Project Milestone 3

Team - Group 4

- Shreya Bakshi *sb59344*
- Rathi Kannan *rk27867*
- Gowtami Khambhampati *gk6952*
- Danqing Wang *dw33369*

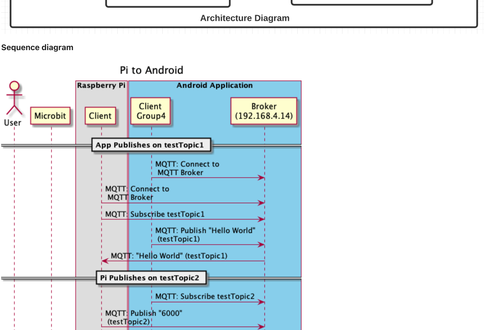
Demo video

Step 5: Submit as described in Step 5

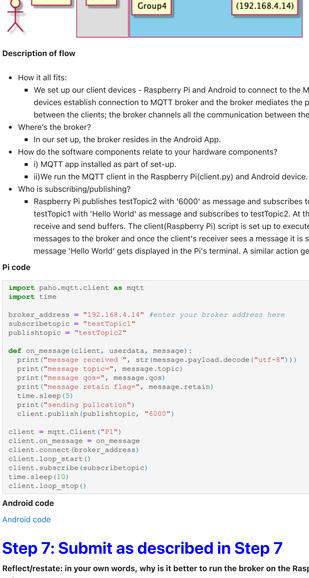
Step 5: Pause. What have you done so far? Draw an architecture diagram that depicts the architectural components of your "system" and how they fit together. Who is subscribing/publishing? Where's the broker? How do the software components relate to your hardware components? You should reference the diagrams 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.

Architecture diagram

Please note: Microbit comes into play in Step 8.



Sequence diagram



Description of flow

- How it all fits:
  - We set up our client devices - Raspberry Pi and Android to connect to the MQTT broker at IP:192.168.4.14. The client devices establish connection to MQTT broker and the broker mediates the pub-sub topics. There is no direct connection between the clients; the broker channels all the communication between the devices.
- Where's the broker?
  - In our set up, the broker resides in the Android App.
- How do the software components relate to your hardware components?
  - i) MQTT app installed as part of set-up.
  - ii)We run the MQTT client in the Raspberry Pi(client.py) and Android device.
- Who is subscribing/publishing?
  - Raspberry Pi publishes testTopic2 with '6000' as message and subscribes to testTopic1. Android device publishes testTopic1 with 'Hello World' as message and subscribes to testTopic2. At the client side, the loop function is set to read the receive and send buffers. The client(Raspberry Pi) script is set up to execute call back function on_message to publish messages to the broker and once the client's receiver sees a message it is set up to subscribe topic testTopic1. The message 'Hello World' gets displayed in the Pi's terminal. A similar action gets executed at the Android device.

Pi code

In [ ]:

```
import paho.mqtt.client as mqtt
import time

broker_address = "192.168.4.14" #enter your broker address here
subscribetopic = "testTopic1"
publishtopic = "testTopic2"

def on_message(client, userdata, message):
  print("message received ", str(message.payload.decode("utf-8")))
  print("message topic=", message.topic)
  print("message qos=", message.qos)
  print("message retain flag=", message.retain)
  time.sleep(5)
  print("sending pulication")
  client.publish(publishtopic, "6000")

client = mqtt.Client("P1")
client.on_message = on_message
client.connect(broker_address)
client.loop_start()
client.subscribe(subscribetopic)
time.sleep(10)
client.loop_stop()
```

Android code

Android code

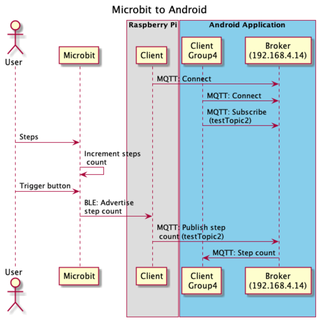
Step 7: Submit as described in Step 7

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.

- From design perspective, both Microbit and Android App connect to Raspberry Pi so it seems logical for Pi to be hosting the broker.
- Secondly, since broker and client were running in Android App it was difficult for us to confirm (partly via logcat and partly via MQTT broker App) if the clients are working as expected.
- We also do not need to restart/connect to MQTT broker on the Android every time we need to pub/sub topics.

Step 8: Submit your design, including a brief textual description and the associated sequence diagram.

Sequence Diagram



Description of flow

In this step, the Raspberry Pi receives BLE signals from microbit. Microbit advertises step count at the press of Button A (trigger button). Again, the clients at Raspberry Pi and Android device are connected to broker at 192.168.4.14. Once the client at Raspberry Pi receives the BLE, Raspberry Pi publishes the step count as testTopic2. The Android device is subscribed to this topic, so when the client at Android device sees the published topic in the receive buffer it executes call back 'messageArrived' to print the steps.

Design - Pi Code

In [ ]:

```
pi@raspberrypi:~ $ cat client_rathi.py

import paho.mqtt.client as mqtt
import time
import aioblescan as aiobs
from aioblescan.plugins import EddyStone
import asyncio
from urllib.parse import urlparse
from datetime import datetime

def _process_packet(data):
    ev = aiobs.HCI_Event()
    xx = ev.decode(data)
    xx = EddyStone().decode(ev)
    broker_address = "192.168.4.14"
    subscribetopic = "testTopic1"
    publishtopic = "testTopic2"

    if xx:
        print("Google beacon:{}".format(xx))
        group = xx.get('url')[0:14]
        if group == 'group4':
          steps =xx.get('url')[26:]


        print('***beacon received******')
        print('Step count:{}'.format(steps))

        print('****MQTT call ***********')
        client = mqtt.Client("P1")

        print('****MQTT call success*****')
        print('******Connecting to broker*****')
        client.connect(broker_address)
        print("*******Publish*****")
        client.publish(publishtopic, steps)
        client.loop_start()
        print("*******Subscribe****")
        client.subscribe(subscribetopic)
        time.sleep(10)
        client.loop_stop()



if __name__=='__main__':

    mydev = 0
    event_loop = asyncio.get_event_loop()
    print(event_loop)
    mysocket = aiobs.create_bt_socket(mydev)
    fac = event_loop._create_connection_transport(mysocket,aiobs.BLEScanRequester,None,None)
    conn, btctrl = event_loop.run_until_complete(fac)
    btctrl.process = _process_packet
    print('after steps display')
    print(btctrl.process)
    btctrl.send_scan_request()
    print('after send scan')

    try:

        event_loop.run_forever()
        print("loop forever")
    except KeyboardInterrupt:
        print('keyboard interrupt')
    finally:
        print('closing event loop')
        btctrl.stop_scan_request()
        conn.close()
        event_loop.close()
```