MOBILE APP

TRAIN AROUND

Where every step counts

1. Introduction

Workout monitoring

- Smartphones and wearables can help users achieving workout motivation and better health care
- These technologies can also be used by athletes to track their training progress
- Sensors information are sometimes useful to trainers to understand how an athlete is perfoming and how to achive better results

Lacks of preexisting applications

1

Extra weight

- Some workout tracking solutions consists in smartphone applications that force users to carry the device in its hand or in a arm holder
- This extra wheight solution could feel unconfortable and lead to a bad user experience

2

Data monitoring

- Collected data are often displayed only to the athlete who is training
- A real-time data sharing with a trainer's device could be useful to the training session

3

Sensors availability

- Smartphones have less available sensors
- Wearables are preferred because they can track more accurately movements

Our solution (I)

Lightweight solution

1

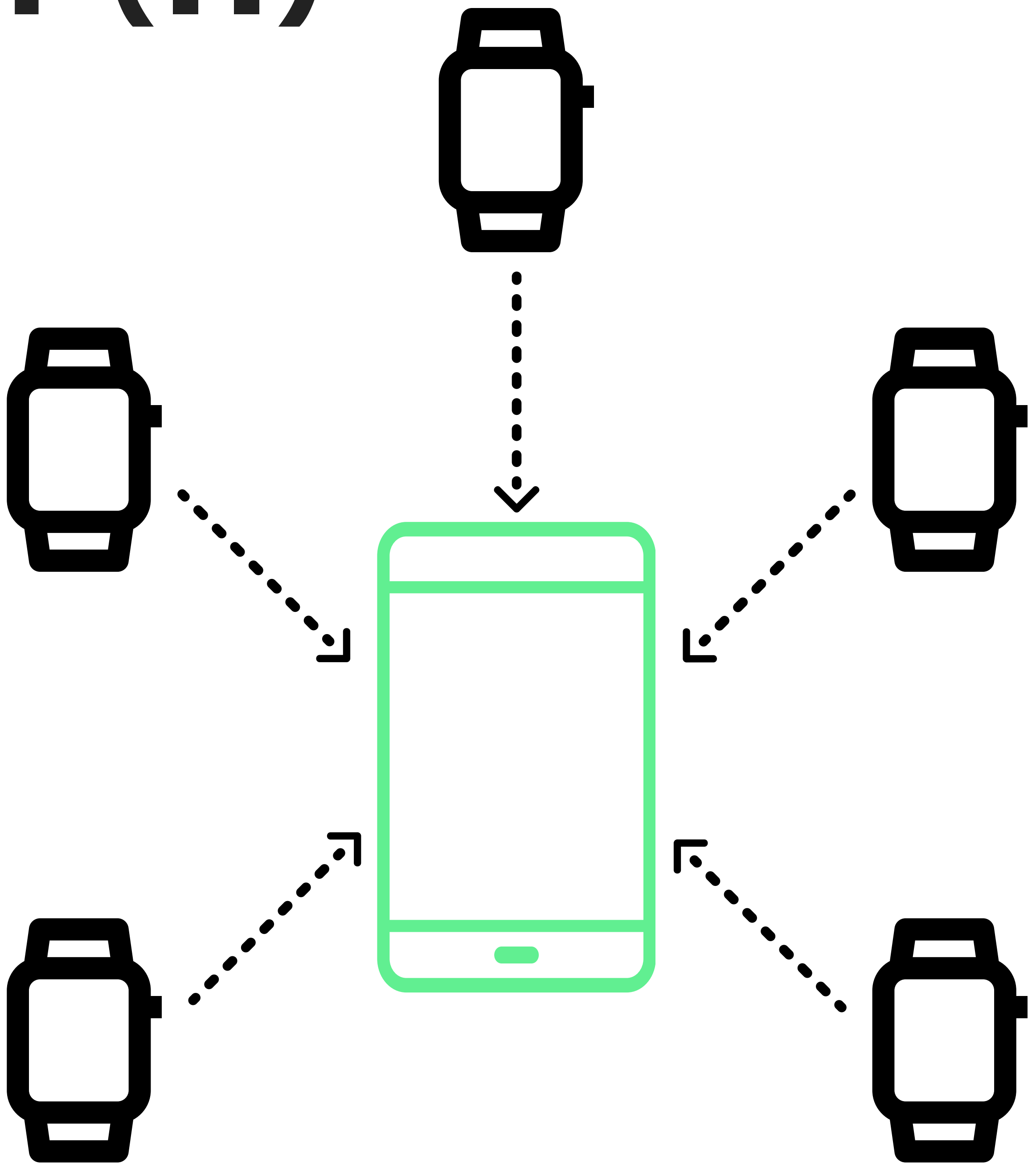
- Smartwatches are used as information gathering instruments
- No need to carry paired phones during the training session
- Wearables could provide more sensor data

Our solution (II)

2

Data sharing

- Change the one-to-one perspective into a one-to-many perspective
- Multiple smartwatches share sensors' data with a central device/node
- A trainer can read useful real-time data from the central device and take decisions about them

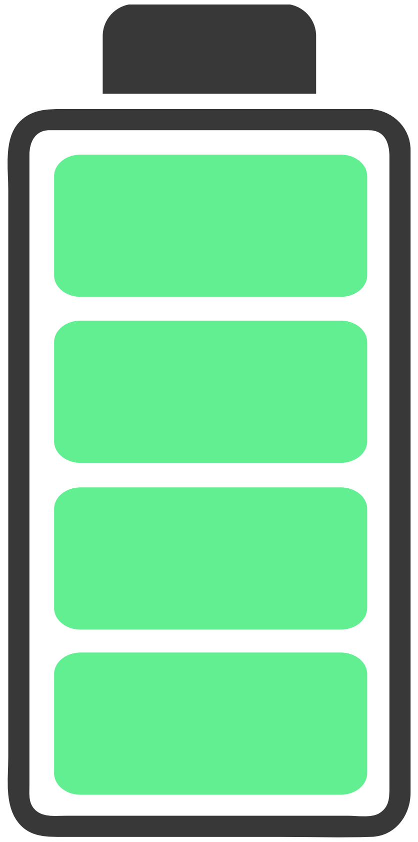


Our solution (III)

3

Battery saving & context awareness

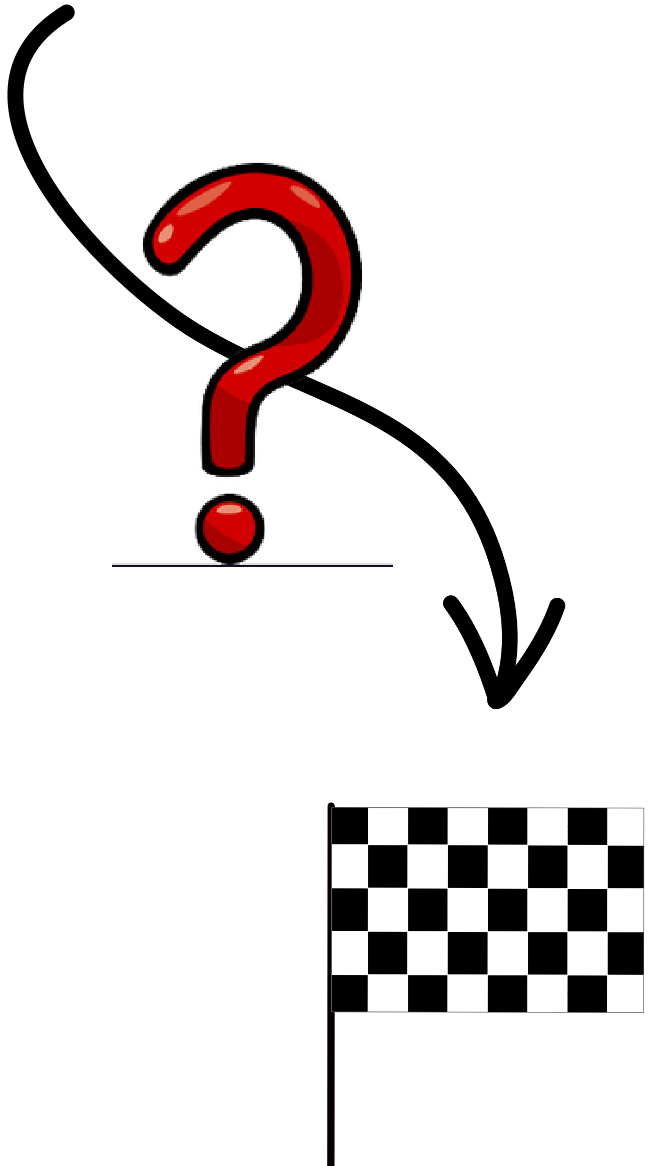
- Avoid internet connection to send data
- Stop sensors' data gathering when the athlete stops its activity



2. Challenges and Design Choices

START

- Build a wearable application
- Exploiting a star topology network
- Real-time monitoring of athletes' activity



Preliminary work on available literature

01

Nearby Connection API

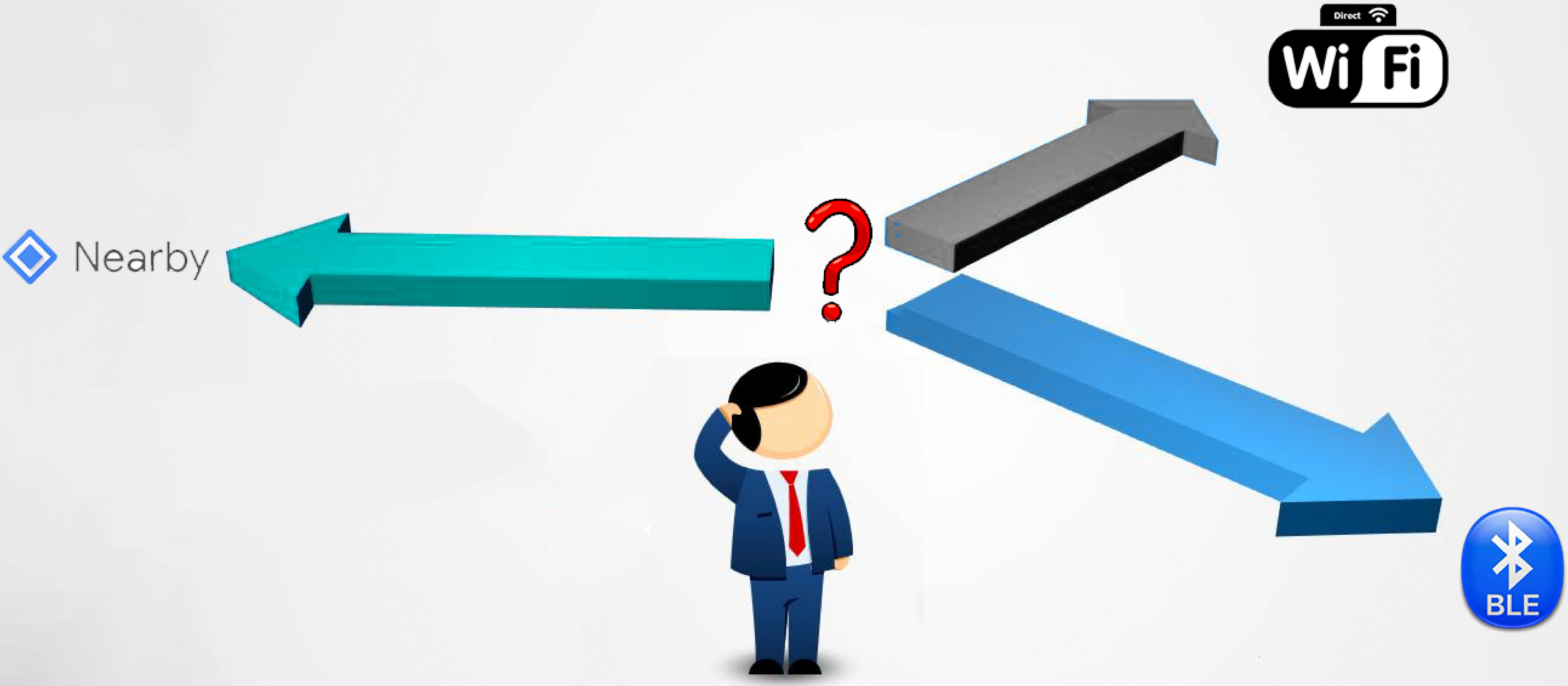
02

Bluetooth Low Energy

03

Wi-Fi Direct

Which one to choose?



1. Nearby

PROS

- Simple
- Versatile
- Energy-efficient

CONS

- Compatibility

2.

BLE

PROS

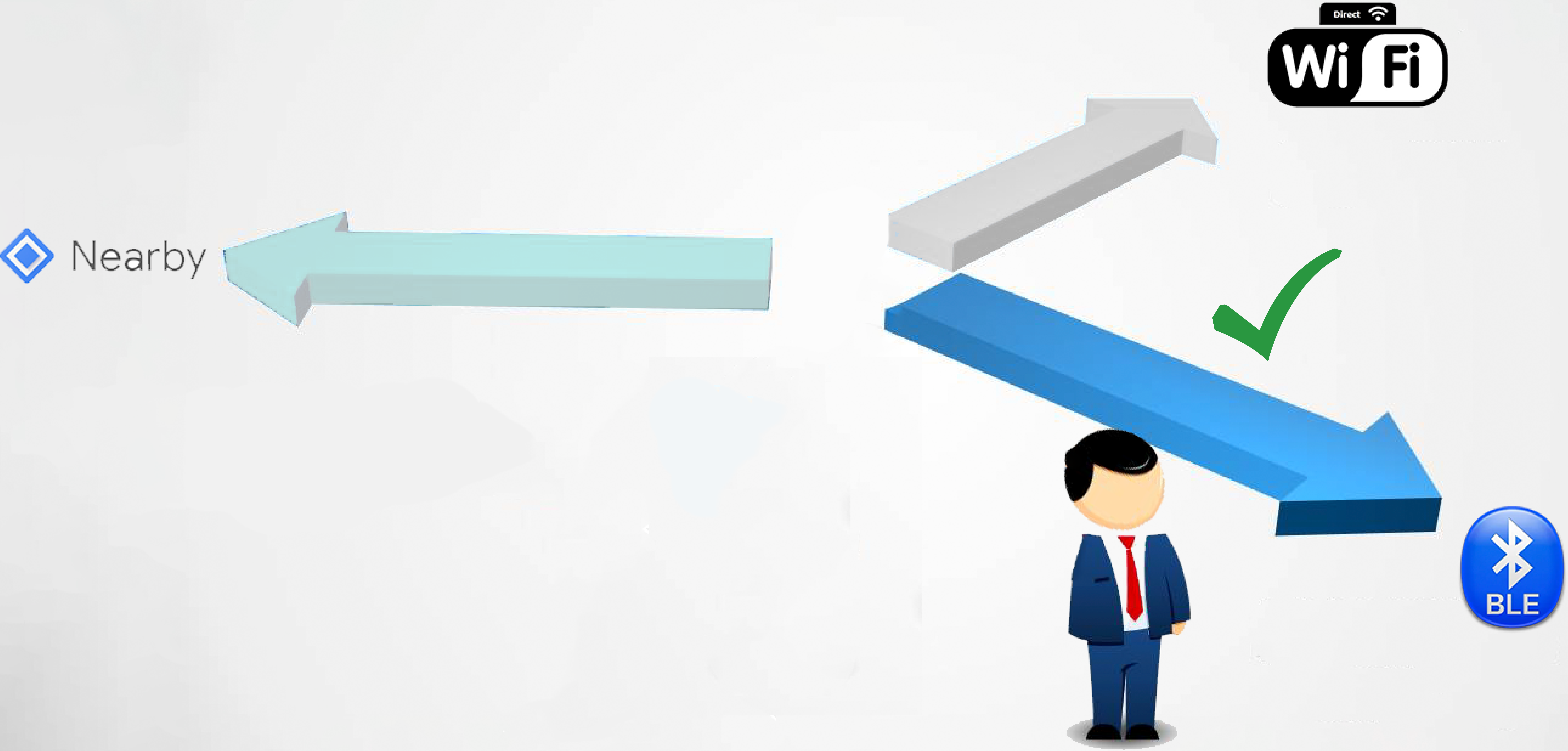
- Energy-efficient
- Compatibility
- Reliability

CONS

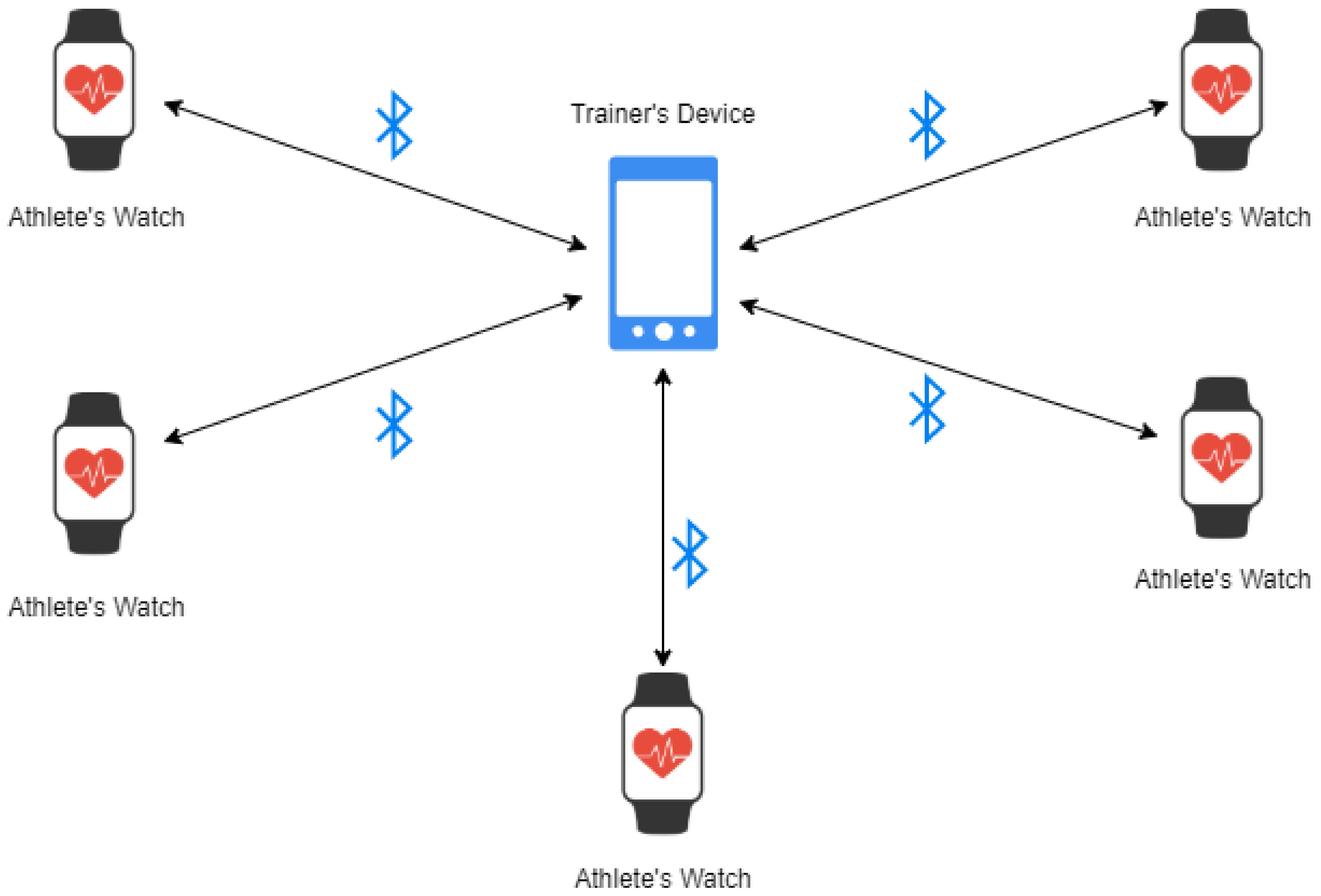
- Lower Bandwidth
- Usability
- Scalability

3. Direct Wi Fi

PROS	CONS
- Fast DTR	- Power Consumption
- High Bandwidth	- Connectivity issues


Nearby
Direct
Wi Fi
✓
BLE

SYSTEM ARCHITECTURE



3. Workout Monitoring

Sensor Choice

- Real time data gathering
- High-Level Sensors and Statistics
  - ○ Fast and accurate
  - ○ Require less data cleaning

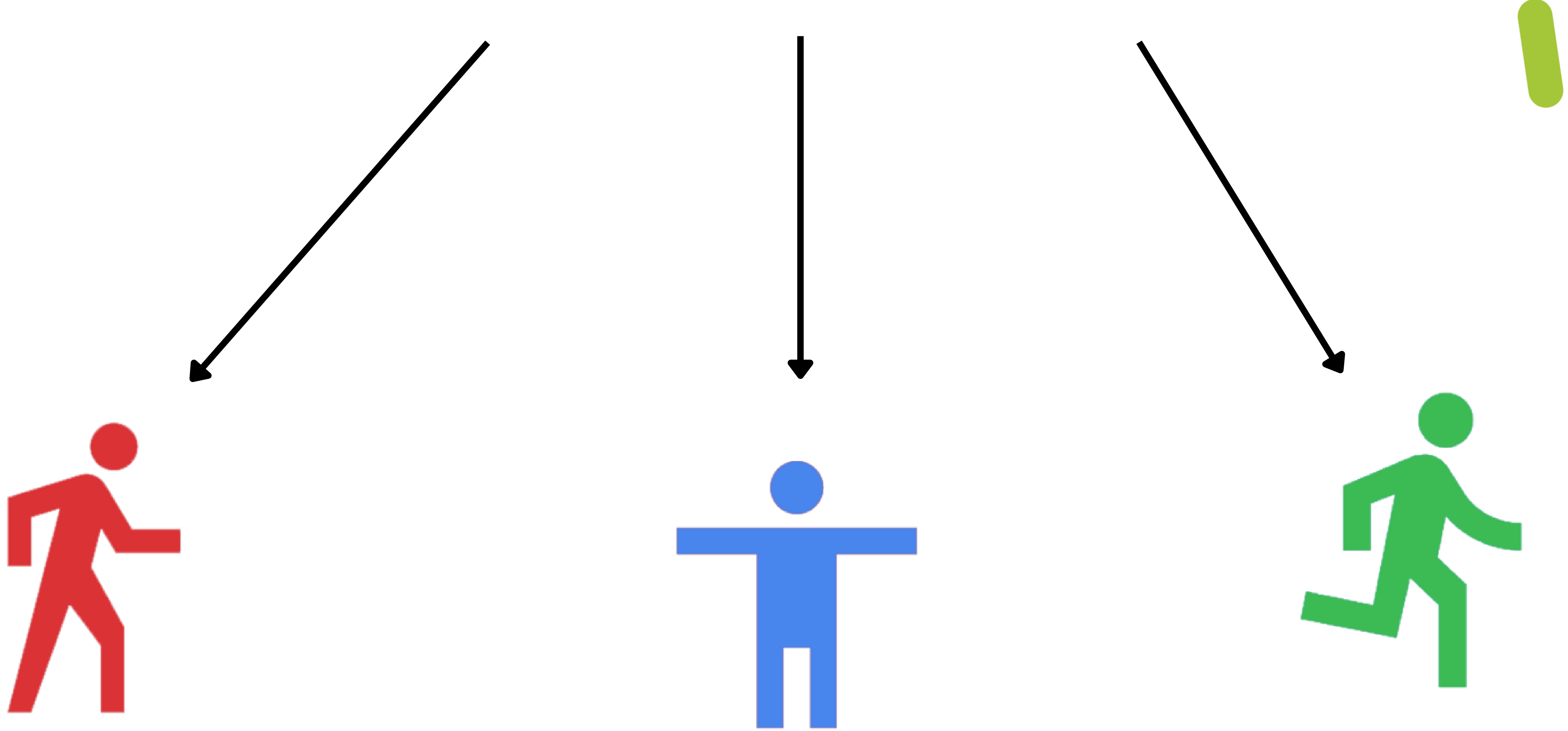
Data Collected

- Build-in Sensors used:
    - Step Counter
    - Heart Rate
- Using GPS, we computed:
    - Speed
    - Distance
    - Pace

$$Pace = \frac{Time}{Distance}$$

Data Collected

Activity Recognition using Android API



Walking

Still

Running

Context-Aware Optimizations

01

Goal:

Reduce the battery consumption of our app depending on a specific context

02

How is it achieved?

We disabled the GPS (that is energy hungry) when the user is not performing any activity and re-enabled when the user starts a new activity

Context-Aware Optimizations

The activity recognition is strictly dependent on the device used:

- Accuracy becomes low when using older devices and activity recognition mechanism becomes slower

Additional Step Counter Check

01

Save the step counter value when the user is detected as STILL by the application

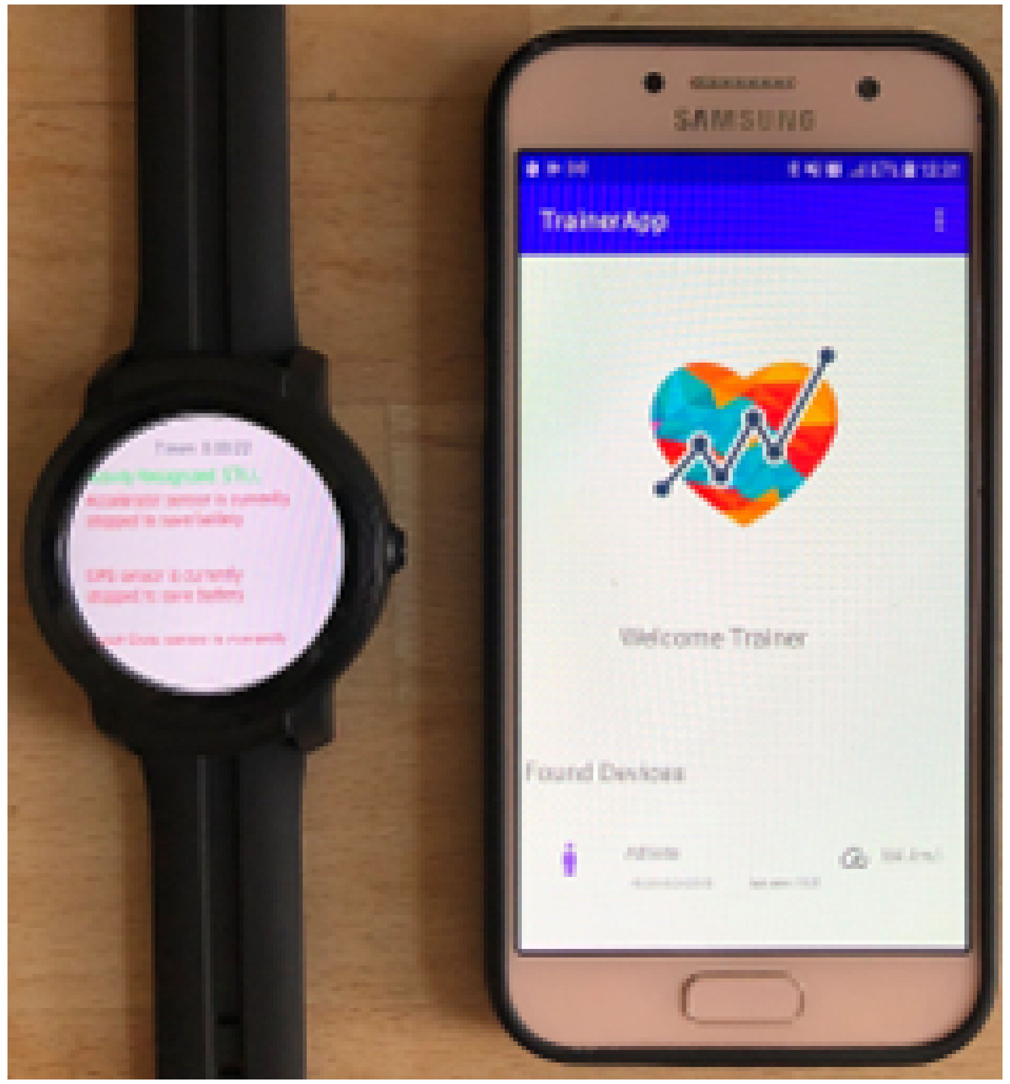
02

When the new step counter value is greater then the saved value + 50 steps. GPS is re-enabled.

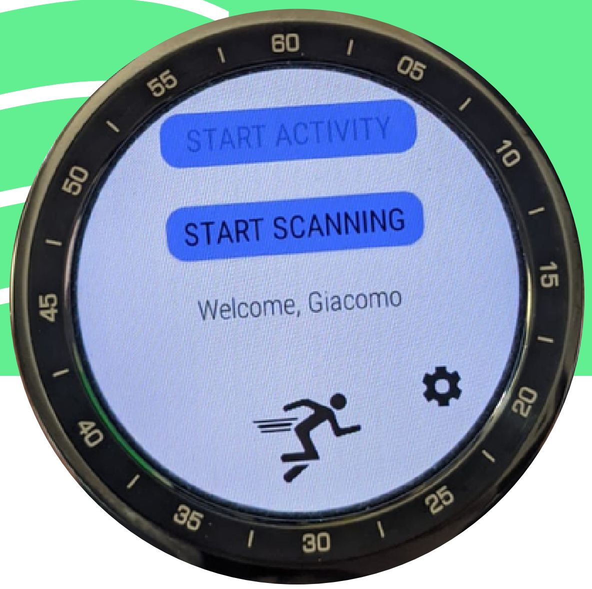
03

In this way even if there are some delays or accuracy errors due to the activity recognition, we can collect GPS data anyway

4. Implementation details



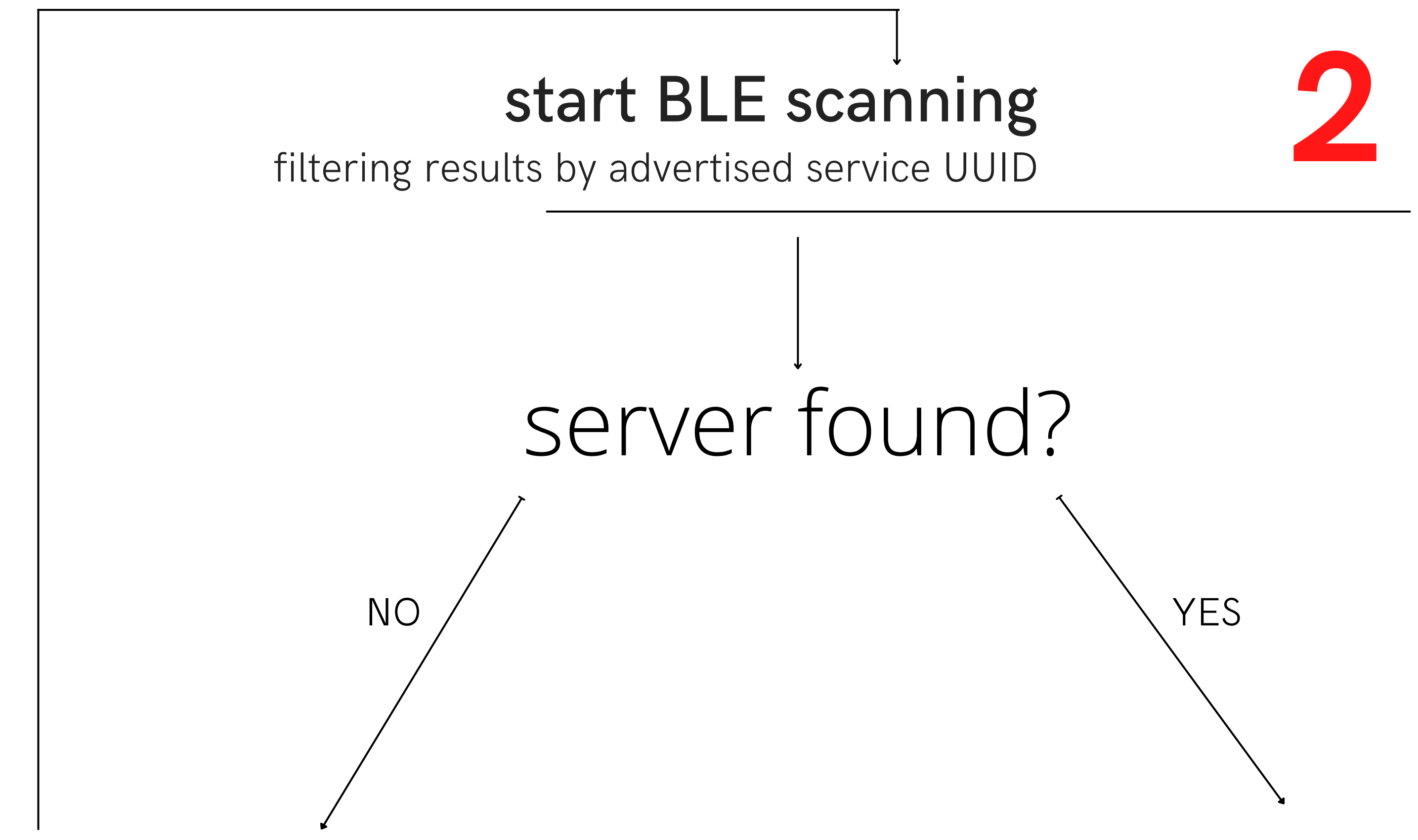
BLE connection strategy



Client side

Server side

1 **start GATT server & BLE advertising**
advertise a chosen service UUID



start BLE scanning
filtering results by advertised service UUID

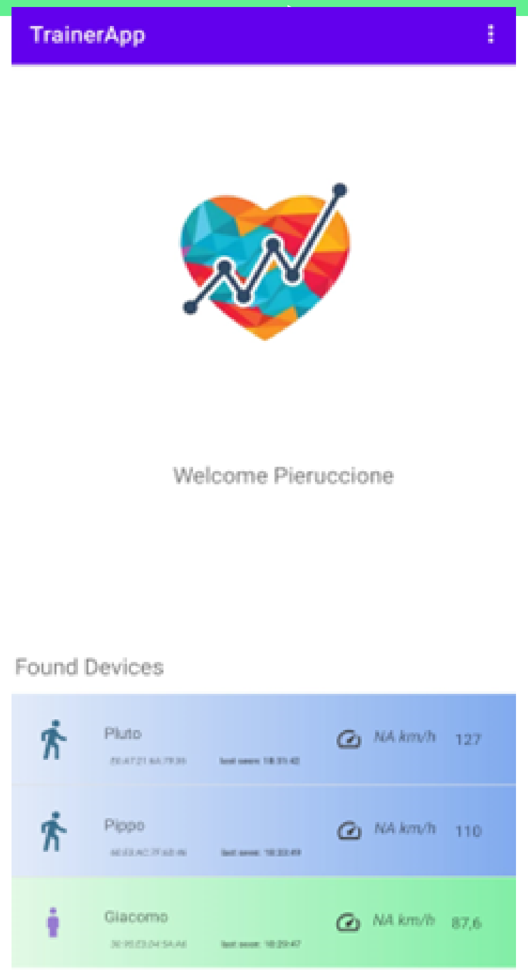
2

server found?

1. notify the user
2. show restart scanning button

Exposed GATT services:

- **AthleteInformationService**
  - AthleteNameCharacteristic
- **HeartRateService**
  - AthleteHeartRateCharacteristic
- **MovementService**
  - AthleteActivityCharacteristic
  - AthleteSpeedCharacteristic
  - AthletePeaceCharacteristic
  - AthleteStepCounterCharacteristic
  - AthleteDistanceCharacteristic



TrainerApp

Welcome Pieruccione

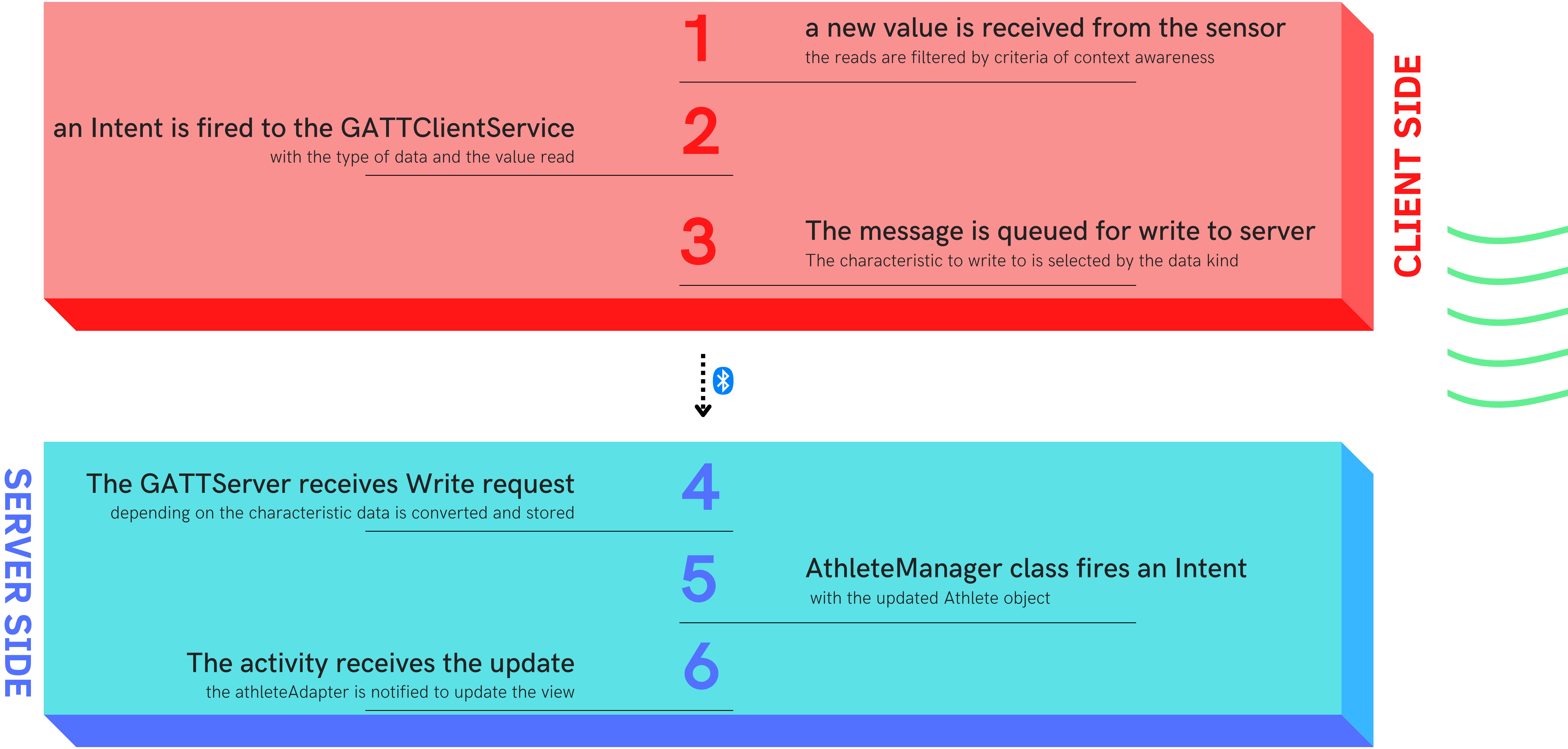
Found Devices

Pluto	NA km/h	127
Pippo	NA km/h	110
Giacomo	NA km/h	87,6

3 **connect to GATT server**
if some of the required services are missing, force service discovery

⚠ this happens because of a wrong caching mechanism in Android BLE GATT library

DATA FLOW



CLIENT SIDE

1 a new value is received from the sensor
the reads are filtered by criteria of context awareness

2 an Intent is fired to the GATTClientService
with the type of data and the value read

3 The message is queued for write to server
The characteristic to write to is selected by the data kind

SERVER SIDE

4 The GATTServer receives Write request
depending on the characteristic data is converted and stored

5 AthleteManager class fires an Intent
with the updated Athlete object

6 The activity receives the update
the athleteAdapter is notified to update the view

TRAINER-SIDE USER INTERFACE

MainActivity

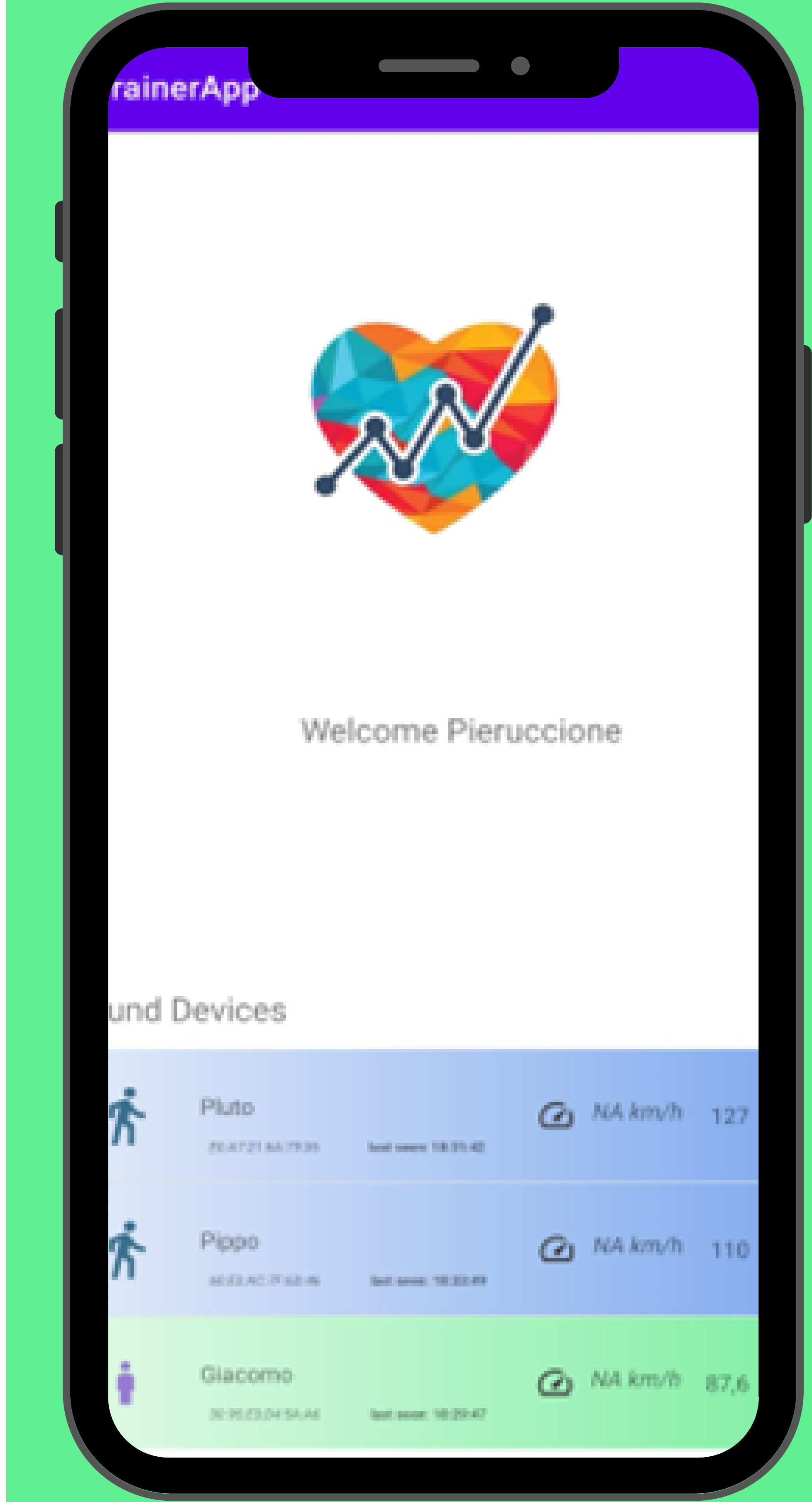
Shows a list of the athletes that joined the training session

AthleteDetailsActivity

All the historical and current statistic for the selected athlete are shown

SettingsActivity

Where the trainer can set its name



rainerApp

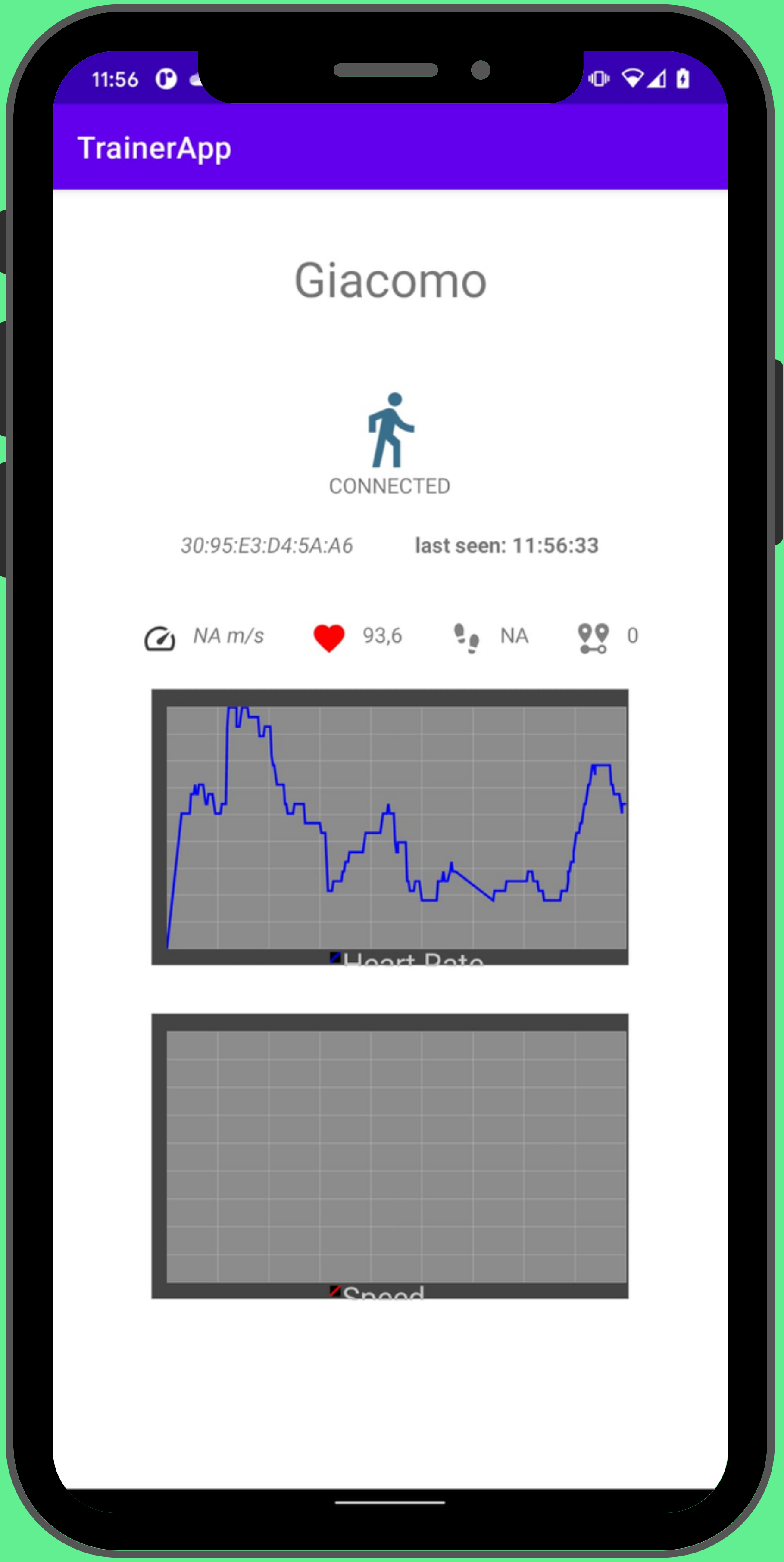
Welcome Pieruccione

und Devices

Pluto [illegible] last seen: 18:31:42 NA km/h 127

Pippo [illegible] last seen: 18:33:49 NA km/h 110

Giacomo 30:95:E3:D4:5A:A6 last seen: 18:29:47 NA km/h 87,6



11:56

TrainerApp

Giacomo

CONNECTED

30:95:E3:D4:5A:A6 last seen: 11:56:33

NA m/s 93,6 NA 0

Observations

1 **WearOS is a constrained version of android also in dev terms**

The libraries and hardware access API available are limited, not all the standard android features are granted.

2 **Android BLE GATT library is not a pleasure**

Developing over Android BLE GATT protocol standard library needs lots of testing phase to discover every network scenario, and every thing has to be managed manually.

Possible future improvements

1

Data processing

1. Improving the sensed data selection phase
2. Storing the past training data in a persistent way
3. Elaborating training summary data for each athlete
4. classifying the training quality of the athlete not by static ranges, but considering its past training data and its physical statistics

2

Network topology

In order to overcome the BLE hardware limit on the number of connected devices, it could be a solution to implement a mesh topology for the bluetooth network, in which every athlete can route the messages to the trainer

Thank you