BREVE MANUAL DE CÓMO USAR LOS PROGRAMAS Y LA RED DE MENSAJERÍA

Los programas se encuentran en el siguiente repositorio:
https://github.com/marinaperez1999/Microfluidics_control

Para cualquier duda: marinaperezdiego@gmail.com

1. Control de caudal sin conexión WiFi:

   Arduino_final_simple

   Este programa mantiene constante un valor de caudal y pausa o reanuda el flujo al pulsar el interruptor de la placa.

   Cada 800 ms (la variable “interval” puede ser ajustada para cambiar ese intervalo) se printea en el *Serial Monitor*: el tiempo en ms, la temperatura, la frecuencia, el voltaje, el caudal, un mensaje de si ha entrado una burbuja, un mensaje de si el voltaje se ha saturado y un mensaje de si el caudal es insuficiente (en ese orden, si estos mensajes son “OK” significa que no hay ninguna alarma).

   Instrucciones:

   - Instalar Arduino: https://www.arduino.cc/en/software
   - Descargar las librerías de Arduino que el programa emplea si aún no están instaladas (SPI.h, SD.h, Wire.h). En realidad solo se usa la Wire.h, las otras dos están incluidas para el caso en el que se desee almacenar los datos en una tarjeta SD (en ese caso habría que modificar un poco el programa).
   - En la primera variable “incValue” (línea 12) especificar el caudal en uL/min que se desea obtener.
   - Conectar la placa de Arduino al PC mediante USB y ejecutar el programa.

2. Control de caudal + Red de mensajería:

   Arduino_final_MQTT + Python (1, 2 y 3)

   La red diseñada cumple con la función anterior (la simple), más la notificación y recibimiento de mensajes a través de una red de mensajería mediante el protocolo de comunicación MQTT.

   Instrucciones:

   - Instalar el broker intermediario “mosquitto”:

https://www.luisllamas.es/como-instalar-mosquitto-el-broker-mqtt/

- Configurar mosquitto:

  La configuración de Mosquitto se guarda en el fichero 'mosquitto.conf' que, en el caso de Windows está en la ubicación donde hayáis instalado Mosquitto, y en Linux en 'etc/mosquito'.

  El fichero 'mosquitto.conf' está dividido en secciones, que controlan los principales aspectos del broker. Para más información, consultar la documentación en https://mosquitto.org/man/mosquitto-conf-5.html

  La configuración que yo realicé en Linux (en negro el comando, en azul lo que escribí en el fichero abierto y en verde la respuesta obtenida de la línea de comandos) es la siguiente:

  sudo nano /etc/mosquitto/acl

  #Allow anonymous access to the sys
  topic read $SYS/#

  sudo nano /etc/mosquitto/conf.d/myconfig.conf

  persistence false

  #mqtt
  listener 1883
  protocol mqtt

  #websockets
  #listener 9001
  #protocol websockets

  allow_anonymous true

  acl_file /etc/mosquitto/acl

  mosquitto -c /etc/mosquitto/mosquitto.conf

  1658941559: Loading config file /etc/mosquitto/conf.d/myconfig.conf
  1658941559: Error: Unable to open log file /var/log/mosquitto/mosquitto.log for writing.
  1658941559: Error: Address already in use

  (estos errores no deberían dar problemas)

- Editar el programa de Arduino:

  En la primera variable “incValue” (línea 13) especificar el caudal en uL/min que se desea obtener.

  En la línea 59 cambiar la variable “broker”: asignarle la dirección IP correspondiente al broker instalado. Al estar instalado en el ordenador local es la misma dirección IP del ordenador bajo la conexión WiFi a la que esté conectado. En linux se puede consultar desde la Terminal con el comando: “ifconfig -a” (la dirección IP es la que está definida como *inet* en *wlp3s0*).

- Instalar Python: https://www.python.org/downloads/

- Instalar las siguientes librerías de Python:
  - paho-mqtt
  - datetime
  - time
  - tkinter
  - requests

- Hay 3 programas de Python y cada uno cumple una función:

  1. python1_register.py

     Crea un registro con las características del sistema (hora, minuto, segundo, tiempo en ms, temperatura, frecuencia, voltaje y caudal) editando un fichero por día.

     Una vez ejecutado debe dejarse correr en el ordenador indefinidamente (hasta que se desee parar de almacenar información).

     Parámetros a editar:

     línea 15 “myHostname”: escribir la dirección IP del ordenador local bajo la conexión WiFi que este tenga

     línea 21 “filename”: escribir la dirección del ordenador donde se desee crear el registro y abrir el fichero

  2. python2_alarms.py

     Este programa crea una interfaz gráfica que indica si ha habido una anomalía y de qué tipo, y envía un mensaje de alarma a Telegram.

     Para recibirlo instalar Telegram (en el ordenador, móvil y/o tablet), crear una cuenta en Telegram y unirse al siguiente grupo: https://t.me/+blzqMbFPv70wYzJk

Una vez ejecutado el programa debe dejarse correr en el ordenador indefinidamente (hasta que se desee parar de recibir alarmas).

línea 76 “broker_address”: escribir la dirección IP del broker bajo la conexión WiFi que este tenga

3. python3_modifications.py

   Este programa debe ser ejecutado cuando el usuario desee realizar una modificación remota sin la necesidad de cargar un nuevo programa en la memoria de Arduino:

   1. Un cebado del sistema: aumentar la frecuencia para eliminar el aire de los tubos
   2. Aumentar 50 uL/min el caudal de referencia
   3. Disminuir 50 uL/min el caudal de referencia
   4. Pausar/Reanudar el flujo

   línea 19 “myHostname”: escribir la dirección IP del ordenador local bajo la conexión WiFi que este tenga

- Conectar la placa de Arduino al PC mediante USB y ejecutar el programa de Arduino.

- Ejecutar el o los programas de Python según la función que se desee realizar. Estos son independientes y pueden ser ejecutados simultáneamente. Por ejemplo, para el funcionamiento completo de la red, los programas 1 y 2 deberían estar corriendo indefinidamente y el 3 debería ser ejecutado sólo cuando se desee realizar la modificación remota.