```
#include <ESP32Servo.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <WiFi.h>
#include <FirebaseESP32.h>

// LCD Configuration
LiquidCrystal_I2C lcd(0x27, 16, 2); // 16x2 display
Servo gateServo;

// WiFi Credentials
#define WIFI_SSID "YOUR_WIFI_SSID"
#define WIFI_PASSWORD "YOUR_WIFI_PASSWORD"

// Firebase Configuration
#define FIREBASE_HOST "YOUR_PROJECT_ID.firebaseio.com"
#define FIREBASE_AUTH "YOUR_DATABASE_SECRET"

// ESP32-WROVER Pin Assignments
#define gateEntrySensor 15  // GPIO15
#define gateExitSensor  13  // GPIO13
#define ir_car1 12        // GPIO12
#define ir_car2 14        // GPIO14
#define ir_car3 27        // GPIO27
#define ir_car4 26        // GPIO26
#define servoPin 18       // GPIO18
```

```
// Constants
const int gateOpenTime = 3000;       // Time gate stays open in ms
const int messageDisplayTime = 2000; // Time to display messages
const int firebaseUpdateInterval = 5000; // Update Firebase every 5 seconds

// Variables
int s1 = 0, s2 = 0, s3 = 0, s4 = 0;
unsigned long lastDetectionTime = 0;
bool gateOpen = false;
String currentMessage = "";
bool lastEntryState = false;
bool lastExitState = false;
bool parkingFull = false;
unsigned long lastDisplayUpdate = 0;
unsigned long lastFirebaseUpdate = 0;

// Firebase Data Object
FirebaseData firebaseData;

void setup() {
  Serial.begin(115200);

  // Initialize WiFi
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
```

```
lcd.print("Connecting WiFi");

while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
  lcd.print(".");
}

lcd.clear();
lcd.setCursor(0, 0);
lcd.print("WiFi Connected!");
lcd.setCursor(0, 1);
lcd.print(WiFi.localIP());
delay(2000);

// Initialize Firebase
Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
Firebase.reconnectWiFi(true);

// Gate control sensors
pinMode(gateEntrySensor, INPUT_PULLUP);
pinMode(gateExitSensor, INPUT_PULLUP);

// Parking slot sensors
pinMode(ir_car1, INPUT_PULLUP);
pinMode(ir_car2, INPUT_PULLUP);
pinMode(ir_car3, INPUT_PULLUP);
```

```
  pinMode(ir_car4, INPUT_PULLUP);

  // Servo initialization
  gateServo.attach(servoPin);
  gateServo.write(90);  // Initial position (closed)

  // LCD initialization
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Car Parking Sys");
  delay(2000);

  // Initial sensor reading
  updateSensorStates();
  checkParkingFull();
  displayStatus();
  updateFirebase(); // Initial Firebase update
}

void loop() {
  updateSensorStates();
  checkParkingFull();

  // Check gate sensors
  int entryDetected = !digitalRead(gateEntrySensor);  // LOW when detected
  int exitDetected = !digitalRead(gateExitSensor);    // LOW when detected
```

```
  // Detect vehicle approach or departure
  if (entryDetected && !lastEntryState) {
    if (!parkingFull) {
      openGate("Welcome!");
    } else {
      currentMessage = "Parking Full!";
      lastDetectionTime = millis();
    }
  }
  else if (exitDetected && !lastExitState) {
    openGate("Goodbye!");
  }

  // Close gate if timeout reached
  if (gateOpen && (millis() - lastDetectionTime > gateOpenTime)) {
    closeGate();
  }

  // Clear message after display time
  if (currentMessage.length() > 0 && !gateOpen && (millis() - lastDetectionTime >
messageDisplayTime)) {
    currentMessage = "";
  }

  // Update display every 500ms
  if (millis() - lastDisplayUpdate > 500) {
    displayStatus();
    lastDisplayUpdate = millis();
```

```
  }

  // Update Firebase every firebaseUpdateInterval
  if (millis() - lastFirebaseUpdate > firebaseUpdateInterval) {
    updateFirebase();
    lastFirebaseUpdate = millis();
  }

  lastEntryState = entryDetected;
  lastExitState = exitDetected;
  delay(50);
}

void openGate(String message) {
  gateServo.write(180);  // Open gate (180 degrees)
  gateOpen = true;
  currentMessage = message;
  lastDetectionTime = millis();
  updateFirebase(); // Immediate update when gate opens
}

void closeGate() {
  gateServo.write(90);  // Close gate (90 degrees)
  gateOpen = false;
  currentMessage = "";
  updateFirebase(); // Immediate update when gate closes
}
```

```
void updateSensorStates() {
  s1 = !digitalRead(ir_car1);
  s2 = !digitalRead(ir_car2);
  s3 = !digitalRead(ir_car3);
  s4 = !digitalRead(ir_car4);
}

void checkParkingFull() {
  parkingFull = (s1 && s2 && s3 && s4);
}

void displayStatus() {
  lcd.clear();

  // First line: Slot status
  lcd.setCursor(0, 0);
  lcd.print("1:");
  lcd.print(s1 ? "Full " : "Empty");

  lcd.setCursor(8, 0);
  lcd.print("2:");
  lcd.print(s2 ? "Full" : "Empty");

  // Second line: Continue slots or show message
  if (currentMessage.length() > 0) {
    lcd.setCursor(0, 1);
```

```
    lcd.print(currentMessage);
  } else {
    lcd.setCursor(0, 1);
    lcd.print("3:");
    lcd.print(s3 ? "Full " : "Empty");

    lcd.setCursor(8, 1);
    lcd.print("4:");
    lcd.print(s4 ? "Full" : "Empty");
  }
}

void updateFirebase() {
  if (WiFi.status() == WL_CONNECTED && Firebase.ready()) {
    // Update individual slots
    Firebase.setInt(firebaseData, "/parking/slots/slot1", s1);
    Firebase.setInt(firebaseData, "/parking/slots/slot2", s2);
    Firebase.setInt(firebaseData, "/parking/slots/slot3", s3);
    Firebase.setInt(firebaseData, "/parking/slots/slot4", s4);

    // Update summary information
    int freeSlots = (!s1) + (!s2) + (!s3) + (!s4);
    Firebase.setInt(firebaseData, "/parking/summary/freeSlots", freeSlots);
    Firebase.setInt(firebaseData, "/parking/summary/totalSlots", 4);
    Firebase.setBool(firebaseData, "/parking/summary/isFull", parkingFull);

    // Update gate status
```

```
    Firebase.setBool(firebaseData, "/parking/gate/isOpen", gateOpen);

    // Update last update timestamp
    Firebase.setString(firebaseData, "/parking/lastUpdate", getTimeStamp());
  } else {
    Serial.println("Firebase update failed - WiFi or Firebase not ready");
  }
}

String getTimeStamp() {
  // Get current time (simplified version)
  unsigned long now = millis();
  unsigned long seconds = now / 1000;
  unsigned long minutes = seconds / 60;
  unsigned long hours = minutes / 60;

  char timeStr[20];
  sprintf(timeStr, "%02d:%02d:%02d", hours % 24, minutes % 60, seconds % 60);
  return String(timeStr);
}
```