

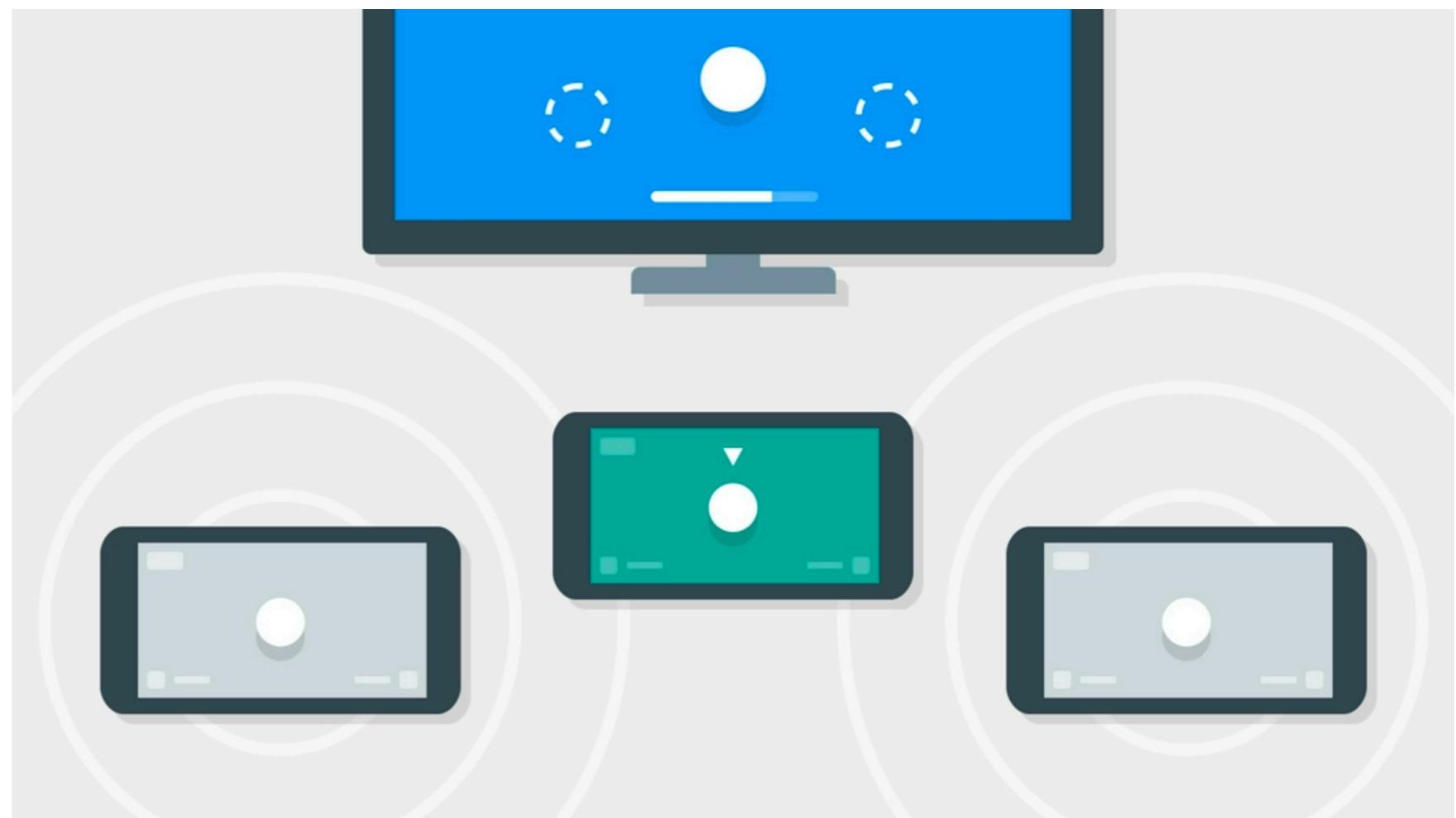
Lecture #8

Nearby & Communication

Services

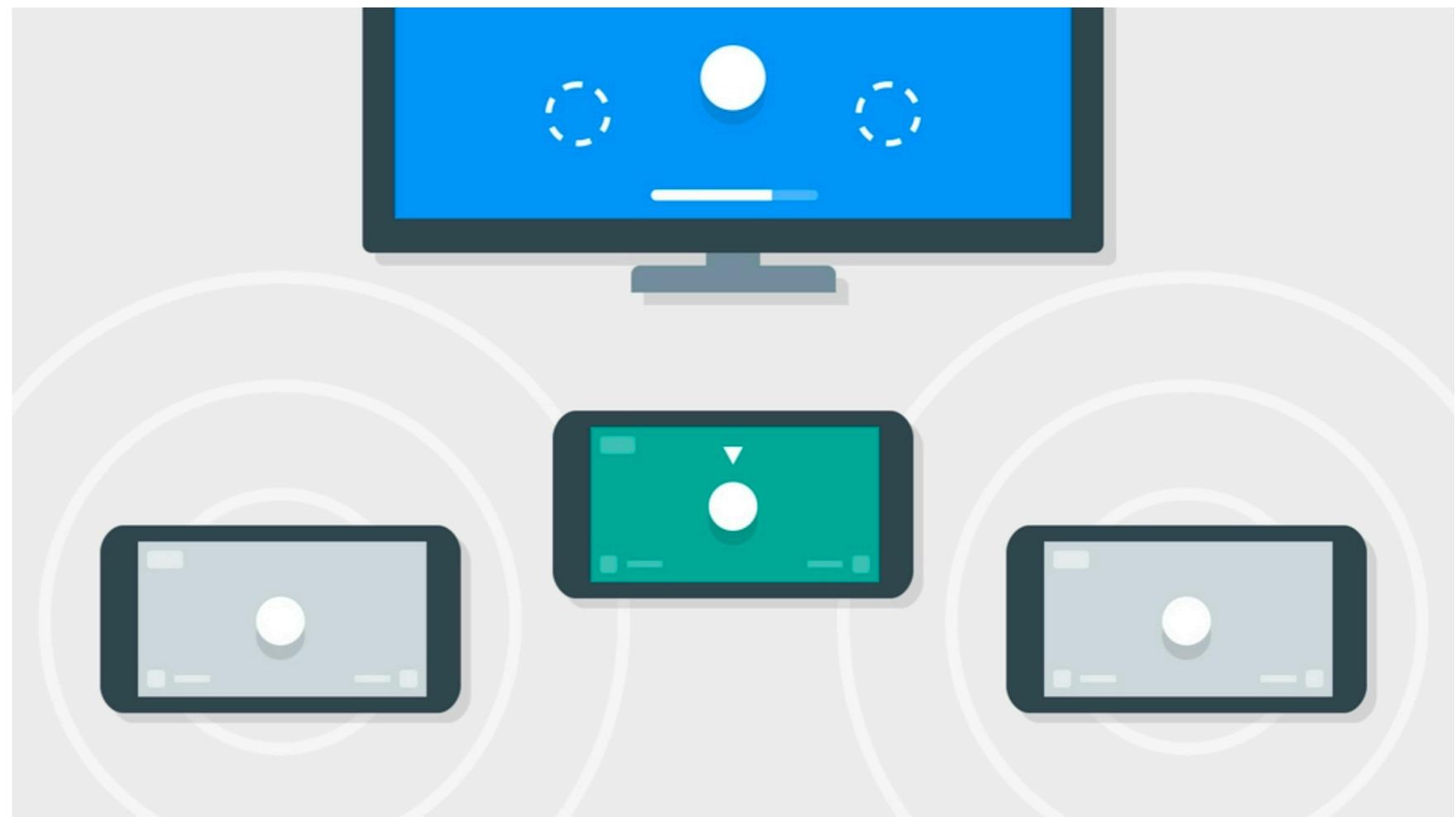
Android Things 2020

Nearby



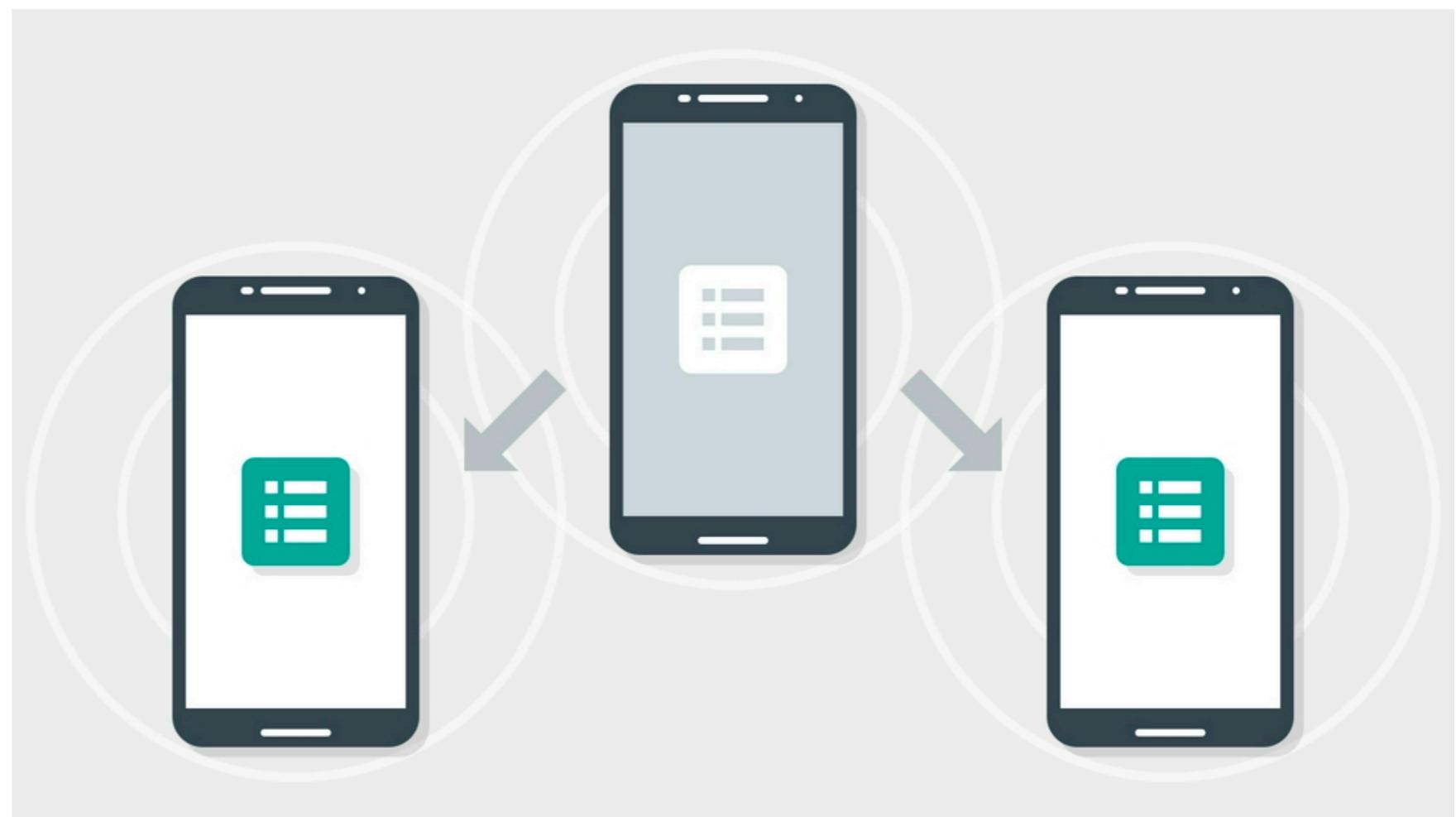
Nearby

- Connections API



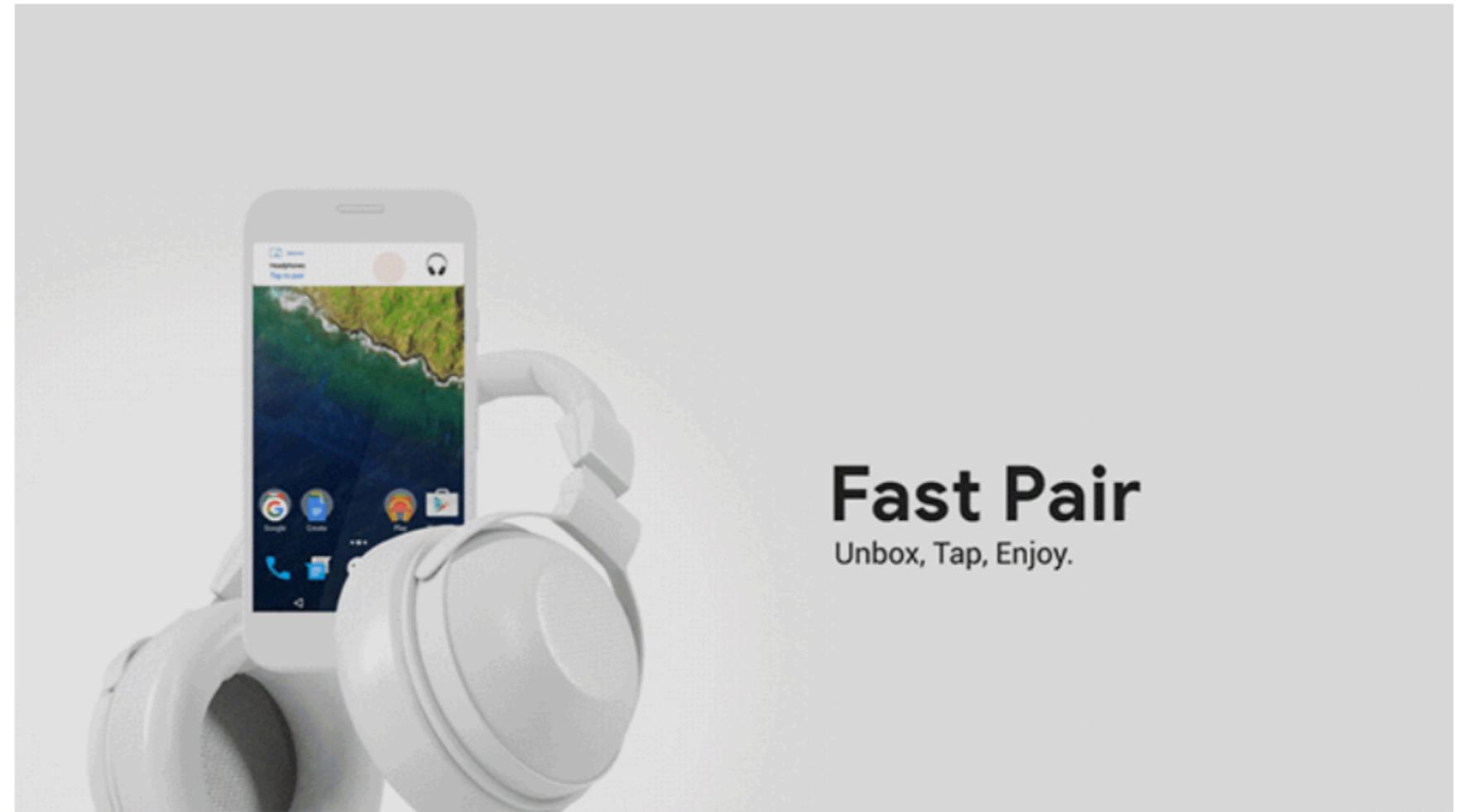
Nearby

- Connections API
- Messages API



Nearby

- Connections API
- Messages API
- Fast Pair



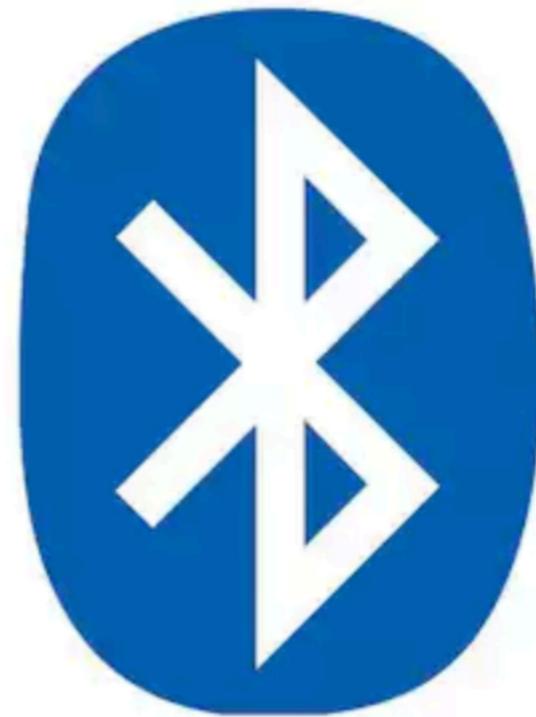
Fast Pair
Unbox, Tap, Enjoy.

Nearby Messages API

- A Publish-Subscribe API.

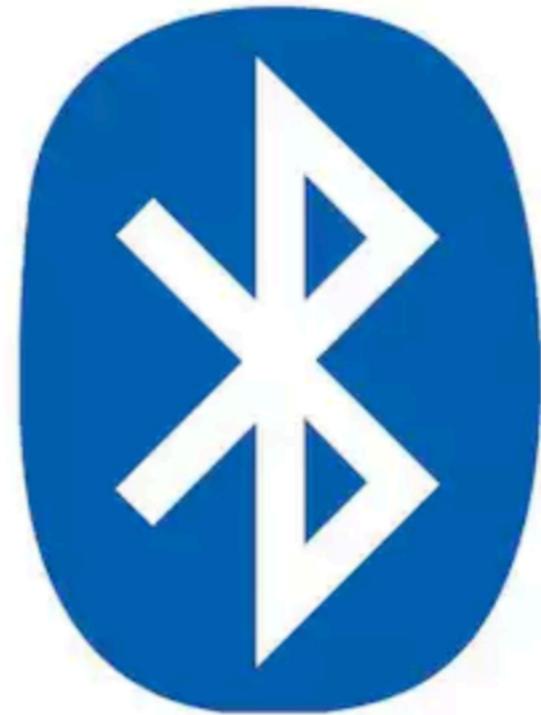


Nearby Messages API



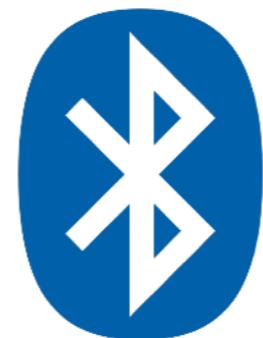
Nearby Messages API

- Bluetooth.



Nearby Messages API

- Bluetooth.
- Bluetooth Low Energy.



Bluetooth
SMART

Nearby Messages API

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

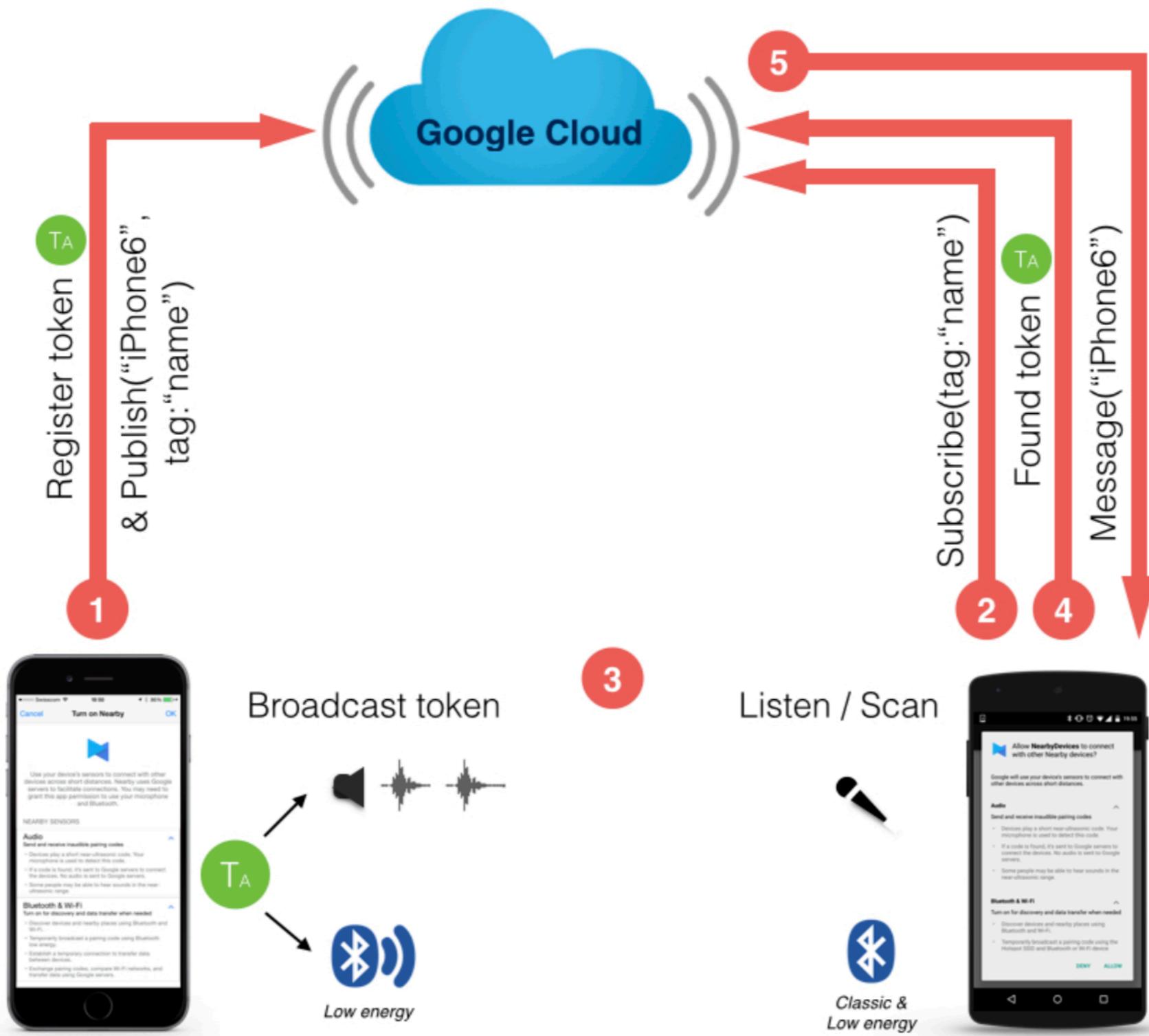


Nearby Messages API

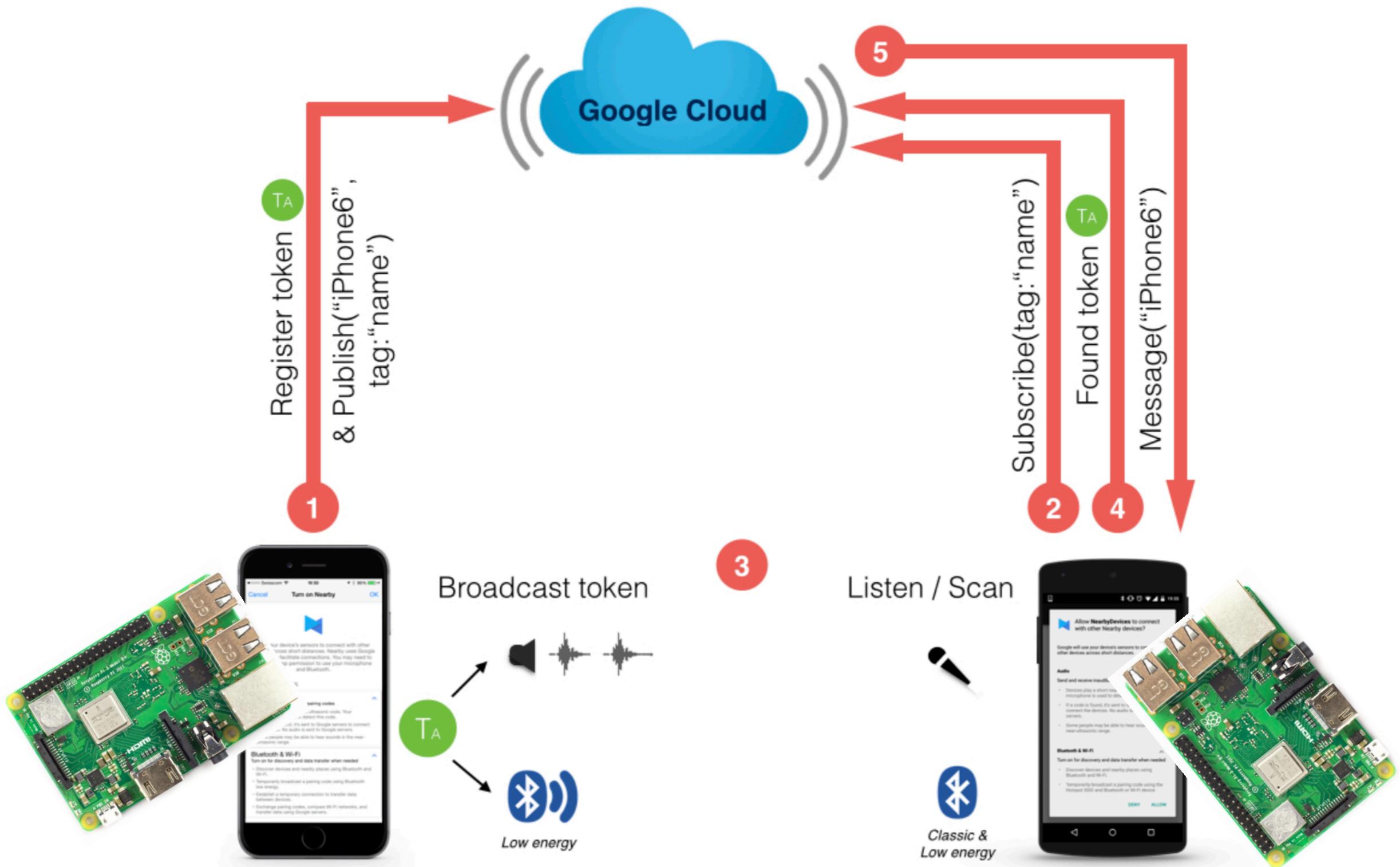
- Bluetooth.
- Bluetooth Low Energy.
- Wi-Fi.
- Near-ultrasonic audio.



Nearby Messages API



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" />  
        <uses-library android:name="com.google.android.things" />  
        <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);  
        }  
    });  
}
```

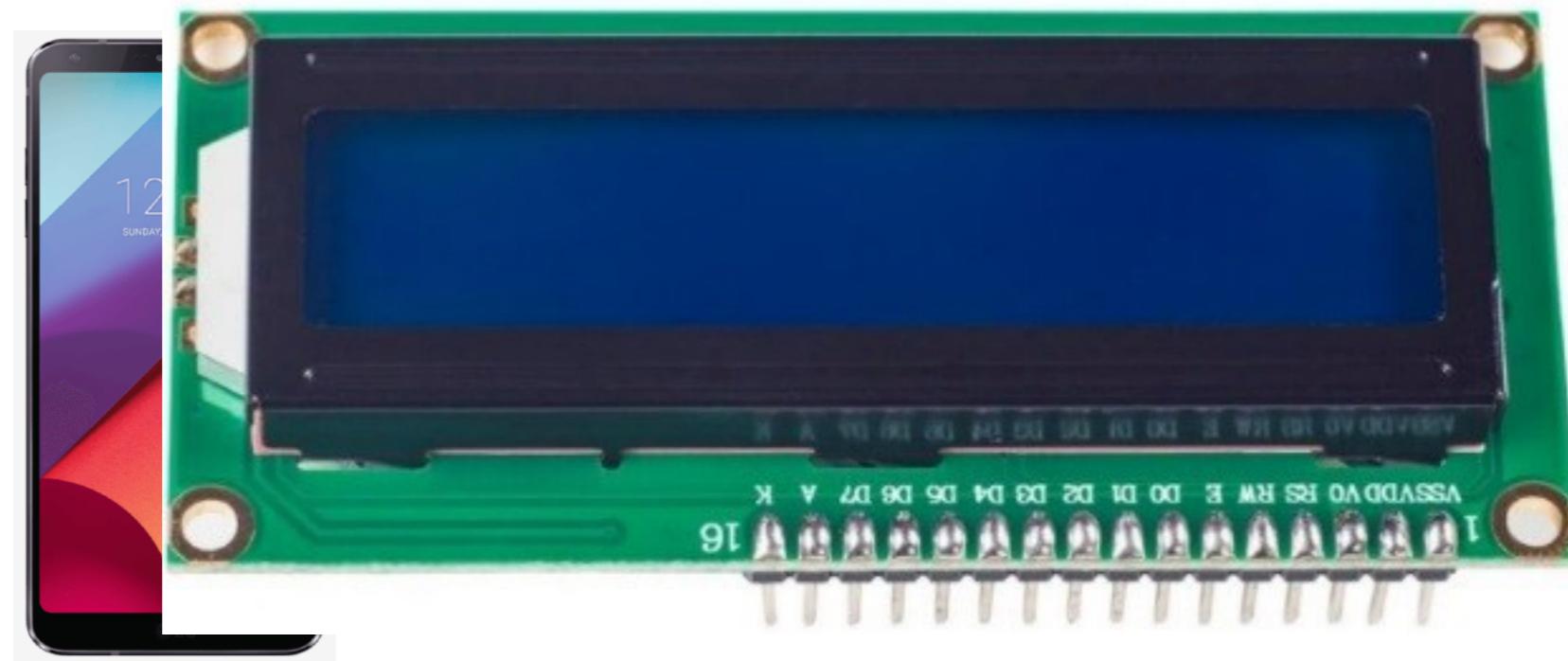
Android Things Nearby Connection



Android Things Nearby Connection



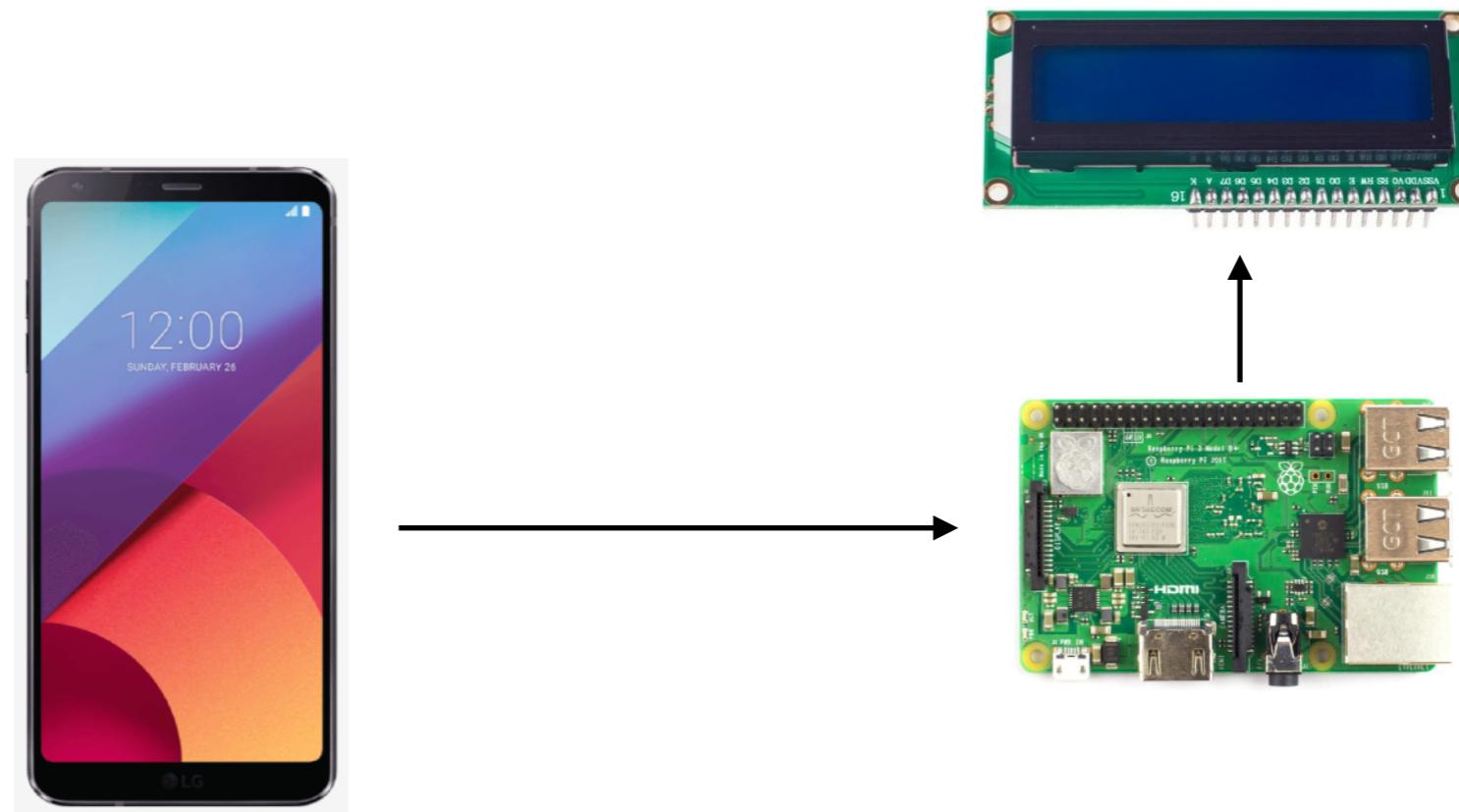
Android Things Nearby Connection



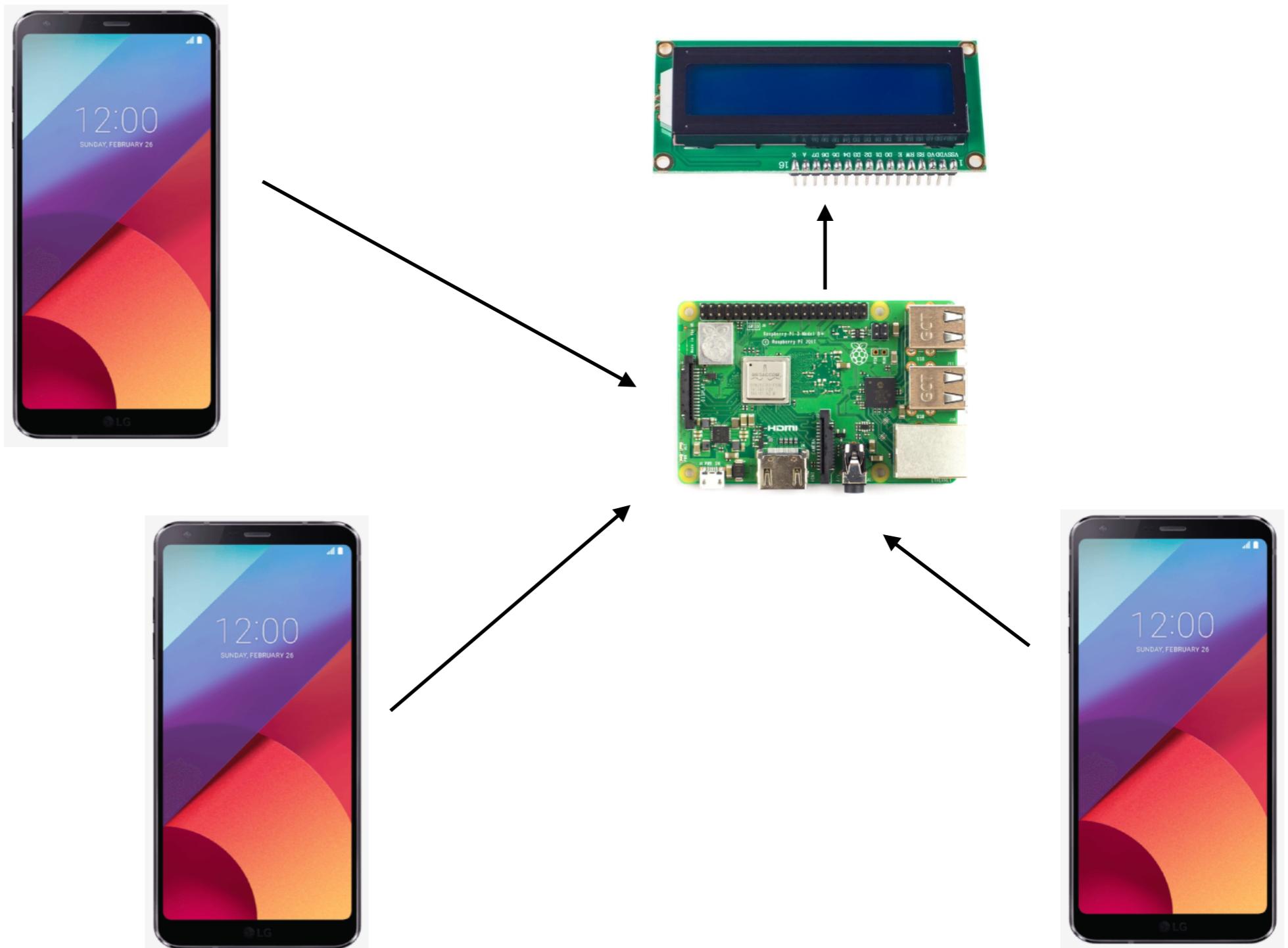
Android Things Nearby Connection



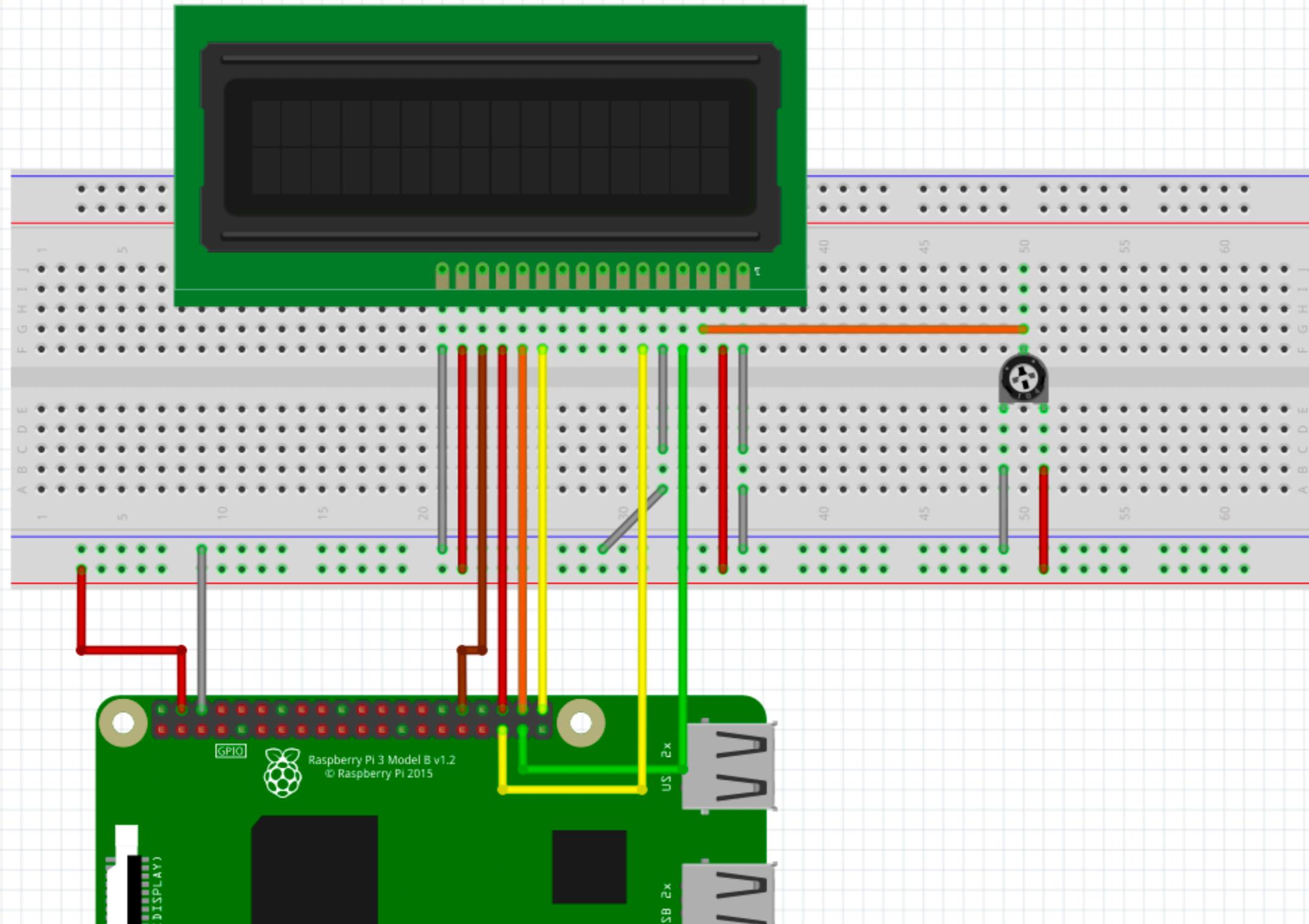
Android Things Nearby Connection



Android Things Nearby Connection



Schema



Advertise

```
val advertisingOptions =  
    AdvertisingOptions.Builder().setStrategy(Strategy.P2P_STAR).build()  
client = Nearby.getConnectionsClient(ctx)  
client.startAdvertising("AndroidThings",  
    SERVICE_ID,  
    connectionLifeCycleCB,  
    advertisingOptions)  
.addOnSuccessListener { logi("OnSuccess...") }  
.addOnFailureListener { e -> loge("OnFailure: ", e) }
```

Discover

```
Nearby.getConnectionsClient(ctx)
    .startDiscovery(SERVICE_ID,
                    endpointDiscoveryCB,
                    discoveryOptions)
    .addOnSuccessListener {
        logi("OnSuccess...")
        listener.startDiscovering()
    }
    .addOnFailureListener { e ->
        loge("OnFailure", e)
    }
```

DEMO

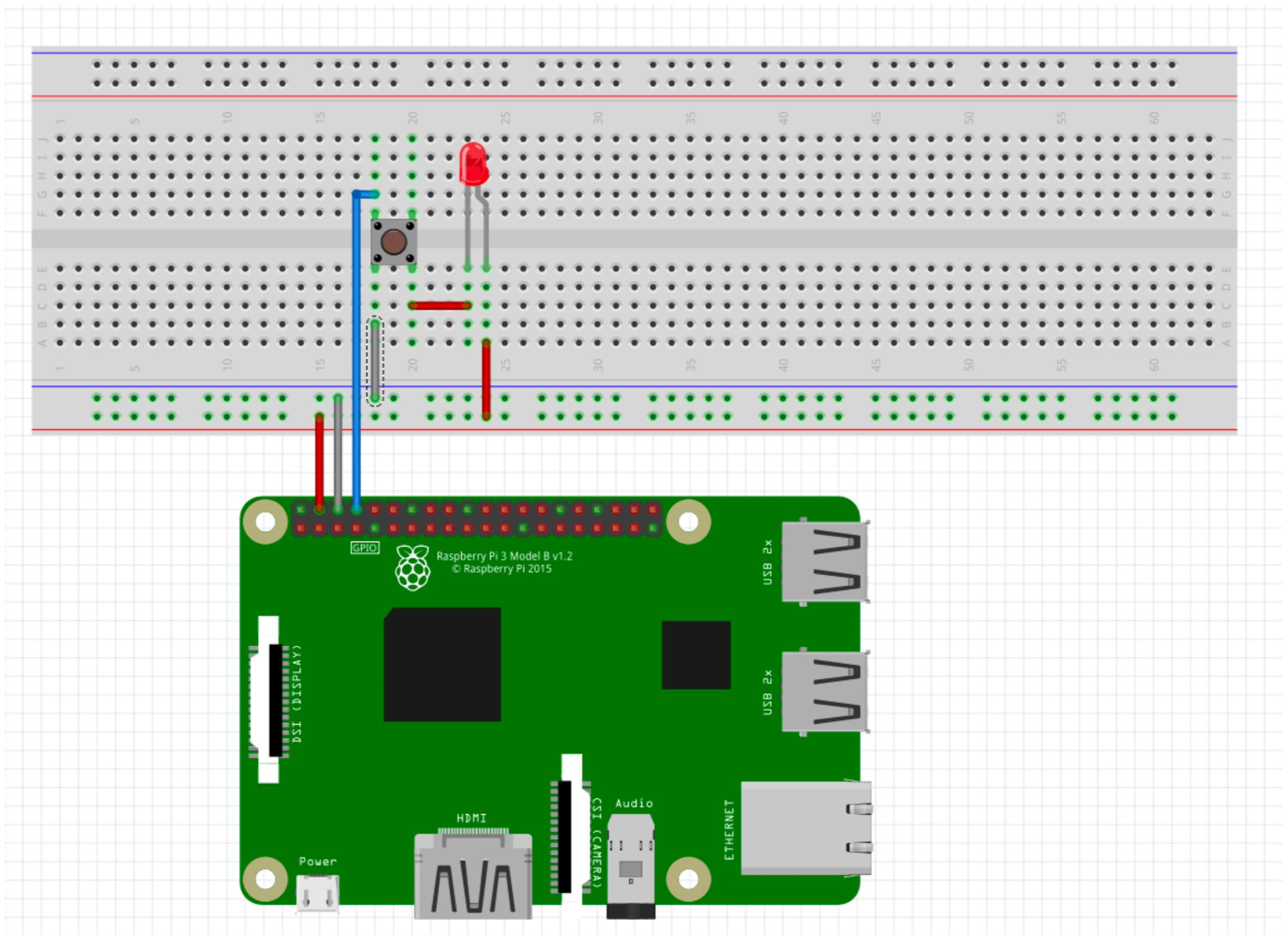
LCD Screen

implementation 'com.nilhcem.androidthings:driver-lcd1602:0.0.3'

```
val GPIO_LCD_RS = "BCM26"
val GPIO_LCD_EN = "BCM19"
val GPIO_LCD_D4 = "BCM21"
val GPIO_LCD_D5 = "BCM20"
val GPIO_LCD_D6 = "BCM16"
val GPIO_LCD_D7 = "BCM12"
```

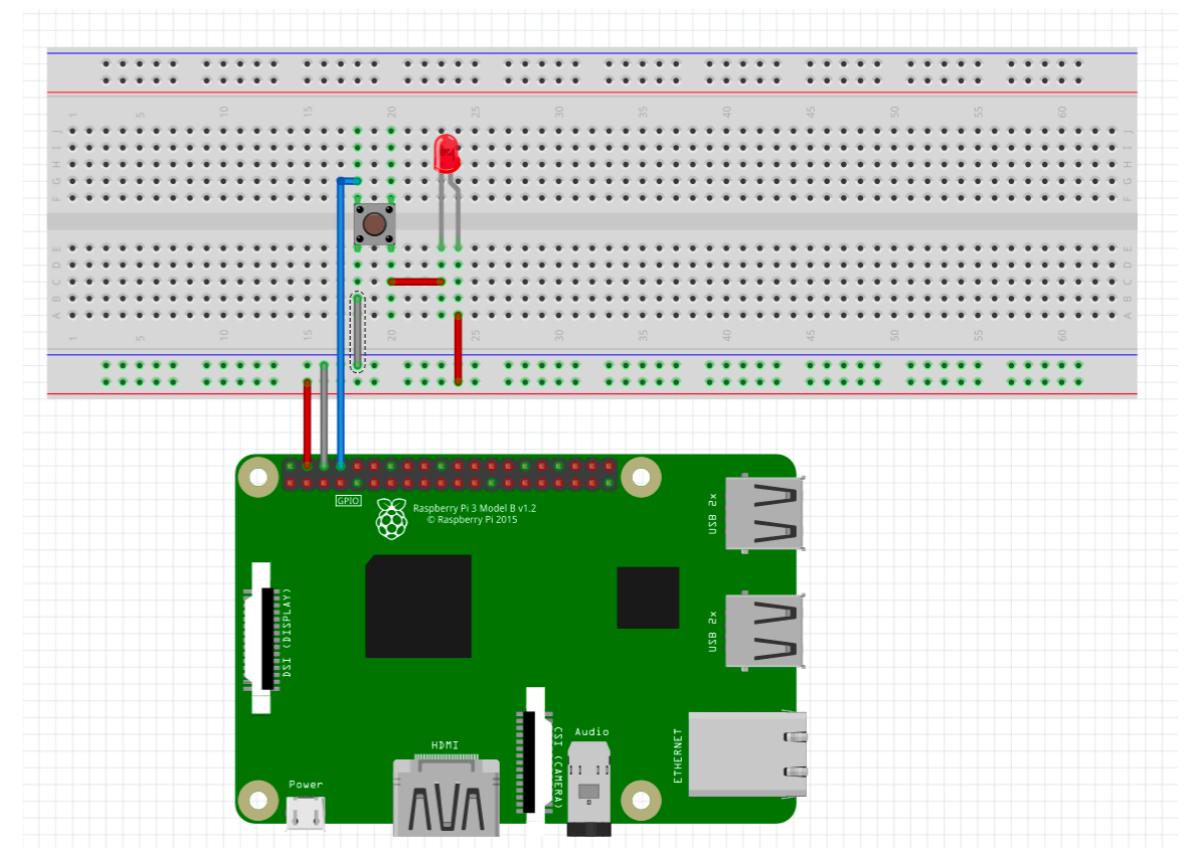
```
mLcd = Lcd1602(GPIO_LCD_RS, GPIO_LCD_EN,
                  GPIO_LCD_D4, GPIO_LCD_D5, GPIO_LCD_D6, GPIO_LCD_D7)
mLcd.clear()
mLcd.print("Hello World!")
```

Panic Button



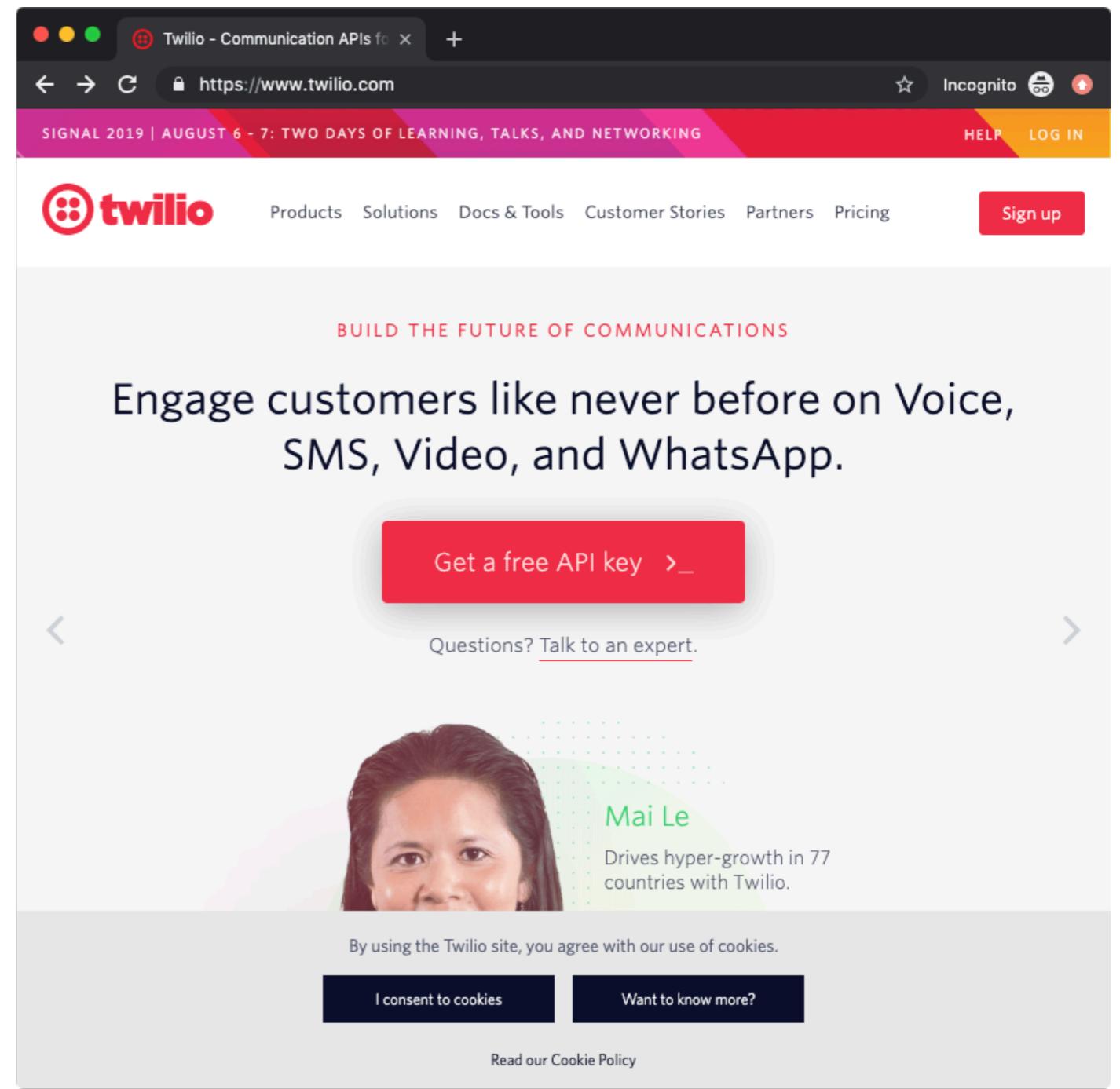
Panic Button

- When pressing the button more than 600ms send an SMS to a predefined number.
- To send the sms we can use, either:
 - <https://www.twilio.com>
 - GSM Sim900A module.



www.twilio.com

- Create a trial account.
- You will get:
 - An API key.
 - A client id code.



Send a SMS Message

```
fun sms(body: String) {  
    val formBody = FormBody.Builder()  
        .add("From", ORIGIN_PHONE_NUMBER)  
        .add("To", DEST_PHONE_NUMBER)  
        .add("Body", body)  
        .build()  
  
    val builder = Request.Builder()  
        .url(TWILLIO_MESSAGE_URL)  
        .post(formBody)  
        execute(builder)  
}
```

Authenticate the Call

```
private fun execute(builder: Request.Builder) {
    val credential = Credentials.basic(TWILLIO_ACCOUNT_SID, TWILLIO_AUTH_TOKEN)
    val request = builder
        .addHeader("Authorization", credential)
        .build()
    httpClient.newCall(request)?.enqueue(object : Callback {
        @Throws(IOException::class)
        override fun onResponse(call: Call, response: Response) {
            if (response.isSuccessful) {
                logd("Twilio request succeed.")
            } else {
                logd("Twilio request failed. Make sure you have the correct configuration")
            }
        }
        override fun onFailure(call: Call, e: IOException) {
            logd("Twilio request failed.", e)
        }
    })
}
```

DEMO

Make a Phone Call

```
fun call() {  
    val url = "https://someurl.com"  
  
    val formBody = FormBody.Builder()  
        .add("From", ORIGIN_PHONE_NUMBER)  
        .add("To", DEST_PHONE_NUMBER)  
        .add("Url", url)  
        .build()  
  
    val builder = Request.Builder()  
        .url(TWILLIO_CALL_URL)  
        .post(formBody)  
  
    execute(builder)  
}
```

Lecture outcomes

- Nearby API.
- Send sms messages.
- Make phone calls.





designed by  freepik