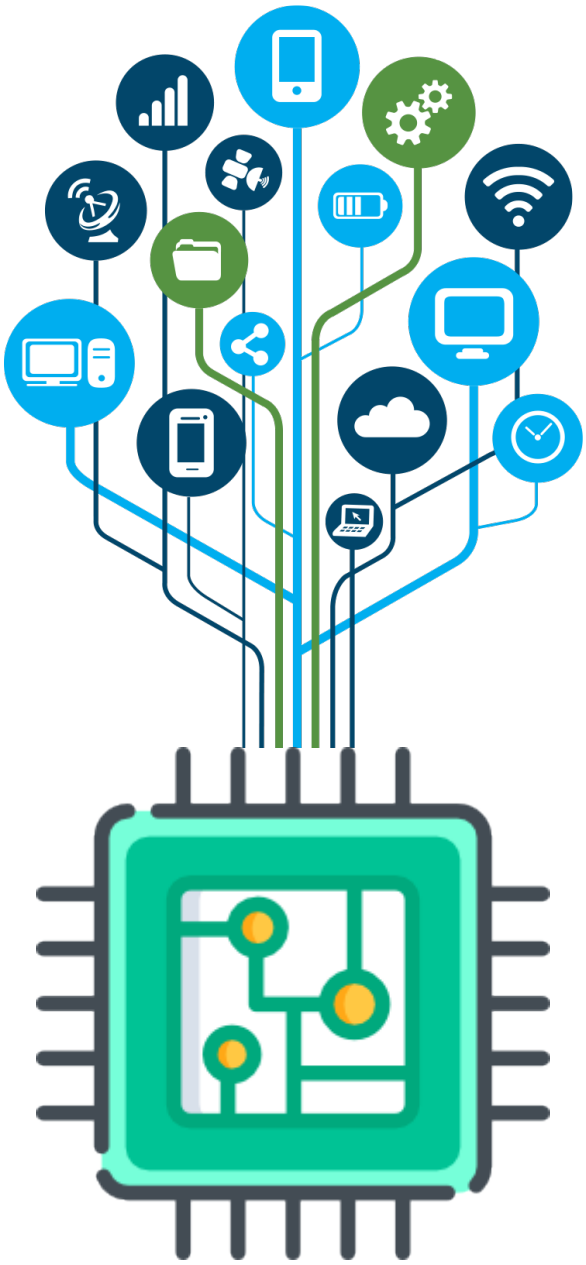
Control d'una ciutat feta amb Unity via MQTT



Jaume Palau Seijas

2nDAM

Projecte final de curs

Professor ponent: Jordi Binefa

Índex de continguts

Resum

El meu treball consisteix en controlar els semàfors i la llum dels fanals d'una ciutat feta amb Unity, emprant una aplicació QML i que el control de la ciutat es vegi reflectit a uns leds gestionats amb un NodeMCU ESP32.

Explicat amb més profunditat la idea és que els semàfors tinguin un control automatitzat, és a dir, cada pocs segons el semàfor canvia; i el mateix passa amb els fanals. A més a més, aquesta configuració es podrà canviar manualment des de l'aplicació.

La placa actuaria en forma de maqueta de la ciutat, vull dir, quan s'encenguin els fanals que s'encengui el LED blanc, quan els semàfors es posin en verd que s'encengui el LED verd...

Que es MQTT?

MQTT o *Message Queuing Telemetry Transport* es un protocol de missatgeria de publicació-subscripció basat en el protocol TCP/IP. Va ser desenvolupat per Andy Stanford-Clark i Arlen Nipper però finalment va ser publicat com a codi obert.

Existeixen molts Brokers MQTT que varien segons la seva funcionalitat, alguns dels més emprats són:

- ActiveMQ
- JoramMQ
- Mosquitto
- RabbitMQ

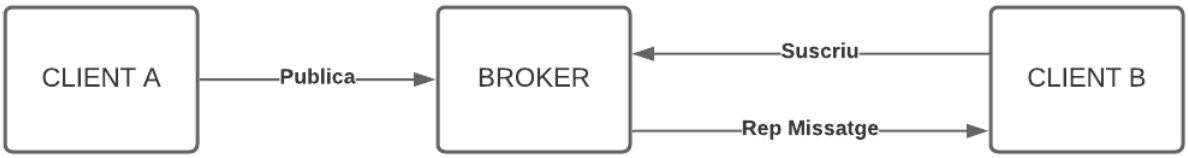
Però, que és un Broker?

Un broker és tan sols un agent de missatges que tradueix els missatges al protocol del emissor al del receptor.

Funcionament del MQTT:

Els missatges que es reben es disposen en «Topics» o temes que s'ordenen jeràrquicament

Un client publica a un tema i un client subscrit a un tema ho rep.



Aplicacions reals de MQTT

Com a curiositat, FaceBook Messenger ha utilitzat aspectes de la comunicació MQTT.

Broker Mosquitto

Com ja he dit abans Mosquitto és un agent de missatges que implementa el protocol MQTT. Aquest proveeix un mètode senzill de gestionar missatges emprant el model de publicació i subscripció.

Aquest broker és el més utilitzat en el àmbit de Internet de les coses amb implementacions a telèfons i sensors.

ESP32 i NodeMCU

L'ESP32 un chip de baix cost i consum de la família SoC(System on a Chip, tendència de integrar tots o gran part dels components d'un computador a un únic circuit integrat o chip) que conté tecnologia WiFI i Bluetoth.

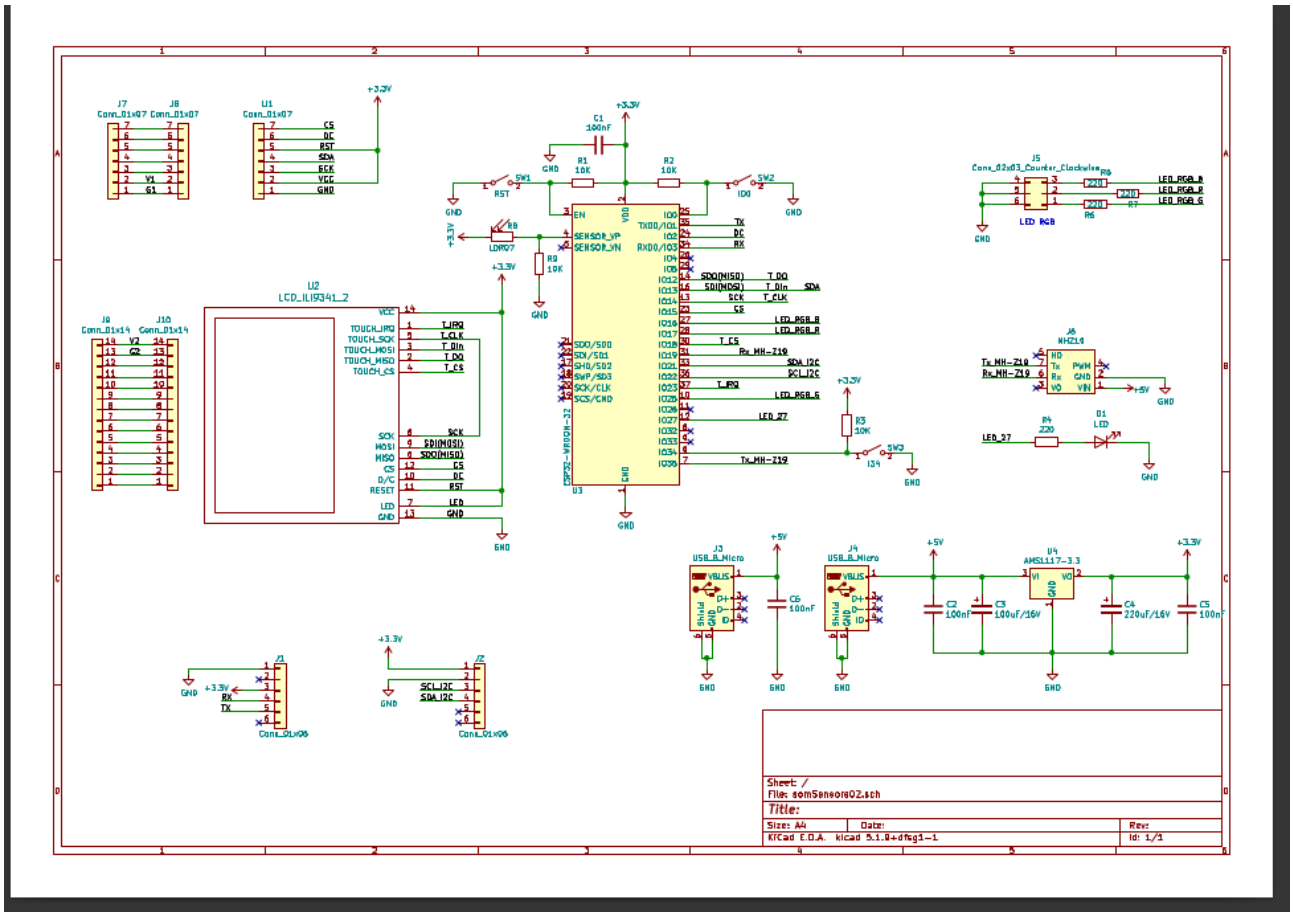
El Node MCU és una plataforma per IoT de codi obert que s'executa a un chip ESP32 o ESP8266. El terme NodeMCU prové del firmware emprat.

Pantalla LVGL

La biblioteca LVGL (Light and Versatile Graphics Library) proporciona tot el necessari per crear una interfície gràfica d'usuari amb elements fàcils d'utilitzar.

La pantalla que he empreat es SPI TFT Module.

Esquema placa

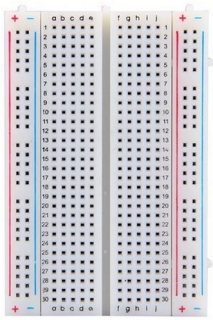


Connexió NODCEMCU LEDS i Joystick

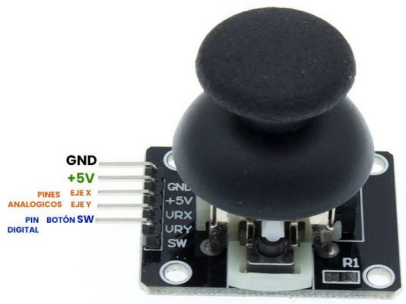
Per realitzar la connexió dels LEDS i el JOYSTICK cal saber que el NODEMCU té una serie de pins anomenats GPIO (Entrada/Sortida de propòsit general) que es poden connectar elements d'entrada i/o de sortida.

Una protoboard o placa de proves o d'inserció és un tauler amb orificis connectats elèctricament entre si.

La connexió amb el NodeMCU, LEDs i la protoboard aniria de la següent forma:



Cada orifici de la a al e en sentit horitzontal, seguixen la mateixa línia. Es a dir, si connectem un pin GPIO a la segona fila i una pota del led hi haurà connexió.



També cal saber que al connectar un LED s'ha de connectar una resistència de 100 ohms per no cremar el LED.

Per connectar el joystick a la placa hem de prendre atenció a les connexions del joystick:

El GND equival al valor negatiu anomenat GND o Massa.
El +5v equival a l'alimentació positiva.
El pin VRX equival al eix X.
El pin VRY equival al eix Y.
El pin SW equival al pulsador.

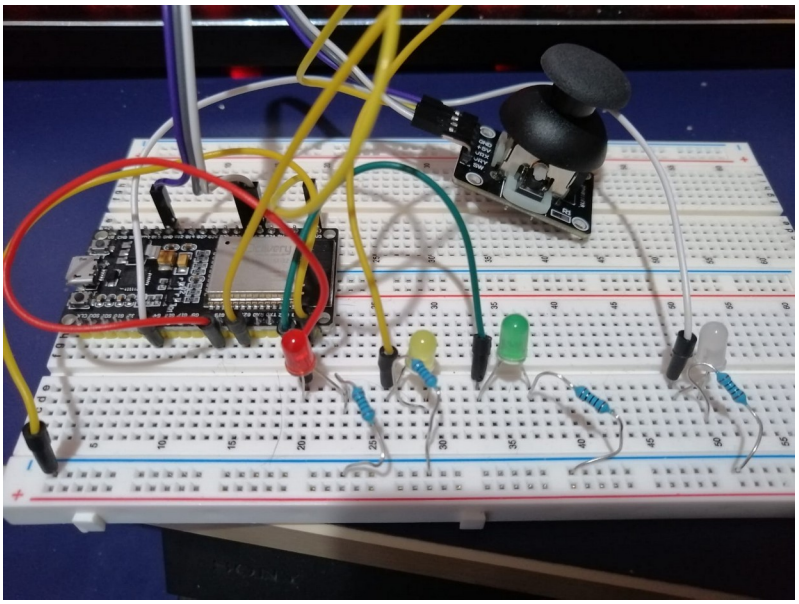
Cada pin ha de tindre uns pins específics al node ja que si no no llegirà correctament els seus valors.

Acontinuació deixo una taula on indico on he connectat els pins a la esp32 respecte a la protoboard.

JoyStick	**ESP32**
GND	GND
+5v	+3.3v
VRX	Pin 32
VRY	Pin 35

LED	**PIN ESP32**
LED White	Pin 4
LED Red	Pin 19
LED Yellow	Pin 21
LED Green	Pin 23

Imatge de referència amb els leds i Joystick connectats.



En el cas dels Leds s'ha d'especificar el mode dels LEDs,si el pin és d'entrada o de sortida, en la funció setup.

```
void setup() {
  pinMode(LED_R, OUTPUT);
  pinMode(LED_W, OUTPUT);
  pinMode(LED_Y, OUTPUT);
  pinMode(LED_G, OUTPUT);
  Serial.begin(115200);
```

Unity

Unity es un Game Engine multiplataforma per el desenvolupament de Videojocs.

Alguns Jocs fets amb Unity

- Among us
- Subnautica
- Hollow Knight
- Cuphead

Realitat virtual

Dins del programa de Unity he implementat realitat virtual per poder veure en primera la ciutat i com canvien els semàfors i fanals.

Com ulleres de realitat virtual he utilitzat el **GoogleCardboar** que no deixa de ser un suport de cartró i lents focals per a un dispositiu mòbil on s'executa el joc amb unity amb el mode de realitat virtual.

Creació d'un Broker MQTT

Com ja he esmentat abans es necessari un Broker MQTT.
Hi ha una serie de brokers MQTT públics que es poden utilitzar com:

- Mosquitto
- Mosca
- Aedes
- HBMQTT
- EMQTT
- RabbitMQ
- HiveMQ CE
- ActiveMQ
- Moquette
- MQTTnet

Pero jo he creat un broker propi a una raspberry amb el sistema operatiu Raspbian basat en Linux però també es pot fer amb Windows o MAC.

Instal·lació mosquitto.

Per instal·lar el servei mosquitto a un sistema linux hem d'executar la següent comanda al terminal:

sudo apt-get install mosquitto

I per instal·lar el client de mosquitto només hem de executar.

sudo apt-get install mosquitto-clients

Ara només hem de configurar el servei per que escolti el port 1883 i no permeti la connexió anònima i especificar un usuari i la contrasenya.

Aquests paràmetres s'han d'especificar al fitxer de configuració de mosquitto.

```
pi@raspberrypi:~ $ cat /etc/mosquitto/mosquitto.conf
# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

pid_file /var/run/mosquitto.pid

listener 1883
allow_anonymous false
persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d
pi@raspberrypi:~ $
```

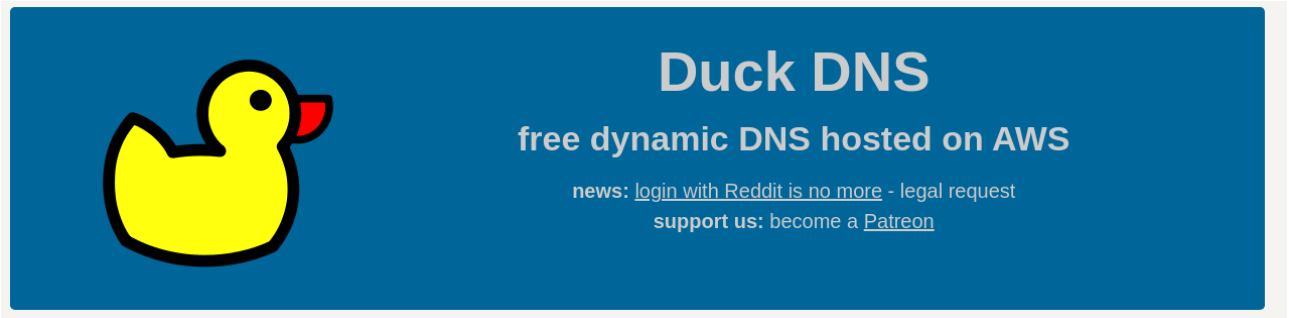
Per especificar el usuari i la contrasenya hem de executar:

passwd <fitxerContrasenyes> -c <usuari>.

I el terminal demana una contrasenya.

Ara només queda tenir un domini gratuït per poder accedir al broker des de qualsevol lloc sense necessitat d'una VPN.

Ens hem de dirigir a la pàgina duckdns.org i crea un compte i un domini.



Un cop tenim el domini hem de seguir les instruccions de la seva pagina per posar en marxa.

Per finalitzar només hem de crear una regla al router per obrir els ports necessaris per poder accedir al equip.

Jo he obert el port 80 i 1883.

MQTT	UDP	1883:1883	1883:1883	192.168.1.40	ON
DuckDNS	TCP	80:80	80:80	192.168.1.40	ON

Unity i MQTT

Per integrar MQTT a Unity s'ha de seguir una serie de passos a seguir:

-Descarregar el projecte d'aquest repositori de GITHUB:

http://blog.jorand.io/2017/08/02/MQTT-on-Unity/

-Obrir el projecte amb Visual Studio Community o Visual Studio Code

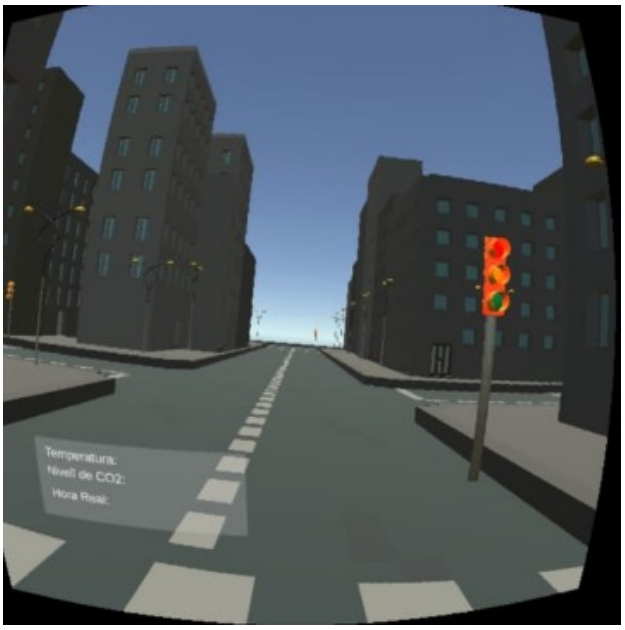
-Executar el projecte, al fer-ho generarà una biblioteca dinàmica

-Només cal arrossegar aquesta biblioteca dins del IDE de Unity i ens deixarà importar les biblioteca necessàries per operar amb MQTT.

Funcionament del projecte

Com ja he comentat la idea es controlar la ciutat via MQTT i que es vegi reflectat . Quan s'enviï un missatge, amb l'ordre d'encesa o d'apagada del fanal, a la ciutat, aquesta enviarà un missatge al ESP32 per que encengui el Led corresponent.

Per establir el cicle de nit i dia de la ciutat ho he fet amb el LDR de la placa virko, cada 15 segons la ciutat envia una petició del nivell de la LDR i segons la resposta rebuda la llum canvia.



→

CICLE DIÜRN **CICLE NOCTURN**

A més, la virko també llegeix el nivell de CO2 i la temperatura, i també ho envia a la ciutat que es pot veure els valor a través de un HUD.

Funcionament QML

L'aplicació QML consta de 3 pestanyes, la primera on estableixes connexió amb el broker, la segona on controles els fanals i la tercera on controles els semàfors.

La connexió amb el broker és immediata i no s'ha de configurar res només cal prémer el botó de connexió. Es important que el dispositiu tingui connexió a internet si no no es podrà connectar.

Control amb Joystick

Per controlar la càmera de la ciutat via MQTT he connectat el joystick als pins del nodeMCU i segons el valor «x» o «y» envia una publicació un altre i segons la publicació que rebi la ciutat, la càmera es mourà cap una direcció o un altre.

Quan el Joystick està quiet els seus valors oscil·len entre 1700 i 1900 aproximadament. Quan s'està aquesta posició s'envia una publicació per que la càmera no avanci.

Si el valor X equival a 0: la càmera es mourà cap a l'esquerra.

Si el valor Y equival a 0: la càmera es mourà cap a endarrere.

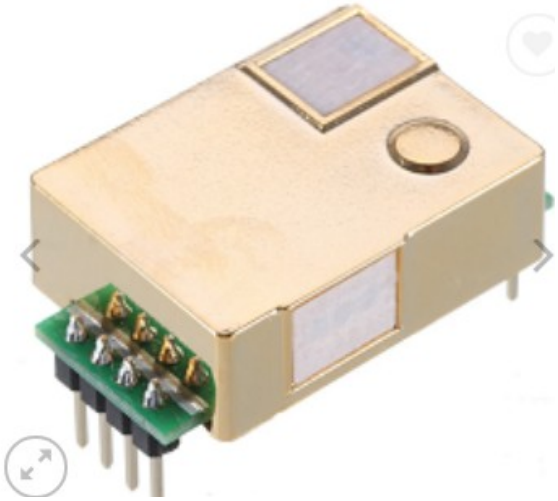
Si el valor X equival a 4095 la càmera es mourà cap a la dreta.

Si el valor Y equival a 4095 la càmera es mourà cap a endavant.

Lectura de CO2 i Temperatura

El sensor que he emprat per la lectura del CO2 és el MH-Z19. Aquest sensor utilitza un sensor d'infrarojos no dispersiu (NDIR) per detectar la presencia de CO2 al aire.

Disposa d'un petit processador intern amb compensació de temperatura integrada.



Per poder utilitzar aquest sensor dins del Arduino IDE s'ha de importar les biblioteques ESPSoftwareSerial i MHZ19.

La biblioteca software serial ha estat desenvolupada per permetre la comunicació serie entre altres pins digitals, I la biblioteca espsoftwareserial permet el mateix però amb plaques ESP8266 i ESP32.

Ús de la placa VirKo

Amb la placa virko he fet un «Weather Station» i poder visualitzar el estat dels semàfors de la ciutat mitjançant la biblioteca LVGL. Primer s'ha d'especificar els pins TFT_MISO, TFT_MOSI, TFT_CS, TFT_DC dins del fitxer User_Setup.h de la biblioteca LVGL.

També s'ha de calibrar, que es pot fer amb un codi de calibració que generà un fitxer de calibració o especificant-li el fitxer de calibració dins del fitxers a pujar a la placa.

Com el protocol MQTT necessita connexió a internet he fet un selector de xarxes sense fils ordenats segons el RSSI.

Un cop s'ha connectat a la xarxa WiFi es mostrarà la pantalla amb l'estació del temps, on es mostra la temperatura, el nivell de diòxid de carboni, el nivell de llum i el estat dels semàfors.

Programació amb OTA

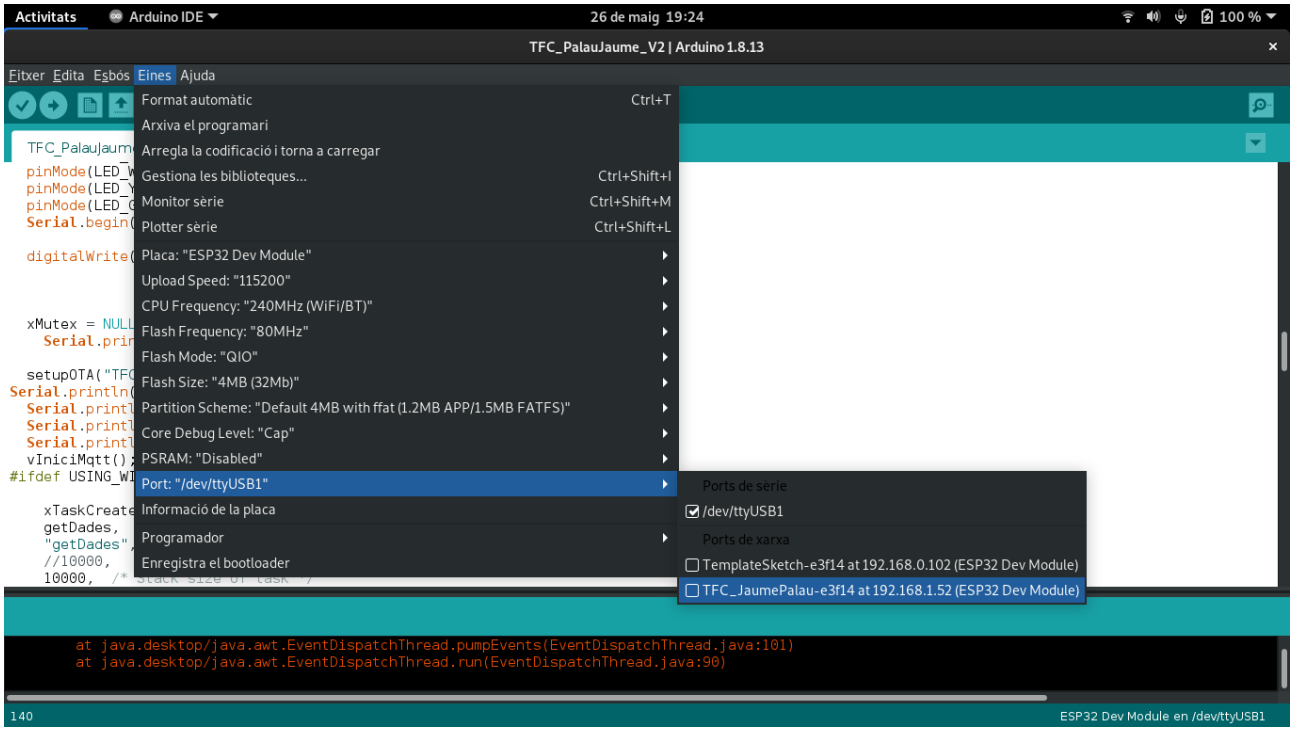
He afegit la possibilitat de programar el NodeMCU remotament gràcies a la tecnologia OTA.

Over the air permet programar a distancia un esp32 mitjançant la connexió WiFi en lloc del port serie .

Es pot fer de diverses formes, des de mitjançant un servidor Web per seleccionar el fitxer binari fins mitjançant l'IDE d'arduino.

Per realitzar he realitzat unes del codi de Andreas Spiess adaptat per al meu projecte. Aquest codi permet ser programable a través d'una IP, es important que el ordinador i el ESP32 estiguin connectats a la mateixa xarxa per que hi hagi comunicació.

Un cop s'ha pujat el codi, en l'apartat Port, dins d'eines del IDE sortirà un nou element:



Això confirma que es pot pujar un codi nou mitjançant la Xarxa, però no es pot veure el que mostra el Serial monitor per això he realitzat un segon codi Idèntic amb la diferencia que es pugui veure el que rep per MQTT mitjançant el protocol Telnet.

```
palau@Palau:~/Baixades/maya22gui-1.1-64bit$ telnet 192.168.1.52
Trying 192.168.1.52...
Connected to 192.168.1.52.
Escape character is '^]'.
Topic: /forward
Missatge:
wTopic: /stop
Missatge:
lTopic: /forward
Missatge:
sTopic: /stop
Missatge:
lTopic: /forward
Missatge:
aTopic: /stop
Missatge:
lTopic: /forward
Missatge:
dTopic: /stop
```

Models de Unity

Els models que he emprat per poder omplir la ciutat són tots extrets de la Unity Asset Store.

Què és l'Asset Store de Unity?

La Asset Store d'Unity és una biblioteca de Assets en expansió.

Tant Unity Technologies com els membres de la comunitat creen aquests actius i els publiquen a la botiga.

Hi ha una barreja d'actius comercials gratuïts i a preus assequibles que pots descarregar directament al teu Projecte d'Unity.

FANALS

https://assetstore.unity.com/packages/3d/props/exterior/street-lights-pack-31644

Com a model de fanals vaig decantar-me pel pack que proporciona gratuïtament un dels creadors de contingut d'Assets de Unity.

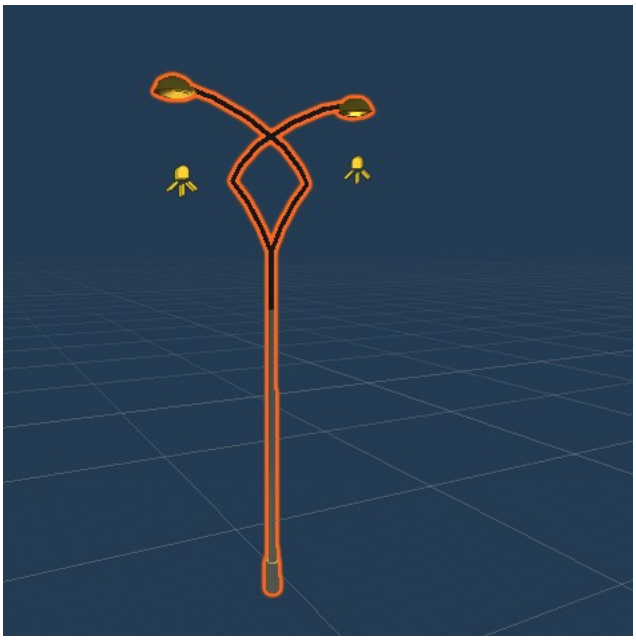
Menció especial al creador:

Mehdi Rabiee

Pàgina del creador:

https://assetstore.unity.com/publishers/3780

Preview dels fanals a la ciutat:



SEMÀFORS

https://assetstore.unity.com/packages/3d/props/free-traffic-essentials-asset-pack-125092

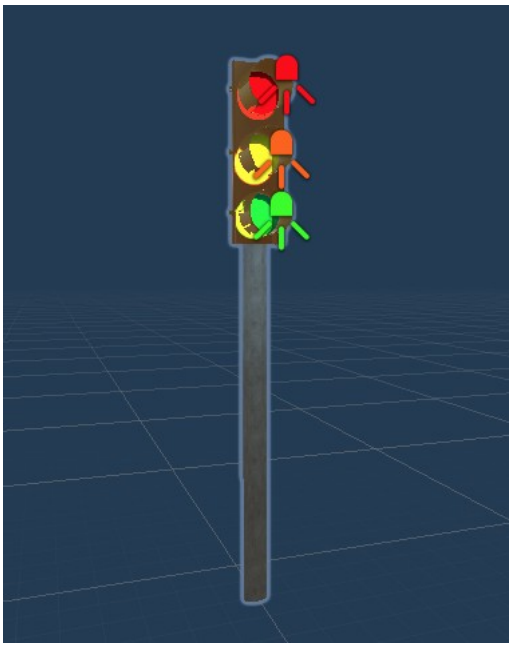
Com a model de semàfor vaig decidir-me pel pack gratuït d'un dels creadors de contingut d'Assets de Unity.

Menció especial al creador:

Ferocious Industries

Pàgina del creador:
https://assetstore.unity.com/publishers/37734

Preview dels semàfors a la ciutat:



CIUTAT (edificis)

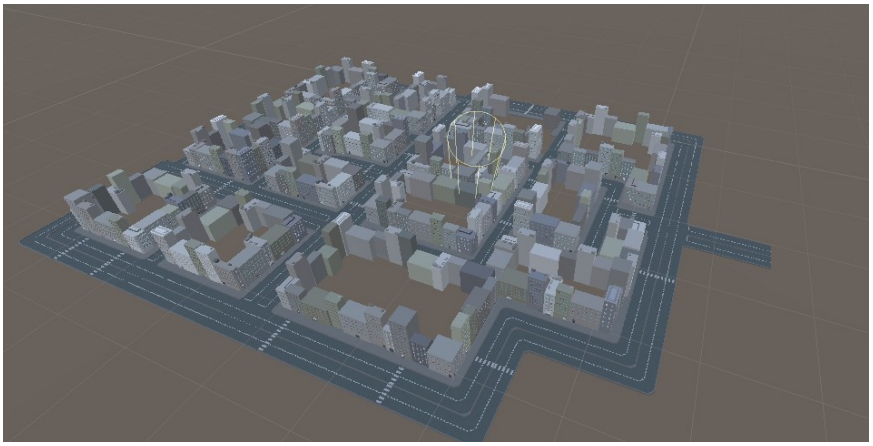
https://assetstore.unity.com/packages/3d/environments/urban/city-voxel-pack-136141

Menció especial al creador:

PoèMe

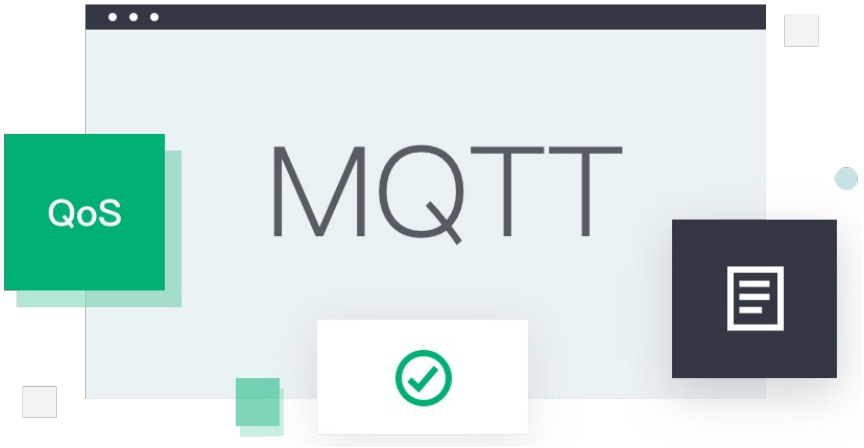
Pàgina del creador:
https://assetstore.unity.com/publishers/39869

Preview de la ciutat:



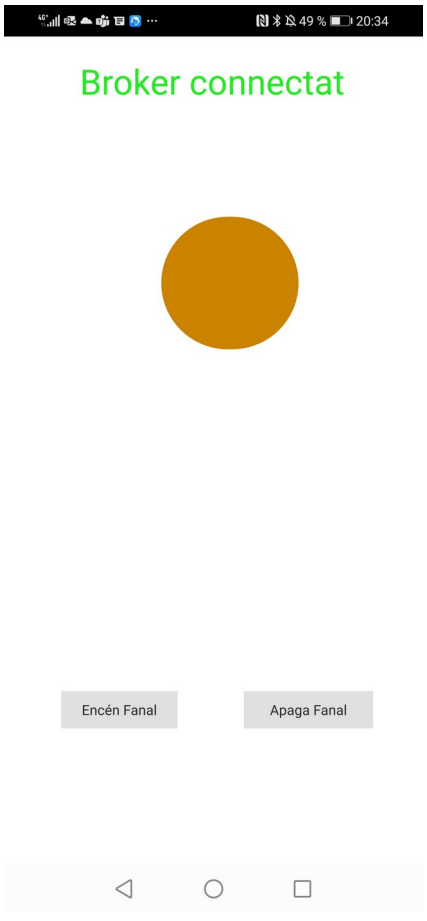
Viabilitat del projecte

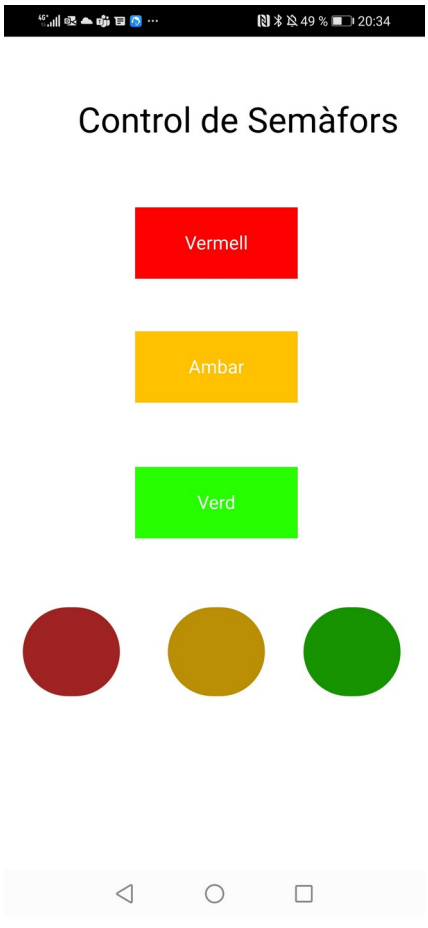
Al tenir el coneixement del motor gràfic de Unity 3D gràcies al projecte realitzat durant el curs en la respectiva UF i existir la biblioteca de **MQTT For UNTIY,** només quedaba ampliar i ampliar més per poder profunditzar en la idea de la monitorització d'una ciutat virtual.



Imatges de la construcció del projecte (QML)

Veiem el SwipeView de la segona plana i la tercera.





Manteniment i futures versions

Com a futures funcions m'agradaria implementar els següents controls i adició de efectes al Unity com ara:

- La pluja.

- La neu.

- El flux del clavegueram.

- Etc...

Funcionalitats que pel tems del projecte i aprofondir en les plaques i altres funcions no s'han afegit.

Accés del projecte

Repositori GITHUB

Tot el contingut del projecte i altres descripcions es poden trobar al repositori GitHub.

Què és GitHub?

GitHub és una plataforma de desenvolupament col·laboratiu de programari per allotjar projectes utilitzant el sistema de control de versions Git.

El codi s'emmagatzema de forma pública, encara que també es pot fer de forma privada, creant un compte de pagament.

LINK: https://github.com/jp0431

WEBGRAFIA I FONTS

https://ca.wikipedia.org/wiki/MQTT

https://ca.wikipedia.org/wiki/Mosquitto

https://www.luisllamas.es/que-es-mqtt-su-importancia-como-protocolo-iot/

https://mosquitto.org/

https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

https://es.wikipedia.org/wiki/NodeMCU

https://docs.lvgl.io/latest/en/html/index.html

https://www.luisllamas.es/principales-broker-mqtt-open-source-para-proyectos-iot/

http://blog.jorand.io/2017/08/02/MQTT-on-Unity/

https://store.prometec.net/producto/sensor-co2-mh-z19/

https://manueldelgadocrespo.blogspot.com/p/biblioteca.html