Middleware Project - IoT
Prof. Mottola
Prof. Guinea

Marco Bacis 873199
Daniele Cattaneo 874757

Problem Statement

The goal of the project is to develop an IPv6-based presence detection system using MQTT:

- People roaming in a given area carry wireless sensors able to sense acceleration
- Every relevant location in the environment, e.g., every room, is equipped with a static wireless device
- This device acts as an IPv6 border router and is wired to a standard host running the IPv6 stack, e.g., through a USB connection
- The border router id acts as an identifier for the corresponding location

Requirements

- The mobile device should do everything possible to minimize energy consumption as long as the acceleration readings tell that the user is still moving
- Once the mobile device detects the user has stopped moving, it must
  - Connect to a nearby border router (if one is reachable)
  - Periodically publish an MQTT message ever K time units Over a topic that includes the id of the border router
  - With a payload that includes the mobile device id

Implementation

TODO: add introduction to the implementation (anything...)

Sensor Finite State Machine

The main sensor thread (client_process) integrates a Finite State Machine (FSM) to manage the mqtt connection based on the reachability of a IP network and the user movements. The FSM process is event-driven (being a contiki protothread), but also has a default timer of one second when waiting for long events (such as connections and disconnections). The FSM can be seen in Figure 1, which highlights the states and relevant transitions. The source and sink state is IDLE, in which the sensor doesn't perform any action apart from waiting for the user to stop moving.

When the user stops moving, the connection is initialized by turning the radio on and waiting for an IP and MQTT connection to be estabilished. After that, a json message is published every K seconds on the root border-router id (which acts as room id). The message format is {client_id, seq_number, last_accel, current_rssi, current_dbm_power, uptime}.

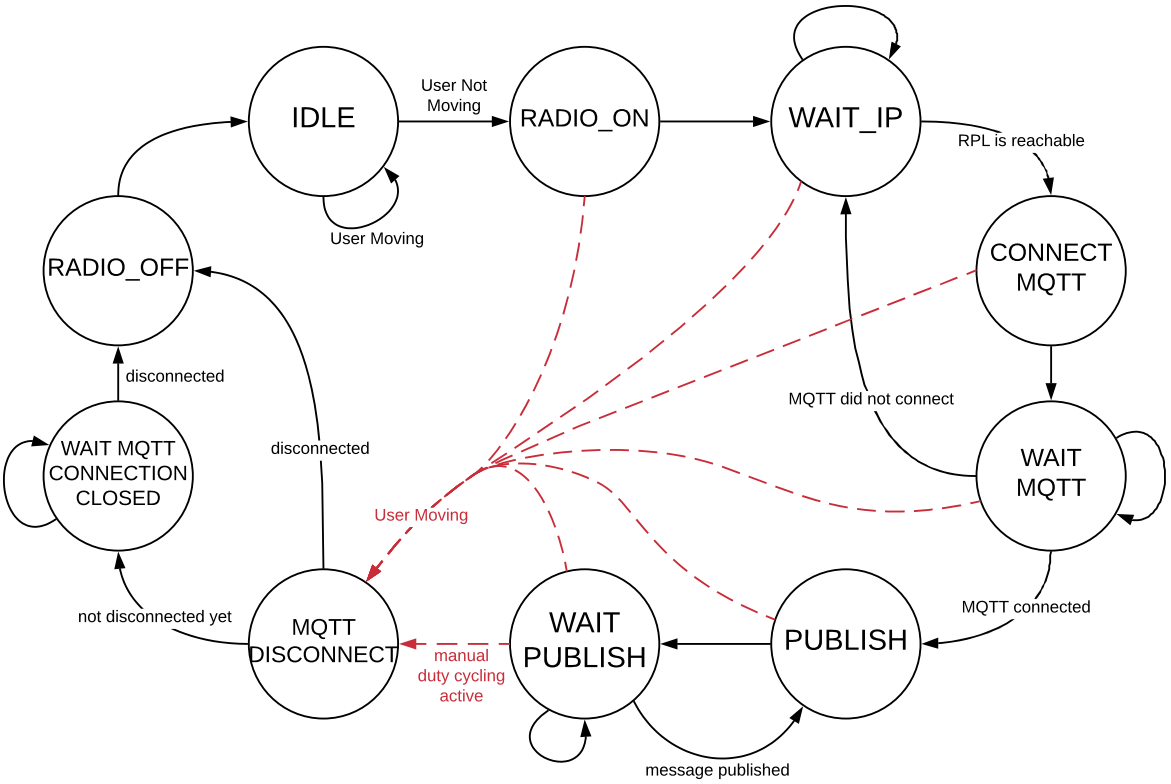

Figure 1: Sensors Finite State Machine responsible for the radio/mqtt connection and publishing

Two global events are also handled. The first is when the user starts moving (detected by the movement monitor process), while the other is connection lost (either MQTT or RPL not reachable). In both cases, the state is changed to *MQTT_DISCONNECT* and the disconnection phase is performed, waiting for the next opportunity to connect.

Parameters Estimation

Two parameters were estimated: T (movement threshold) and G (movement read timeout while not moving).

For both parameters, real data were collected using the sensortag board, and analyzed in order to get a fair estimate. The data were collected while a user moved in his house with the sensortag attached. From the X, Y and Z acceleration readings other parameters (such as euler angles and acceleration diffs) were computed to see when the user was moving.

The movement threshold T is applied to the absolute value of the acceleration module minus the gravity acceleration module, so it was estimated by trying to match the estimated movements with the real ones. We estimated $T = 1000$ roughly. Also, a second threshold T_DMOD is applied to the difference between the current and last modules, and has been set to half the general treshold.

The reading timeout G is to be done yet

TODO: real data movement graph showing moving periods

Energy Consumption minimization

In order to reduce the energy consumption of the device, two features were added to the client code, one for acceleration readings and the other for the network connection.

As for the data movement reading, the movement sesor (MPU 9250) is activated only when necessary (only while reading).

The network connections impacts heavily on the client energy consumption. To reduce this, two options can be configured in the client code: TSCH or manual CSMA duty cycling.

With TSCH, the duty cycling mechanism is handled transparently underneath RPL, meaning that the radio is controlled with a coarse grained approach (user moving or not moving implies client connected or not).

Instead, when manual duty cycling is enabled in CSMA mode, the radio is kept on only for the duration of time needed to connect to the MQTT broker and send a single message, then it is turned off immediately. Moreover, to avoid excessive power consumption for small values of K, when manual duty cycling is on, K is effectively increased by the amount of time needed to connect to the RPL network after the radio is turned on.

TODO: add images to compare the methods (nothing, tsch and manual duty cycling...moving vs standing or not?)

Attachments

Lab Notes, HelloWorld.ic, FooBar.ic