

Parking Management in a Mall

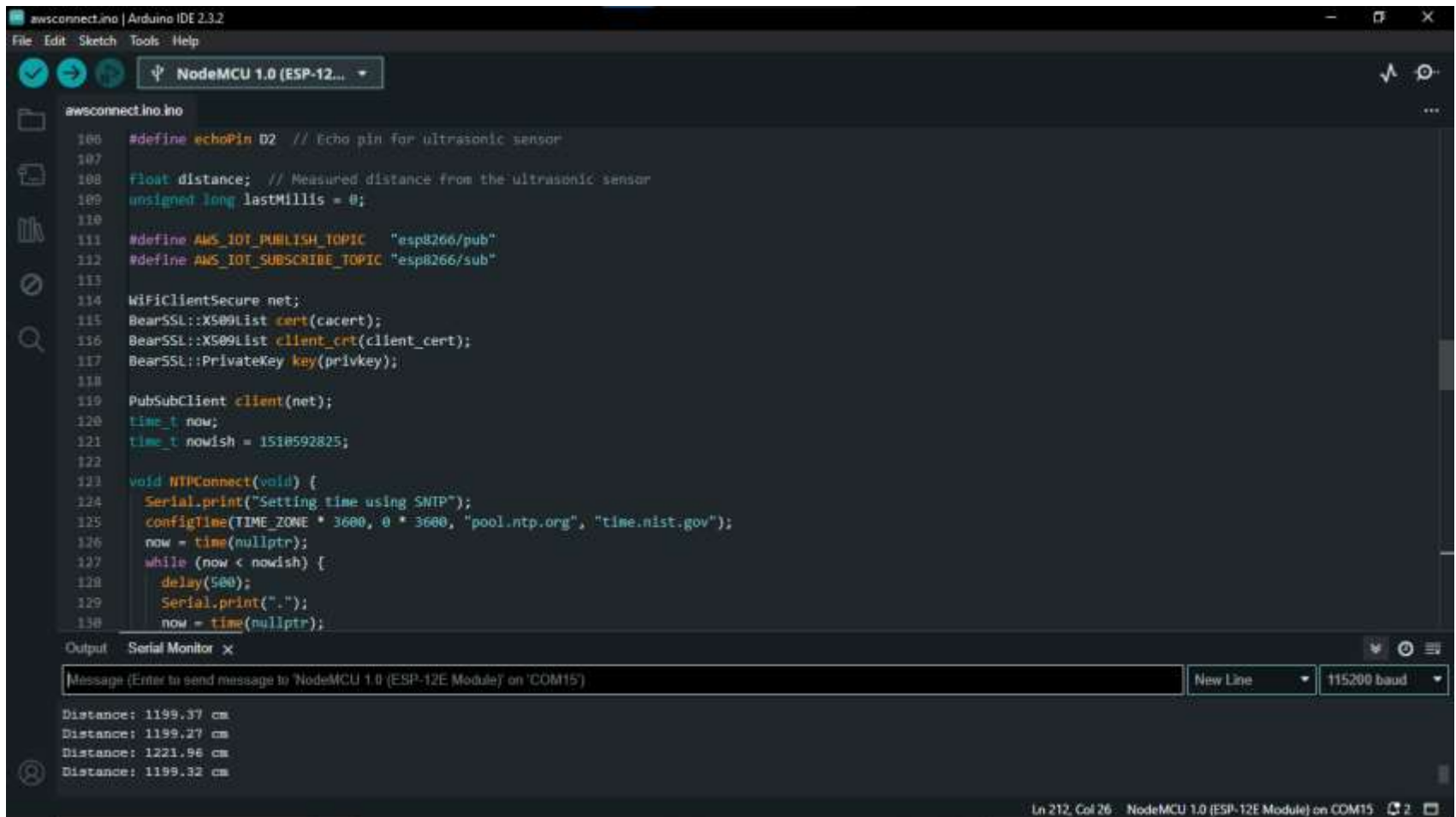
Objective

- Detects parking space availability using IoT sensors.
- Provides real-time updates of parking spots through a web application.
- Displaying the availability .
- Integrates AWS services such as DynamoDB, Lambda, and IoT Core for backend management

SYSTEM ARCHITECTURE

1. IoT Devices (Ultrasonic Sensors): These sensors detect whether a parking spot is occupied or available.
2. AWS IoT Core: Facilitates the communication between IoT devices and the cloud infrastructure.
3. AWS Lambda: Executes the serverless backend logic, including sensor data handling, database interactions, and API responses.
4. DynamoDB: A NoSQL database used to store parking space data, including real-time sensor status and reservation information.
5. API Gateway: Exposes API endpoints for the web application to interact with the backend.
6. Web Application (Frontend): Displays real-time parking information

Arduino IDE program for Esp8266



```
awsconnect.ino | Arduino IDE 2.3.2
File Edit Sketch Tools Help
NodeMCU 1.0 (ESP-12...)
awsconnect.ino
106 #define echoPin D2 // Echo pin for ultrasonic sensor
107
108 float distance; // Measured distance from the ultrasonic sensor
109 unsigned long lastMillis = 0;
110
111 #define AWS_IOT_PUBLISH_TOPIC "esp8266/pub"
112 #define AWS_IOT_SUBSCRIBE_TOPIC "esp8266/sub"
113
114 WiFiClientSecure net;
115 BearSSL::X509List cert(cacert);
116 BearSSL::X509List client_cert(client_cert);
117 BearSSL::PrivateKey key(privkey);
118
119 PubSubClient client(net);
120 time_t now;
121 time_t nowish = 1518592825;
122
123 void NTPConnect(void) {
124   Serial.print("Setting time using NTP");
125   configTime(TIME_ZONE * 3600, 0 * 3600, "pool.ntp.org", "time.nist.gov");
126   now = time(nullptr);
127   while (now < nowish) {
128     delay(500);
129     Serial.print(".");
130   }
131   now = time(nullptr);
132 }

Output Serial Monitor x
Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM15') New Line 115200 baud
Distance: 1199.37 cm
Distance: 1199.27 cm
Distance: 1221.96 cm
Distance: 1199.32 cm
Ln 212, Col 26 NodeMCU 1.0 (ESP-12E Module) on COM15
```

Aws IOT Core - MQTT

The screenshot displays the AWS IoT console interface for the 'eu-north-1' region. The left sidebar contains navigation links for 'Monitor', 'Connect', 'Test', and 'Manage'. The 'Test' section is active, showing the 'MQTT test client' option. The main content area features a 'Subscribe' button at the top. Below it, the 'Subscriptions' section shows a subscription for 'esp8266/pub'. To the right of this subscription are buttons for 'Pause', 'Clear', 'Export', and 'Edit'. The 'Message payload' field contains a JSON object:

```
{  "message": "Hello from AWS IoT console"}
```

. Below the payload field is an 'Additional configuration' section with a 'Publish' button. At the bottom, a log entry for the 'esp8266/pub' subscription shows a timestamp of 'October 05, 2024, 21:56:47 (UTC+0530)' and a JSON response:

```
{  "spaceID": "1",  "status": "available",  "time": 287256}
```

. The footer of the console includes 'CloudShell', 'Feedback', and copyright information for Amazon Web Services.

AWS IoT

Monitor

Connect

- Connect one device
- ▶ Connect many devices
- Domain configurations [Updated](#)

Test

[MQTT test client](#)

Manage

- ▶ All devices
- Software packages [New](#)
- ▶ Remote actions
- ▶ Message routing
- Retained messages

Subscriptions **esp8266/pub** Pause Clear Export Edit

esp8266/pub ♥ ✕

Message payload

```
{  "message": "Hello from AWS IoT console"}
```

▶ Additional configuration

Publish

▼ esp8266/pub October 05, 2024, 21:56:47 (UTC+0530)

```
{  "spaceID": "1",  "status": "available",  "time": 287256}
```

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Lambda – parking data

The screenshot displays the AWS Lambda console for the function **ParkingDataProcessor** in the **eu-north-1** region. The breadcrumb navigation shows **Lambda > Functions > ParkingDataProcessor**. The function name **ParkingDataProcessor** is prominently displayed at the top, with buttons for **Throttle**, **Copy ARN**, and **Actions**.

The **Function overview** section is active, showing a **Diagram** tab. The diagram illustrates the function's architecture: **ParkingDataProcessor** is connected to **AWS IoT** and **API Gateway** triggers. Below the diagram are buttons for **+ Add trigger** and **+ Add destination**. A **Layers** section shows **(0)** layers attached.

On the right, a metadata panel provides details:

- Description**: -
- Last modified**: 7 days ago
- Function ARN**: `arn:aws:lambda:eu-north-1:311141566481:function:ParkingDataProcessor`
- Function URL**: [Info](#)

The bottom navigation bar includes **Code**, **Test**, **Monitor**, **Configuration**, **Aliases**, and **Versions**. The footer contains **CloudShell**, **Feedback**, and copyright information for Amazon Web Services, Inc. or its affiliates, along with links for **Privacy**, **Terms**, and **Cookie preferences**.

DynamoDB – for Parking Data

The screenshot displays the AWS Management Console for DynamoDB. The left-hand navigation menu includes options like Dashboard, Tables, Explore items (selected), PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations, Reserved capacity, Settings, DAX, Clusters, Subnet groups, Parameter groups, and Events. The top navigation bar shows the AWS logo, Services, a search bar, and the user's profile. The main content area is titled 'DynamoDB' and shows the 'Explore items' view for the 'ParkingSpaces' table. A blue banner at the top reads 'Introducing Favorite Tables with Amazon DynamoDB'. Below this, the 'Tables (2)' section lists 'ParkingReservations' and 'ParkingSpaces' (selected). The 'Scan or query items' section shows a status message: 'Completed, Read capacity units consumed: 0.5'. The 'Items returned (1)' section displays a table with one item:

spaceID (String)	status	time
1	occupied	115239

The bottom of the console shows the 'CloudShell' button, 'Feedback' link, and copyright information: '© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences'.

API gateway – parking data

The screenshot displays the AWS API Gateway console for the 'ParkingAPI' (chtm78bc6b) in the eu-north-1 region. The left sidebar shows the navigation menu with 'Resources' selected under 'API: ParkingAPI'. The main content area is titled 'Resources' and shows a tree view with a root resource '/' and a child resource '/GetParkingSpots'. The '/GetParkingSpots' resource has two methods: 'GET' and 'OPTIONS'. The 'Resource details' section shows the path '/GetParkingSpots' and the resource ID 'zjnz67'. The 'Methods (2)' section shows a table with two methods: 'GET' (Lambda integration) and 'OPTIONS' (Mock integration).

API Gateway

APIs
Custom domain names
VPC links

▼ API: ParkingAPI

- Resources
- Stages
- Authorizers
- Gateway responses
- Models
- Resource policy
- Documentation
- Dashboard
- API settings
- Usage plans
- API keys

API Gateway > APIs > Resources - ParkingAPI (chtm78bc6b)

Resources

Create resource

- /
- /GetParkingSpots
 - GET
 - OPTIONS

Resource details

Delete Update documentation Enable CORS

Path: /GetParkingSpots Resource ID: zjnz67

Methods (2)

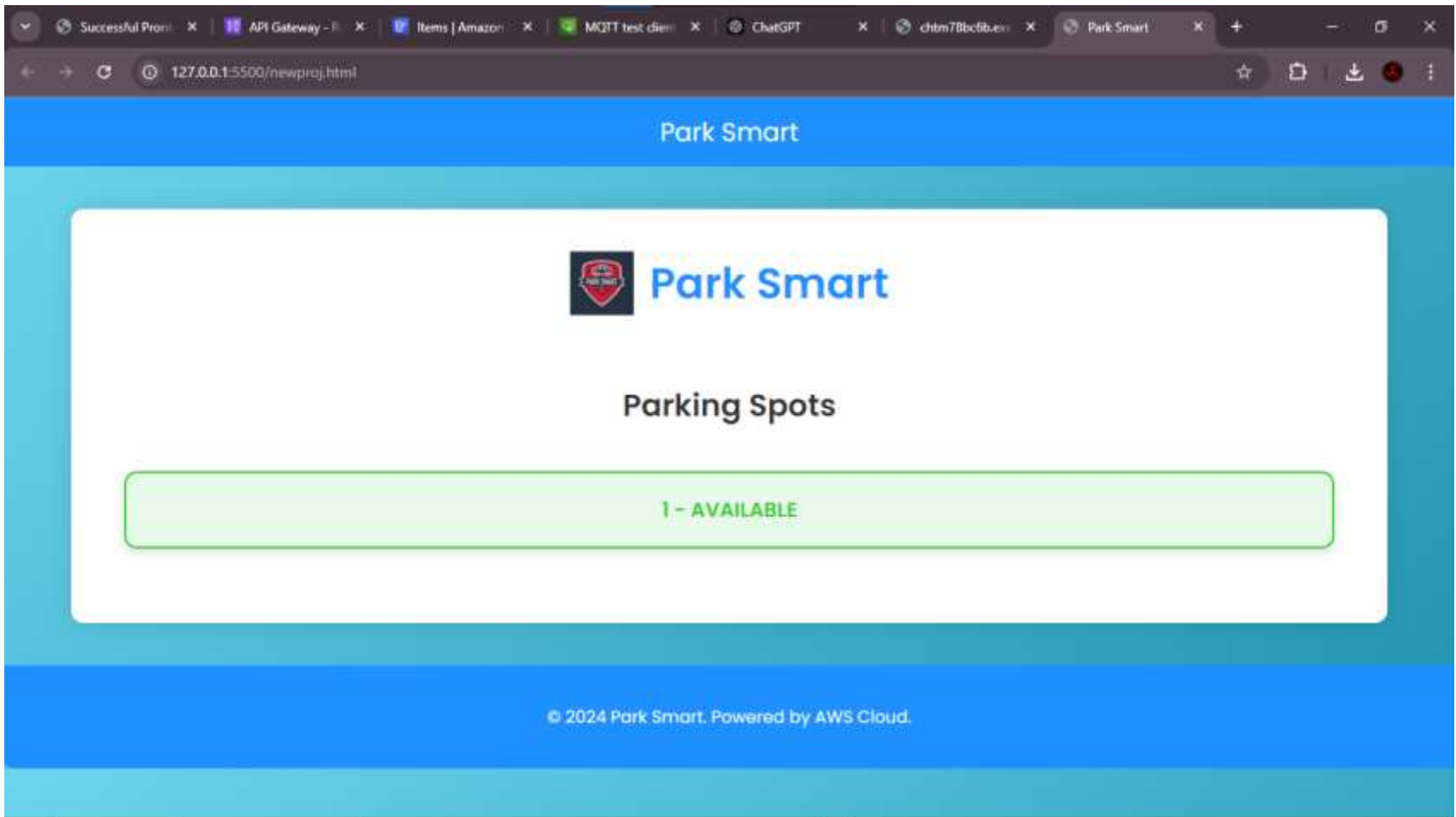
Delete Create method

	Method type ▲	Integration type ▼	Authorization ▼	API key ▼
<input type="radio"/>	GET	Lambda	None	Not required
<input type="radio"/>	OPTIONS	Mock	None	Not required

CloudShell Feedback

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Web



Web

