ECE 558 IoT MQTT Project

Assignment version 3.0: ESP32 version
due to GitHub and Canvas (see Deliverables section for details)
on February 26, 2022 at 10:00 PM

Introduction

Project #2 requires you to write an Android-based control app communicating via MQTT with an ESP32 board connected to sensors and simple electronic components. A cloud-based or PC-based MQTT server acts as the host, and both the Android and the board are MQTT clients. All the messages are sent over WiFi using the MQTT messaging protocol.

MQTT Overview

MQTT is a messaging protocol where all messages are sent to a central hub, known as the MQTT server or broker. Messages always have a topic. MQTT clients can either publish or subscribe to topics.

MQTT Server

A free, public MQTT server you can use for this project is HiveMQ..

- Since this is a public MQTT server, your topics must be very specific. Your MQTT client should not subscribe or publish to a topic as generic as “temperature,” or you will receive messages from other MQTT publishers unrelated to your project, as well as messages from other teams in the class.
- To differentiate your unique topic, you might prefix your messages with something like the class number and your initials (the exact format is up to you). For example, I would use: ECE558_temperature_ED
- Do not publish any secrets on HiveMQ, as anyone can read any message.

https://www.hivemq.com/public-mqtt-broker/

Alternatively, if you prefer to set up a personal MQTT network, you may use a PC-hosted MQTT server. Mosquitto MQTT is a widely-used, open-source MQTT broker from the Eclipse community: https://mosquitto.org/

MQTT Clients

Your project will have two MQTT clients: your Android and your ESP32 board. Each of these clients will run inside a program that you write, and the program will do other things as well such as interact with the user or interact with the hardware.

- The Android MQTT client will **publish** topics for the following:
  - interval at which to take temperature and humidity readings (user-entered value in the Android app)
  - LED (the user can choose to turn LED on or off by pressing a button on the Android app)
- The Android MQTT client will **subscribe** to the following topics:
  - Button Status (“pressed” or “not pressed”)
  - Temperature
  - Humidity

- The MQTT client on the ESP32 will **publish** topics for the following:
    - Temperature
    - Humidity
    - Button Status (“pressed” or “not pressed”)
- The MQTT client on the ESP32 will **subscribe** to the following topics:
    - LED (turn on or turn off)
    - interval (how often to take sensor readings)

Android App

Your Android app must do the following. All other design choices are up to you:

- Let the user turn the LED on or off (button, switch, etc)
- Let the user enter the interval at which to take sensor readings. There should be a default value which can then be updated by the user.
- Display the temperature and humidity, which are updated at the specified interval.
- Display the status of the push button (Pressed or Not Pressed)

All those actions will depend on its communication via MQTT with the board.

ESP32 Board

This project was originally developed to use a RaspberryPi as the board. Due to shortages, we have redesigned it with an ESP32-based board instead. If you have a RaspberryPi, you may still use it as your board, either with this version of the assignment (using a public MQTT server) or in its original version (where the MQTT broker is hosted on the RaspberryPi).

If you use an ESP32-based board, you may choose your programming environment. ESP32 supports CircuitPython. An advantage of CircuitPython is that it supports a RESTful interface, which could be used to expand this project into your final project.

Alternatively, the Feather Huzzah also can use the Arduino IDE, which uses a programming language based on C++. There is more support and examples available for the Arduino IDE with the Feather Huzzah, so this might be an easier option for this board specifically.

Lastly, if you wish to use a board other than an ESP32 or RaspberryPi, please check in first, but other boards are possible. The project “support” from the TA will be mostly limited to ESP32 and RaspberryPi.

MicroPython Path

Please see the Micropython ESP32 references at the end of this document. I recommend that you test code in the REPL, then incrementally add working code to the main.py file and upload that over the serial connection to the ESP32. Do this for everything from connecting the board to the WiFi to interacting with MQTT and the hardware.

I also noticed that the board was most responsive immediately after start up, and tended to get wonky after running for 10-15 minutes. If you run into error messages when installing libraries, try restarting the board and see if that helps.

I had trouble installing the AHT20 and Blinka libraries to the ESP32. This may have been a problem unique to my setup and you might avoid this issue entirely.

GitHub and GitHub Classroom

Please turn in all code to GitHub Classroom. You may turn in your report (as a PDF or DOC file) to GitHub or to Canvas. You may wish to use GitHub for version control as you work. IntelliJ (and Android Studio which is based on IntelliJ) provides excellent integration with GitHub.

Deliverables

- Project Report (details below)
- Android Studio Project
- MQTT Client Program that runs on the board (depending on your choice of board and of program language, this could be a Python or C++ file).
- Video or live demo of your project

Project Report

Your project report can be brief, but it must include at minimum the following sections:

- Project Description, including which option you chose for MQTT server, which board you used, and which programming language you used to write the board's MQTT client program.
- Theory of Operation (https://en.wikipedia.org/wiki/Theory_of_operation)
- Link to your GitHub repository

Bill of Materials

- Android
- Android power supply
- ESP32 board (we recommend Adafruit Huzzah32: https://www.adafruit.com/product/3405
  (If you plan to use a different board, please email Professor Kravitz to make sure it is ok)
- Micro USB cord to connect to and power the board
- temperature/humidity sensor
  - Sample project used the Adafruit AHT20 but you may use another if you prefer: https://www.adafruit.com/product/4209
  - You will need to either solder headers to the AHT20 or use a Stemma QT cable (https://www.adafruit.com/product/3955) to connect it to your breadboard.
- red LED (or color of your choice, but change the value of the resistor accordingly)
- One 330 Ω resistor
- One 220 Ω resistor
- push button
- breadboard
- Wires for the breadboard

Task List

1) Procure the hardware and software needed for the project.
2) Set up the ESP32 board with your choice of programming environment (Arduino IDE or CircuitPython). See the Useful Links section for examples and references for each option.
3) Choose your MQTT server (HiveMQ's public broker or your own PC-hosted MQTT server.)
4) Install the MQTT client libraries for Android and for your ESP32's IDE (either the Arduino MQTT client library or the MQTT client library for Python). Links are in the Useful Links section.
5) Install the Adafruit AHTx0 library for your board (either the Arduino or CircuitPython library; see links in the Useful Links section).
6) Wire the hardware. If you used the Huzzah Feather, you may use the wiring diagram in this document. If you used a different board, adjust the wiring to suit your board. Develop and run test programs to ensure everything is connected and working properly.
7) Write the MQTT client program to run on the ESP32 board. This program will interact with the hardware (read the temperature and humidity, light the LED, detect if the button is pressed) and will communicate about the hardware via MQTT.
8) Develop an Android app that monitors and displays the sensors, controls how frequently the temperature and humidity are sampled (with a user-entered "interval"), displays whether the button is pressed, and allows the user to turn on the LED by pressing a button on the Android app. Integrate the Android app with the MQTT server.
9) Prepare and submit your deliverables. Deliverables include all the source code you developed, a project report, and a demo video (or sign up for a live demo).

Useful Links

MQTT in general:

- Paho MQTT client libraries https://www.eclipse.org/paho/
- Playing with MQTT by Android Part 1: https://www.youtube.com/watch?v=BAkGm02WBc0
- Part 2: https://www.youtube.com/watch?v=6AE4D8INs_U&t=3s
- Paho Android Service – MQTT Client Library Encyclopedia https://www.hivemq.com/blog/mqtt-client-library-enyclopedia-paho-android-service/

- Paho Python – MQTT Client Library Encyclopedia
  https://www.hivemq.com/blog/mqtt-client-library-paho-python/
- Installing the Arduino MQTT client library
  https://docs.arduino.cc/tutorials/uno-wifi-rev2/uno-wifi-r2-mqtt-device-to-device
- MQTT Essentials - A Lightweight IoT Protocol by Gaston Hiller
  https://subscription.packtpub.com/book/application-development/9781787287815

Arduino IDE (option 1 for the board's programming environment):

- Arduino IDE for Huzzah Feather board:
  https://learn.adafruit.com/adafruit-huzzah32-esp32-feather/using-with-arduino-ide
- Adafruit AHTx0 library for Arduino: https://learn.adafruit.com/adafruit-aht20/arduino
- ESP32 tutorials (helpful for the hardware part, but do not follow the Adafruit IO tutorial; use HiveMQ instead) https://makeabilitylab.github.io/physcomp/esp32/
- ESP32 MQTT Client (this will manage MQTT and WiFi)
  https://github.com/plapointe6/EspMQTTClient
- AHT20 Temperature Humidity Sensor Example project on an Arduino
  https://learn.adafruit.com/adafruit-aht20/arduino

MicroPython (option 2 for the board's programming environment):

- Installing and setting up MicroPython on ESP32:
  https://github.com/pvanallen/esp32-getstarted
- To connect the board to WiFi, I had good luck typing help() in the REPL, then following the instructions given there (import network, then 4 connection instructions). Once you've done these successfully once in the REPL, add these 4 instructions to main.py so they execute automatically on startup.
- If you use WebREPL, I noticed that mine did not work on Google Chrome but it did work on Safari, so changing browsers may help. Alternatively, you can run a REPL through an extra Terminal window.
- Using MQTT with Micropython: https://boneskull.com/micropython-on-esp32-part-2/
- MicroPython driver for the AHT10 and AHT20 temperature and humidity sensors:
  https://pypi.org/project/micropython-ahtx0/#modal-close

Circuit Wiring Diagram