Internet of Things

Homework: exercise n. 1

Andrea Caravano, Alberto Cantele

Academic Year 2024–25

Exercise text

Context

A logistics company operates a warehouse composed of a 500 m^2 **underground** indoor area and a 1 km^2 outdoor yard.
Electric forklifts are used across both zones and return to specific docking stations to recharge.

Objective

Design a low-cost IoT system to:

1. localize forklifts in real time
2. monitor their status, including daily distance traveled, maximum and average speed, and impact detection.

Requirements

Your design must include:

- A description of the hardware installed on each forklift, including sensors, processing unit, and communication module.
- The chosen connectivity strategy, communication protocol, and data transmission frequency.
- A backend architecture detailing how data is ingested, processed, stored, and visualized.
- Pseudocode outlining the core logic running on each forklift (data collection, impact detection, communication)
- **A graphical system-wide building block diagram** showing the architecture from edge devices to the backend platform, including key data flows and system components.

Solutions should prioritize simplicity, low cost, and scalability.
Design choices must be clearly justified.

1 Forklift hardware planning

As will be discussed later, the chosen communication technology may be a dictating factor for some hardware component: in our IoT implementation, however, most hardware devices will be independent of the communication details: we will complete the remaining elements upon planning of the data-exchange strategy.

In the following, a sketched hardware plan is outlined:

Objective	**Hardware device**
Custom monitoring and localization routines	Micro-controller and related circuitry
Localization + communication	Wireless connectivity module
Monitoring	Inertial Measurement Unit (IMU)
Impact detection	Ultrasonic distance sensor
Outdoor tracking support	Global Positioning System (GPS) sensor
Regular maintenance	Docking station adapter and integrated storage

Let's now be more precise in the description of the proposed usage for each component:

- Micro-controller board and related circuitry (e.g. ESP-32/Arduino): the processing unit and communication manager of the overall system.

  It will implement the complete monitoring and localization strategies sketched later.

- Wireless connectivity module (e.g. LoRaWAN/Wi-Fi): wireless communication parameters (e.g. RSSI) can also be used for localization through the FINGERPRINTING or TRIANGULATION technique, detailed further on.

- Intertial Measurement Unit (IMU): an IMU is defined as a 9 axis sensor that measures orientation, velocity and gravitational forces by combining an accelerometer, a gyroscope and a magnetometer.

  Its recent iterations are smaller than ever and therefore particularly well suited for micro-controllers and development boards.

  In its essence, an IMU can be used to determine the forklift's orientation and speed for monitoring.

  Note, however, that it may not be a particularly precise solution in the long run: a re-calibration and/or filtering strategy provide optimal results, especially if combined with the docking/recharging procedures.

- Ultrasonic distance sensor (e.g. HC-SR04): an ultrasonic distance sensor is an affordable distance measurement sensor, with a range from $\sim 2\ cm$ to $\sim 4\ m$. Its proposed purpose is the implementation of the impact detection routine.

  A measurement cycle is proportional to the duration of the ECHO pulse through a conversion factor.

- Global Positioning System (GPS) sensor: the GLOBAL POSITIONING SYSTEM is the well-known tracking and navigation structure, exhibiting a medium localization precision. It requires, of course, satellite coverage, which excludes the UNDERGROUND area.

  Its current implementation may not be precise enough to track forklifts in the environment: it can serve as a rough support for outdoor tracking, being also sensibly cheap.

- Docking station adapter and integrated storage: the docking station serves the purpose of providing energy to the overall system, as expected.

  The integrated storage may be flushed at recharge times to perform a monitoring comparison and complete eventually unavailable data due to communicative inabilities (collisions, congestion, ...).

2 Localization in wireless networks

Almost all location systems are composed of two parts:

- Anchor nodes: devices in known positions (e.g. GPS satellites)
- Target nodes: devices to be localized (e.g. forklifts)

Anchor nodes may transmit or receive from the target nodes, depending on the application perspective.

Localization systems may be further classified based on:

- Signal type: in our case, the most interesting setting are radio-frequency signals.
- Type of input data: in our case, the most interesting setting are power-based parameters (e.g. RECEIVED SIGNAL STRENGTH, RSS).
- Localization technique:
  1. Parametric: model-based
  2. Non-parametric: fingerprinting

2.1 Model-based RSS localization

2.1.1 Ranging

At the receiver, estimate the distance d of the transmitter given the signal strength s.
Typically, the LOG-DISTANCE PATH LOSS MODEL is used:

$$s = s_0 - 10 \cdot a \cdot log_{10}\frac{d}{d_0} + b$$

In which:

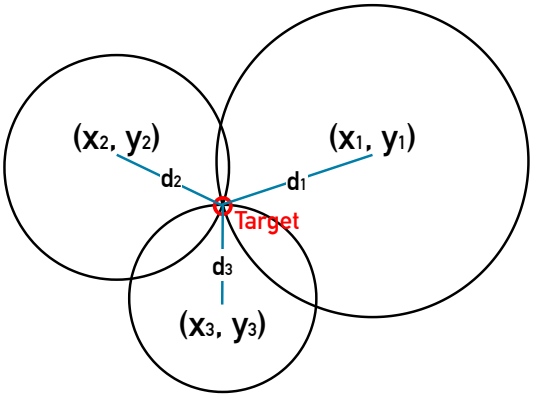
- s_0 is the power emitted at d_0 meters
- a is the path loss exponent, b is a noise term
- s_0, a are estimated, b is generally set to 0

And to finally retrieve d:

$$d = 10^{\frac{s_0 - s}{10 \cdot a}}$$

2.1.2 Triangulation

The target estimates distances $d_1, d_2, \ldots, d_n$ from n anchor nodes in positions $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$.



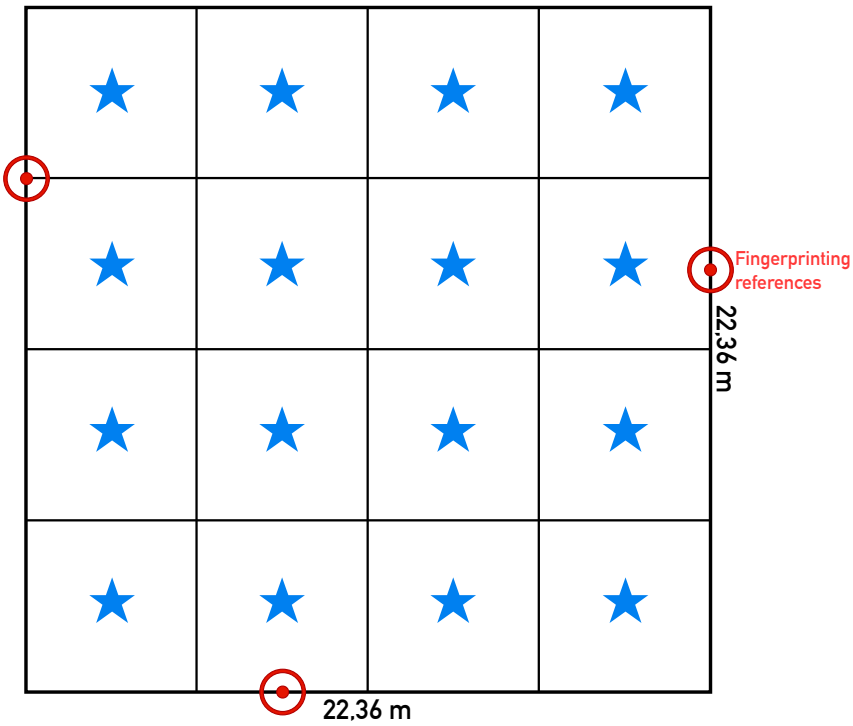
Target position is the intersection of the (at least three, of course) circumferences having $d_1, d_2, \ldots, d_n$ as radius and $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ as their centers.

2.2 Fingerprint-based localization

The FINGERPRINTING approach consists of two basic steps:

1. Construct a radio map (FINGERPRINT DATABASE) of the environment, storing for each position (x, y) the RSS from all visible monitoring points $(s_1, s_2, \ldots, s_n)$.
2. When the target needs to be localized, transmit the RSS from monitoring points to the database and get the position with the most similar fingerprint.

A sketched example of a FINGERPRINTING DATABASE construction on a sample underground field is shown in the following



3 Communication and connectivity strategy

3.1 Communication technologies: a brief overview

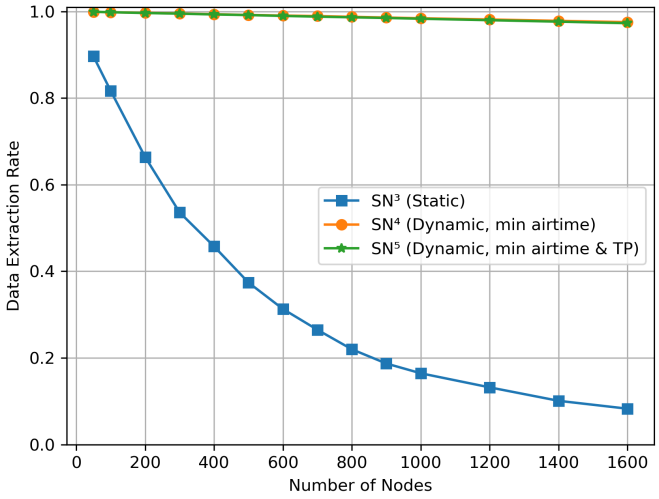
The most relevant communication technology for our case should definitely stem from the considerations about the fields' size and overall running environment.

A WI-FI or BLUETOOTH network seems difficult to fit in such an implementation scenario: being both areas quite large and mobile devices pervasive in the environment, our attention may be focused towards solutions employing LONG RANGE technologies, that also empathize on system's lifetime.

Moreover, as known, **WI-FI or BLUETOOTH traffic heavily suffers from interference with standard human usage and collisions**, especially at its lowest frequency band (2,4 GHz), being heavily employed in human activities.

Short range technologies instead, like ZIGBEE, could be adequate only if planning correctly the environment and its distribution, because of the limited transmissive power.

As inspected during the latest CHALLENGE activity, **LORAWAN deployments can hugely benefit from optimizations** on airtime through configuration of transmit parameters, as shown by the following experimental run through LORASIM, that exhibits a DATA EXTRACTION RATE (DER) $\geq 0{,}9$ in highly polluted transmission environments, while also covering considerable distances ($\sim$ hundreds meters).



Also, being the **outdoor yard much larger than the underground one**, a long-range technology has the additional benefit of requiring a single technical and communicative deployment, rather than replicating it with a shorter range one, that could, however, provide additional benefits and simplicity to the underground area, as mentioned.

An ideal fit for such a layout is therefore represented by LoRaWAN or LoRa-based custom point-to-point technologies.

3.2 The Things Network and The Things Stack

The Things Network (TTN) is an open-source infrastructure (and network server, the most intelligent part of a LoRa network) aiming at providing free LoRaWAN network coverage.

The project is actively maintained by a growing community across the world, on a voluntary basis.

Gateways are incrementally being deployed across the globe, providing an increasing network coverage.

The Things Network plays, more or less, the same role that a mobile radio network operator plays for mobile User Terminals: it provides the user with the connectivity and managerial services of their LoRaWAN provisions.

An implicit assumption of the underground environment, however, is that network coverage cannot be directly provided by The Things Network's gateways: we will probably have to install a new one ourselves, also exploiting mains power being freely available.

The gateway will also need network connectivity to forward collected measurements to either a local or remote data broker (e.g. MQTT).

- If cable connectivity (e.g. Ethernet) is a viable option, we could directly exploit it and forward collected data to a remote processing broker, similarly to the inspected ThingSpeak implementation.
- If instead, only mains power is directly available, we could also add a LoRaWAN Network Server.

  Thankfully, the organization behind The Things Network open sourced its Network Server implementation (`"The Things Stack"`), making it freely available to all interested entities for both global and small networks.

Furthermore, the hardware costs for LoRa gateways and Network Servers are decreasing more and more, becoming a quite affordable solution for modern implementations.

Finally, the installation should take care of physical and communicative protection of the device and, if necessary, inspect transmission characteristics with one or multiple sink nodes.

3.3 Localization strategy

Depending on the required precision and liveness metrics of the overall system, both TRIANGULATION and FINGERPRINTING localization strategies can be applied to our case.

In the TRIANGULATION case, an additional number of anchor nodes is required in the environment: they are, in practice, inactive nodes, that just serves the purpose of providing the target with a distance estimation to perform triangulation with an adequate rate.

If mains power can also be provided to the anchor nodes, they should constitute the ideal solution.

In case mains power is not available, a maintenance strategy should be adopted in this case and the FINGERPRINTING approach would benefit from the easier implementation strategy: note, however, that the overall precision will always be affected by the accuracy of the initial FINGERPRINTING DATABASE construction.

Our assumption, in the following, is that mains power is also available for anchor nodes to perform triangulation locally to the micro-controller node: in such a case, transmission parameters can also be adapted accordingly.

Measurement rate is assumed to live in the order of tens of seconds: much greater than the LORAWAN optimized case.

The average speed of a forklift is assumed to reach, at its maximum, $\sim 10\ km/h$: an aggressive transmission rate should not be useful, in such a case.

Ultimately, optimizations on the effective machine's movement and directionality should be further taken into consideration, as shown in the final pseudocode snippet.

A small VERSIONING SYSTEM should therefore be simulated by the data management procedures.

3.4 Data-exchange approach

As shown during the CHALLENGE activities, both MQTT and HTTP or COAP protocols are adequate for our case: they are all normally employed in the IoT field and, since our expected payload size would not represent a problem for either of those, all three can be quite equally chosen.

An ideal fit should therefore mimic the scenario inspected when using THINGSPEAK, whose aim, among data collection, also resides in powerful data processing through MATLAB technologies.

Our assumption, in the following, is therefore to use the MQTT or MQTT-SN protocols, that also put an additional strength towards the IoT scenario, also optimizing on message sizes and transport management.

4 Backend data processing and storage solution

A forklift node exchanges planning is provided in the following, correlated by an ideal data management pipeline.

- RSSI: integrated in all LoRaWAN messages, it is used to perform triangulation through the model-based localization approach described.
- Directionality, speed and orientation: collected by the Inertial Measurement Unit (IMU), they are used also to implement a sketched versioning system, detecting meaningful changes for updates.
- Distance from obstacles: collected by the Ultrasonic distance sensor, they are used to implement the impact detection strategy, which also serves the purpose of providing the maintenance teams with useful insights about potential impacts.
- Additional tracking and status support information: collected, for example, by the GPS communication module if GPS coverage is found to be sufficient (e.g. in the outdoor yard).

  Interesting additional pieces of information are represented, for example, by a periodic battery status update, to plan for docking stations usage.

Ultimately, upon docking/charging procedures, an internal storage flush can be performed, to complete monitoring outcomes and solve eventually unavailable measurements due to communicative inabilities (collisions, congestion, ...).

Depending on LoRa coverage and cabling availability, both the Gateway and Network Server can be deployed remotely, exploiting, for example, The Things Network's provisioning, as mentioned.

Our assumption, in the following, is that connectivity to The Things Network's Network Server is available through appropriate cabling, while the LoRa Gateway should be deployed locally, due to poor coverage of the underground area.

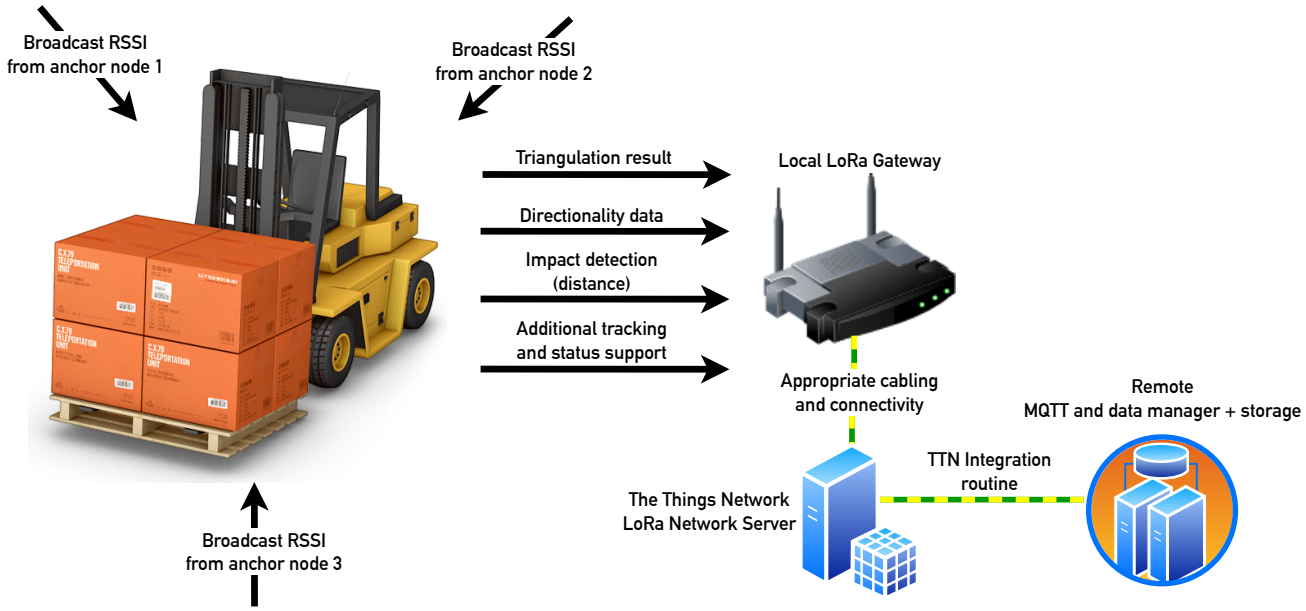
The MQTT data broker can be deployed both locally or remotely, depending, most likely, on the bandwith availability and storage requirements of the overall solution.

Our assumption, in the following, is that the MQTT data broker is deployed remotely, through a cloud service provider, that can also provide for additional reliability and scalability of the overall solution.

The cloud service provider will therefore also manage data storage and maintenance, presumably replicating collected measurements both locally to the datacenter, through RAID storage solutions and consequently distributing it in a geographically-meaningful fault tolerance strategy.

5 Overall system architecture

The resulting overall system's architecture sketch is provided in the following, alongside the planned delivery paths and the assumed implementation details.



The described solution ultimately empathizes on:

- Simplicity: no more than a micro-controller/development board per forklift should be expected.

  A single LoRa gateway is expected to cover the whole underground area. Depending on the physical characteristics and coverage of the warehouse, additional ones could be required for the outdoor yard, if The Things Network's coverage is not expected.

  Processing details are covered by The Things Network's integration and a supporting cloud provider, which resolves the need for additional processing and data storage management.

  It finally ensures a reasonable accuracy for both the localization and monitoring requirements, through the strategically positioned anchor nodes.

- Low cost: LoRa end-nodes (micro-controller/development boards) and Gateways are becoming more and more affordable, though they may require an additive term due to the proprietary modulation scheme.
- Scalability: As also inspected during the latest Challenge activity, LoRaWAN deployments can hugely benefit from optimizations on airtime through configuration of transmit parameters, as shown by the experimental run through LoRaSim, that exhibits a Data Extraction Rate (DER) $\geq 0{,}9$ in highly polluted transmission environments, while also covering considerable distances ($\sim$ hundreds meters).

  The outdoor yard, as mentioned, may require additional LoRa Gateways, to both ensure additional stability and localization accuracy, via triangulation.

6 Forklift logic: pseudocode snippet

```
#include <IMU.h>  // Inertial Measurement Unit (IMU) library
#include <LoRaWAN.h>

LoRaModem modem;

void setup() {
    Serial.begin(115200);

    modem.begin(EU868);
    modem.joinOTAA(appEui, appKey);

    // additional transmission parameters...
    // in case of a custom gateway implementation, transmission intervals can be
    // lowered...
    Serial.println("Setup completed!");
}

bool computeImpact() {
    digitalWrite(PIN_TRIGGER, LOW);
    delayMicroseconds(2);
    digitalWrite(PIN_TRIGGER, HIGH);
    delayMicroseconds(10);
    digitalWrite(PIN_TRIGGER, LOW);

    // Read the pulse duration and compute the proportion:
    int duration = pulseIn(PIN_ECHO, HIGH);

    float distance = duration / 58.0;  // floating point division

    return (distance > DISTANCE_DISCRIMINANT);
}

const ANCHOR_NODES = 3;  // number of anchor nodes (>= 3, see theory)
float triangulation_RSSI[ANCHOR_NODES];
float previous_position[2];  // (x, y)
float current_position[2];

float IMU_accelerometer_measurements[3];
float IMU_gyroscope_measurements[3];
const PREVIOUS_OUTCOMES_VERSIONS = 5;  // versioning system implementation
```

```
float previous_IMU_outcomes[PREVIOUS_OUTCOMES_VERSIONS];
float current_IMU_outcome;

Forklift forklift;
int assigned_docking_station;

void loop() {
    enum GPS_coverage = check_GPS_coverage();
    if (GPS_coverage == GOOD) {
        current_position = get_GPS_position();
    } else {
        // Perform triangulation
        for (int i = 0; i < ANCHOR_NODES; i++) {
            // collect the RSSI from broadcast anchor nodes' transmissions
            triangulation_RSSI[i] = collect_RSSI(i);
        }

        // And finally compute an output position (x, y)
        current_position = triangulate(triangulation_RSSI);
    }

    // Let's now use the Inertial Measurement Unit to collect directionality
    IMU_accelerometer_measurements[0] = IMU.getAccelerometerOffset(X_AXIS);
    IMU_accelerometer_measurements[1] = IMU.getAccelerometerOffset(Y_AXIS);
    IMU_accelerometer_measurements[2] = IMU.getAccelerometerOffset(Z_AXIS);
    IMU_gyroscope_measurements[0] = IMU.getGyroOffset(X_AXIS);
    IMU_gyroscope_measurements[1] = IMU.getGyroOffset(Y_AXIS);
    IMU_gyroscope_measurements[2] = IMU.getGyroOffset(Z_AXIS);

    // And finally compute a meaningful outcome from the IMU
    current_IMU_outcome = compute_IMU_model(IMU_accelerometer_measurements,
                                            IMU_gyroscope_measurements);
    // In which we assumed a meaningful mathematical model describing the IMU's
    // measurements have been collectively used to come out with a final outcome

    // Let's implement a sketched versioning system: we will first compute the
    // average of the previous 5 measurements coming out of the IMU
    float previous_IMU_avg = 0;
    for (int i = 0; i < PREVIOUS_OUTCOMES_VERSIONS; i++) {
        previous_IMU_avg += previous_IMU_outcomes[i];
    }
    previous_IMU_avg /= PREVIOUS_OUTCOMES_VERSIONS;

    // and check for meaningful differences coming out of the IMU, impact
    // detection, triangulation or GPS tracking
    bool meaningful_difference_IMU =
        compare_IMU(previous_IMU_avg, current_IMU_outcome);
    bool meaningful_difference_position =
        euclidean_distance(current_position, previous_position);
    bool impact_detected = computeImpact();
    bool battery_status = forklift.battery.getStatus();

    if (meaningful_difference_IMU || meaningful_difference_position ||
        impact_detected) {
        msg = {
```

```
            IMU_outcome = current_IMU_outcome,
            position = current_position,
            impact = impact_detected,
            battery = battery_status
        }

        modem.beginPacket();
        modem.print(msg);
        modem.endPacket(true);
    }

    // And finally update local storage for the next versioning cycle
    update_local_state(current_IMU_outcome, current_position, impact_detected,
    ↪ battery_status);

    // return back to the docking station if battery status is poor
    if (battery_status == LOW) {
        forklift.move(assigned_docking_station);
    }

    delay(...);
}
```