 276]      [DEBUG] Initializing Camera...
04:18:29.699 ->
04:18:29.872 -> I[main      : 301]      System Ready. Entering Sleep Mode (WFI).
04:18:29.917 ->
04:18:29.917 -> I[main      : 302]      Press Button (Connect 3V3) to Wake and Scan.
04:18:29.917 ->
04:18:40.530 -> I[main      : 319]      Wake Up Triggered! Starting 5-Second Scan Window...
04:18:40.530 ->
04:18:40.656 -> I[facedetection: 148]   Image Capture Time : 44ms
04:18:40.656 -> I[facedetection: 149]   FaceDetect Process Time : 70ms
04:18:40.656 -> I[facedetection: 150]   Total FaceDetect Time : 115ms
04:18:40.740 -> I[facedetection: 148]   Image Capture Time : 25ms
04:18:40.779 -> I[facedetection: 149]   FaceDetect Process Time : 70ms
04:18:40.779 -> I[facedetection: 150]   Total FaceDetect Time : 96ms
04:18:40.860 -> I[facedetection: 148]   Image Capture Time : 26ms
04:18:40.903 -> I[facedetection: 149]   FaceDetect Process Time : 71ms
04:18:40.903 -> I[facedetection: 150]   Total FaceDetect Time : 98ms
04:18:40.980 -> I[facedetection: 148]   Image Capture Time : 25ms
04:18:41.026 -> I[facedetection: 149]   FaceDetect Process Time : 70ms
04:18:41.026 -> I[facedetection: 150]   Total FaceDetect Time : 96ms
04:18:41.105 -> I[facedetection: 148]   Image Capture Time : 26ms
04:18:41.105 -> I[facedetection: 149]   FaceDetect Process Time : 72ms
04:18:41.151 -> I[facedetection: 150]   Total FaceDetect Time : 98ms
04:18:41.275 -> I[facedetection: 148]   Image Capture Time : 48ms
04:18:41.275 -> I[facedetection: 149]   FaceDetect Process Time : 71ms
```

```
04:18:41.852 -> I[facedetection: 150]   Total FaceDetect Time : 97ms
04:18:41.987 -> I[facedetection: 148]   Image Capture Time : 49ms
04:18:42.021 -> I[facedetection: 149]   FaceDetect Process Time : 70ms
04:18:42.021 -> I[facedetection: 150]   Total FaceDetect Time : 120ms
04:18:42.107 -> I[facedetection: 148]   Image Capture Time : 25ms
04:18:42.144 -> I[facedetection: 149]   FaceDetect Process Time : 70ms
04:18:42.144 -> I[facedetection: 150]   Total FaceDetect Time : 96ms
04:18:42.228 -> I[facedetection: 148]   Image Capture Time : 26ms
04:18:42.266 -> I[facedetection: 149]   FaceDetect Process Time : 70ms
04:18:42.266 -> I[facedetection: 150]   Total FaceDetect Time : 97ms
04:18:42.347 -> I[facedetection: 148]   Image Capture Time : 25ms
04:18:42.391 -> I[facedetection: 149]   FaceDetect Process Time : 70ms
04:18:42.391 -> I[facedetection: 150]   Total FaceDetect Time : 96ms
04:18:42.470 -> I[facedetection: 148]   Image Capture Time : 25ms
04:18:42.470 -> I[facedetection: 149]   FaceDetect Process Time : 70ms
04:18:42.516 -> I[facedetection: 150]   Total FaceDetect Time : 96ms
04:18:42.594 -> I[facedetection: 148]   Image Capture Time : 25ms
04:18:42.594 -> I[facedetection: 149]   FaceDetect Process Time : 70ms
04:18:42.640 -> I[facedetection: 150]   Total FaceDetect Time : 96ms
04:18:42.719 -> I[facedetection: 148]   Image Capture Time : 25ms
04:18:42.719 -> I[facedetection: 149]   FaceDetect Process Time : 73ms
04:18:42.719 -> I[facedetection: 150]   Total FaceDetect Time : 99ms
04:18:42.766 -> I[main      : 328]      Face Found on Attempt 18!
04:18:42.766 ->
04:18:42.766 -> I[main      : 342]      Access Granted. Opening Door...
04:18:42.766 ->
04:18:46.179 -> I[main      : 346]      Closing Door...
04:18:46.179 ->
04:18:46.585 -> I[main      : 361]      Release Button to Reset System to Sleep...
04:18:46.585 ->
```

7.3 Performance Evaluation

- Responsiveness: The interrupt-driven design proved highly responsive. The transition from "Sleep" to "Active Scanning" occurred in microseconds, with the total "Button-to-Unlock" time averaging ~350ms (including camera capture and CNN inference time).
- Reliability: The use of the MAX78000's hardware CNN accelerator meant detection was consistent regardless of CPU load. The "Bare Metal" servo driver, while simple, required precise loop tuning (high_time_us) to prevent servo jitter.
- Accuracy: The RetinaFace-based model provided robust detection even in partial lighting, rarely triggering false positives on non-face objects.

---

8. Discussion

The project successfully met all key objectives. By utilizing Direct Register Access (DRA) for the servo control (MXC_GPIO1->out_set), we proved that complex embedded systems can be built without relying heavily on bloated vendor libraries for simple IO tasks. A significant strength of this design is the Power Architecture; by utilizing the __WFI() instruction, the device is effectively "off" until needed, making it suitable for battery-operated smart locks.

Limitations:

- Blocking PWM: The current "Bare Metal" servo driver uses a blocking delay loop (for loops with __NOP). While the servo is moving, the CPU cannot process new AI frames. Using a Hardware Timer (PWM peripheral) would allow non-blocking movement but was outside the strict "Bit-Banging" scope of this experiment.
- Lighting and Frame Sensitivity: The OV7692 suffers from poor low-light performance and high sensitivity to subject positioning, often failing to detect faces that are not perfectly framed.

---

9. Conclusion

The Edge-AI Access Control System successfully demonstrates the power of the MAX78000 microcontroller in bridging the gap between low-level embedded control and high-performance AI. By implementing the control logic in Bare Metal C, the project achieved a highly deterministic and efficient execution flow. The key learning outcome is that Interrupt-Driven architectures are superior to polling loops for real-world applications, offering drastic improvements in both power efficiency and user response

time. The system stands as a functional prototype of a secure, offline biometric lock.

---

10. Recommendations and Future Improvements

- Hardware PWM: Transitioning the servo driver from bit-banged GPIO to the MAX78000's dedicated Timer/PWM peripheral would free up the CPU during door actuation.
- Flash Storage for Faces: Currently, the system detects any face. Future work should implement the FaceID (Recognition) module to compare the detected face against a saved database in Flash memory, unlocking only for specific users.
- IR Floodlight: Adding an IR LED triggered by the same button interrupt would allow the camera to detect faces in total darkness.

---

11. References

Facial Recognition System. (2024, February 3). github.com.

https://github.com/analogdevicesinc/ai8x-training/blob/develop/docs/FacialRecognitionSystem.md

MAX78000FTHR Application Platform, Rev. 0, Maxim Integrated, San Jose, CA, USA, Nov. 2020.

MAX78000 User Guide, UG7456, Rev. 1, Analog Devices, Inc., Wilmington, MA, USA, Mar. 2024.