UNIVERSITÀ DI PISA

Progetto Internet of Things

Medical Monitoring

Secondulfo Valerio – Matricola: 475827

Index

Introduction 3
Architecture 3
Implementation 4
Message format 4
MQTT 4
CoAP 5
Application side 6
Fault Detection Procedure 7
Machine Learning Model 10
MySQL Database 11
Grafana 12
Conclusions and improvements 13
References 13

Introduction

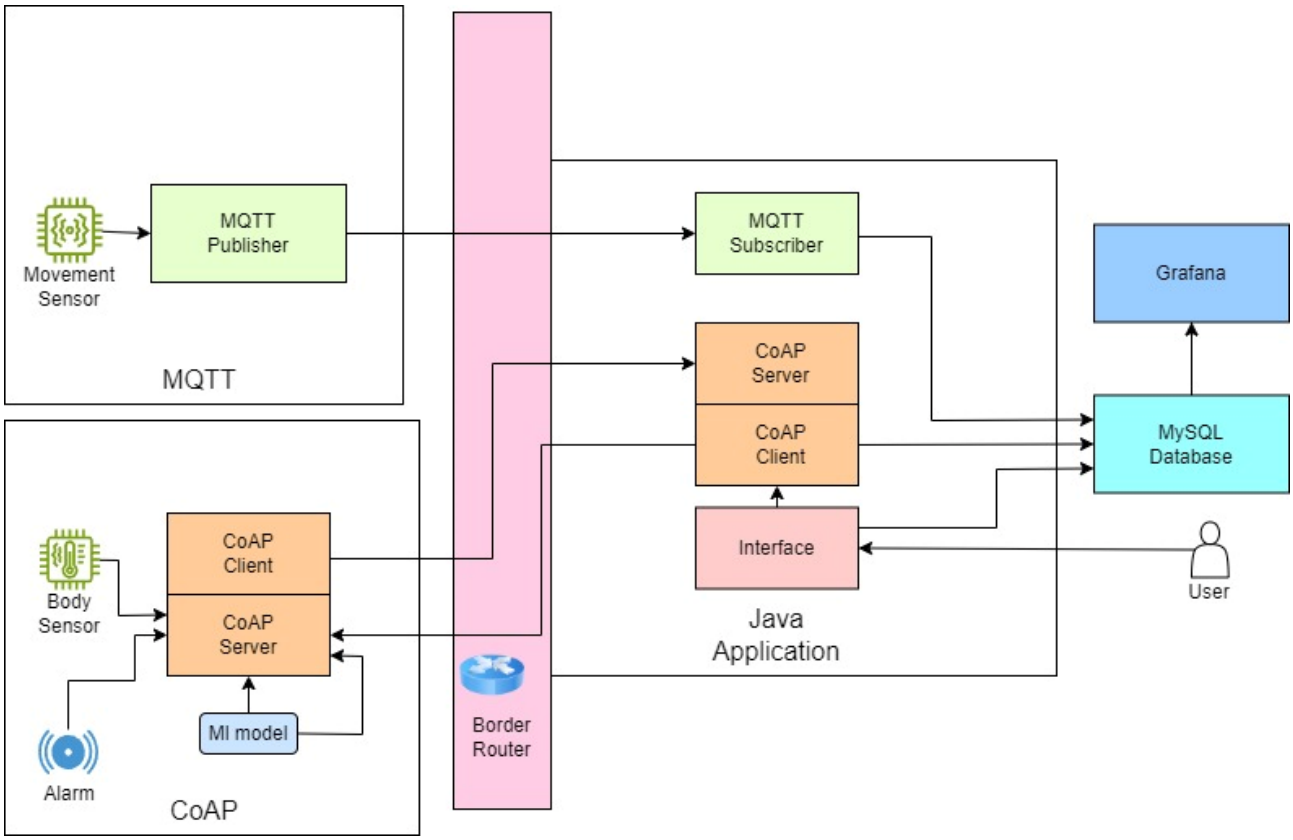
One of the fields covered nowadays for smart objects is the medical one.

Elderly people, especially, needs constant care and attention and, for that, the use of smart objects to control their status and activity may be very useful and, in some cases, save lives.

This project aims to create two sensors for monitoring patient movements and some body values such as heartbeat, electrocardiographic measurement at rest, blood pressure and many others in order to detect accidental falls by the patient or heart problems. In one, or both, these cases it will trigger an alarm. The values of the sensors are saved in a Database to keep track of the status of the patient. Along with that, a simple control java application has been developed to let a user to check the patient status.

Architecture

The architecture is shown in the next UML diagram.



We have a sensor for monitoring the movements of the patient, this sensor uses MQTT protocol for communication.

The second sensor monitors the body values of the patient such as Blood pressure, Heartbeat Rate, Electrocardiogram at rest and Fasting Blood Sugar FBS. In the sensor is implemented a machine learning model that can predict the heart status of the patient using the sampled parameters.

The CoAP node is used also as an actuator to trigger an alarm is a fall or a heart deses is detected.

Both MQTT and CoAP communicate with a Java Application through a border router.

MQTT, CoAP and Border Router have been implemented into three different Nordic Semiconductor nrf52840 dongles.

The Java Application subscribes to a MQTT topic to listen for MQTT sensor data and is used both as a CoAP Server to let the CoAP sensor registration into a MySQL database and as CoAP Client in order to send commands to the CoAP Actuator to set off the alarm and get the last sampled values from the sensor.

The Java Application communicates with a MySQL database to store sensors values and to retriever data for the user.

A simple user interface is also implemented to let the user read the patient status, the body values and to turn off the alarm in case of false warnings.

The MySQL DB communicates with the web application Grafana which shows some interesting metrics about the patient stats retrieved.

Implementation

Message format

For both MQTT and CoAP, JSON has been chosen as message format due to it's compatibility with multiple programming languages and platforms.

Along with that, it's simple to integrate in IoT devices, more human readable and less verbose than XML.

Considering the needs of the application, a lighter message structure like JSON was chosen over XML due to its efficiency in communication, reducing the burden on the limited processing power and memory of the devices.

XML's main advantage is its support for validation schema; however, for this application, data validation was less stringent so a flexible and lightweight approach like JSON is preferred.

MQTT

The MQTT sensor is used to generate 6 values related to the linear and angular values generated by the patient during his movements or at rest laying on a surface.

These 6 values are related to the x y z axis for both linear and angular values and will be used to calculate the corresponding linear and angular acceleration.

Before starting to generate the values, the MQTT node connects to the boarder router and if the connection is established it starts to send the values through the following example of JSON format:

```
{
        "app": "MedicalMonitoring"
        "Patient_ID":1
        "xla":  1
        "yla":  1
        "zla":  1
        "xaa": 2
        "yaa": 2
        "zaa": 2
        “MAC”: “xxxxxxxx”
}
```

- App: is used in order to allow multiple applications coexist in the same environment.
- Patient ID: refers to a single patient to have multiple patients monitored at the same time.
- XYZ la values: refers to the linear acceleration on the 3 axis.
- XYZ aa values: refers to the angular acceleration on the 3 axis.
- MAC: is used to associate the values registered to a unique sensor in order to guarantee also fault tolerance.

MQTT node uses a button pressor to trigger two 2 different modalities with witch it generates the movement values:

- GREEN LIGHT: Standing mode, it generates values for the patient on his feet while moving.
- RED LIGHT: Laying mode, it generates values for the patient laying on a surface.

The switching between these 2 modalities is used to simulate a variation in the movements that triggers a fall alarm.

CoAP

CoAP node is used both as a client and as a Server.

On the client side, before starting any action, the node tries to register itself on the CoAP server sending a message to the java application containing it’s IP address, along with the registration, the node IP is saved in a dedicated table in the Database in order to use it for further communications and also for distinguish it with different nodes (even if in this project there’s only one).

Doing so the Java app will also be able to retrieve the sensor IP in order to use it to send messages to the CoAP node.

Along with that, the node generates body values sampled using different monitoring sensors on the patient to keep constant track of his status.

These samples alternate their values over the time to simulate both a normal status and a heart disease status in order to allow a complete simulation of the capacities of the whole created system.

In the CoAP node has been implemented also a ML model trained on a small dataset of patients with many body values sampled and is used to classify the status of the patient between “normal” and “disease”.

After generating the body values, the node uses the model to classify the status of the patient and send those stats to the java application using a JSON message in the following format:

```
{
        "stat" = 1
        "trestbps" = 111
        "fbs" = 111
        "restecg" = 1
        "thalach" = 111
}
```

- Stat: is the status of the patient retrieved with the ml model classification, it can be 0 if the patient conditions are normal or 1 if the patient has a disease.
- Trestbps: Blood pressure at rest.
- Fbs: Fastng Blood Sugar.
- Restecg: Electrocardiogram at rest.
- Thalach: Cardiac frequency reached.

As regarding the CoAP actuator role, the CoAP node receives a message from the java application with both the fall calculated value and the patient body status value in a JSON format as the following:

```
{
        "command": 0
        "alarm": 0
}
```

- Command: it's a value that indicates the fall status, 0 if the patient is normal, 1 if a fall has been detected.
  It can also have a value equal to 3 that triggers the CoAP alarm to shut down.
- Alarm: it refers to the patient body status, its taken from the CoAP client JSON message and it's send back along with the fall/command value.

The alarm has 4 possible statuses:

- GREEN LIGTH: patient is stable, no fall or heart disease detected.
- BLUE LIGHT: it's used both for a fall or a body disease detection, the status can be retrieved by the client through the java app interface but also a warning message is sent.
- RED LIGHT: patient has fallen and a heart disease is detected.

Application side

The java application is written in Java, it offers both functionalities to be in communication with the MQTT and CoAP nodes and to a MySQL DB, also it implements a simple User interface in order to allow a user to do some short simple actions.

It's divided in the following classes:

- Actuator: This class is used to store mqtt values to the db, behave as a CoAP server in order to send CoAP messages to the CoAP actuator and trigger or not the alarm. It also temporaneally stores the

last retrieved body values in order to allow the client to read them quickly without asking the DB for a response. This was thought in order to have the last body values always avaiable so, in case of a DB malfunction, if no other monitoring/reading systems are avaiable, for the user could still read them.

- CoAP_Handler: This class is responsible for CoAP sensor IP registration to the Database, it handles the values retrieved from the CoAP sensors, stores them in the database and save them in the Actuator class.
- Database: the class responsible for comunicating to the DB adding new values in the corresponding tables and retrieving values from them.
- Fall_Detection: this class implements the calcuations performed using the MQTT node retrieved values to calculate if a fall has been detected or not.
- MedicalMonitoring: It's the main class, it connect the app to the Database, starts MQTT listening for the specified topic, starts the CoAP server and launch the User interface.
- MqttSubscriber: This class start the comunication with the broker connection to the 127.0.0.1 address and subscribes to topic "patient_values" which is the topic where MQTT is sending values to.
- User Control: Is the user interface, it implements 5 options:

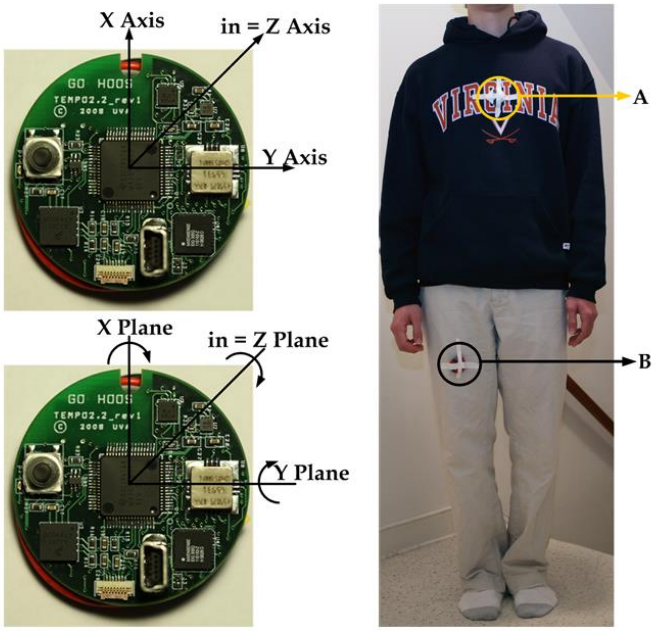
    1. Patient Info: retrieve patient standard informations from the patient info tables from the DB.
    2. Patient Heart Monitoring: retrieve the last patient body values retrieved from CoAP node.
    3. Patient Status: retrieve the last patient status detected by the CoAP node.
    4. Set off Alarm: set off the CoAP actuator alarm in case of a false warning.
    5. Exit: close the application.

Fault Detection Procedure

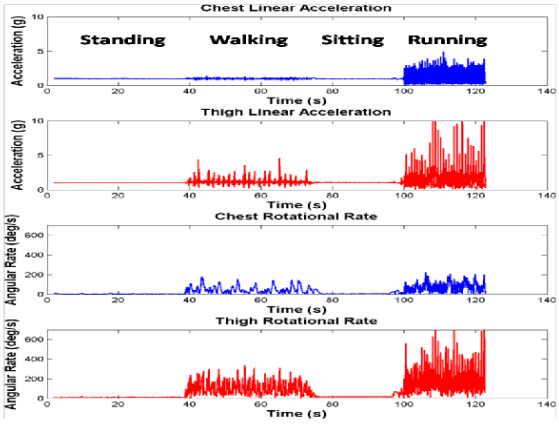
The fall detection procedure is based on the paper "Accurate, Fast Fall Detection Using Gyroscopes and Accelerometer-Derived Posture Information" by Qiang Li, John A. Stankovic, Mark Hanson, Adam Barth, John Lach from University of Virginia and Gang Zhou from College of William and Mary.

The paper presented some experiments conducted on people movements calculating their linear and angular accelerations during both the movement and laying phases.

They found a correlation between the gap of linear and angular acceleration when passing through a standing phase to a laying phase for which they determined a threshold with which classify if the passage from standing to laying was intentional or not.



The values considered in the paper were retrieved from both an accelerometer and a gyroscope positioned on the patient chest and leg.



Sampled values of linear/angular values retrieved in walking/lying phase

After several experiments the values were used to derive an algorithm with which detect a fall.

Algorithm 1: The three-phase fall detection process

▷ Monitor if people are static or dynamic during the present time segment.
if $|a_{A_{max}} - a_{A_{min}}| < 0.4g \wedge |a_{B_{max}} - a_{B_{min}}| < 0.4g \wedge |\omega_{A_{max}} - \omega_{A_{min}}| < 60\%s \wedge |\omega_{B_{max}} - \omega_{B_{min}}| < 60\%s$ **then**
    ▷ Recognize the present static posture: is it lying?
    if $\theta_A > 35° \wedge \theta_B > 35°$ **then**
        ▷ Determine if the transition before the present lying posture is intentional.
        if $a_{A_{max}} > T_{a_A} \wedge a_{B_{max}} > T_{a_B} \wedge \omega_{A_{max}} > T_{\omega_A} \wedge \omega_{B_{max}} > T_{\omega_B}$ **then**
            return *Yes*
return *No*

With TaA = 30g , TaB = 25g, TA = 200s and TB = 340s being the chosen thresholds for linear and angular acceleration on A and B sensors for detection an intentional/unintentional fall.

With linear and angular acceleration on A (chest sensor) and B (leg sensor) calculated as:

$$\begin{aligned} a_A &= \sqrt{a_{A_x}^2 + a_{A_y}^2 + a_{A_z}^2} \\ a_B &= \sqrt{a_{B_x}^2 + a_{B_y}^2 + a_{B_z}^2} \\ \omega_A &= \sqrt{\omega_{A_x}^2 + \omega_{A_y}^2 + \omega_{A_z}^2} \\ \omega_B &= \sqrt{\omega_{B_x}^2 + \omega_{B_y}^2 + \omega_{B_z}^2} \end{aligned}$$

Angular position of the patient:

$$\theta_A = \arccos \frac{a_{A_x}}{g}, \theta_B = \arccos \frac{a_{B_x}}{g}$$

Used to the detect the patient status:

θ_A (deg)	θ_B (deg)	Posture
< 35	< 35	Standing
> 35	< 35	Bending
< 35	> 35	Sitting
> 35	> 35	Lying

The application uses the same process but on only considering the Accelerometer and Gyroscope positioned on the patient leg since the sensor at disposal where limited and because on the leg we have the best detection for determining the fall event.

The calculation made for this experiment consider the variation over the last 5 seconds in the patient movements. However, in order to let the simulation be more understandable the generating timing for the MQTT values has been delayed in order to reduce the fastness of these events just for the sake of the demonstration.

Machine Learning Model

To let the CoAP node detect a body disease, a simple neural network It's been created.

The dataset used is taken from: https://archive.ics.uci.edu/dataset/45/heart+disease .

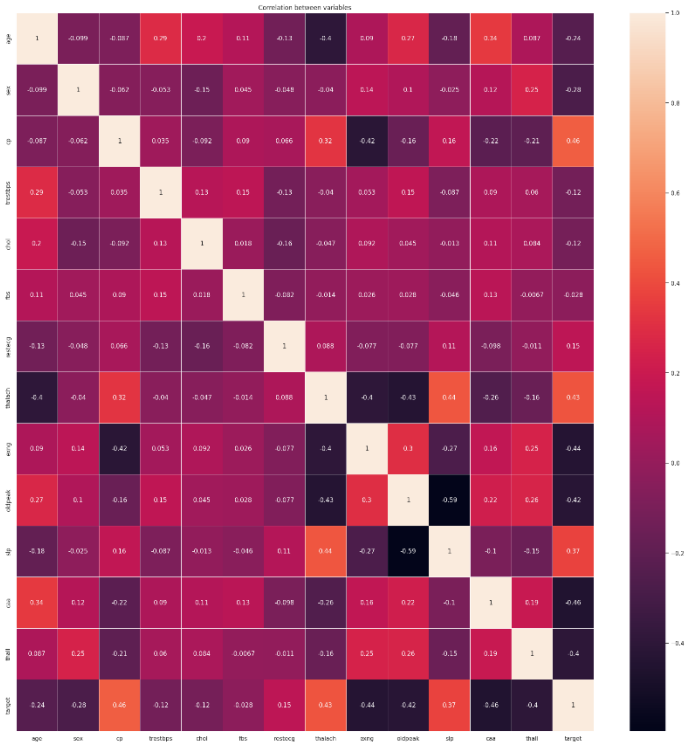
The dataset contained many files but only one was good enough to be used for the process.

The dataset is composed of about 304 patient data from Cleveland with the following attributes:

age, sex, cp, trestbps, chol, fbs, restecg, thalach, exng, oldpeak, slp, caa, thall, target

where target is 1 or 0 if the patient was considered to have a heart disease or not.

Of these 14 attributes only some of them were maintained taking in consideration that only sampleable values had to been considered and also because by applying the correlation matrix only few of them where really important for the classification.



Considering this, only Trestbps, fbs, restecg and thalach have been considered.

Even if attributes such as Cp (chest pain) were relevant in the correlation matrix, only values that could be sampled by real sensors were chosen also because chest pain is a value retrieved from the patient feelings so it couldn't be calculated in any way.

A simple preprocessing phase eliminated rows with empty or inconsistent values.

The net used had 40 neurons and was trained for 80 epochs retrieving a final accuracy around 67%, which is not an excellent value but being the dataset very small, better results couldn't have been achieved.

MySQL Database

MySQL dababase contains 4 tables:

MovementData

It contains the movements values registered by the MQTT node, is divided in topic, patient id, linear and angular acceleration and MAC value of the MQTT node, also timestamp has been included to guarantee the uniqueness of data related to the patient id.

id	Topic	Patient_ID	lastlin_acc	last_ang_acc	MAC	timestamp
1	patient_movement	1	0	1365.2402718935593	000400040004	2024-05-31 12:55:00
2	patient_movement	1	0	1280.6381221875288	000400040004	2024-05-31 12:55:03
3	patient_movement	1	0	1253.8560523441276	000400040004	2024-05-31 12:55:05
4	patient_movement	1	0	1252.5857256092295	000400040004	2024-05-31 12:55:08
5	patient_movement	1	0	1136.9890061034012	000400040004	2024-05-31 13:09:51

Bodydata

It contains the body values reported by the CoAP node, tge PatientID and a timestamp with the time in which the monitoring was executed.

id	Patient_ID	trestbps	fbs	restecg	thalach	timestamp
1	1	106	0	0	121	2024-05-31 13:09:51
2	1	127	0	0	108	2024-05-31 13:09:54
3	1	100	0	0	111	2024-05-31 13:09:56
4	1	121	0	0	112	2024-05-31 13:09:59
5	1	107	0	0	123	2024-05-31 13:16:23
6	1	122	0	0	127	2024-05-31 13:16:25
7	1	102	0	0	113	2024-05-31 13:16:28
8	1	124	0	0	104	2024-05-31 13:16:30
9	1	117	0	0	109	2024-05-31 13:16:33
10	1	129	0	0	112	2024-05-31 13:16:35

Actuators

It contains the list of CoAP nodes registered in the database, in this case only 1 is added.

Actuator_ID	Patient_ID
fd00:0:0:0:f6ce:36d4:b744:3ef3	1

Patients

The list of monitored patients. In this case only 1 is considered.

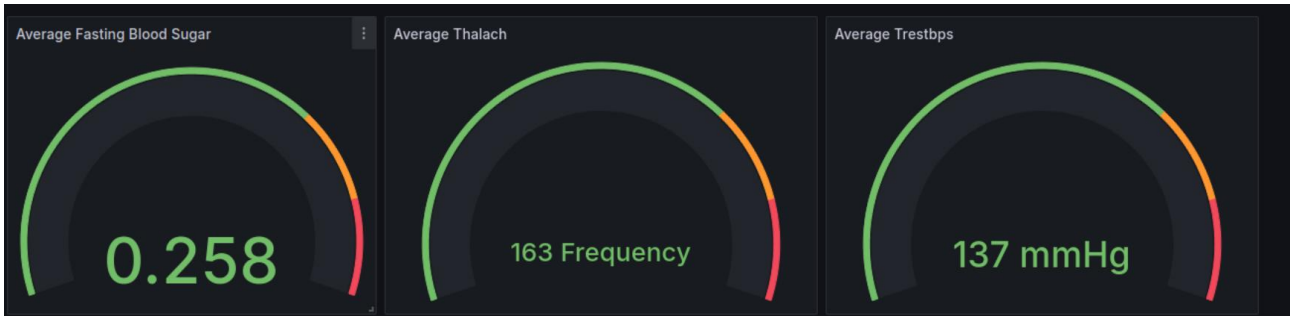
Patient_ID	Name	Surname	Age
1	Mario	Rossi	88

Grafana

In order to display some interesting statistics over the sampled values of the patient, Grafana, a web application, was used to retrieve those values on the Databases showing some graph statistics.

In the following picture are reported some examples of Grafana implementation.

In this case it was used to have an average value of the Heartbeat rate, blood pressure and fast blood sugar, along with that also the last 50 values retrieved from the patient are shown in order to let the user check the last changes of these parameters in an easy way.



Have been also reported the patient movements in order to check it's behaviour in the last 50 retrieved accelerations so if a Fall alert is reported the user could check what happened from these graphs.

Conclusions and improvements

The project can provide a simple patient monitoring sensors and application.

Even if in the practice it's not ready to be used in a real situation due to its basic construction, it is a valid starting point for a future and more solid system to be used in a real scenario.

But for doing so some improvements have to be performed.

For what concerns the ML model, it needs really more values in order to create a reliable model to be applied into the CoAP node. A possible solution for that is also to use the patient data retrieved by the same sensors even if they need first a handmade classification of the patient status first.

Along with that it can be applied also another sensor to detect also the chest linear and angular acceleration in order to stay allow a full implementation of the paper experiment.

Also, the MQTT part can be moved to a CoAP node and with another ML model implemented in it to automatically detect falls instead of doing mathematical calculations by the java application.

It can also be implemented in the alert system an automatic call to devices like phones for an ambulance or assistance call or in an hospital scenario a call to the medic or nurse pagers to request an intervention.

The monitoring can also be expanded adding more sensors to detect more variables from the patient.

References

- Nordic Semiconductor. nrf52840
- https://www.cs.virginia.edu/~stankovic/psfiles/bsn09-1.pdf
- https://archive.ics.uci.edu/ml/datasets/Heart+Disease