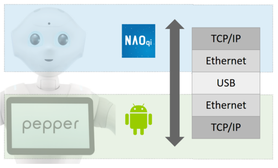
Synchronous or Asynchronous?

Pepper Running a different process for a fortnight:

- One in the tablet is.
- One in the robot head.

There is an exchange of information transfer between the two CPUs via USB using the TCP/IP protocol :



You have the flexibility to design your code and decide whether you want to handle this communication synchronously or asynchronously.

However, the choice between the best way to work synchronously or asynchronously depends on a number of factors that we will describe below.

In the UI thread

Android allows you to access UI threads using different Activity lifecycle callbacks, such as:

```
@Override
    protected void onCreate(Bundle savedInstanceState){
        // Executes on the UI thread.
        }
    @Override
    protected void onResume() {
        // Executes on the UI thread.
    }
    ...
```

Performing synchronous calls on the UI thread can block the UI and lead to a bad user experience.

To prevent this, you must use **QiSDK** asynchronous calls when dealing with the UI thread, otherwise `NetworkOnMainThreadException` will throw the.

in practice

omission:

```
// UI thread.
Say say = SayBuilder.with(qiContext)
        .withText("Hello")
        .build(); // Throws a NetworkOnMainThreadException.
```

```
// UI thread.
goTo.run(); // Throws a NetworkOnMainThreadException.

serve as:



 // UI thread.
 Future<Say> sayBuilding = SayBuilder.with(qiContext)
        .withText("Hello")
        .buildAsync(); // OK.



 // UI thread.
 goTo.async().run(); // OK.
```

Working on a worker thread

You can work on working threads on Android in a number of ways, using your own threa

`QiSDK` provides several ways to put work on the worker thread:

Bot Lifecycle Robot Lifecycle:

```
 @Override
 public void onRobotFocusGained(QiContext qiContext) {
   // Executes on a worker thread.
 }
 @Override
 public void onRobotFocusLost() {
     // Executes on a worker thread.
 }
```

Futures cable length (= 1 (chaining) :

```
 future.thenConsume(future -> {
     // Executes on a worker thread.
 });
```

listener Listeners:

```
 qiContext.getHumanAwareness().addOnHumansAroundChangedListener(humans -> {
     // Executes on a worker thread.
 });
```

In practice

If you want to handle cancellations:

Use asynchronous calls .

```
 Future<Void> goToFuture = goTo.async().run();
 ...
 goToFuture.requestCancellation();
```

If you want to run multipleactions:

Use asynchronous calls.

```
 Future<Void> sayFuture = say.async().run();
 Future<Void> goToFuture = goTo.async().run();
```

In other cases:

Use synchronous or asynchronous calls.

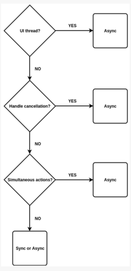
```
 say.run();
```

or

```
 Future<Void> sayFuture = say.async().run();
```

Summary

The following diagram illustrates the types of calls to be used depending on the con



See also

javadoc