Explicación de códigos e instrucciones de acceso a la Base de Datos

I.- Explicación del funcionamiento de los códigos

I.I Bibliotecas utilizadas

Con respecto a los códigos es importante aclarar en primer lugar las bibliotecas que se debieron utilizar, para ambos códigos se utilizaron las siguientes:

a) pyfirmata: esta biblioteca es quizás la más utilizada a lo largo de ambos códigos, su utilidad es permitir a la placa Raspberry Pi recibir (mediante puerto serial) los datos que ambas placas Arduino capturan de los sensores conectados a sus pines análogos y digitales.

b) time: la biblioteca time se utiliza principalmente para dar un tiempo de espera a los sensores antes de que entregasen su siguiente lectura (debido a que muchos de los sensores no eran capaces de entregarlas correctamente de manera ininterrumpida si es que no existía algún intervalo de por medio)

c) pymongo: tal como su nombre lo indica la biblioteca pymongo es la que permite realizar el enlace entre la base de datos MongoDB y el código escrito en Python. Gracias a esta librería se pudieron realizar las distintas inserciones y consultas a dicha base de datos.

I.II.- Variables importantes

Como se puede ver la mayor parte de estos códigos corresponde a funciones dentro de las cuales se encuentran las tareas más relevantes de los mismos siendo el resto de código poco más que llamadas a estas funciones con distintos argumentos según corresponda y alguna asignación de variables, de todas maneras, de estas variables cabe mencionar:

a) client: la variable client es quien almacena la dirección en donde se encuentra la base de datos para ello se utiliza la función MongoClient de Pymongo.

b) db: es la encargada de almacenar el nombre de la base de datos en donde se encuentran las colecciones con las que se trabajara.

c) colec: existen diversas variables “colec” (+”algo”) en el código, estas variables hacen referencia a “colecciones” que en mongoDB son similares a lo que es una “tabla” en una base de datos relacional por ello es que existe una para cada tipo de sensor.

d) board: es la encargada de almacenar el puerto (serial) en donde se encuentra conectada la placa Arduino para ello se utiliza la función Arduino de pyfirmata. Posteriormente esto permite configurar los pines para recibir señales análogas o digitales mediante la función get_pin y asignar cada pin a una variable con la que se pueda trabajar obteniendo su lectura con la función read.

I.III Funciones

Aclaradas las variables quedaría la explicación de las funciones

I.III.I Comunes (presentes en ambos códigos)

a) EntregarNuevaId: normalmente mongoDB debiese entregar IDs distintas para cada inserción realizada, sin embargo, en la práctica surgieron diversos errores de tipo duplicateKey. Probablemente esto se produjo por trabajar con múltiples sensores que eventualmente podían realizar una inserción casi al mismo tiempo. Por ello la función EntregarNuevaID suma antes de cada inserción una unidad a la última inserción de la base de datos (que se obtiene con la función ObtenerUltimaInsercion).

b) ObtenerUltimaInsercion: a grandes rasgos esta función verifica si es que existen inserciones en la base de datos (para ello recibe como argumento la colección que se quiere revisar), de no existir crea una primera inserción (de ID=0) con la finalidad de que el programa pueda sumar la unidad a la última inserción independiente de si esa inserción está o no presente al momento de ejecutar el código. Luego de esto simplemente se utiliza un sort que ordena las inserciones según su ID para tomar el último elemento de esa lista.

c) RegistroHistorico: tal como se puede apreciar en las otras funciones que realizan inserciones (como registro de Arduino2.py) lo que se inserta finalmente a cada colección en mongoDB son diccionarios. El formato base de ese diccionario (que contiene aspectos como fecha, hora, lectura, id del sensor, etc.) es pasado a estas funciones como argumento (ejemplo) a lo que adicionalmente se le suman argumentos para poder identificar que sensor realizo la lectura en esta función dicho argumento es un diccionario que contiene el ID de cada sensor como llave y su nombre y contador como valor. Con estos elementos se pueden realizar las inserciones que como tales se llevan a cabo cada 5 minutos (el intervalo se establece con el uso de la función time) independiente de la lectura del sensor.

I.III.II Arduino1.py

a) guardarenlocal: dentro del sistema estaba presente un módulo relee, su funcionalidad consistía en encender o apagar dos luces leds según el horario y día que se indicase en la base de datos. Como estos horarios no serían cambiados de manera constante por el usuario se decidió mantener los horarios en una base de datos local de Raspberry que se iría actualizando cada 30 minutos. Para esta inserción se debían rescatar primero los datos insertados en la base de datos y para ello se crearon las funciones consultaRele y rescatarDatos.

b) ConsultaRele: su única funcionalidad es rescatar los datos desde la base de datos en la web y guardarlos en una lista.

c) rescatarDatos: rescatarDatos más que hacer lo que su nombre indica tiene como funcionalidad asignar a variables locales los datos rescatados por ConsultaRele para que estos puedan pasar como parámetro a la función guardarenlocal. De todas maneras, antes de guardar las variables se verifica si es que las horas de encendido y apagado deben o no "funcionar" ese día para ello cada día de la semana tiene un valor booleano asociado.

d) saltarcero: como tal saltarcero no es muy distinta en su funcionamiento a RegistroHistorico (igualmente se trabaja con diccionarios, id de sensores, etc) pero existe una diferencia importante. El nombre saltarCero se debe a que el sensor magnético puede eventualmente sufrir interferencias que alteren sus lecturas, interferencias que se manifiestan como un valor 0.0 cuando los sensores están separados (en cuyo caso el valor siempre debería ser mayor a 0.0) por ello es que antes de determinar un cierre o apertura se debe repetir el valor 0.0 o bien un valor mayor a él al menos 2 veces (pues en todas las pruebas realizadas nunca existió más de un 0.0 sucesivo producido por interferencia) "saltando" de esta manera los 0 producidos por interferencia.

I.III.III Arduino2.py

a) Registro: funciona de manera análoga a RegistroHistorico, la salvedad es que las inserciones se realizan únicamente cuando existen "cambios de estado" (por ello existen argumentos como el valor umbral necesarios para determinar esos cambios) de esta manera se evita que sensores como mq7 o funduino realicen inserciones de manera ininterrumpida sino que más bien realicen inserciones más relevantes para el usuario como puede ser el momento en el que se produce alguna fuga y el momento en el que está ya no está presente. La única excepción a esto es el sensor de corriente que, por su naturaleza de indicar consumo eléctrico, más que alguna fuga o alerta, realiza inserciones en intervalos definidos de 5 minutos.

II.- Acceso a la Base de Datos en mLab

Para acceder a la base de datos mLab que contiene la información que se capturo a lo largo de la práctica se debe utilizar el siguiente link:

https://mlab.com/login/

nombre de usuario: hha172

contraseña: clave123

En mongoDB Deployments se debe acceder a pruebaaa lo que mostrara las distintas colecciones que contienen los documentos con la información.