Smart Parking Garage Controller (Two DMC8 CPUs + Handshaking, DEEDS)

https://github.com/mahfoudh-sidi/deeds-smart-parking/tree/main/project

Course: **Microprocessors and Microcontrollers**
Instructor: Dr. **AHMET GÜRHANLI**
Date: 05 / 01/ 2026

Full Name	Student ID	Signature
Yousuf Jamal Abdurahman	22040102185	
MOHAMMED BELHAIBA	22040102094	
WAEL BENALI	22040102050	
MOHAMED AL MOHAMED ALNASER	22040102155	
Emmanuel Ikechukwu Chukwu	22040102198	
Mahfoudh Sidi oumar	22040102171	

Project Brief

This project implements a Smart Parking Garage Controller in DEEDS using DMC8 assembly. The garage capacity is 5 cars. The system detects entry/exit requests, updates the car count, controls status indicators, and transfers the count to a second processor for display. The design is implemented as a two-CPU system to demonstrate inter-processor communication.

System Architecture (Two CPUs)

CPU_A (Controller): reads sensors, applies parking rules, drives LEDs, and transmits the current count.
CPU_B (Display): receives the count and shows it on a one-digit display. This separation reflects real systems where the control unit and display unit are independent.

Components and Connections

Inputs (to CPU_A):

- **S_IN (entry) → CPU_A IA.0**
- **S_OUT (exit) → CPU_A IA.1**

Outputs (from CPU_A):

- **GATE LED → OC.0 (short pulse on a valid entry/exit)**
- **FULL LED → OC.1 (ON when count = 5)**
- **BLOCKED LED → OC.2 (ON when entry is attempted while full)**

Display (from CPU_B):

- **One Hex Digit display → CPU_B OC (low nibble) showing 0..5**

Inter-CPU Communication

The system uses polling to monitor sensors and handshake lines, and it uses a bidirectional STB/ACK handshaking protocol for reliable CPU-to-CPU communication.

The output of CPU_A is used as the input of CPU_B through a parallel bus and two control lines:

- **Data: CPU_A OA[7..0] → CPU_B IA[7..0]**
- **STB (strobe): CPU_A OB.0 → CPU_B IB.0**
- **ACK (acknowledge): CPU_B OB.0 → CPU_A IB.0**

Protocol summary: CPU_A places the count on OA and raises STB. CPU_B detects STB, reads the count, updates the display, then raises ACK. CPU_A clears STB, and CPU_B clears ACK. This prevents missed/overwritten data.

Software Behavior (Summary)

CPU_A:

- **Polls IA.0/IA.1 and uses edge detection so each press is counted once.**
- **Maintains COUNT in the range 0..5.**
- **Valid entry: COUNT++, gate pulse.**
- **Valid exit: COUNT--, gate pulse.**
- **At COUNT=5: FULL ON.**
- **Entry attempt while full: BLOCKED ON; count does not increase.**
- **After each event, sends COUNT to CPU_B using the handshake.**

CPU_B:

- **Waits for STB, reads COUNT from IA, outputs it to the one-digit display, then ACKs.**

Testing and Expected Output

1. **Entry test: toggle S_IN repeatedly → display: 0→1→2→3→4→5, GATE pulses each time.**
2. **Full condition: at 5 → FULL LED stays ON.**
3. **Entry while full: press S_IN → BLOCKED LED turns ON; display remains 5; entry is denied.**
4. **Exit test: press S_OUT → display decreases (5→4→…), GATE pulses; FULL turns OFF when count < 5.**