Lecture #11
Awareness and Nearby

Mobile Applications 2019-2020

Google Awareness API

Google Awareness API

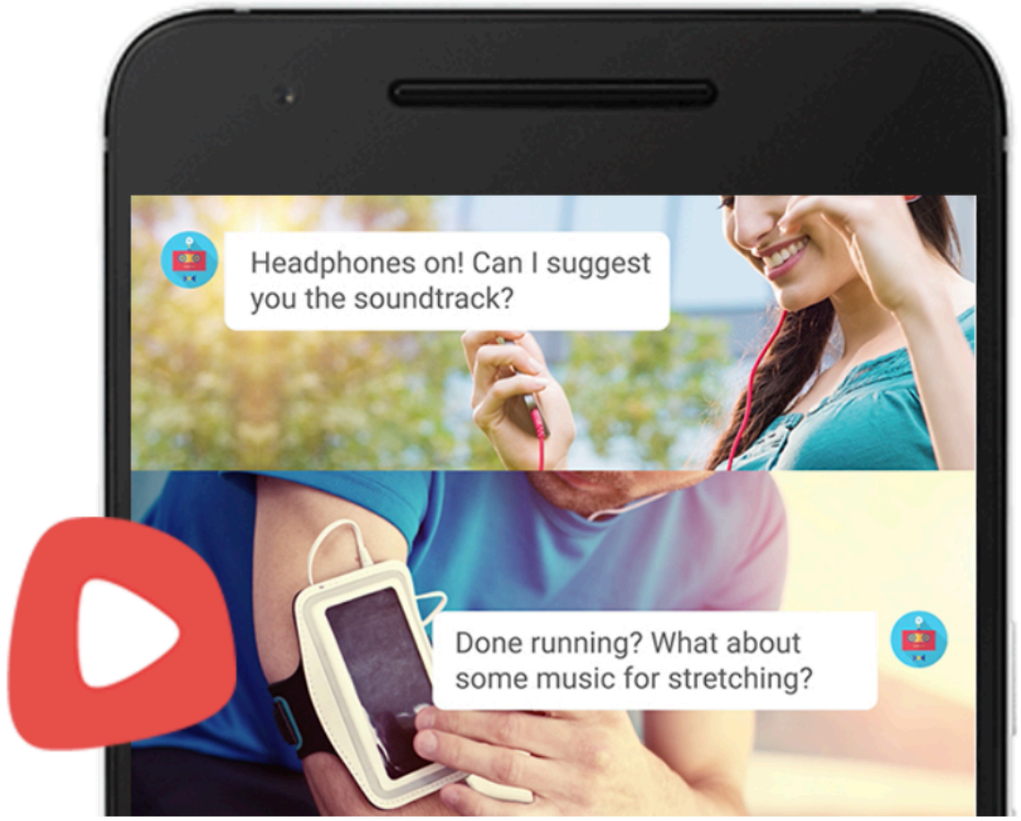
- Part of Google Play Services.

Google Awareness API

- Part of Google Play Services.
- Many signals, one API.

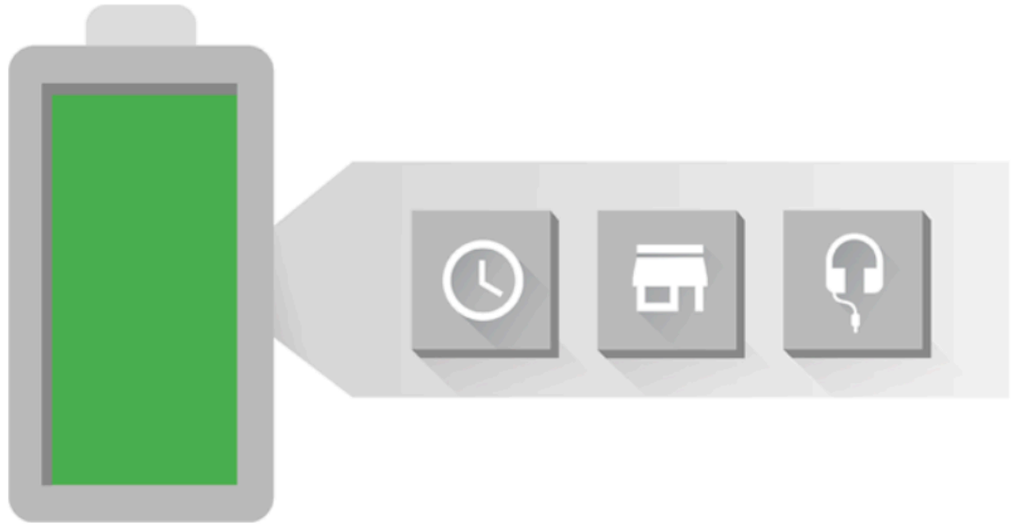
Google Awareness API

- Part of Google Play Services.
- Many signals, one API.
- High quality data.



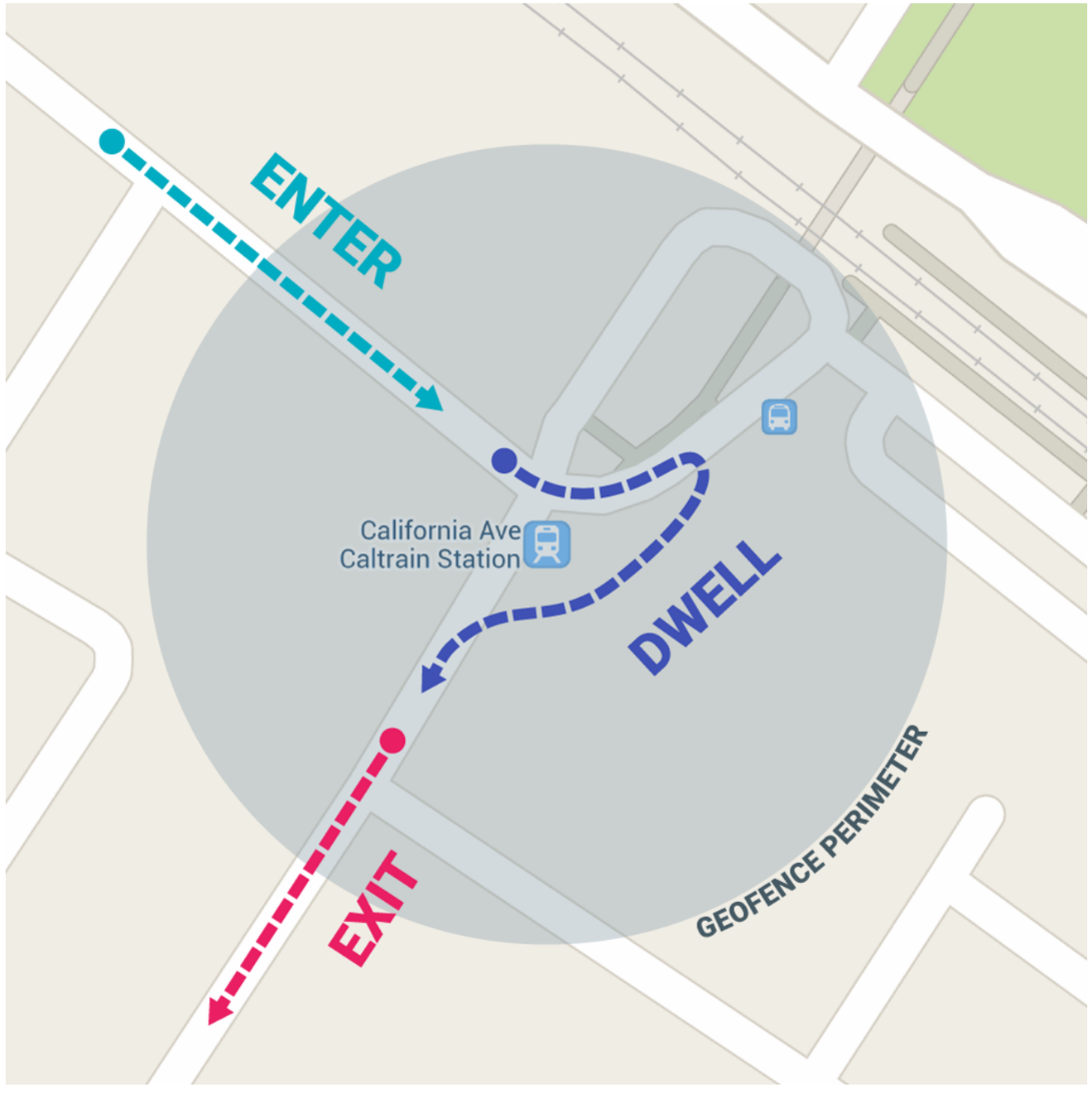
Google Awareness API

- Part of Google Play Services.
- Many signals, one API.
- High quality data.
- Smart battery savings.



Fence API

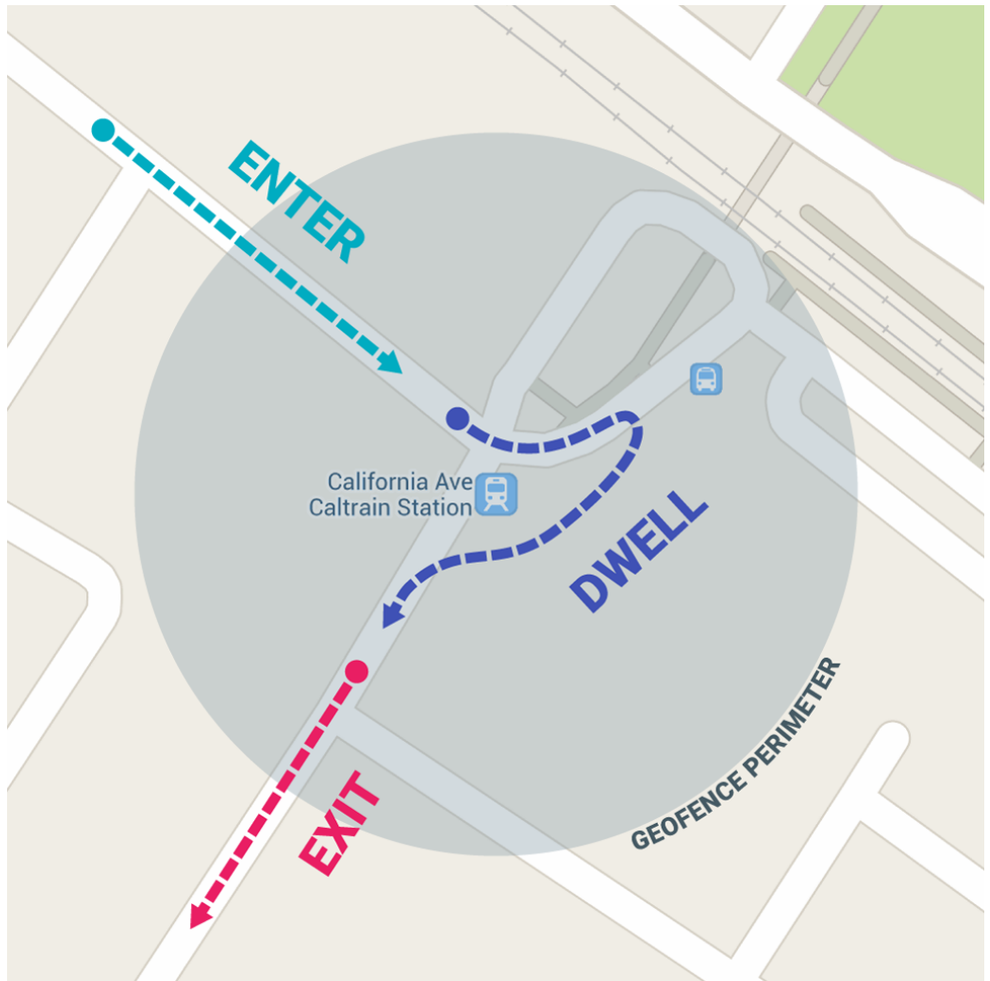
Fence API



https://developer.android.com/training/location/geofencing

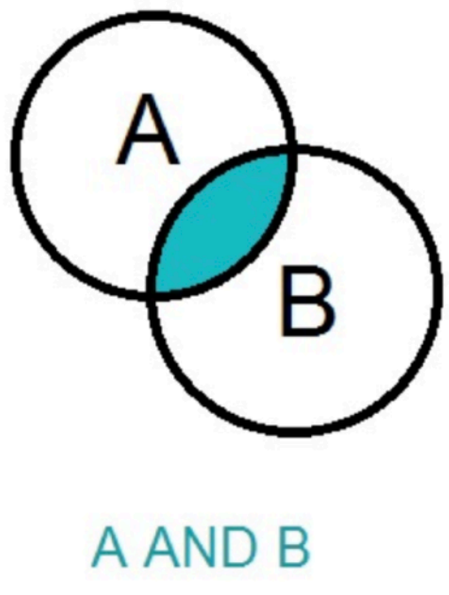
Fence API

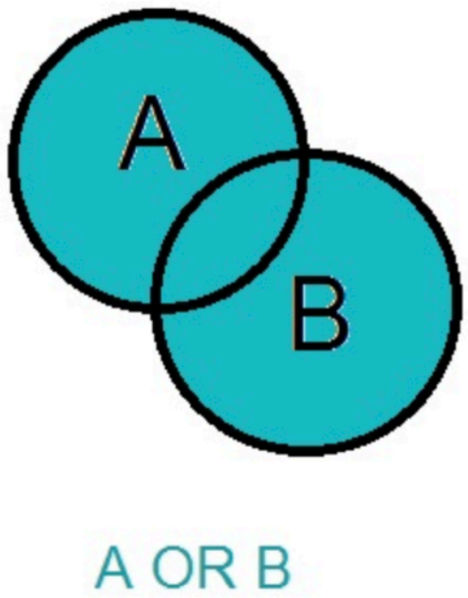
- The user's current location (lat/lng).
- The user's current activity (walking, driving, etc.).
- Device-specific conditions, such as whether the headphones are plugged in.
- Proximity to nearby beacons.

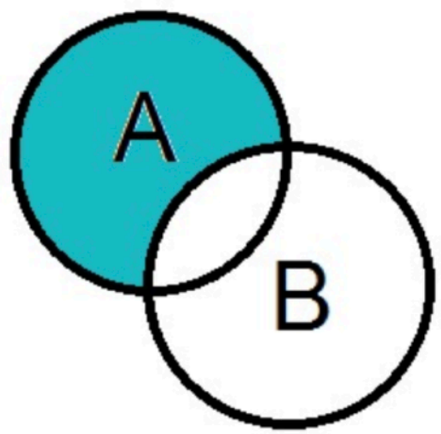


Fence API

- User plugs in headphones and starts walking.
- User enters a 100-meter geofence before 5 p.m. on a weekday.
- User enters range of a specific BLE beacon.







Configuration

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>

<!-- Required if your app targets Android 10 (API level 29) or higher -->
<uses-permission android:name="android.permission.ACCESS_BACKGROUND_LOCATION"/>
```

Configuration

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>

<!-- Required if your app targets Android 10 (API level 29) or higher -->
<uses-permission android:name="android.permission.ACCESS_BACKGROUND_LOCATION"/>
```

```
<application
  android:allowBackup="true">
  ...
  <receiver android:name=".GeofenceBroadcastReceiver"/>
<application/>
```

Fence API

```
lateinit var geofencingClient: GeofencingClient

override fun onCreate(savedInstanceState: Bundle?) {
    // ...
    geofencingClient = LocationServices.getGeofencingClient(this)
}
```

Fence API

```
geofenceList.add(Geofence.Builder()
        // Set the request ID of the geofence. This is a string to identify this
        // geofence.
        .setRequestId(entry.key)
        // Set the circular region of this geofence.
        .setCircularRegion(
                entry.value.latitude,
                entry.value.longitude,
                Constants.GEOFENCE_RADIUS_IN_METERS
)
// Set the expiration duration of the geofence. This geofence gets automatically
// removed after this period of time.
.setExpirationDuration(Constants.GEOFENCE_EXPIRATION_IN_MILLISECONDS)
// Set the transition types of interest. Alerts are only generated for these
// transition. We track entry and exit transitions in this sample.
.setTransitionTypes(Geofence.GEOFENCE_TRANSITION_ENTER or Geofence.GEOFENCE_TRANSITION_EXIT)
// Create the geofence.
.build())
```

Fence API

```
geofenceList.add(Geofence.Builder()
        // Set the request ID of the geofence. This is a string to identify this
        // geofence.
        .setRequestId(entry.key)
        // Set the circular region of this geofence.
        .setCircularRegion(
                entry.value.latitude,
                entry.value.longitude,
                Constants.GEOFENCE_RADIUS_IN_METERS
)
// Set the expiration duration of the geofence. This geofence gets automatically
// removed after this period of time.
.setExpirationDuration(Constants.GEOFENCE_EXPIRATION_IN_MILLISECONDS)
// Set the transition types of interest. Alerts are only generated for these
// transition. We track entry and exit transitions in this sample.
.setTransitionTypes(Geofence.GEOFENCE_TRANSITION_ENTER or Geofence.GEOFENCE_TRANSITION_EXIT)
// Create the geofence.
.build())
```

★ **Note:** On single-user devices, there is a limit of 100 geofences per app. For multi-user devices, the limit is 100 geofences per app per device user.

Fence API

```
private fun getGeofencingRequest(): GeofencingRequest {
    return GeofencingRequest.Builder().apply {
        setInitialTrigger(GeofencingRequest.INITIAL_TRIGGER_ENTER)
        addGeofences(geofenceList)
    }.build()
}
```

Define a broadcast receiver

```
class MainActivity : AppCompatActivity() {
    // ...
    private val geofencePendingIntent: PendingIntent by lazy {
        val intent = Intent(this, GeofenceBroadcastReceiver::class.java)
        // We use FLAG_UPDATE_CURRENT so that we get the same pending intent back when calling
        // addGeofences() and removeGeofences().
        PendingIntent.getBroadcast(this, 0, intent, PendingIntent.FLAG_UPDATE_CURRENT)
    }
}
```

Connect the Pieces

```
geofencingClient?.addGeofences(getGeofencingRequest(), geofencePendingIntent)?.run {
    addOnSuccessListener {
        // Geofences added
        // ...
    }
    addOnFailureListener {
        // Failed to add geofences
        // ...
    }
}
```

Handle Transitions

```
class GeofenceBroadcastReceiver : BroadcastReceiver() {
    // ...
    override fun onReceive(context: Context?, intent: Intent?) {
        val geofencingEvent = GeofencingEvent.fromIntent(intent)
        if (geofencingEvent.hasError()) {
            val errorMessage = GeofenceErrorMessages.getErrorString(this,
                    geofencingEvent.errorCode)
            Log.e(TAG, errorMessage)
            return
        }

        // Get the transition type.
        val geofenceTransition = geofencingEvent.geofenceTransition

        // Test that the reported transition was of interest.
        if (geofenceTransition == Geofence.GEOFENCE_TRANSITION_ENTER ||
                geofenceTransition == Geofence.GEOFENCE_TRANSITION_EXIT) {
```

```
class GeofenceBroadcastReceiver : BroadcastReceiver() {
    // ...
    override fun onReceive(context: Context?, intent: Intent?) {
        val geofencingEvent = GeofencingEvent.fromIntent(intent)
        if (geofencingEvent.hasError()) {
            val errorMessage = GeofenceErrorMessages.getErrorString(this,
                    geofencingEvent.errorCode)
            Log.e(TAG, errorMessage)
            return
        }

        // Get the transition type.
        val geofenceTransition = geofencingEvent.geofenceTransition

        // Test that the reported transition was of interest.
        if (geofenceTransition == Geofence.GEOFENCE_TRANSITION_ENTER ||
                geofenceTransition == Geofence.GEOFENCE_TRANSITION_EXIT) {

            // Get the geofences that were triggered. A single event can trigger
            // multiple geofences.
            val triggeringGeofences = geofencingEvent.triggeringGeofences

            // Get the transition details as a String.
            val geofenceTransitionDetails = getGeofenceTransitionDetails(
                    this,
```

```
        // Test that the reported transition was of interest.
        if (geofenceTransition == Geofence.GEOFENCE_TRANSITION_ENTER ||
                geofenceTransition == Geofence.GEOFENCE_TRANSITION_EXIT) {

            // Get the geofences that were triggered. A single event can trigger
            // multiple geofences.
            val triggeringGeofences = geofencingEvent.triggeringGeofences

            // Get the transition details as a String.
            val geofenceTransitionDetails = getGeofenceTransitionDetails(
                    this,
                    geofenceTransition,
                    triggeringGeofences
                    )

                    // Send notification and log the transition details.
                    sendNotification(geofenceTransitionDetails)
                    Log.i(TAG, geofenceTransitionDetails)
                } else {
                    // Log the error.
                    Log.e(TAG, getString(R.string.geofence_transition_invalid_type,
                            geofenceTransition))
                }
            }
        }
```

Stop Monitoring

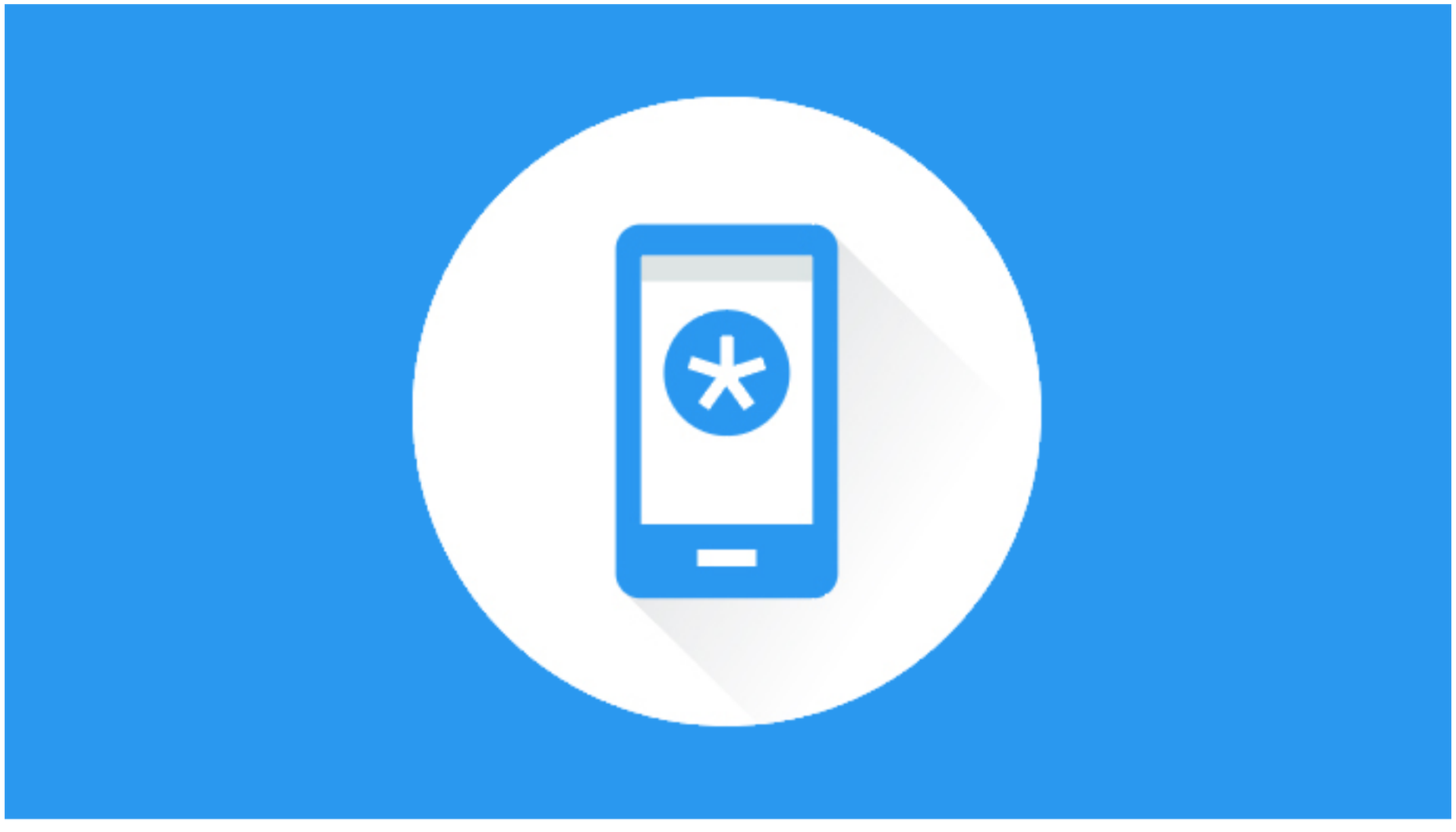
```
geofencingClient?.removeGeofences(geofencePendingIntent)?.run {
    addOnSuccessListener {
        // Geofences removed
        // ...
    }
    addOnFailureListener {
        // Failed to remove geofences
        // ...
    }
}
```

```
geofencingClient?.removeGeofences(geofencePendingIntent)?.run {
    addOnSuccessListener {
        // Geofences removed
        // ...
    }
    addOnFailureListener {
        // Failed to remove geofences
        // ...
    }
}
```


```
geofencingClient?.removeGeofences(geofencePendingIntent)?.run {
    addOnSuccessListener {
        // Geofences removed
        // ...
    }
    addOnFailureListener {
        // Failed to remove geofences
        // ...
    }
}
```

https://developers.google.com/awareness/android-api/fence-api-overview

Snapshot API



Snapshot API

- Detected user activity, such as walking or driving.
- Nearby beacons that you have registered.
- Headphone state (plugged in or not).
- Location, including latitude and longitude.
- Place where the user is currently located.
- Weather conditions in the user's current location.

Get an API key from the Google Developers Console

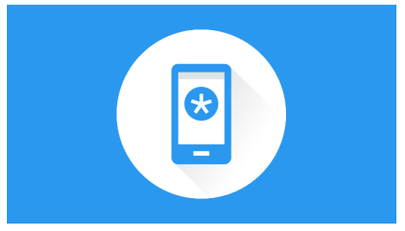
- Go to the Google Developers Console.
- Select a project, or create a new one.
- Click Continue to enable the Awareness API.
- On the Credentials page, create an Android key and set the API credentials.
- In the create key dialog, restrict your usage to Android apps — enter your app's SHA-1 fingerprint and package name.
- Click Create.

Get an API key from the Google Developers Console

- Go to the Google Developers Console.
- Select a project, or create a new one.
- Click Continue to enable the Awareness API.
- On the Credentials page, create an Android key and set the API credentials.
- In the create key dialog, restrict your usage to Android apps — enter your app's SHA-1 fingerprint and package name.
- Click Create.

```
AIzaSyBdVl-cTICSwYKrZ95LoVuw7dbMuDt1KG0
```

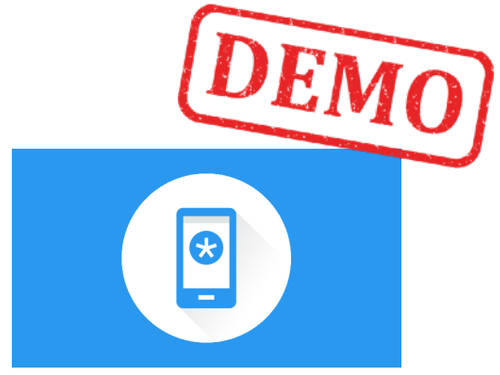
Snapshot API



```
Awareness.getSnapshotClient(this).detectedActivity
   .addOnSuccessListener { dar →
     val arr = dar.activityRecognitionResult
     // getMostProbableActivity() is good enough for basic Activity detection.
     // To work within a threshold of confidence,
     // use ActivityRecognitionResult.getProbableActivities() to get a list of
     // potential current activities, and check the confidence of each one.
     val probableActivity = arr.mostProbableActivity

     val confidence = probableActivity.confidence
     val activityStr = probableActivity.toString()
     mLogFragment.logView.println("Activity: $activityStr,
                                   Confidence: $confidence/100")
  }
.addOnFailureListener { e → Log.e(TAG, "Could not detect activity: $e") }
```

Snapshot API




```
Awareness.getSnapshotClient(this).detectedActivity
   .addOnSuccessListener { dar →
     val arr = dar.activityRecognitionResult
     // getMostProbableActivity() is good enough for basic Activity detection.
     // To work within a threshold of confidence,
     // use ActivityRecognitionResult.getProbableActivities() to get a list of
     // potential current activities, and check the confidence of each one.
     val probableActivity = arr.mostProbableActivity

     val confidence = probableActivity.confidence
     val activityStr = probableActivity.toString()
     mLogFragment.logView.println("Activity: $activityStr,
                                   Confidence: $confidence/100")
  }
.addOnFailureListener { e → Log.e(TAG, "Could not detect activity: $e") }
```

https://developers.google.com/awareness/android-api/snapshot-api-overview

Nearby Messages API

- A Publish-Subscribe API.

Nearby Messages API

Nearby Messages API

- Bluetooth.

Nearby Messages API

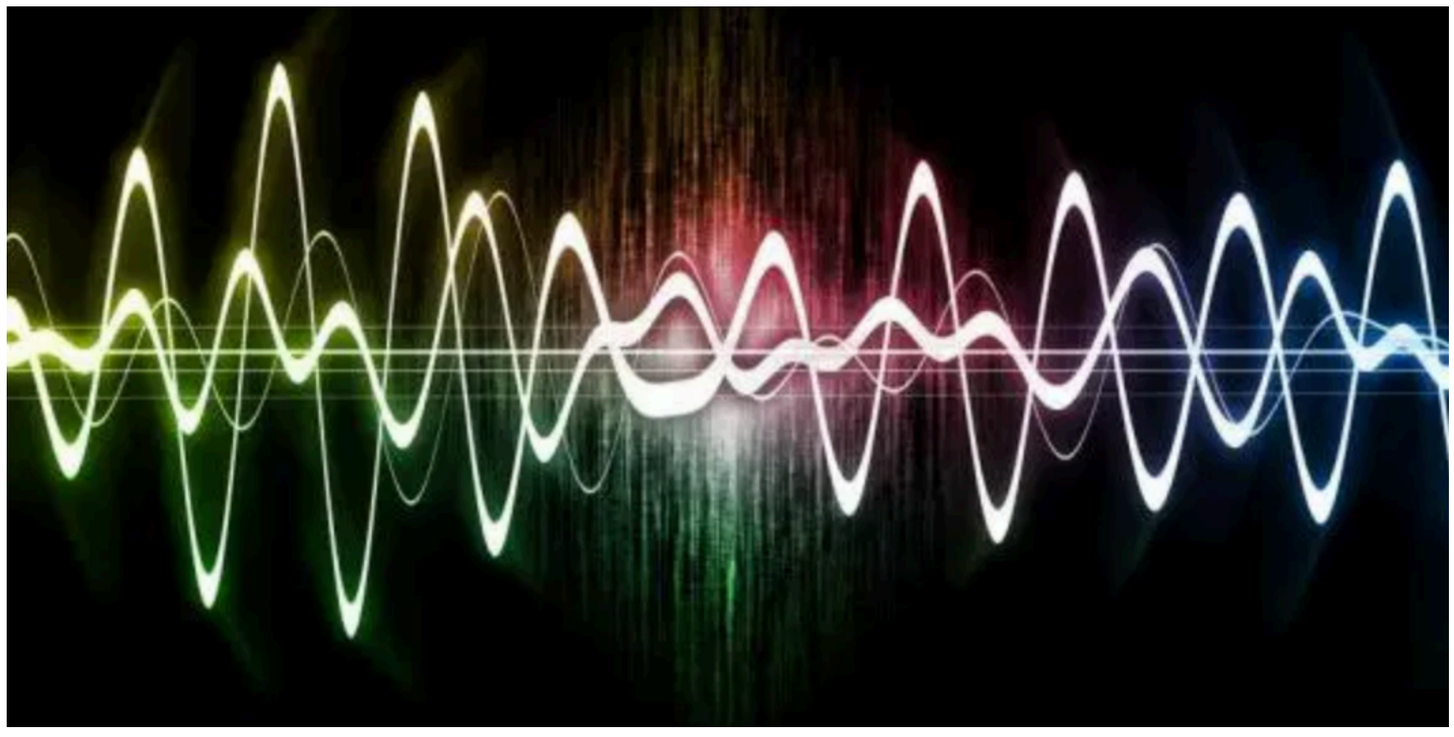
- Bluetooth.
- Bluetooth Low Energy.

Nearby Messages API

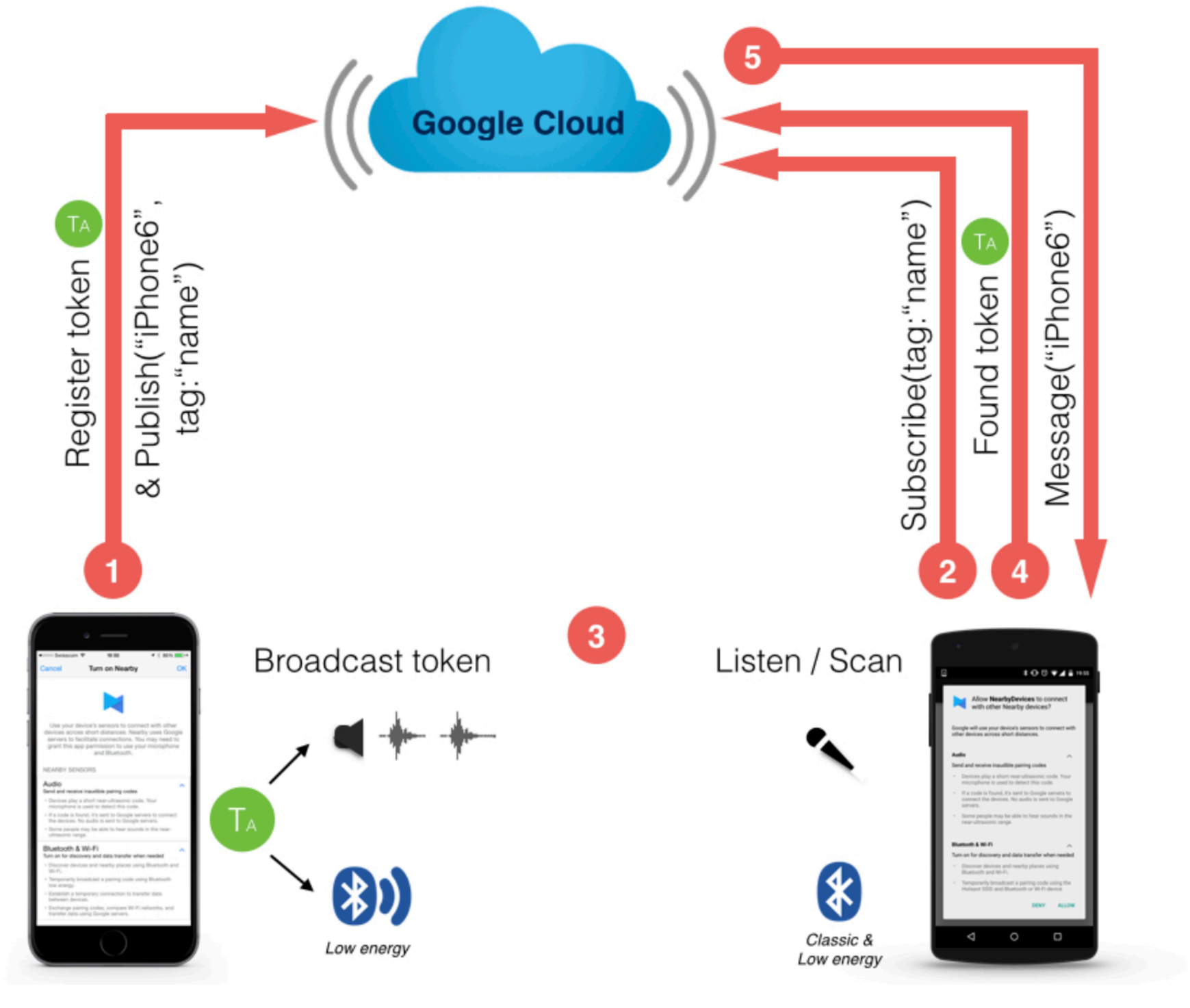
- Bluetooth.
- Bluetooth Low Energy.
- Wi-Fi.

Nearby Messages API

- Bluetooth.
- Bluetooth Low Energy.
- Wi-Fi.
- Near-ultrasonic audio.



Nearby Messages API



Configuration

```
apply plugin: 'android'
...
dependencies {
  compile 'com.google.android.gms:play-services-nearby:16.0.0'
}
```

Configuration

```
apply plugin: 'android'
...

dependencies {
  compile 'com.google.android.gms:play-services-nearby:16.0.0'
}
```

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.google.sample.app" >
    <application ...>
        <meta-data
            android:name="com.google.android.nearby.messages.API_KEY"
            android:value="API_KEY" />
        <activity>
        ...
        </activity>
    </application>
</manifest>
```

Publish and Subscribe

```
@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  ...
  mMessageListener = new MessageListener() {
    @Override
    public void onFound(Message message) {
      Log.d(TAG, "Found message: " + new String(message.getContent()));
    }
    @Override
    public void onLost(Message message) {
      Log.d(TAG, "Lost sight of message: " + new String(message.getContent()));
    }
  }

  mMessage = new Message("Hello World".getBytes());
}

@Override
public void onStart() {
  super.onStart();
  ...
  Nearby.getMessagesClient(this).publish(mMessage);
  Nearby.getMessagesClient(this).subscribe(mMessageListener);
```

```
  ...
  mMessageListener = new MessageListener() {
    @Override
    public void onFound(Message message) {
      Log.d(TAG, "Found message: " + new String(message.getContent()));
    }

    @Override
    public void onLost(Message message) {
      Log.d(TAG, "Lost sight of message: " + new String(message.getContent()));
    }
  }

  mMessage = new Message("Hello World".getBytes());
}
```

```
@Override
public void onStart() {
  super.onStart();
  ...
  Nearby.getMessagesClient(this).publish(mMessage);
  Nearby.getMessagesClient(this).subscribe(mMessageListener);
}

@Override
public void onStop() {
  Nearby.getMessagesClient(this).unpublish(mMessage);
  Nearby.getMessagesClient(this).unsubscribe(mMessageListener);
  ...
  super.onStop();
}
```

Handling User Consent

```
if (ContextCompat.checkSelfPermission(this,
    Manifest.permission.ACCESS_FINE_LOCATION)
        == PackageManager.PERMISSION_GRANTED) {
    mMessagesClient = Nearby.getMessagesClient(this,
       new MessagesOptions.Builder()
     .setPermissions(NearbyPermissions.BLE)
     .build());
}
```

Get Beacon Messages


```
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  ...
  mMessageListener = new MessageListener() {
    @Override
    public void onFound(Message message) {
      Log.d(TAG, "Found message: " + new String(message.getContent()));
    }

    @Override
    public void onLost(Message message) {
      Log.d(TAG, "Lost sight of message: " + new String(message.getContent()));
    }
  }
}

// Subscribe to receive messages.
private void subscribe() {
  Log.i(TAG, "Subscribing.");
  SubscribeOptions options = new SubscribeOptions.Builder()
    .setStrategy(Strategy.BLE_ONLY)
    .build();
  Nearby.getMessagesClient(this).subscribe(mMessageListener, options);
}
```

Get Beacon Messages

Subscribe in the background

```
// Subscribe to messages in the background.
private void backgroundSubscribe() {
    Log.i(TAG, "Subscribing for background updates.");
    SubscribeOptions options = new SubscribeOptions.Builder()
            .setStrategy(Strategy.BLE_ONLY)
            .build();
    Nearby.getMessagesClient(this).subscribe(getPendingIntent(), options);
}

private PendingIntent getPendingIntent() {
    return PendingIntent.getBroadcast(this, 0, new Intent(this,
        BeaconMessageReceiver.class), PendingIntent.FLAG_UPDATE_CURRENT);
}
```

Get Beacon Messages

Subscribe in the background

```
@Override
public void onReceive(Context context, Intent intent) {
  Nearby.getMessagesClient(context).handleIntent(intent, new MessageListener() {
    @Override
    public void onFound(Message message) {
      Log.i(TAG, "Found message via PendingIntent: " + message);
    }

    @Override
    public void onLost(Message message) {
      Log.i(TAG, "Lost message via PendingIntent: " + message);
    }
  });
}
```

https://developers.google.com/nearby/messages/android/get-beacon-messages

Lecture outcomes

- Create applications to be aware of multiple aspects of a users context, while managing battery and memory health.
- React to changes in the user's environment.
- Get instant details about the user's current environment, by accessing 7 signals from one simple API surface.
- Create nearby notifications to helps users discover what's around them.