1. Project Title and Team Information

Project Title: SmartPark: An Automated Parking Guidance and Slot Display System

Team Members:

- Noe S Setenta Jr (noejr.setenta@g.msuiit.edu.ph)
- Jah Isaac Cagula (jahisaac.cagula@g.msuiit.edu.ph)

Github Repository:
https://github.com/MrNoe70/SmartPark-An-Automated-Parking-Guidance-and-Slot-Display-System/blob/main/MileStone_2/Milestone%202%20Report%20Document.pdf

---

2. Updated Project Description

SmartPark is a real-time parking management system that automatically detects vehicle occupancy in six parking slots using ultrasonic sensors, displays slot status on an LCD, and provides visual feedback via individual LEDs. This enables drivers to quickly identify available slots, reducing the time spent searching for parking, minimizing congestion, and enhancing user convenience.

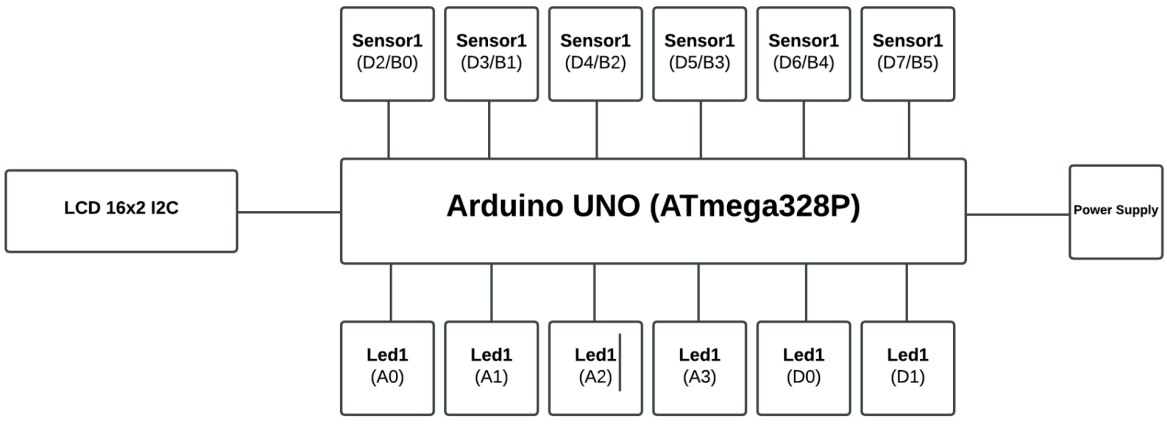
Changes from MileStone 1:

- Expanded from 1 to 6 ultrasonic sensors testing
- Added individual LED indicators for each slot
- Implemented Finite State Machine (FSM) for each parking slot

---

3. System Architecture (Updated)

Block Diagram:



Process Flow:

1. Initialize LCD, sensors, and LEDs.

2. Trigger all sensors and wait for echo pulses.

3. Calculate distance to detect car presence.

4. Update LEDs and LCD in real-time.

5. Display “FULL PARKING” when all slots are occupied.

Key System Features:

- **MCU:** ATmega328P (Arduino Uno)
- **Sensors:** 6× HC-SR04 Ultrasonic (Trigger: D2-D7, Echo: D8-D13)
- **Actuators:** 6× LEDs (A0-A3, D0-D1), 16×2 LCD with I2C
- **Communication:** I2C for LCD, Pin Change Interrupts for sensors
- **Timers:** Timer1 for microsecond pulse timing (0.5μs resolution)

---

4. Current Progress Summary

Implemented (90% Complete):

1. All 6 ultrasonic sensor drivers with interrupt-based timing
2. Individual LED indicators for each parking slot
3. LCD display with real-time slot status updates
4. Finite State Machine for each parking slot
5. "FULL PARKING" display mode
6. Hardware-Software integration

Remaining Task (10%):

1. Final enclosure design

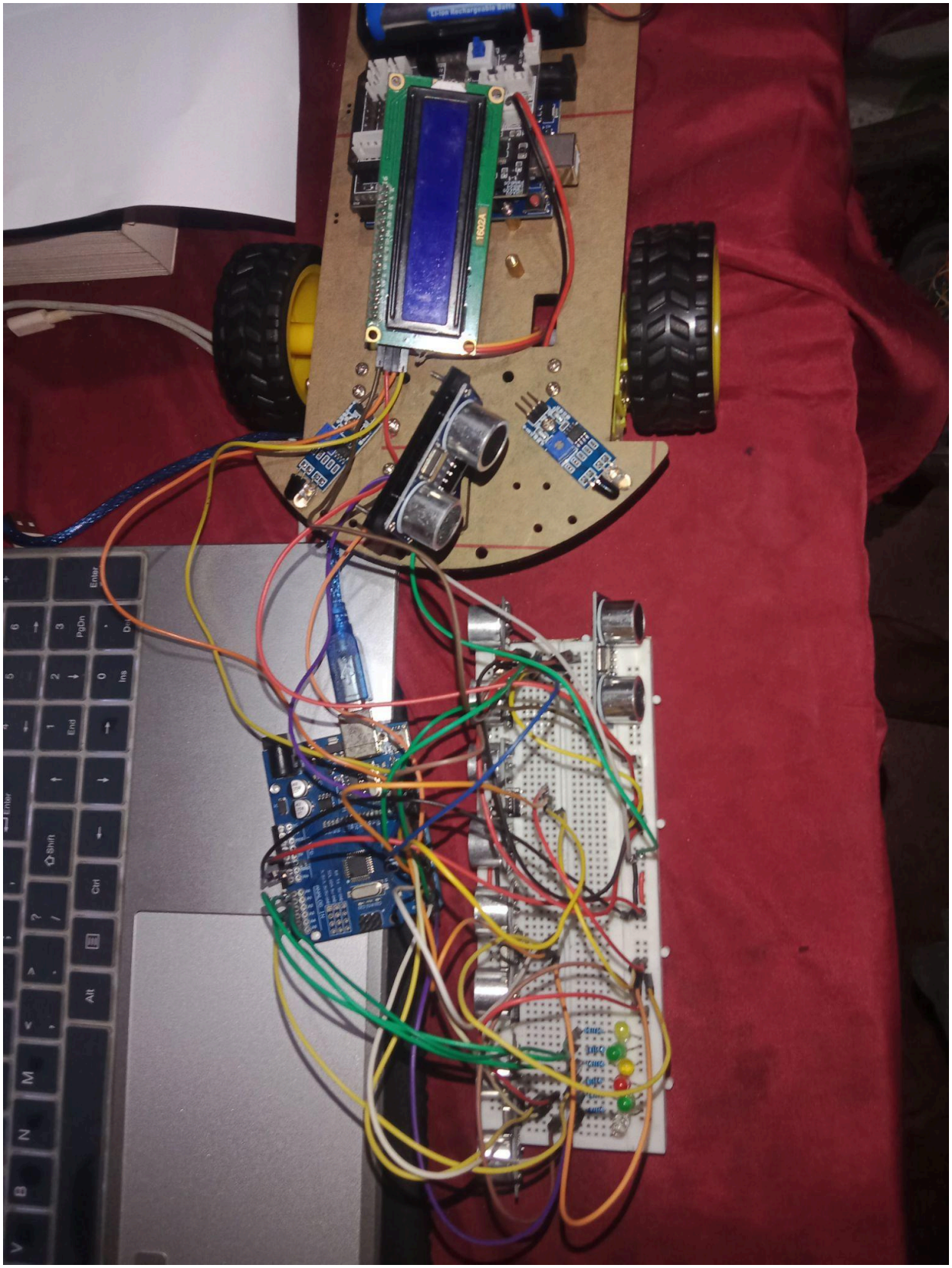
Integration Progress:

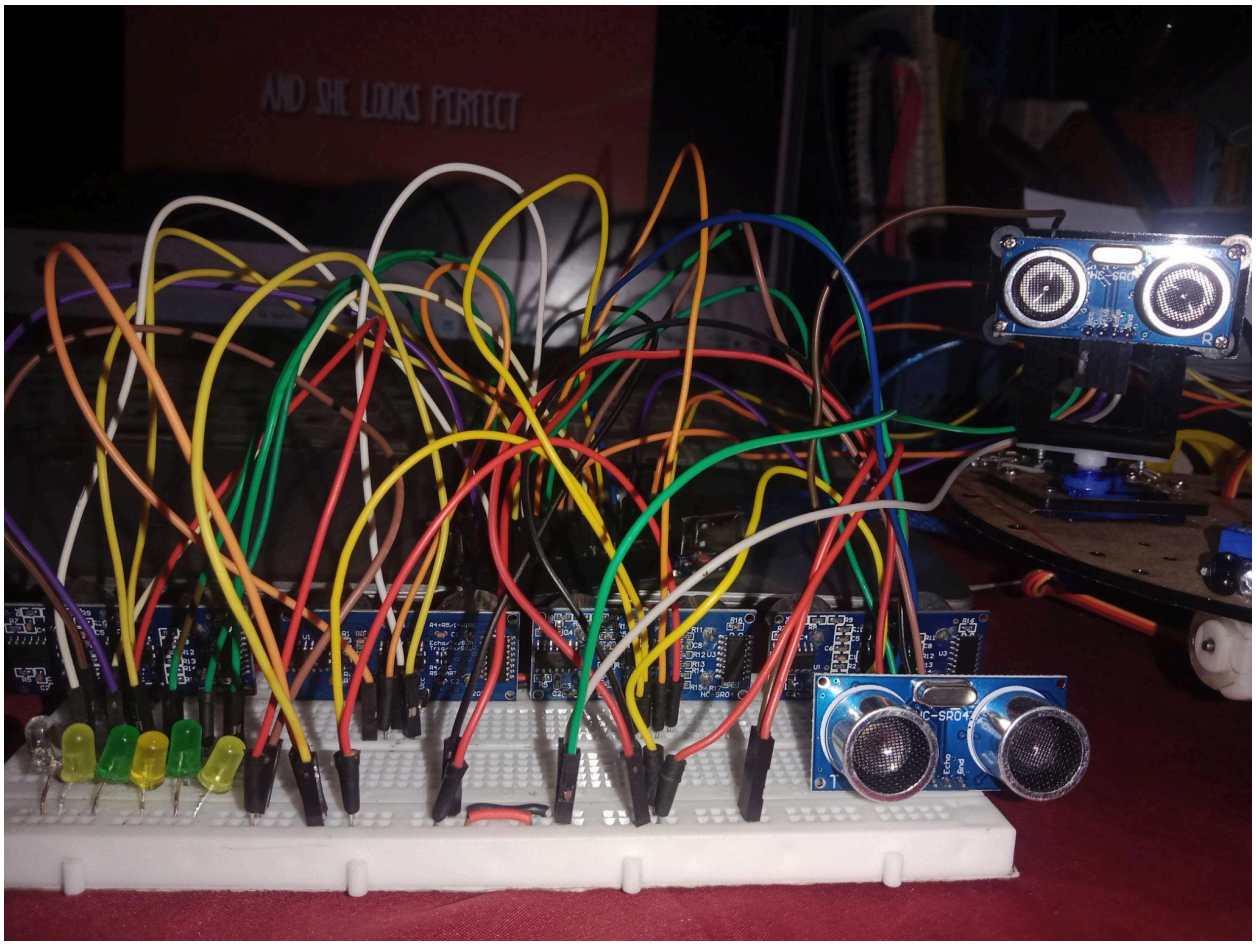
- Hardware assembly: 100% complete
- Firmware development: 90% complete
- System integration: 85% complete

---

5. Hardware Implementation (So Far)

Initial Prototype Photos:





Wiring Diagrams:

ULTRASONIC SENSORS:

Sensor 1: Trig=D2, Echo=D8, LED=A0

Sensor 2: Trig=D3, Echo=D9, LED=A1

Sensor 3: Trig=D4, Echo=D10, LED=A2

Sensor 4: Trig=D5, Echo=D11, LED=A3

Sensor 5: Trig=D6, Echo=D12, LED=D0

Sensor 6: Trig=D7, Echo=D13, LED=D1

LCD (I2C 16x2):

SDA=A4, SCL=A5

POWER:

All sensors: VCC=5V, GND=GND

LEDs: 2000Ω resistors in series

Components List:

1. Arduino Uno R3 (ATmega328P) ×1
2. HC-SR04 Ultrasonic Sensors ×6
3. LEDs ×6
4. 2000Ω Resistors ×6
5. 16×2 LCD with I2C ×1
6. Breadboard ×1
7. Jumper wires ×40
8. 5V Power supply ×1

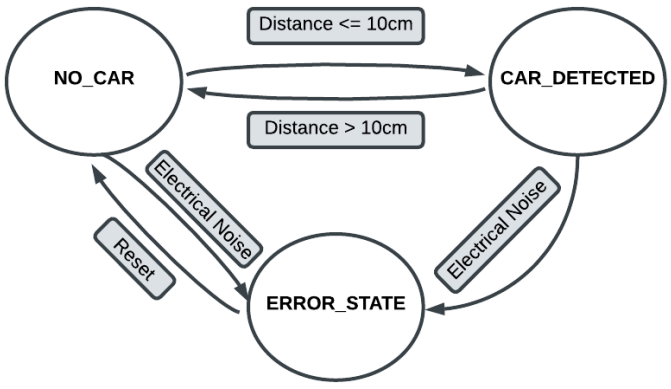
---

6. Software / Firmware Implementation

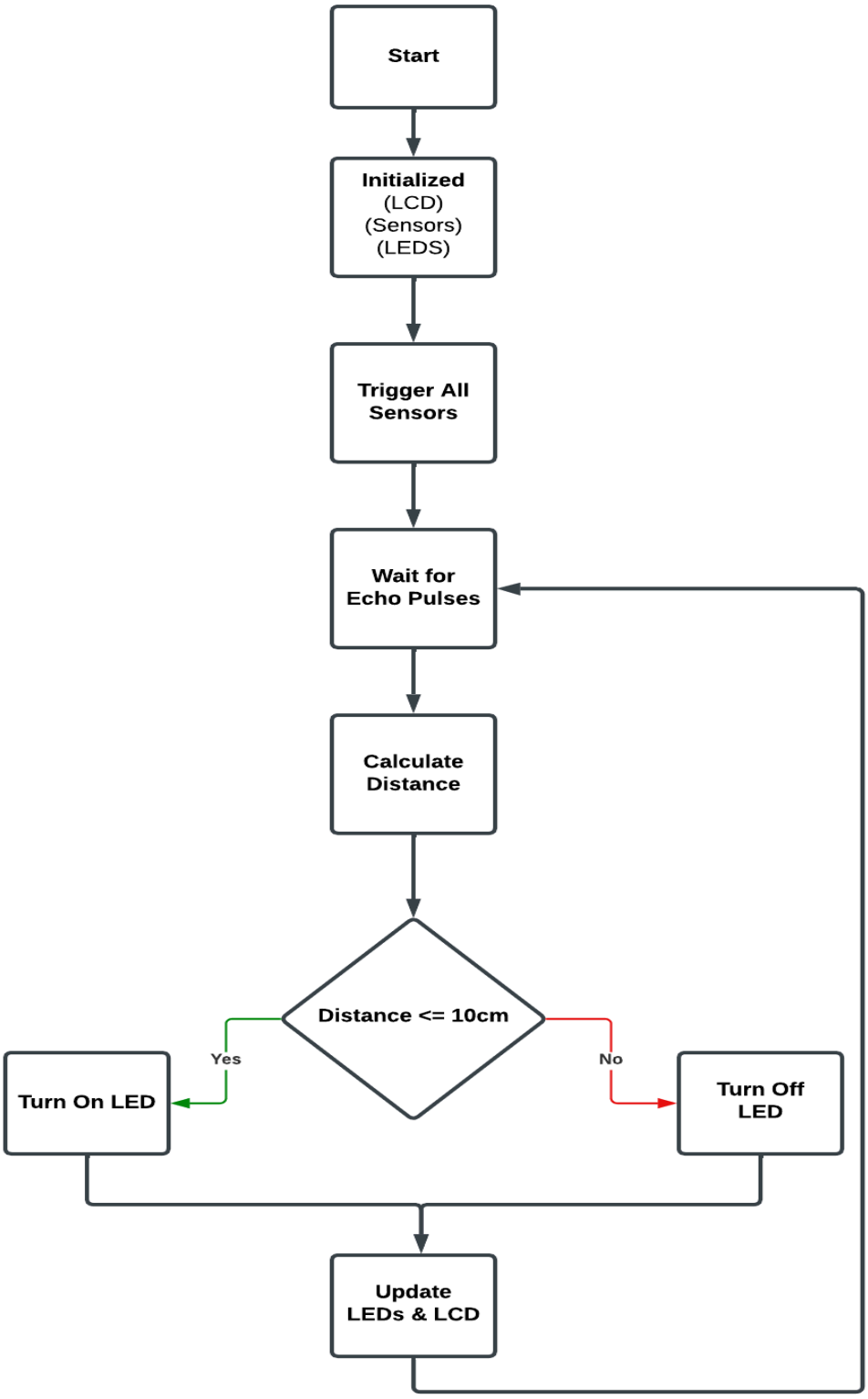
Code Structure:

```
src/
├── main.c              # Main application with FSM
├── ultrasonic.c/h      # 6-sensor driver with interrupts
├── gpio.c/h            # GPIO abstraction layer
├── lcd.c/h             # I2C LCD driver
└── Makefile            # Build configuration
```

Finite State Machine:



Flowchart:


Start
Initialized
(LCD)
(Sensors)
(LEDS)
Trigger All Sensors
Wait for Echo Pulses
Calculate Distance
Distance <= 10cm
Yes
No
Turn On LED
Turn Off LED
Update LEDs & LCD

Key Implemented Modules:

1. **Interrupt-Driven Ultrasonic Timing:**

```
// Configure Timer1 for microsecond timing (prescaler 8)
TCCR1A = 0;
TCCR1B = (1 << CS11); // Prescaler = 8 (0.5µs per tick at 16MHz)

// Configure Pin Change Interrupts for all sensors (PORTB)
PCICR |= (1 << PCIE0);                    // Enable PCINT0_vect (PORTB)
PCMSK0 |= (1 << PCINT0) |                 // PB0 (Sensor 1)
          (1 << PCINT1) |                 // PB1 (Sensor 2)
          (1 << PCINT2) |                 // PB2 (Sensor 3)
          (1 << PCINT3) |                 // PB3 (Sensor 4)
          (1 << PCINT4) |                 // PB4 (Sensor 5) - D12
          (1 << PCINT5);                  // PB5 (Sensor 6) - D13
```

2. **Finite State Machine:**

```
// FSM States for Each Slot
typedef enum {
    STATE_NO_CAR,
    STATE_CAR_DETECTED,
    STATE_ERROR
} ParkingState_t;
```

NO_CAR → CAR_DETECTED when distance ≤ 10cm

CAR_DETECTED → NO_CAR when distance > 10cm

ERROR_STATE for invalid readings

3. **Real-time LCD Updates:**

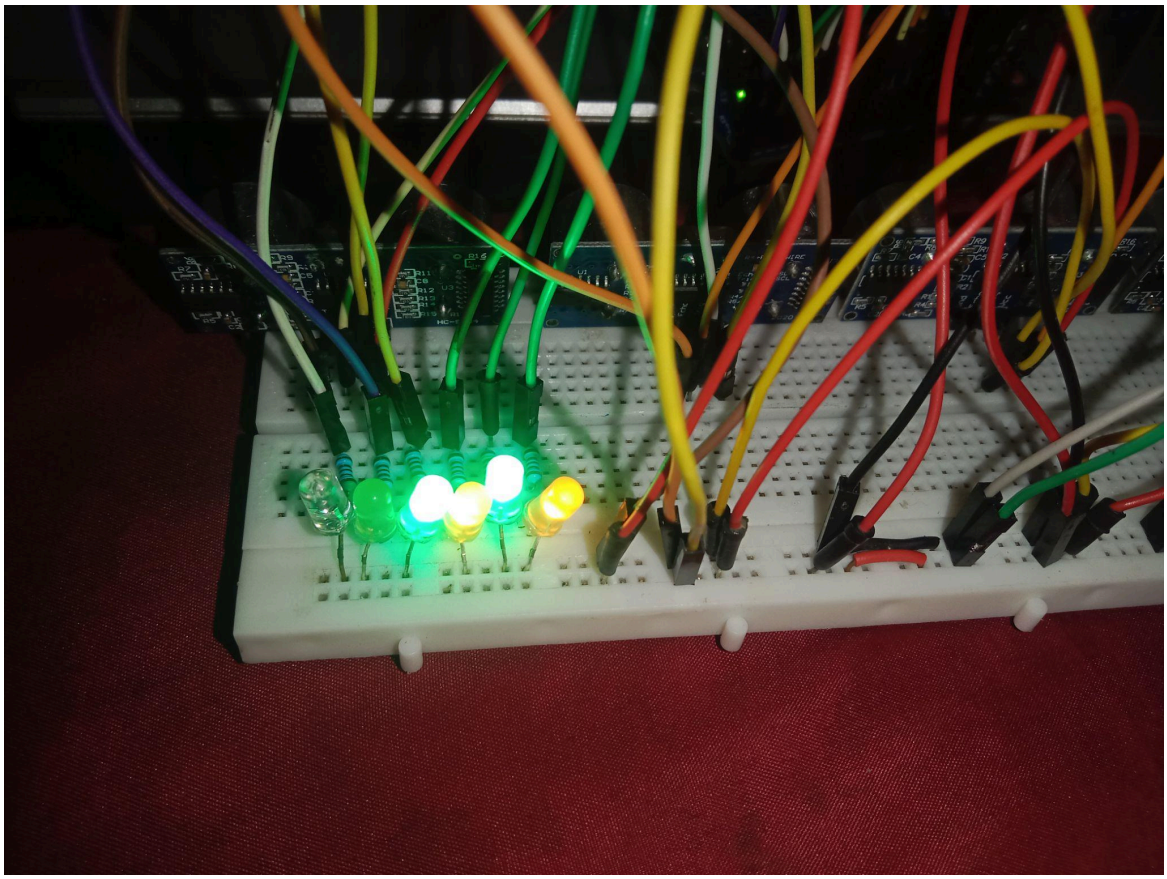
```
// Display format:
// Line 1: P0:0 P1:0 P2:0
// Line 2: P3:0 P4:0 P5:0
// 0 = Empty, 1 = Occupied
```

Working System Screenshots:

1. LCD showing "P0:1 P1:0 P2:0 / P3:0 P4:1 P5:0"

2. LCD showing "FULL PARKING"

3. LEDs illuminated for occupied slots



7. Testing and Initial Results

What Already Works Reliably:

1 .All 6 Ultrasonic Sensor:

- **Individual Detection:** Each sensor detects objects within 10cm range
- **Simultaneous Operation:** All 6 sensors trigger and measure simultaneously
- **Accurate Timing:** Interrupt-driven measurements with 0.5µs resolution
- **Consistent Response:** <200ms from detection to system update

2. All LED Indicator System:

- **Individual Control:** Each LED independently controlled
- **Instant Response:** Lights within 150ms of detection
- **Error Indication:** Blinking LED for sensor errors (tested)

3. LCD Display System:

- **Real-time Updates:** Instantly shows P0:0 → P0:1 when car detected
- **"FULL PARKING" Mode:** Automatically switches when all slots occupied
- **Clear Visibility:** Readable from 2+ meters distance

4. Finite State Machine (FSM):

- **6 Independent FSMs:** Each slot has its own state machine
- **Correct Transitions:** NO_CAR ↔ CAR_DETECTED based on distance
- **Error State:** Properly enters STATE_ERROR on invalid readings
- **State Persistence:** Maintains state between measurement cycles

5. Interrupt System:

- **Pin Change Interrupts:** All 6 echo pins (D8-D13) trigger interrupts
- **Timer1 Operation:** 0.5µs timing resolution working correctly
- **ISR Handling:** Rising/falling edge detection accurate
- **No Interrupt Conflicts:** I2C and PCINT work together without issues

Issues Encountered and Solutions:

Issue: LCD updates Too Slow

Solution: I changed to event-driven updates

Issue: C99 Compilation Errors

Solution: I added -std=gnu99 in the Makefile

Issue: Shared Pins for LCD I2C and LEDs

Solution: I move the LEDs to difference pins

Show that your system works:

- Results of tests you already performed
- Screenshots, logs, or sensor output
- What already works reliably
- Issues encountered and how you plan to fix them

---

8. Remaining Tasks Before Final Submission

FInal Polishing Steps:

1. Build final enclosure
2. Code clean up and optimization
3. Complete final report
4. Prepare presentation materials

---

9. Appendix (Optional)

Future Enhancement:

1. Wireless status reporting via Bluetooth
2. Mobile app interface
3. Parking history logging
4. Solar power capability
5. Weather-resistant enclosure