Trabajo Práctico Nº1

Ejercicio 1

Luego de declarar las dependencias en el archivo de texto requierements.txt, procedemos a instalarlas ejecutando el comando sudo pip install -r requierements.txt. Las dependencias que instalamos son el servidor Flask y la base de datos de MySQL.

Un ejemplo de uso del servidor Flask es el siguiente:

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
	return 'Hola amigos de Geeky Theory!'

if __name__ == '__main__':
	app.run(host='0.0.0.0')
```

Se puede observar que luego que el cliente ingresa a través de un explorador de web la ip 0.0.0.0 de localhost se envia una petición de acceso al servidor el cual crea una instancia de *app* con el argumento ***__main__*** necesario para que Flask busque files, plantillas y otros archivos. Luego el cliente recibirá la respuesta generada por el servidor definida en la función de la ruta "/" la cual retorna un string en pantalla del navegador.

Ejercicio 2

Una de las formas de demostrar como llega la información de python del lado del cliente en forma de HTML es crear una función que le retorne al cliente distintos datos que ha ingresado en la pantalla.

El siguiente codigo pertenece al archivo de python:

```
@app.route('/form', methods = ['GET'])
def action_form (frecuencia= None):
	frecuencia = int(request.args["frecuencia"])
	return render_template('response.html', frec= frecuencia)
```

Para analizar la información que llega al cliente se muestra el código HTML de response.html en el cual se obtiene la variable frec pasada por parámetro:

```
<html>
    <head>
        <h1>Datos Recibido</h1>
    </head>

      <body>
            <p> La variable recibida de python se informa de la
            siguiente manera {{ frec }} </p>
      </body>
</html>
```

Ejercicio 3

Para realizar esta solución se crearon dos archivos en python los cuales uno actua como productor de datos (productor.py). Las acciones que realiza este proceso es:

1. El cliente debe ingresar el periodo de muestreo.
2. Conectar la base de datos MySQL.
3. Generar dato aleatorio.
4. Realizar la consulta en MySQL para ingresar el dato generado.
5. Realizar commit de la consulta realizada para que el dato generado quede permanente en la base de datos.
6. Vuelve a realizar la acción del punto 2. hasta que se hayan ingresado 10 datos.
7. Cerrar la conexión con la base de datos.
8. Informar que se terminó de agregar datos.
9. Mostrar un botón que permita insertar más datos en la base de datos.

El otro proceso es el consume los datos de la base de datos MySQL (consumidor.py). Las acciones que realiza este proceso es:

1. El cliente debe ingresar el periodo de muestreo.
2. Conectar la base de datos MySQL.
3. Realizar la consulta en MySQL para todos los datos actuales presentes en la misma.
4. Enviar los datos al cliente en formato HTML.
5. El script del lado de HTML procesa todos los datos recibidos y va mostrando de a uno cada cierto tiempo de muestreo definido por el cliente.
6. Por último está la opción de oprimir un botón para procesar más datos de la base de datos.

Ejercicio 4

Para este ejercicio se utilizó la misma solución que el ejercicio anterior. La única modificación realizada es la plantilla inicial (“form.html”). En la misma se insertó el tag select para que el usuario elija entre las opciones predefinidas para el periodo de muestreo.

Ejercicio 5

La problemática de la concurrencia existente en la simulación se halla en que ambos procesos intentan acceder al mismo tiempo a la base de datos (utilizar el mismo recurso), lo que produciría un bloqueo en el acceso de uno de los procesos. Al mismo tiempo, al agregar la posibilidad de cambiar el período de muestreo, lo que ocurriría es que puede que uno de los dos procesos este a una frecuencia mayor que el otro. Lo que esto conlleva a que uno intente leer información que el otro no escribió todavía.

En el caso de que sea en un ambiente real en el cual estamos queriendo medir diversas variables a una frecuencia indicada, deberíamos analizar la frecuencia a la que el sensor nos envía los datos para así configurar el microcontrolador a la misma frecuencia para no perder datos, como así también, no leer datos erróneos o señales sin sentido. Esto ocurriría en el caso de que uno intenta escribir información que el otro no alcanza a leer y también se repite el caso de error mencionado en la simulación.

Los problemas de concurrencia existentes en el ambiente real podrían darse en el caso de que existan varios sensores conectados a la placa intentando enviarle información por el mismo puerto debería coordinarse los tiempos de envío porque sino existirían colisiones entre los valores enviados, produciendo así datos incorrectos. Otro posible problema se da cuando todos los sensores están conectados a distintos puertos y el microcontrolador los va sensando de a uno pero cuando este intenta enviarlos al servidor (y este está siendo accedido por lectores) no puede acceder, por lo tanto, almacena los valores hasta tener acceso. Esto dependerá de la memoria que se disponga, si esta se agota por el tiempo de espera se perderán valores.
Posibles problemas de tiempo real que podrían producirse en general:

- No tener responsividad. Los sensores envían sus datos mediante interrupciones, generando así anidamientos para atenderlas e informando así datos críticos cuando ya el suceso pasó.
- No tener determinismo. Llegó un valor de un sensor por el cual había que activar un actuador de una manera determinada y la señal hacia este no llegó, produciendo así una falla en la funcionalidad final del sistema.

- No tener confiabilidad. El sistema en ambientes electromagnéticos o a determinada temperatura no anda o los sensores envían datos erróneos.

Ejercicio 6

En la simulación planteada por nosotros tenemos una aplicación que genera datos corriendo en un puerto, la cual escribe en una base de datos y, a su vez, tenemos una segunda aplicación que lee la base de datos y muestrea a una determinada frecuencia los datos. En este caso no existirían errores como los analizados ya que la base de datos no es leída por más de un proceso en simultáneo, de la misma manera que no se pierden datos nunca y no existen datos erróneos.
En el caso real podrían producirse datos erroneos por algun problema en sensores o por colisiones en las redes de comunicación utilizadas y, a su vez, se podría perder información porque los sensores no mantienen los datos por tiempo infinito y si el microcontrolador no los sensa en el momento apropiado estos se perderían.