MIS 284N: PROJECT MILESTONE 3

Team Members: Vikram Gupta, Yeggi Lee, Chetna Singhal

STEP 6: Draw an architecture diagram that depicts the architectural components of your "system" and how they fit together. Who is subscribing/publishing? Where's the broker? How do the software components relate to your hardware components? You should reference the diagrams in the lecture about MQTT, but you will have additional information. Write an accompanying paragraph for your diagram to describe (in your own words) what's happening.

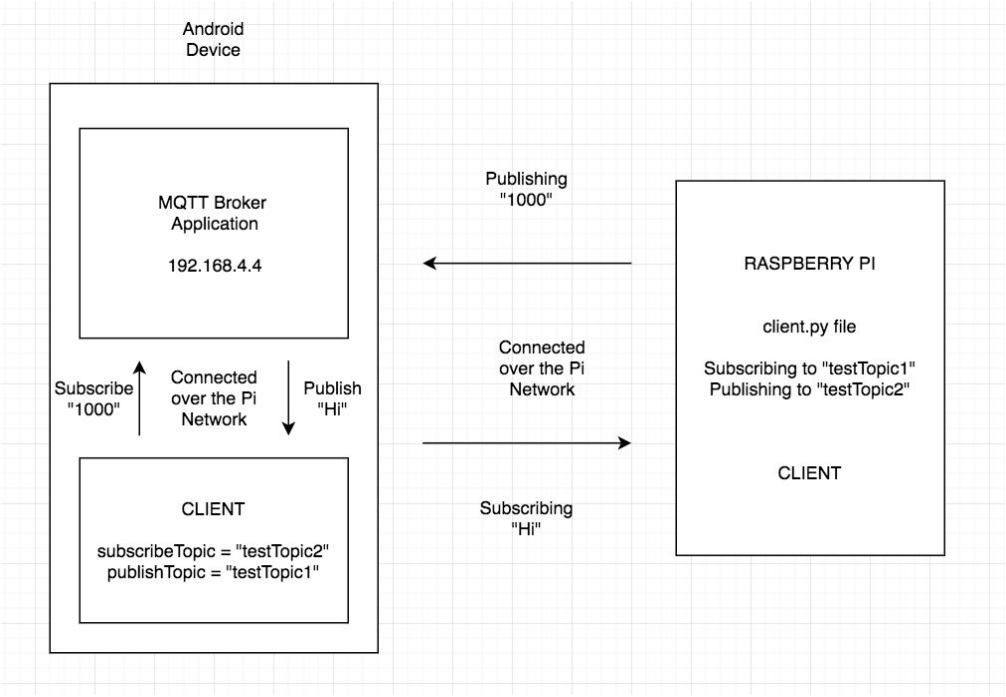

Figure 1. Architecture Diagram on Step 6

In this step, we have two clients: one in the Android device and the other in the Raspberry Pi. The client inside of the Android device is publishing the topic "testTopic1" and subscribed to the Raspberry Pi's topic "testTopic2". On the other hand, the client in the Raspberry Pi is publishing "testTopic2" and subscribing to "testTopic1". **The two clients communicate with each other through the broker inside the MQTT Broker Application (hosted on Android device) which is connected over the Pi network.** Therefore, the Raspberry Pi client is able to receive and read messages that the client in the Android Device publishes and vice versa. So essentially the Raspberry Pi is publishing a message containing "1000" which the client inside the Android device receives. The Android Device publishes a message containing "Hi" which the Raspberry Pi client can access and read.

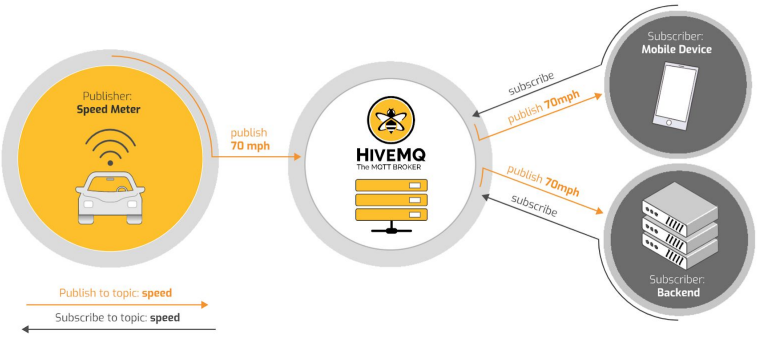

Figure 2. Similar Architecture Design

This setup is very similar to the one above except that the publisher and subscriber are the clients instead of a separate publisher. It is also important to note that clients can't speak directly to each other but only through a broker.

STEP 8: Reflect/restate: in your own words, why is it better to run the broker on the Raspberry Pi instead of the Android device? Give at least two reasons.

Reason 1:Network flexibility for the Android device
In the scenario where the Android device is both hosting the broker and acting as a client and it also needs to establish a connection with another client i.e. Raspberry Pi, it would have to join Raspberry Pi's network. Now, neither Raspberry Pi nor Android device is connected to the Internet. In case Android client wants to connect to the Internet, the Android device would necessarily have to switch networks which would break the connection between the broker & the client (Raspberry Pi). Alternatively, if the broker is hosted on the Raspberry Pi, the Android client could switch between the local Raspberry Pi network and the Internet anytime without disrupting the communication network between MQTT broker and its clients.

Reason 2: Mobility for the Android device
In order to stay connected to the Raspberry Pi client, the Android device which is hosting the broker needs to be physically present in Raspberry Pi's network range (few meters) at all times which would be inconvenient. Whereas, if the broker is hosted on Raspberry Pi, the Android device/client is free to move out of Raspberry Pi's network range without disrupting the MQTT broker-client network.

Reason 3: Static IP (broker hosted on Raspberry Pi) vs Dynamic IP (broker hosted on Android device)

STEP 9: Submit your design, including a brief textual description and the associated sequence diagram.

Our design re-implements our pedometer.

As before, we use our Microbit to actually count the number of steps and a Raspberry Pi to receive the number of steps. This was done by means of a Bluetooth Eddystone beacon.

The difference this time is that the end device, or the device on which the user reads the number of steps, is an Android phone. For brevity, we will only elaborate on the steps that are unique to this milestone. If needs be, the steps common with the previous milestone may be inferred from our diagram.

Our Raspberry Pi hosts an MQTT broker implementation named Mosquitto. The Android phone is configured to subscribe to the broker (for us this was 192.168.4.1:1883) on a topic "stepCounter".

We modified the previously used Eddystone beacon receiver python code so that in addition to simply decoding and displaying the number of steps, it also publishes a message on the topic with the text being the number of steps.

Concisely, the setup works in the following order:

1. The microbit button B is pressed, which advertises the URL containing the number of steps as an Eddystone beacon.
2. Receiving the beacon, the Raspberry Pi decodes the URL and publishes the number of steps on the chosen topic using the Mosquitto broker.
3. Our Android phone, which is listening on that topic, receives and displays the step count.

(sequence diagram on next page)

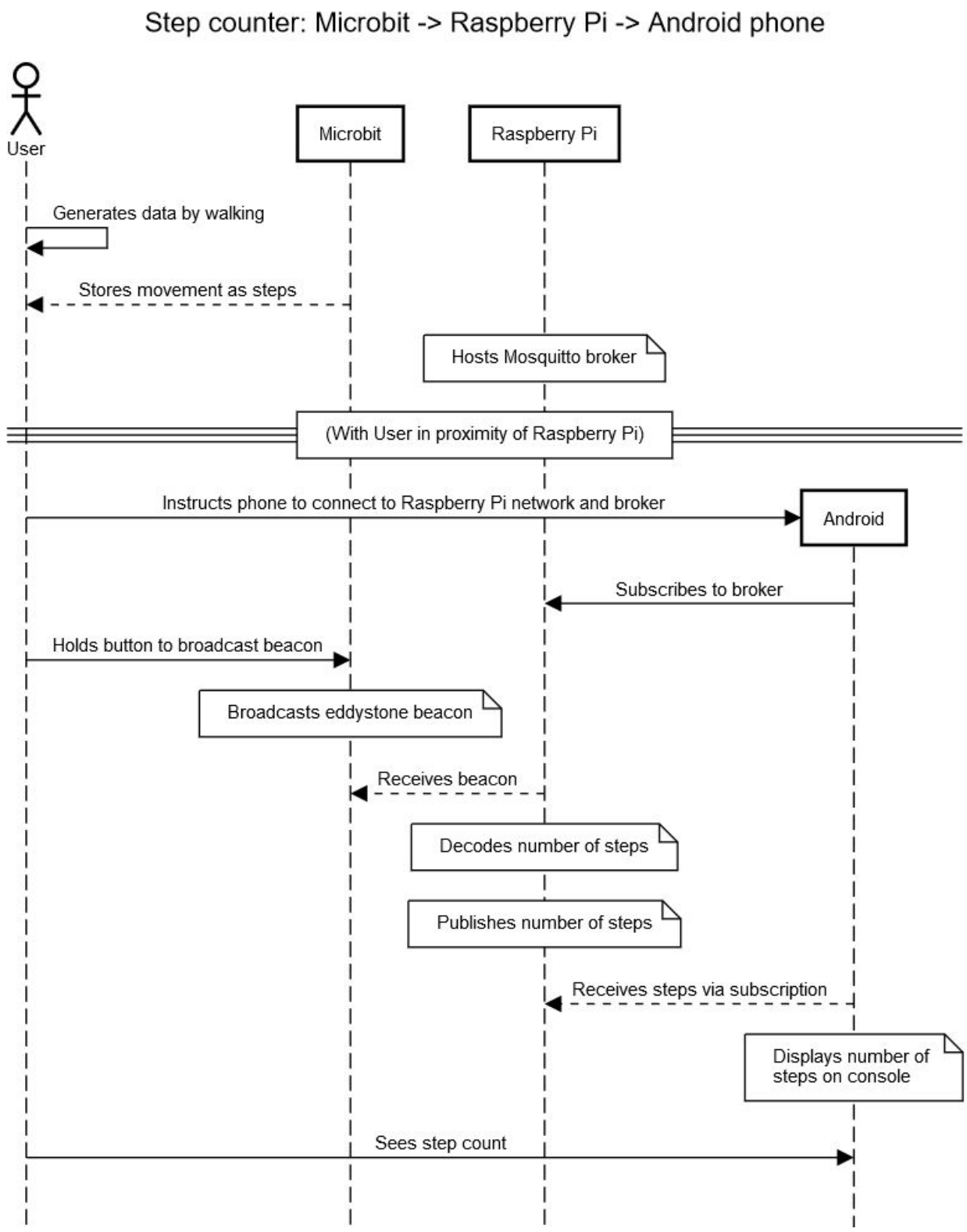

Figure 2. Sequence Diagram of Application