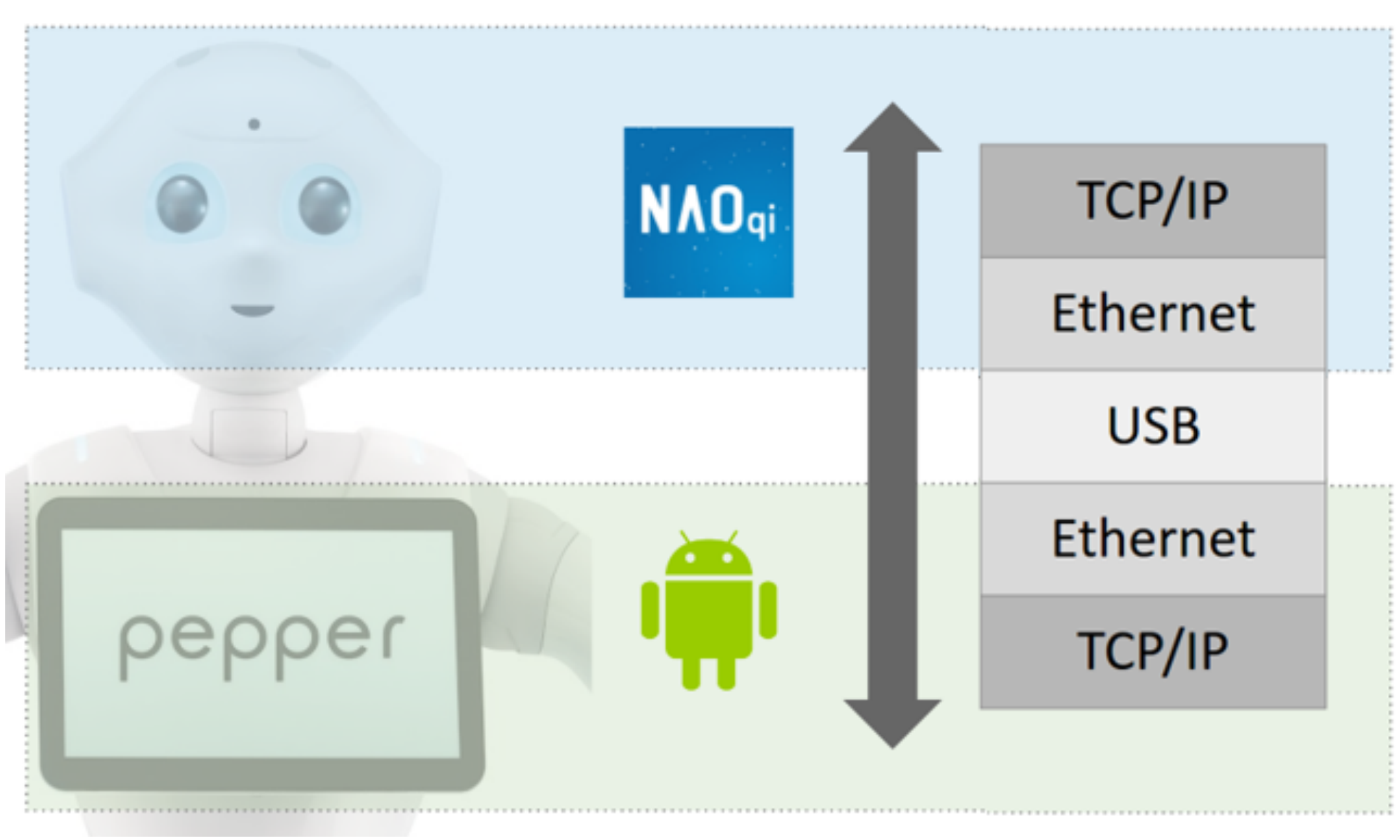
同步还是异步?

Pepper 运行两周不同的进程:

- 一个在平板是,
- 一个在机器人头部中.

在两个CPU之间有通过USB使用TCP/IP协议的信息传递交换 :



您可以灵活地设计代码，并决定是要同步还是异步处理此通信.

但是，存在最佳的方式，同步或异步工作之间的选择取决于我们将在下面描述的一些因素.

在UI线程中

Android系统允许您使用不同的Activity生命周期回调访问UI线程，例如:

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

如果在UI线程上执行同步调用，则会阻塞UI并导致糟糕的用户体验.

为了防止这种情况，当您处理UI线程时，必须使用**QiSDK**异步调用，否则 `NetworkOnMainThreadException` 将抛出.

在实践中

不做:

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

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

做:

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

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

在工作线程上

您可以通过多种方式在Android上的工作线程上工作，使用您自己的线程管理系统或为专门处理此类问题的库.

QiSDK 提供多种方式来把工作放在工作线程上:

机器人生命周期 Robot Lifecycle:

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

Futures 链 (chaining) :

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

监听器 Listeners:

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

在实践中

如果你想处理取消:

使用异步调用.

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

如果要同时运行多个actions:

使用异步调用.

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

在其他情况下:

使用同步或异步调用.

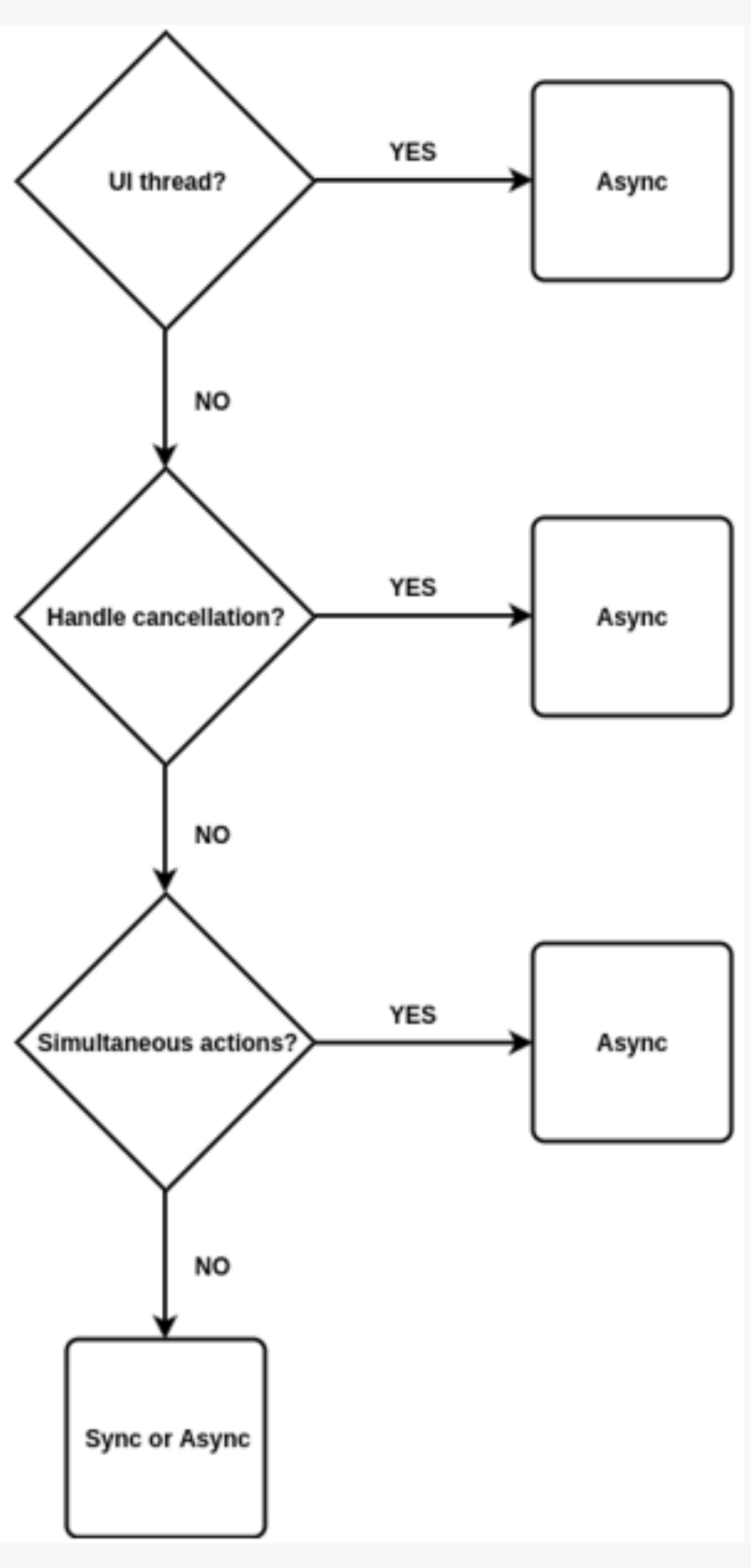
```
  say.run();
```

或者

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

总结

下图说明了要根据上下文使用的调用类型:



See also

javadoc