RÉSEAUX LOCAUX ET INDUSTRIELS
Lab. Session 2 & 3 - Elevator with MODBUS

In this second lab, the main goal is to introduce you to

- Visual Basic programming
- *MODBUS TCP Protocol*

1 Grading Procedure

Ask me to check your lab when you have finished. I also ask you to:

- submit your project on http://campus.ece.fr
- in **one week (your lab ends at 16h45 on Tuesday ? You have until Tuesday next week, before 16h45, to submit your file)** after your session
- in **ZIP** file format with the name:

[RLI][TP2][Gr**TD-SubGroup**] Name1 & Name2 & Name3

“TD” is the group number according to the timetable (1, 2, 3 etc) and “SubGroup” is your team group number assigned this year. **Any files that do not meet these constraints will be ignored**

Ok, so, imagine you are Mr Durant in group 4, and we assigned to you and your partner (Mr Al-Kâtib) the team number 13.
Your file will be : **[RLI][TP2][Gr4-13] Al-Kâtib & Durant**
No [GrSEI4-13], nor [4-13], nor [GR4-13], neither DURANT etc...

Each day late will be penalized by **-1 point**. This lab is graded on **20 points.**

2 Questions

For this lab, you will work on a prepared project. It is an elevator controlled with an automaton. This automaton has **2** outputs/Coils (move the elevator **up** or **down**) and **5** inputs/Sensors (a sensor **BETWEEN** each floor. So, if the elevator is exactly at floor 1, no sensors are used. When the elevator move from floor 1 to floor 2, the sensor between these two floors is used). Download this zipped project at http://campus.ece.fr.

The program, in Visual Basic, will represent the automaton. In the first part of this lab, I ask you to implement the functionalities of this automaton, the sensors and the coils. In the second part, one computer will be the automaton and another the elevator. They will communicate through a socket with the *MODBUS* Protocol. The connection is already implemented in "ClientClass.vb" and "ServerClass.vb". You only need to work on the file "Elevator.vb".

2.1 Implementation of the automaton

On point 1.a and 1.b, you have to implement the real function of the system in terms of **wires**. Imagine a real industrial system: when you check the "CheckBox UP", you send 1 on a wire connected to a motor. When the elevator is in front of a sensor, you will receive a "1" on the corresponding wire.

On point 2, you implement the function of the automaton. According to information coming from sensors (a "1" or a "0") and option on buttons "Call Floor", you have to choose the action on outputs : up, down or stop.

1. **Wired Part** (Server/Slave : Sensors+Coils)
   (a) In the "Coils/Outputs":
   - While the "CheckBox" is checked, the elevator has to move. But it can not fly nor dig...
   - To change the position of the elevator, e.g. *Me.ElevatorPhys.Location = New Point(Me.ElevatorPhys.Location.X, Me.ElevatorPhys.Location.Y - 1)*

  - To test the "CheckBox", you have to look at boolean *Me.CoilUP.Checked* e.g.
  - To simulate a real time movement, you can start or stop a timer e.g. Do **NOT** use a **WHILE LOOP**!

(b) In the "Sensors/Inputs":
  - While the elevator is in front of the sensor (**between** two floors, you could use *Me.PositionSensor1.Location.Y* to get the position of the Sensor1), the led has to be turned on
  - To change the color of LedSensor, e.g. *LedSensor0.BackColor = Color.Green*

2. **Automaton part** (Client/Master : Decision+Call floor buttons):
   - Each time one of these buttons is pushed, the elevator has to move to the required location
   - **According to sensors**, you have to move the elevator by changing the outputs (the **checkboxes** "CoilUp" and "CoilDown"). **Your are the automaton**, not God (I hope so...): all you can see is the states of the sensors (color of "LedSensor0.BackColor" etc), not the real position of the elevator.
   - You can add some options as multiple calls, lights in the elevator (property "BackColor" e.g.)...

2.2 MODBUS TCP Protocol

You have to read a description of *MODBUS TCP Protocol* on http://campus.ece.fr. The principle is quite simple. Each "bit" is a connector on a board. A 1 means *ON*, 0 *OFF*. I give you the following connections :

Input		Output	
Element	Connected Wire	Element	Connected Wire
Sensor 0	0	CoilUP	0
Sensor 1	1	CoilDown	1
Sensor 2	2		
Sensor 3	3		
Sensor 4	4		

For example, if the client (the master) wants to know the state of the first eight sensors (e.g. connectors/bits 0 to 7), it will send a datagram of 12 bytes with a FC2 function:

Byte	Field name	Example
Byte 0,1	Transaction identifier	0x0000
Byte 2,3	Protocol identifier	0x0000
Byte 4,5	Length field	0x0006
Byte 6	Unit identifier	0x01 not used
Byte 7	MODBUS function code	0x02
Byte 8,9	Reference number	0x0000
Byte 10,11	Bit count	0x0008

Table 1: FC2 – Read Discrete Input

Bytes 4 and 5 indicate the total number of bytes starting from byte 6. Bytes 10 and 11 indicate the total number of **bits** wanted in the response. Byte 7 indicates the function (FC2 : Read Discrete Input). Bytes 8 and 9 indicate the address where to start the function. Thus, if you want to read connectors/bits 3 and 4 on the slave, you will set Bytes(8,9) = 0003, Bytes(10,11) = 0002.

The slave will respond a datagram with the same values in the first 7 bytes and complete with the following:
Byte 7 repeats the function called by the master. Byte 8 indicates the total number of **bytes** in the response. Byte 9 is the response where each bit represents a connector/sensor.

1. You have to use the MODBUS TCP Protocol to communicate between your client/master and your server/slave
2. On the master, you will be able to call the elevator but the elevator will not move. It will ask for sensors states and force coils states on the slave using MODBUS.
3. On the slave, you will see the elevator move according to master controls, display real sensor states and up/down checkbox states.
4. You have to implements MODBUS functions FC1, FC2, FC5 and FC15.

Some useful indications on Bytes:

Byte	Field name	Example
Byte 0,1	Transaction identifier	0x0000
Byte 2,3	Protocol identifier	0x0000
Byte 4,5	Length field	0x0004
Byte 6	Unit identifier	0x01 not used
Byte 7	MODBUS function code	0x02
Byte 8	Byte count	0x01
Byte 9	Bit values	0x12

Table 2: FC2 – Response

```
Dim test As Byte() = new Byte(4) {} 'Declare a table of 5 Bytes
test(0) = 0 'Set the first byte to 0, in decimal
test(1) = &H10 'Set the second byte to 10, in hexadecimal
test(0) = test(0) Or 8 'Set the fourth bit of the first byte to 1
test(1) = test(1) And Not 16 'Set the fifth bit of the second byte to 0
Encoding.ASCII.GetString(msgBytes) 'Get a String from a table of Bytes
Encoding.ASCII.GetBytes("COUCOU") 'Get a table of Bytes from a String
```

3 Evaluation

[14pts] Implementation of the automaton

4pts In the "Coils/Outputs":
- 2pts + 1pt While the "CheckBox" is checked, the elevator has to move [2pts]. But it can not fly nor dig... [1pt]
- 1pts To simulate a real time movement, you can start or stop a timer each time an output changes e.g.

4pts In the "Sensors/Inputs":
- 2pts + 1pt While [2pts] the elevator is crossing a sensor (between two floors [1pts]), its led has to be turned on
- 1pt To change the color of LedSensor, e.g. *LedSensor0.BackColor = Color.Green*

6pts With "Call Floor" buttons (**Automaton part**):
- 4pts Each time one floor is called, the elevator has to move to the required location [1pt for each floor]
- 2pts According to sensors, you have to move the elevator by changing the outputs (the checkboxes "CoilUp" and "CoilDown")
- 0.5pt Bonus You can add some options such as multiple calls, lights in the elevator (property "BackColor" e.g.)...

[6pts] MODBUS TCP Protocol

4pts MODBUS functions used ([2pts for each function])
2pts Interpretation of messages and functionality of the network part