Universidad
Carlos III de Madrid

Trabajos de Informática Industrial 1

Grado en Ingeniería Electrónica Industrial y Automática

2013-2014

Profesores: Todos los profesores de la asignatura (2013-14)

Normativa

Los trabajos propuestos se podrán realizar tanto individualmente como en grupos de dos alumnos como máximo**. Los trabajos tendrán un cupo máximo limitado y se asignarán por orden de solicitud.**

Será obligatorio entregar tanto el código fuente como la documentación, que constará de un informe explicativo, los diagramas **UML** y de la documentación autogenerada por **Doxygen** si se usa este último. Será recomendable tener un manual de usuario para el manejo del programa.

Los alumnos deberán notificar a su profesor de grupo pequeño, antes del viernes 11 de octubre, quienes son los componentes de cada grupo y qué trabajo van a realizar (con una segunda opción de trabajo si el primero no está disponible). La lista de asignación definitiva se publicará el **martes 15 de octubre**. Los trabajos tendrán que ser defendidos bajo la supervisión del profesor. Las fechas oficiales para la defensa de los trabajos serán los días comprendidos entre el **16 y 20 de diciembre del 2013**. Se deberá acordar una cita previa con el profesor para la defensa. Si algún grupo desea presentar antes de la fecha fijada, tendrá que acordar un día y hora con el profesor responsable. La fecha tope para entregar todos los documentos finales y el código fuente será el día de la defensa acordada. Existirá un cupo máximo de trabajos por día que se asignará por orden de solicitud.

Se deberá hacer uso de la **sintaxis de C++**, pudiéndose usar las librerías necesarias, y seguir el estilo **Doxygen** para los comentarios del código fuente. El diseño y la implementación de los trabajos deberán ser orientados a objetos. Entre otras cosas, se valorará especialmente el diseño de la solución propuesta, la calidad, la claridad, la reutilización de código, la robustez y el control de errores mediante excepciones.

Para la generación del diagramado UML se recomienda la utilización de la herramienta de modelado software de código abierto, **Dia**.

Documentación a entregar

- Tras la asignación del trabajo y en un plazo de una semana deberá hacerse la primera entrega, que contendrá una descripción breve del planteamiento del proyecto (un par de hojas máximo, una propuesta inicial de diagrama de clases, la descripción de algunos casos de uso).
- Para la entrega final:
    - Informe de contenido.
    - Modelado del software en UML (diagramas que se consideren necesarios).
    - Código fuente y ejecutable con todos los ficheros necesarios para su funcionamiento. Deberá ser un proyecto en QtCreator el cual pueda ser cargado para su posterior análisis y ejecución.
    -

Notas:

- El trabajo a evaluar será el entregado en Aula Global 2, no aceptándose así cualquier modificación posterior. Cualquier indicio de copia entre trabajos hará que los grupos implicados suspendan en el acto.
- Se recomienda documentar el código usando el formato Doxygen. Doxygen podrá ser utilizado (no es obligatorio) para general la documentación final a entregar.

Trabajo 1: Sistema de Control Domótico (SCD) para una vivienda

Se propone realizar un programa que permita configurar y controlar los diferentes dispositivos domóticos de una vivienda.

Las principales tareas que deberá realizar son las que se detallan a continuación:

- Encender/apagar luces, subir/bajar persianas, encender/apagar calefacción y aire acondicionado, controlar el termostato (temperatura, velocidad del viento, etc.), abrir/cerrar ventanas y puertas, encender/apagar la TV y la radio, y detectar un intruso. Monitorizar temperatura, humedad, luminosidad, ruido, alarmas de agua/gas/fuego.
- Para ello se creará un programa que tenga las siguientes partes:
    - **Configuración inicial**: el usuario configura su casa indicando las estancias que la componen y añadiendo dispositivos de una lista.
    - **Telecontrol**: una vez configurada la casa, el usuario podrá controlar y monitorizar todos los dispositivos desde la interfaz que se haya diseñado (modo texto o gráfica).

Se recomienda hacer:

- Implementación de un módulo de gestión de usuarios, de manera que cada usuario tendrá su perfil y la casa se configurará en función de sus preferencias. Cada usuario tendrá unos privilegios que le permitirán acceder o no a ciertos dispositivos. Como mínimo habrá dos tipos de usuarios, Administrador y Usuario general.
- Además, el sistema será capaz de tomar ciertas decisiones por sí mismo, de manera que si hay poca luminosidad encenderá las luces y/o subirá las persianas, si hay alarma de agua cerrará la llave del agua, etc. Como mínimo habrá que implementar cinco controles automáticos.

Se valorará positivamente:

- Implementación de una función de simulación de presencia, de manera que el sistema simule la presencia de usuarios no identificados (para evitar robos) mediante el encendido y apagado de dispositivos.
- Se valorará positivamente el uso de interfaz gráfica pero no compensará una baja calidad del trabajo

Trabajo 2: Sistema de optimización del tráfico de una ciudad

El ayuntamiento de una ciudad quiere mejorar el tráfico en las horas punta. Por ese motivo, propone la implementación de un sistema que ayude a los conductores a elegir la ruta más óptima para llegar a su destino.

Por las calles de la ciudad transitan vehículos desde una intersección origen a una intersección destino. Estos vehículos pueden ser coches, motocicletas y camiones de reparto. La velocidad máxima de una calle dependerá del barrio donde está situado. De las calles se conoce su longitud y el grado de congestión.

a) b)

Figura 1. Ejemplo de ciudad donde cada segmento es una calle; a) Grado de congestión: rojo = alto, amarillo = medio, verde = bajo; b) Barrios: azul = centro, naranja = residencial, verde = industrial.

Se pide diseñar e implementar un sistema que permita a un vehículo, dada su posición actual (una intersección de calles), obtener la ruta más óptima hasta su destino (otra intersección) teniendo en cuenta las condiciones del enunciado. Para esto es necesaria la implementación de un interfaz que deberá mostrar además el estado actual de la ciudad y la ruta sugerida.

Se valorará positivamente el uso de herencia y abstracción siempre que sea posible y plantillas de código.

Se valorará además la posibilidad de modificar en tiempo de ejecución las condiciones de congestión de la ciudad.

Trabajo 3: Evitar colisiones con los obstáculos para manipuladores

Tenemos un brazo robótico de dos barras y la tercera es el elemento terminal. Las barras del robot están conectadas entre sí mediante pares cinemáticas (uniones). Los obstáculos están formados por tres patrones: esfera, cilindro, y prisma rectangular. Los obstáculos tienen posiciones estáticas en el entorno de trabajo del manipulador. Un camino está compuesto por una serie de configuraciones y cada configuración se define con los ángulos de las uniones (ver figura inferior). Considerando que el camino del brazo está previamente calculado, el trabajo a realizar por los alumnos será la detección de las posibles colisiones entre las barras del brazo y los obstáculos en el entorno de trabajo.

El brazo robótico se moverá siempre en el plano X,Y mientras que los obstáculos están situados en el espacio 3D. OPCIONAL: Permitir que el brazo robótico se mueva en 3D (para ello se proporcionará otra versión del robot).

Para modelar el problema se proponen las siguientes clases, aunque es trabajo del alumno identificar las relaciones entre ellas:Las principales tareas que deberá realizar son las que se detallan a continuación:

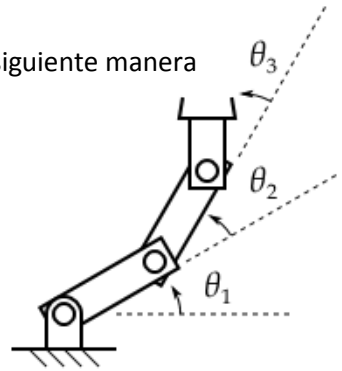
Link: define las características físicas de las barras del brazo (longitud,…).

Configuration: define una configuración del brazo en base a los ángulos de las uniones. Calcular las coordenadas Cartesianas de cualquier punto del manipulador mediante otro objeto se llama **DirectKinematic**. OPCIONAL: construir una clase para modelar la cinemática inversa que se llame InvKinematics.

ROBOT: que define un brazo.

Point: define cualquier punto en 3D.

Obstacle: define los obstáculos en el entorno del trabajo. Debe definirse de la siguiente manera




```
Template <class T = double>
class Obstacle {
public:
        virtual bool isPointInObstacle(Point & p) = 0;
        virtual bool isLinkInObstacle(Point & p1, Point & p2, T r) = 0;
        virtual void print(ostream&) = 0;
};
Template <class T = double> class Sphere: public Obstacle { …… }
Template <class T = double> class Sphere: public Obstacle { …… }
Template <class T = double> class Sphere: public Obstacle { …… }
```

Para detectar las colisiones consideramos la esfera como un punto en el espacio y las barras del robot y los cilindros como segmentos. Aun así, a la hora de calcular la distancia mínima entre los obstáculos y las barras del brazo debe tenerse en cuenta el tamaño real de los objetos y del brazo. Se proponen tres funciones para calcular las distancias mínimas. Tenéis que saber la mejor posición de esas funciones:

```
T dist_Sphere_Link(Point SP0, Point LP1, Point LP2);
T dist_Cylinder_Link(Point CP1, Point CP2, Point LP1, Point LP2);
T dist_Plane_Link(Point PP1, Point PP2, Point PP3, Point PP4, Point LP1, Point LP2);
```

Nota: Las ecuaciones para el cálculo de la cinemática serán facilitadas a los alumnos.

Trabajo 4: Sistema de Administración de un Banco

El objetivo de este proyecto es desarrollar un sistema de gestión para un banco que permita la creación y el mantenimiento de las cuentas. El sistema proporcionará la capacidad de realizar transacciones, depósitos y retiradas de efectivo. Es necesario también mantener una base de datos de cuentas y transacciones para permitir realizar consultas sobre el estado del sistema. A continuación se presentan las principales características del sistema:

1) El usuario debe ser capaz de crear una nueva cuenta, que deberá contener un número de cuenta (código numérico único de seis dígitos autogenerado), fecha de creación de la cuenta (día, mes, año), Título (Sr., Srta., Sra.), nombre y apellidos del titular de la cuenta, tipo de cuenta (cuentas de ahorro o cuenta corriente) y el saldo.

2) Una cuenta también debe tener un campo adicional para almacenar la fecha de cierre de la cuenta (día, mes, año). Este campo debe ser NULL, mientras la cuenta esté activa.

3) El usuario debe ser capaz de depositar dinero en una cuenta.

4) El usuario debe ser capaz de retirar dinero de una cuenta. La cantidad retirada debe ser menor que o igual a la saldo en la cuenta.

5) El usuario debe ser capaz de verificar el saldo de una cuenta.

6) El usuario debe ser capaz de modificar los campos de título, nombre, apellido y tipo de cuenta asociados a una cuenta.

7) El usuario debe ser capaz de cerrar una cuenta. El cierre de una cuenta debe iniciar automáticamente una operación de retirar si el saldo es superior a 0 euros. Además, sobre el cierre de la cuenta, el campo fecha de cierre de la cuenta, debe modificarse. Este campo no puede contener una fecha que sea anterior a la fecha del campo fecha de creación de la cuenta.

8) Cada transacción (un depósito o una retirada de efectivo) debe ser registrada y vinculada a la cuenta en cuestión. Cada transacción debe tener campos como número de la cuenta, tipo de transacción (depósito o retirada), montante de la transacción, y saldo.

9) El usuario debe ser capaz de enviar una consulta para mostrar todas las cuentas activas en el sistema, según el tipo de cuenta (ahorros, corriente o todos). El resultado debe contener el número de cuenta, título, nombre, apellido y saldo de cada cuenta.

10) Para un número de cuenta determinado, el usuario debe ser capaz de consultar todas las transacciones según el tipo de transacción (depósitos, retirada o todos). El resultado debe contener el tipo de transacción, monto de la transacción y saldo de cada transacción.

11) El usuario podrá buscar entre todas las transacciones en el sistema según el tipo de transacción (depósitos, retirada o todos). El resultado debe contener el número de cuenta, tipo de transacción, monto de la transacción y saldo.

12) El usuario debe ser capaz de salir del sistema. Al salir, el estado del sistema (cuentas y transacciones) se debe escribir en un fichero. Cuando se vuelve a abrir el programa, el programa primero debe leer este fichero para cargar las listas de cuentas y transacciones del sistema.

Trabajo 5: Simular una red de telecomunicaciones simple

Se propone simular una red de telecomunicaciones simple, en la que varios abonados pueden darse de alta o de baja. Para ello, se debe de partir de los siguientes requisitos:

- La red de telecomunicaciones estará formada por un nodo principal (NP) y varios nodos secundarios (NS), de forma que todos los nodos secundarios se conectan al principal.

- Los abonados se conectan a su nodo secundario correspondiente; un abonado deberá de ser identificado por un identificador (número de 4 cifras).

- Los abonados pueden disponer del siguiente equipamiento para realizar las comunicaciones: Teléfono fijo (TF), teléfono 4G wifi conectado a router (TW), PC con programa de conferencias multimedia de tipo Netmeeting (PC).

El programa deberá de dotar a cada abonado de un equipamiento específico, existiendo tres tipos de abonados:

- Abonado tradicional: solo dispone de un teléfono fijo (TF)

- Abonado moderno: dispone de un teléfono 4G wifi con router (TW)

- Abonado avanzado dispone de un teléfono 4G wifi (TW) y PC con Netmeeting (PC).

Las comunicaciones que se pueden realizar son las siguientes:

- Los teléfonos fijos solo pueden hacer llamadas de voz hacia otros teléfonos fijos o teléfonos Wifi 4G. Y solo pueden recibir llamadas de voz de otros teléfonos fijos o teléfonos 4G wifi.

- Los teléfonos 4G wifi pueden hacer llamadas de voz a teléfonos 4Gwifi y teléfonos fijos, y recibir llamadas de teléfonos fijos y teléfonos 4G wifi. Además, puede iniciar videoconferencias hacia otros teléfonos 4G wifi y PCs, así como recibir peticiones de inicio de videoconferencia de otros teléfonos 4g wifi y PCs

- PCs pueden realizar videoconferencias hacia otros PCs y teléfonos 4G wifi, y recibir peticiones de videoconferencia de otros PCs o teléfonos 4G wifi. Nunca pueden realizar llamadas de voz.

El programa deberá de sacar por pantalla y en fichero (fichero “factura.txt”) una factura, en la que se mostrará para cada abonado el coste de sus comunicaciones. Como unidad monetaria se utilizará la moneda ficticia “bolo”, de forma que una llamada telefónica cuesta 1 bolo, y una videconferencia cuesta 2 bolos. Se añadirá a todos los abonados una tarifa de enganche de línea por valor de 12 bolos. Cada vez que una comunicación precise acceder al NP, se añadirá 0,5 bolos a la comunicación. Como parte opcional, se propone la posibilidad de que los teléfonos 4G wifi (TW) y los teléfonos fijos (TF) tengan la posibilidad de disponer del servicio suplementario “Conferencia a tres”. La tarificación de una conferencia a tres será de 3 bolos. Si en cualquier parte de la comunicación se precisa pasar a través del NP, se añadirá a la tarificación 4 bolos por comunicación.

Si un abonado se da de baja, se debe de mantener su información de tarificación. Todos los mensajes relativos a las comunicaciones deberán de salir por pantalla y se almacenarán en un fichero “comunic.txt”. Se podrá solicitar a través de línea de comandos la tarificación de los abonados. Antes de salir del programa, éste deberá de almacenar la tarificación de todos los abonados en el fichero “factura.txt”.

Trabajo 6: Gestor de Tarjetas de Fidelización

Este trabajo consiste en hacer un programa que ejecuta unas de las tareas relacionadas con la gestión de las Tarjetas de Fidelización (TF) de una empresa. Más concretamente se pide:

1. Procesar los Registros de Ventas (RV) de la empresa para actualizar la lista de los datos de los clientes (por ejemplo puntos acumulados según compras etc.).
2. Actualizar la lista de las TF (por ejemplo cambiar el estado de TF, introducir una TF en el sistema, etc.).
3. Generar ofertas/descuentos particularizados para cada cliente según sus puntos acumulados y en relación con los productos que consume (ofreciéndole productos del mismo subgrupo y rango de precio).

Los resultados de las tareas se almacenarán en distintos ficheros de texto (donde se indicará en cabecera el tipo de parámetro de cada columna). La información (registros) que alimentará los procesos pedidos se suministrará en ficheros de texto.

Para ejecutar las tareas pedidas se utilizarán por lo menos los elementos siguientes:

1. Tarjeta, que contendrá:
   1.1. Identificador único de tarjeta en el sistema.
   1.2. Fecha de suministro.
   1.3. Fecha de envío al cliente.
   1.4. Fecha de baja.
   1.5. Estado (en stock, enviada al cliente (en uso), baja, etc.).
2. Cliente, que contendrá:
   2.1. Identificador único de cliente en el sistema.
   2.2. Dirección de domicilio completa.
   2.3. Fecha de alta.
   2.4. Fecha de baja.
   2.5. Puntos acumulados por compras realizadas.
3. Producto comercializado, que contendrá:
   3.1. Identificador único de producto.
   3.2. Descripción.
   3.3. Fecha de inicio de comercialización.
   3.4. Fecha de fin de comercialización.
   3.5. Característica 1.
   3.6. Característica 2.
   3.7. Característica 3.
   3.8. Característica 4.
   3.9. Grupo (1, 2, 3 etc.).
   3.10. Subgrupo (1, 2, 3, etc.).
   3.11. Rango de precio (1, 2, 3 etc.).

Los registros de compra contendrán los campos siguientes:

1. Lugar de operación.
2. Fecha de operación.
3. Hora de operación.
4. Identificador de TF.
5. Tipo de operación (compra o devolución).
6. Resultado (ON, NOK).
7. Códigos de productos comprados.
8. Importe total de operación.
9. Modo de pago (efectivo, tarjeta bancaria o TF es decir pago con puntos acumulados).
10. Puntos correspondientes a la operación (Si es un pago con TF, indica los puntos utilizados. Si es una devolución, indica los puntos devueltos).