IoT / WI joint projects

This document describes projects that students attending both the Internet of Things (IoT) and Wireless Internet (WI) courses can choose to implement.

Main rules:

- Projects can be taken individually or in groups of maximum 2 students.
- Projects are evaluated a maximum of 4 points for both the IoT and WI courses. That is, with a single project you get a bonus of 4 points for the IoT exam and a bonus of 4 points for the WI exam.
- The evaluation takes into account the quality of the solution, cleanliness of the written code and technical depth of the final report.
- The hard deadline for selecting projects is 28/05/2021. After this deadline, students who did not choose a joint project CANNOT take it.
- The hard deadline for joint project delivery is 01/09/2021. After this deadline, no bonus points will be given.

Project 1: Finding Nemo

Nemo is a young clownfish who lives with his father Marlin in an anemone in the Great Barrier Reef. Unfortunately, on the first day of school, Nemo is captured by a pair of scuba divers.

Marlin, who is a really anxious father, had already prepared a system to find his son. He had placed 6 special anchors in the four corners of the Great Barrier Reef and installed a hacked operative system on Nemo's phone. Marlin had mapped all the Great Barrier Reef with the anchors, creating a fingerprint dataset composed of the Received Signal Strength Indicator (RSSI), a smartphone in some location of the Great Barrier Reef.

Now, he only needs to capture the probe requests coming from Nemo's smartphone and map them into his fingerprint dataset. For this task, he needs you. Would you help Marlin in this adventure?

In this project, you are requested to find the coordinates (X,Y) of the position of Nemo using the dataset of RSSI values from the six sniffer-anchors. The dataset is, however, hidden and fragmented inside MQTT publishes/subscriptions and CoAP requests packets. Every correct MQTT and CoAP request will give you a piece of the dataset, and, in the end, you will be able to reconstruct it and eventually find Nemo's coordinates. The dataset is a fingerprint dataset, and for each (X,Y) coordinates, it contains an array of RSSI values corresponding to the values captured by the anchors in that position. Once found the RSSIs emitted by Nemo's smartphone, you have to determine his location by comparing the RSSI measurement with the database's entries.

To help you, you should implement the following points:

1. Find <u>all</u> the fragments of the dataset. The more pieces you find, the easier it will be to find Nemo's positions.
2. Those fragments are hidden inside:
    a. The payload of the publications of MQTT topics
    b. CoAP GET/POST/PUT/DELETE resources response
    c. CoAP resource observe
    d. HTTP resources

3. Reconstruct the dataset and find Nemo
    a. If needed, filter, clean, remove outliers, etc…
    b. Compare Nemo's RSSI with the ones in the dataset
    c. Obtain the (X,Y) position of Nemo

<u>Useful resources:</u>

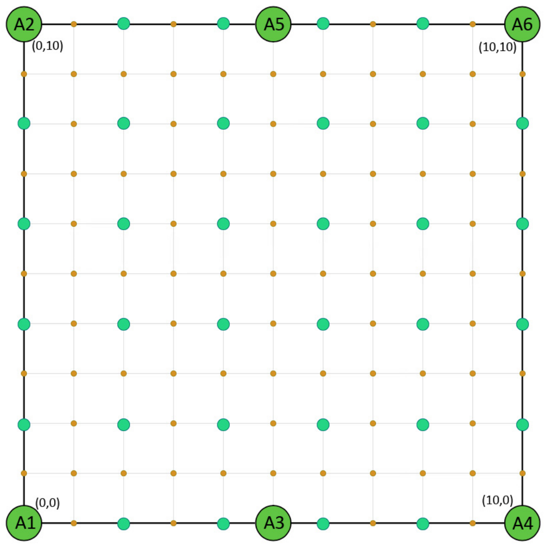
- MQTT and CoAP server address: TBA

- Anchors' coordinates:
    - A1: (0,0)
    - A2: (0, 10)
    - A3: (5, 0)
    - A4: (10, 0)
    - A5: (5,10)
    - A6: (10,10)

- An example of a frame is

X,Y Position	**A1**	**A2**	**A3**	**A4**	**A5**	**A6**
(6,4)	[-83, -43, -67, -57, -98]	[-83, -43, -67, -57, -98]	[-83, -43, -67, -57, -98]	[-83, -43, -67, -57, -98]	[-83, -43, -67, -57, -98]	[-83, -43, -67, -57, -98]

- In the dataset, you can find small green dots ● (only even positions). However, Nemo can be both in even and odd positions.



You are free to use your favourites tools (i.e. the ones seen during laboratories), your favourite programming language, editor, etc.

Final report
The final report must include the code, the fingerprint dataset, a report that explains the approach and, of course, Nemo's coordinates.

Project 2: The (hidden) Terminal

The hidden terminal problem is common to all wireless communication technology. In particular, the IEEE 802.11 (WiFi) standard uses a virtual carrier sensing methodology to overcome the problem: the RTS/CTS exchange can be in fact used to inform hidden terminals of the presence of a transmission, thus helping in avoiding collisions.

In this project, you are requested to implement a lightweight version of the RTS/CTS protocol in a Wireless Sensor Network operating on the IEEE 802.15.4 protocol. In particular, the following points should be implemented:

1. Use the WSN simulator of your choice (TOSSIM or COOJA) to simulate a network with 1 base station and 5 nodes, with at least 2 hidden terminals.
2. Implement a baseline transmission protocol: each node randomly transmits data to the base station. Each node transmits sequentially numbered packets according to a Poisson distribution with parameter Lambda.
3. At the base station, compute each node's Packet Error Rate. You should tune simulation parameters so that the hidden terminal problem is highlighted.
4. Implement the RTS/CTS mechanism. Each node willing to transmit a packet transmits first a RTS message to the base station. Nodes receiving the RTS message will stop their transmission for a fixed time of X seconds. The base station replies with a CTS message. Nodes receiving the CTS message will stop their transmission for a fixed time of X seconds.
5. At the base station, compute each node's Packet Error Rate in this case. Change the value of X to Y and repeat the test. (X and Y of your choice)

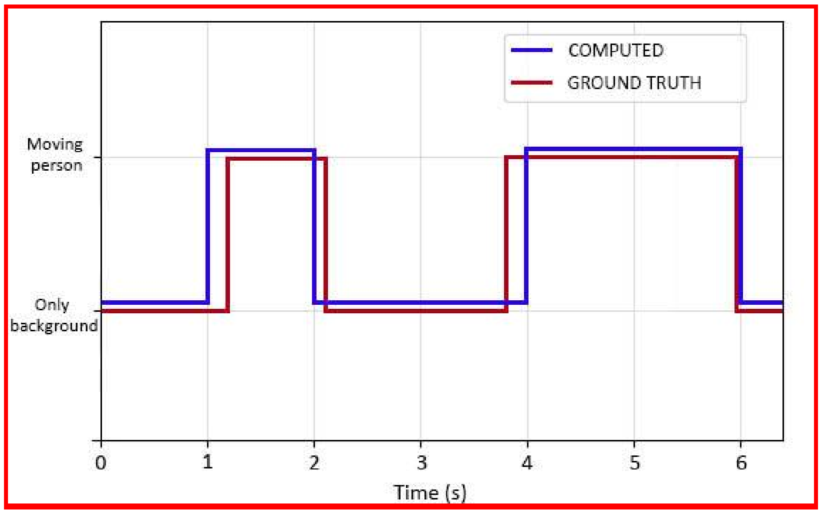
Final report
The final report must include the full TinyOS code, the log of the simulation (.log file in case of TOSSIM and some screenshot for Cooja) and a report that explains the approach and the assumptions done.

Project 3: Spy your mate

Have you ever thought about what happened in the background during a webcam call or in your smart surveillance camera? Can the network traffic reveal some useful information even if encrypted?

In this project, you are requested to analyse the traffic generated by a webcam call and, from such traffic, identify whether a person or only the background is recorded in the video. You have to approach this problem from a Machine Learning (ML) perspective.
To help you, you may follow the following approach:

1) Start a video call with your project's mate with your favourite program: skype, teams, zoom, etc. If you're doing the project alone, you can call yourself with two different devices.
2) Capture the network traffic with Wireshark and filtering all the video call packets (watch out for inbound and outbound video traffic)
3) During the call, have frames with the person in the camera and frames with only the background. Be extra careful in this process: here, you create your ground-truth labels (the real information provided by direct observation). Stay at least 1 minute in each position, and then switch it.
4) In order to record the ground truth events, you have to use Node-RED. Create a simple dashboard that allows you to store the ground-truth value (e.g. 1 a for person, 0 for no-person). Save everything in a csv file for further ML analysis.
5) Import your data (Wireshark capture + ground truth) in your favourite ML environment (pure Python, R, Matlab, Jupyter notebook, etc). It may be useful to convert your Wireshark file into a csv.
6) Play with some ML recognizer of your choice (at least two). Your final goal is to identify for each time interval whether the person was in the video or not.
7) Report your inferences in a Node-RED chart as in Figure 1 (down here).



<u>Machine Learning tips and tricks</u>:

- Aggregate wireshark packets with a time window of T seconds (T is of your choice and may be 100ms < T < 900ms). Be consistent with the time interval used in the ground truth process!
- For each time window, some interesting features to extract might be:
    - The interval between packets in ms (you can even split it in inbound and outbound)
    - Packet number (same as above)
    - Packet size (same as above)
    - Inbound/outbound rate
    - Etc.
- For each feature, you can extract average, standard deviation, median, etc..
- Eventually, you'll end up with a dataset like this (but with more features!):

Time Interval	Avg pkts interval	Inbound pkts	Outbound pkts	Avg pkt size	Person
0ms-500ms	0.384 s	37	3748	23237 byte	0
500ms-1min	0.5655 s	1239	9837	8232827 byte	1

<u>Final report</u>

The final report must include the code, the node-red export, the input dataframe (Wireshark capture + ground truth), discussion of the results and the final graph done with Node-RED. In the discussion, you are strongly encouraged to use graphics to explain your results (e.g. confusion matrix, learning curve, etc.).