UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Politècnica Superior d'Enginyeria de Manresa

Treball Final de Grau

DISSENY I CONFIGURACIÓ D'UN SCADA PER A UNA PLANTA DE LABORATORI

Grau en Enginyeria Electrònica Industrial i Automàtica

Curs 20/21

Autor: Guillem Pla Pujols

Director: Teresa Escobet Canal

Data: 08/10/2021

Localitat: Avinyó

RESUM DEL PROJECTE

En aquest projecte es fa la creació i millora d'una aplicació industrial per al control d'un procés de biolixiviació. Aquest projecte continua la tasca feta anteriorment en la idea d'automatitzar una planta de biolixiviació, que s'està desenvolupant en el grup de recerca BIOGAP – Grup de Tractament Biològic de Contaminants Gasosos i Olors – de l'EPSEM. Per fer-ho, es parteix del hardware i el software que es van realitzar en 2 treballs de final de grau anteriors. Pel que fa al hardware, es manté igual. Però, pel que fa al software, aquest es remodela i s'introdueixen noves funcionalitats, es simplifica l'aplicació i també s'alleugereix de càrrega. Pel que fa al sistema operatiu de la Raspberry on s'allotjarà la aplicació, aquest es canvia i es fa des del sistema operatiu Windows IoT.

ABSTRACT

In this project, the creation and performance of an industrial application to control a biolixiviation process are done. This project continues the task done previously in the idea of automatizing a biolixiviation plant, which is being developed by the EPSEM's BIOGAP research group – Grup de Tractament Biològic de Contaminants Gasosos i Olors. To do it, we start from the hardware and software done in 2 previous final degree projects. Regarding the hardware, it stays the same. But, regarding the software, this is remodeled and new functionalities are added, the application is simplified and also the load is lightened up. Speaking about the Raspberry's operating system where the application will be hosted, it is changed and is done from the Windows IoT operating system.

ÍNDEX

ÍNDEX DE FIGURES

ÍNDEX DE TAULES

1. INTRODUCCIÓ

Aquest treball de final de grau és la continuació de dos projectes de final de grau anteriors. En aquests treballs es va realitzar el control del procés de biolixiviació, es va dissenyar un hardware i un software per controlar el procés i també es va dissenyar una base de dades per emmagatzemar informació de la aplicació.

En aquesta tercera fase, el projecte té com a objectiu general la creació i/o millora de l'aplicació SCADA per visualitzar la planta de biolixiviació. L'objectiu general s'ha desglossat en els següents objectius parcials:

- Alleugerir la càrrega de l'aplicació canviant fotos pesades per eines simples.
- Canvi de sistema operatiu a Windows IoT.
- Simplificació de l'aplicació per ser més comprensible de cara a un possible operari.

El maquinari sobre el que s'ha realitzat el projecte ens ha vingut imposat i concretament es disposa d'un Rasberry Pi connectada amb una pantalla HMI.

Per assolir els objectius s'han realitzat les següents tasques:

1. S'ha fet una cerca de com de com instal·lar el sistema operatiu Windows IoT en una a Rasberry Pi.
2. S'han instal·lat sobre de la Rasberry Pi, el sistema operatiu Windows IoT i el programa Visual Studio 2019, aquest últim ha permès desenvolupar el nou projecte.
3. Per el nou desenvolupament s'ha utilitzat la interfície d'usuari Windows Presentation Foundation, que permet desenvolupar un entorn gràfic interactiu.
4. Familiarització amb l'eina Visual Studio.
5. S'ha fet l'edició d'una pantalla gràfica per a la monitorització de la planta de biolixiviació.

Pel que fa a la pantalla gràfica creada, ens permet realitzar diverses funcions:

- Veure l'estat dels sensors de nivell.
- Veure l'estat de les bombes.
- Veure els valor en directe dels diversos sensors analògics.
- Veure les alarmes de l'aplicació en directe.
- Veure una llista amb els temps de funcionament d'algunes bombes.
- Veure un historial dels valors dels sensors guardats cada 20 segons.
- Veure el color dels sensor de color en directe.

S'inclouen en l'apartat Annexos, els codis desenvolupats. Totes les figures del procés de biolixiviació són les consensudades en el marc del projecte de recerca Optimització i validació d'un prototip amb base biotecnològica per a la recuperació de metalls valuosos en residus electrònics, Projecte R+D+I competitiu, AGUR (2019-2021).

2. ANTECEDENTS

2.1. TREBALLS PREVIS

El present TFG té com a punt de partida els treballs previs realitzats a la planta de biolixiviació. En concret s'han fet dos TFG diferents.

El treball realitzat per Oscar Arrabal anomenat "*M-Duino. Estudi dels PLC enfocats a la indústria 4.0"* va tenir per objectiu la creació d'un software i d'un hardware per controlar el procés i la planta de biolixiviació.

El hardware que es va utilitzar va ser el següent: Una caixa que inclou un PLC M-Duino 58, el qual controla la planta; una Raspberry, la qual emmagatzema la aplicació o software i una pantalla, la qual mostra l'aplicació continguda a la Raspberry. A part d'això s'incloïen altres aparells com sensors, bombes, font d'alimentació...

Pel que fa al software o aplicació:

- Va utilitzar com a sistema operatiu dins de la Raspberry Ubuntu.
- Va utilitzar Arduino per dissenyar el control de la planta del PLC.
- Va utilitzar Visual Studio per crear la aplicació.
- Va utilitzar Windows Forms dins de Visual Studio.

El treball realitzat per Enrique Flores anomenat "*Monitorització de processos i integració de dades a una base de dades"* va tenir per objectiu l'obtenció d'informes i visualització de dades a través d'internet referents a la planta de biolixiviació.

Per a realitzar-lo, es va mantenir l'M-Duino 58 com a PLC, però, pel que fa al software, es va canviar de Windows Forms a WPF, ja que té millors prestacions de cara a la configuració de bases de dades.

En aquest segon treball es va crear una nova aplicació amb WPF i també es va crear una base de dades amb SQL Server.

2.2. RESUM DE LA PLANTA DE BIOLIXIVIACIÓ

La planta de biolixiviació consisteix en un seguit de processos químics amb l'objectiu de recuperar metalls d'aparells electrònics en desús.

La planta consta de 5 etapes:

- Bioreactor
- Etapa de lixiviació
- Etapa de recuperació del coure
- Dipòsit pulmó número 1
- Dipòsit pulmó número 2.

En cadascuna d'aquestes etapes hi apareixen unes variables i elements controladors amb la següent llegenda:

Taula 1: Variables de control

Variables de Control	Explicació
Lv	Nivell del líquid. (HLv per nivell superior i LLv per nivell inferior)
pH	Mesura pH
R	Redox
C	Mesura del color de la mostra
T	Mesura de la temperatura
IT	Mesura de la intensitat o corrent

Taula 2: Equips

Equips	Explicació
B	Bioreactor
L	Etapa de lixiviació
R	Etapa de recuperació del coure
FT	Dipòsit pulmó 1
ST	Dipòsit pulmó 2

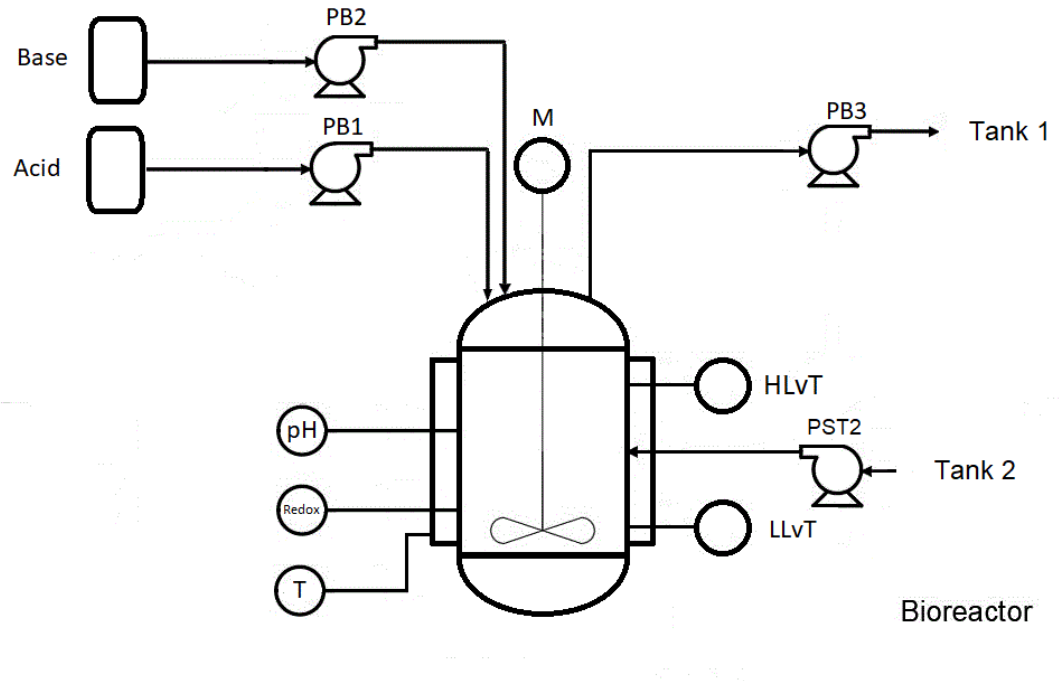
Taula 3: Elements controladors

Variables de Control	Explicació
T	Transmissor de senyal
I	Lector/display de senyal
C	Controlador
Y	Operador
P	Bombes
M	Motor agitador

La planta de biolixiviació, com ja hem explicat, consta de 5 etapes, que seguidament explicarem més detalladament, juntament amb un esquema de cadascuna d'elles.

La primera etapa, la del Bioreactor, s'encarrega, principalment, de mesurar i registrar el Redox del reactor. També disposa d'un injector d'aire, una camisa calefactora que permet controlar la temperatura a l'interior del reactor i un motor agitador que funcionen sempre. En aquesta etapa, a més, també s'han de complir un seguit de requisits, com per exemple que el pH es mesura i ha d'estar entre 1.6 i 1.8 pel correcte funcionament del bioreactor.

L'esquema següent (Il·lustració 1) equival a l'etapa del Bioreactor:

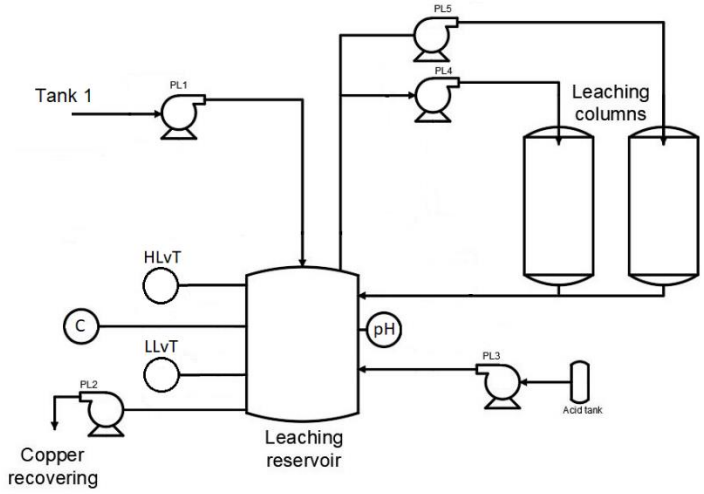


Il·lustració 1: Esquema de l'etapa del Bioreactor

La segona etapa, la de Lixiviació, principalment, té per objectiu controlar, mitjançant un sensor de color, el color actual dins del tanc de lixiviació.

Tal com en l'etapa anterior també s'han de seguir tot un seguit de requisits, entre els quals consta d'un sensor de pH i també controla que aquest es mantingui en el rang entre 1.8 i 1.6.

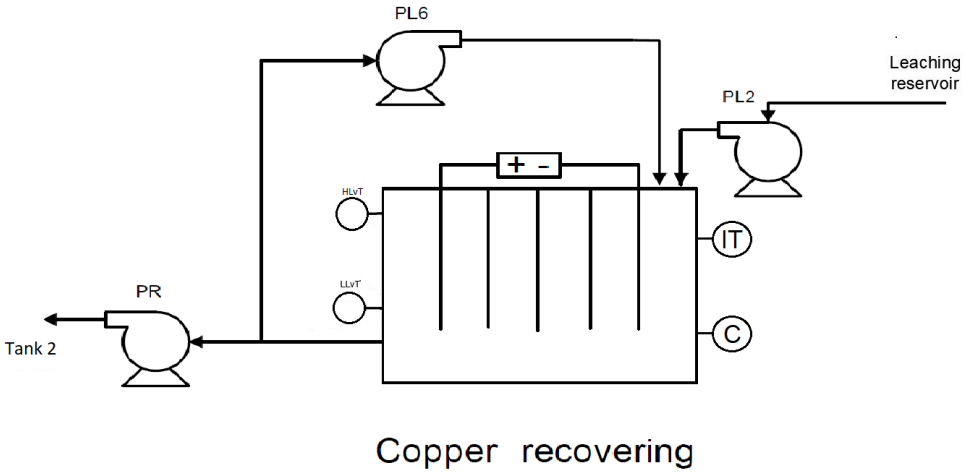
L'esquema de l'etapa de lixiviació (Il·lustració 2) es mostra a continuació:



Il·lustració 2: Esquema de l'etapa de lixiviació

Pel que fa a l'etapa 3, s'anomena etapa de recuperació del coure i el seu objectiu és controlar la reacció de coure mitjançant un sensor de color.

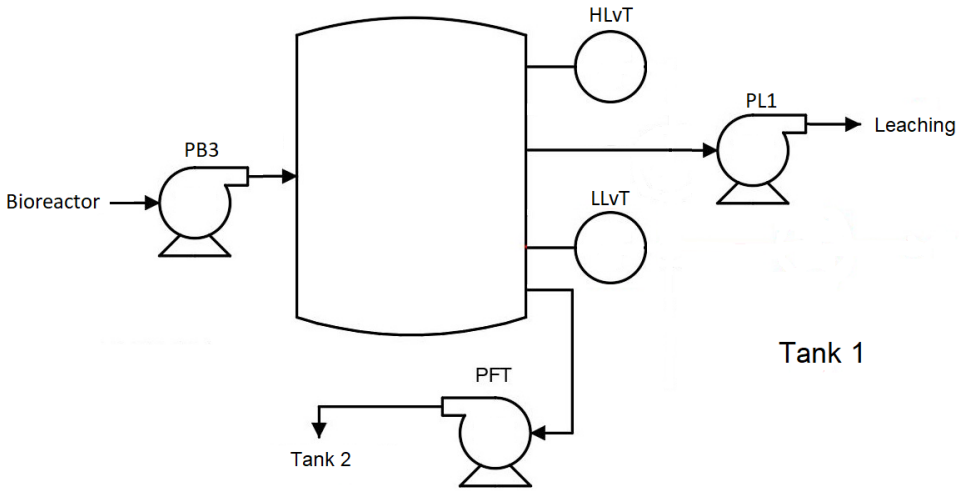
L'esquema d'aquesta etapa (Il·lustració 3) és el següent:



Il·lustració 3: Esquema de l'etapa de Recuperació del Coure

La quarta etapa és el tanc 1 o el pulmó 1. En aquesta etapa s'observa el sensor de nivell alt i si aquest està activat, doncs es procedeix al buidatge del tanc.

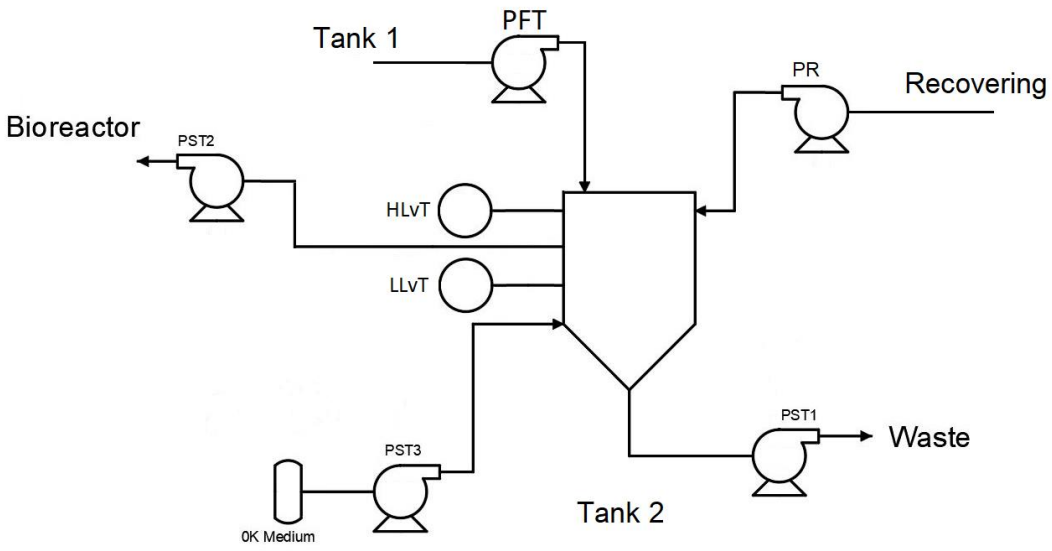
L'esquema relatiu a aquesta etapa (Il·lustració 4) és el següent:



Il·lustració 4: Esquema de l'etapa del Tanc 1

La última etapa és el tanc o pulmó 2. En aquesta etapa es realitza un procés de buidatge constant i, també igual que les altres consta d'un seguit de funcions per al correcte funcionament de la etapa.

L'esquema equivalent a aquesta etapa (Il·lustració 5) és el següent:



Il·lustració 5: Esquema de l'etapa del Tanc 2

2.3. RESUM DE LA MAQUINÀRIA

Per a el control de la planta de biolixiviació, s'utilitza tot un seguit de maquinària (Il·lustració 6):

- **PLC M-Duino**: S'encarrega del control de la planta de biolixiviació i es programa mitjançant l'IDE d'Arduino.
- **Pantalla:** Dispositiu on es visualitza l'aplicació HMI.
- **Raspberry Pi:** Dispositiu que emmagatzema l'aplicació HMI i del qual se'n parlarà més endavant.
- Ventilador, connector d'alimentació, polsador d'emergència, etc.

Tot això queda englobat dins d'una caixa de plàstic (Il·lustració 7) i conforma la maquinària per controlar el procés de biolixiviació realitzada en el TFG d'Oscar Arrabal.

Il·lustració 6: Interior de la caixa de plàstic amb la maquinària

Il·lustració 7: Caixa de plàstic per fora

2.4. MUNTATGE DE RASPBERRY

Abans de començar amb temes de Sistemes operatius i Software, faré una petita introducció sobre el hardware que utilitzarem. Utilitzarem la Raspberry Pi 4 Model B 4 GB.

Què és Raspberry Pi?

La Raspberry Pi és una placa computadora, o el que és el mateix, un mini ordinador. Té un baix cost i és de tamany reduït. Fa les mateixes funcions que faria un PC, per això està composta pels mateixos components: un SoC, CPU, una RAM, ports IN/OUT d'àudio i vídeo, connectivitat de xarxa, ranura SD per emmagatzematge, rellotge, toma per alimentació, connexions per perifèrics de baix nivell…

En el meu cas, disposo d'un kit de muntatge de la Raspberry anomenat: Raspberry Pi 4 Starter Kit de la marca. multicomp PRO.

En aquest kit hi ha inclòs: Raspberry Pi 4 Model 4 GB, microSD card precarregada amb Rasberrian OS, carcassa blanca per protegir la Raspberry Pi, cable d'alimentació de la Raspberry Pi, cable microHDMI a HDMI.

Per poder treballar amb la Raspberry també necessitarem perifèrics com una pantalla, un ratolí i un teclat.

Més instruccions sobre el kit a la **Referència 1**.

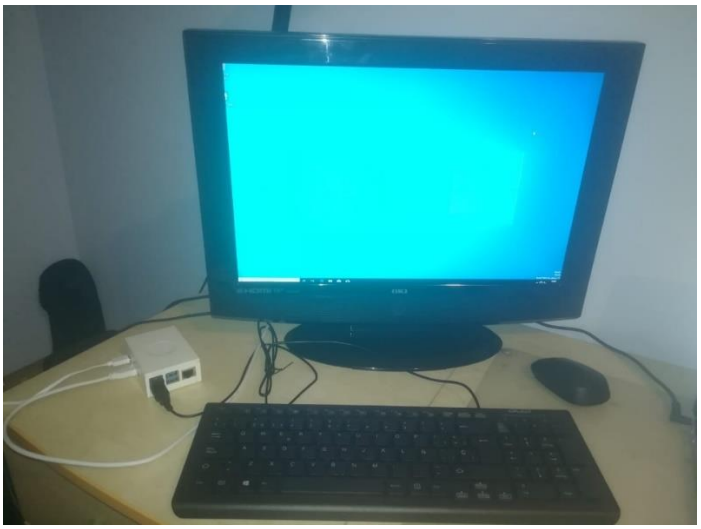
Primer de tot posarem la Raspberry dins de la seva carcassa per protegir-la (Il·lustració 8, Il·lustració 9 i Il·lustració 10):

Il·lustració 8: Raspberry Pi amb la part inferior de la carcassa

Il·lustració 9: Raspberry Pi amb la part superior de la carcassa

Il·lustració 10: Raspberry Pi amb la tapa de la carcassa

Seguidament, cal connectar el ratolí, el teclat, la pantalla i finalment, cal alimentar la Raspberry. El resultat final és el següent (Il·lustració 11):



Il·lustració 11: Equip complet

3. SISTEMA OPERATIU DE LA RASPBERRY

3.1. TRIA DE LA SOLUCIÓ

Un dels objectius d'aquest TFG és la implementació d'un software gràfic fet en un sistema operatiu Windows. Cal recordar que l'actual HMI està realitzat amb un sistema operatiu Linux.

Per a fer-ho, com que la nostra aplicació gràfica estarà continguda en una Raspberry Pi, haurem de bootejar aquesta amb un sistema operatiu Windows.

Primer de tot, cal dir que a dia 10 d'abril de 2021 no hi ha cap versió oficial de Windows IoT (Sistema operatiu Windows dissenyat per l'internet de les coses i especialment per dispositius intel·ligents petits i protegits, el que el fa ideal per implantar-lo a la nostra Raspberry Pi). Per tant, per poder posar el sistema operatiu Windows dins de la nostra Raspberry ho haurem de fer de manera no oficial mitjançant aproximacions de Windows IoT creades per usuaris anònims.

Buscant a internet, cal dir que existeixen diferents versions no oficials de Windows IoT. Aquestes tenen, per això, algunes limitacions. Depenent de la web veiem que alguns sistemes operatius tenen limitació a 1 GB de RAM i en els millors, de fins a 3 i 4 GB de RAM.

Al veure que, encara que no hi hagi versió oficial, és una opció la instal·lació del Windows IoT per la Raspberry, doncs l'he instal·lat en una targeta microSD (per posteriorment insertar-la en la Raspberry).

Primer, disposava d'una targeta microSD de 8 GB, però en veure que tots els tutorials i webs recomanaven molta més capacitat, finalment he utilitzat una microSD de 128 GB.

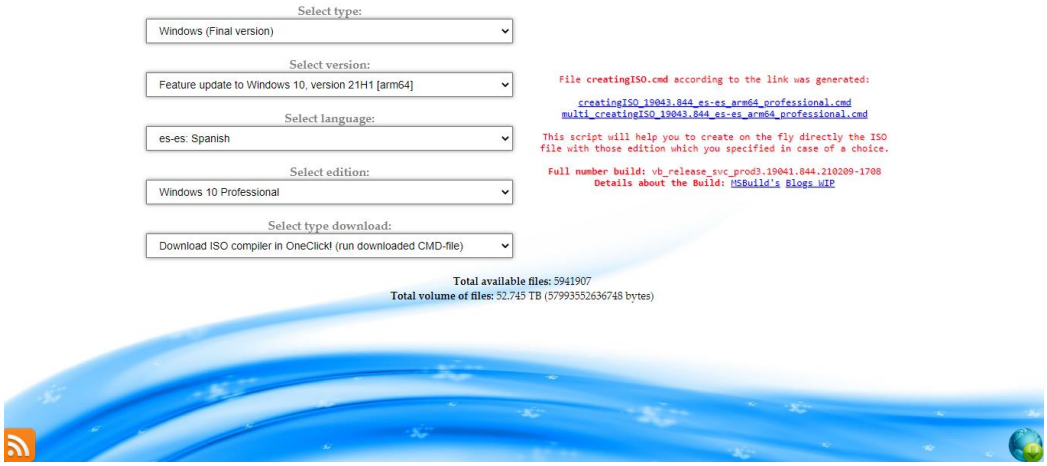
Per instal·lar el Windows IoT a la microSD, necessitem accedir a 2 pàgines web, que mencionarem tot seguit.

3.1.1. WEB DE DESCÀRREGA DE LA IMATGE ISO

La primera web on cal anar és a la **Referència 2**.

En aquesta web (Referència 2) és on descarregarem l'ISO del Windows IoT.

Una imatge ISO és un arxiu informàtic on s'emmagatzema una copia o imatge exacta d'un sistema d'arxius, en el nostre cas un sistema operatiu. Es regeix per l'estàndard ISO 9660, que li dona nom.



Il·lustració 12: Captura de pantalla del web UUP RG

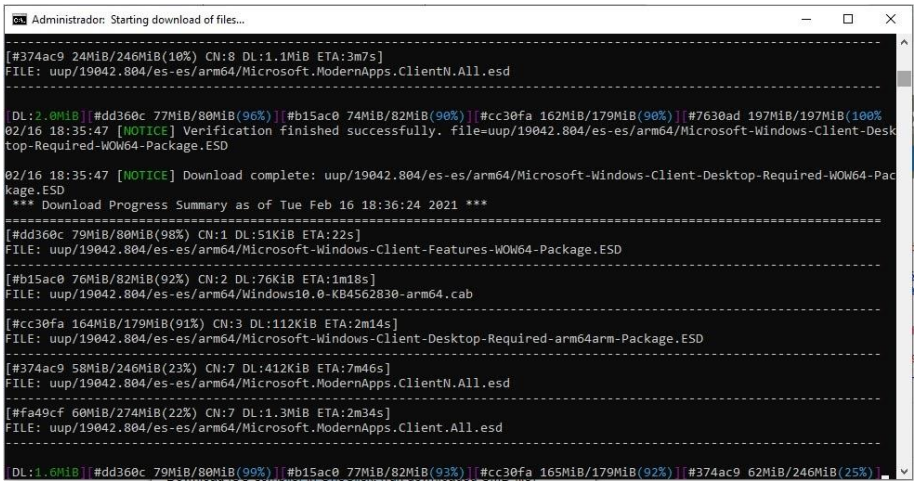
En la web (Referència 2), seleccionem el tipus d'arxiu que volem, la versió (la més actualitzada i amb arm64), la llengua, l'edició i el tipus de descàrrega (jo, tal com he comentat, seleccionaré la imatge ISO).

Un cop seleccionades les opcions, cliquem sobre el link de la part dreta de la pantalla i se'ns descarregarà el següent link en la carpeta de descàrregues:

<*creatingISO_19043.844_es-es_arm64_professional.cmd*>

A continuació, accedim a la carpeta de descàrregues del nostre ordinador, cliquem a sobre de l'arxiu descarregat i seleccionem amb el botó dret la opció "Executa com a administrador".

Seguidament, s'executarà l'arxiu cmd (Il·lustració 13).



Il·lustració 13: Captura de pantalla de l'execució de l'arxiu cmd

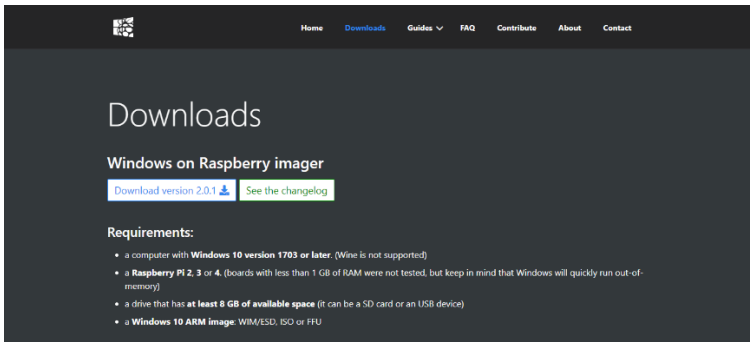
Un cop s'hagi acabat d'executar l'arxiu cmd i l'arxiu ISO estigui compilat, en la mateixa carpet de descàrregues ens apareixerà el següent arxiu, que és la imatge ISO del Windows IoT:

<19041.844.210209-1708.VB_RELEASE_SVC_PROD3_CLIENTPRO_OEMRET_A64FRE_ES-ES.iso>

3.1.2. WEB DE DESCÀRREGA DE L'APLICATIU D'INSTAL·LACIÓ

Un cop tenim la imatge ISO descarregada, cal descarregar el WoR des de la **Referència 3**.

El WoR és un aplicatiu que significa Windows on Raspberry i ens permet gravar una imatge ISO prèviament descarregada (en el meu cas, descarregada del web anterior) a la microSD, per seguidament instal·lar el sistema operatiu a la Raspberry.

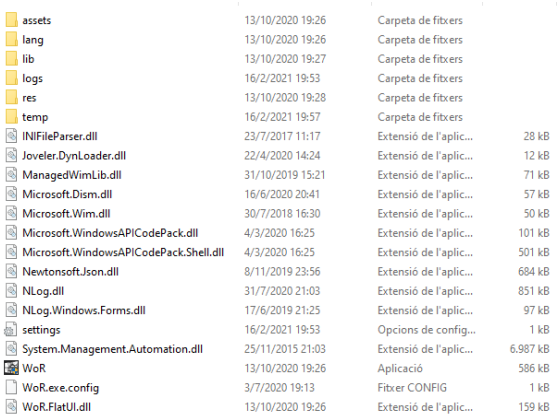


Il·lustració 14: Pantalla de descàrrega de l'aplicatiu WoR

Un cop hem accedit a l'enllaç previ (Referència 3), i, per tant, un cop som a la pantalla de la Il·lustració 14, hem de clicar a sobre de "Download version 2.0.1" i, un cop fet això, ens apareixerà l'arxiu següent a la carpeta de descàrregues del nostre ordinador:

<WoR_Release_2.0.1.zip>

I, seguidament, hem de descomprimir la carpeta zip i obtindrem la mateixa carpeta descomprimida amb el mateix nom i amb diversos arxius a dins (Il·lustració 15).

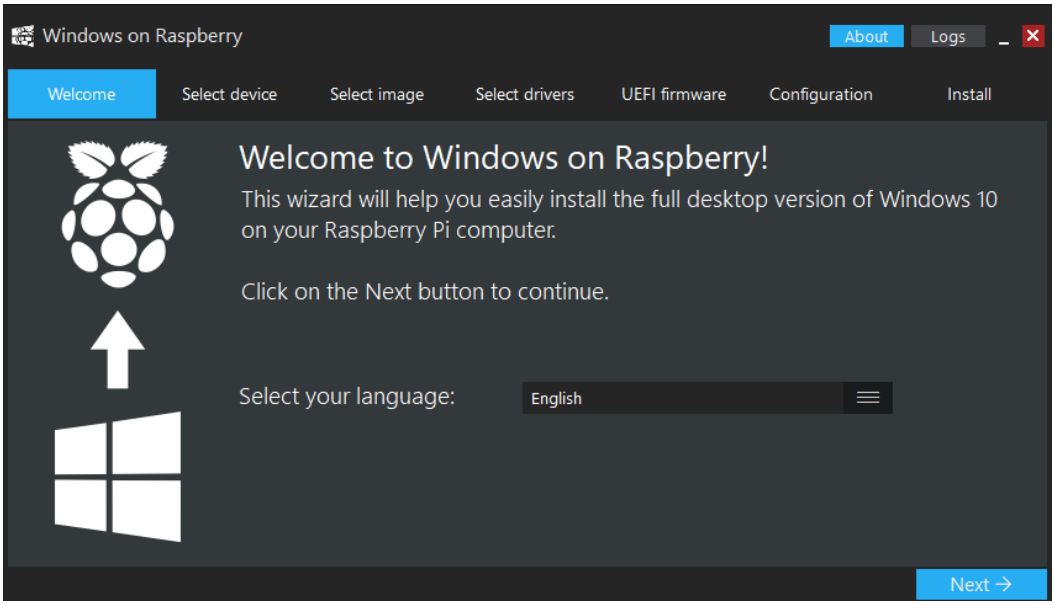


Il·lustració 15: Captura de pantalla de l'interior de la carpeta WoR_Release_2.0.1.zip

Finalment, un cop dins de la carpeta, hem de seleccionar l'aplicació WoR.exe i l'hem de executar. Un cop executada, ens apareixerà una pantalla inicial, a partir de la qual començarà un procés d'instal·lació de la imatge ISO a la targeta microSD.

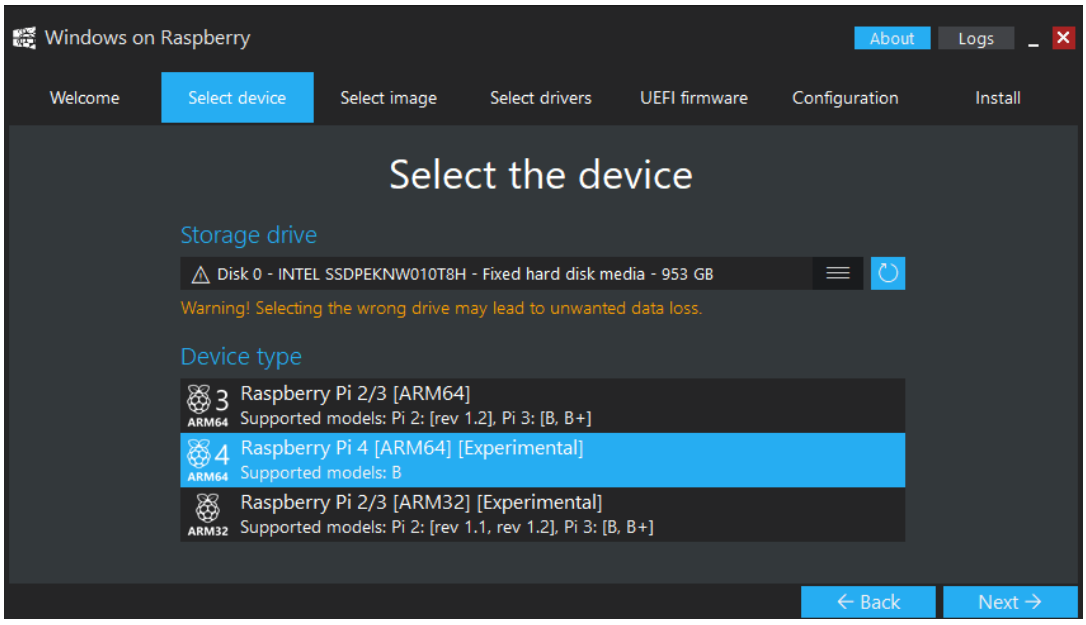
3.2. INSTAL·LACIÓ DEL SISTEMA OPERATIU

Un cop som dins de la pantalla principal de l'aplicatiu WoR (Il·lustració 16), se'ns demanarà la selecció de la llengua pel procés d'instal·lació:



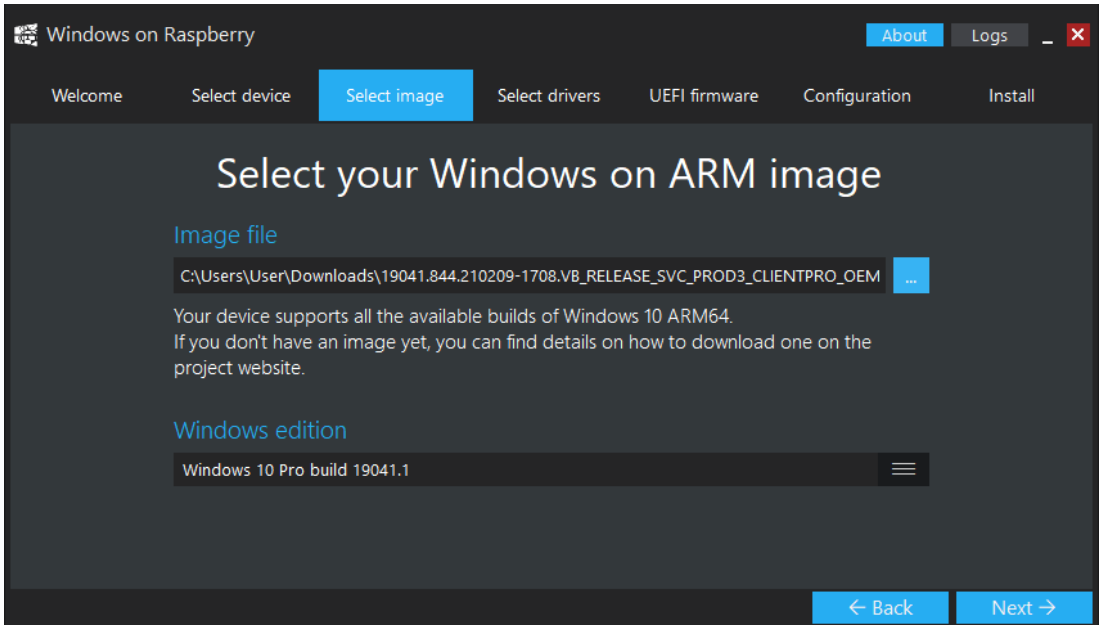
Il·lustració 16: Pantalla Principal del WoR

Seleccionem la llengua i cliquem sobre de "Next". Ara (Il·lustració 17), hem de seleccionar el lloc on volem guardar la nostra imatge ISO. En el nostre cas, la volem guardar a la targeta microSD que prèviament haurem introduït al nostre PC. També ens demanarà el tipus de Raspberry que tenim. En el nostre cas, la Raspberry Pi 4.



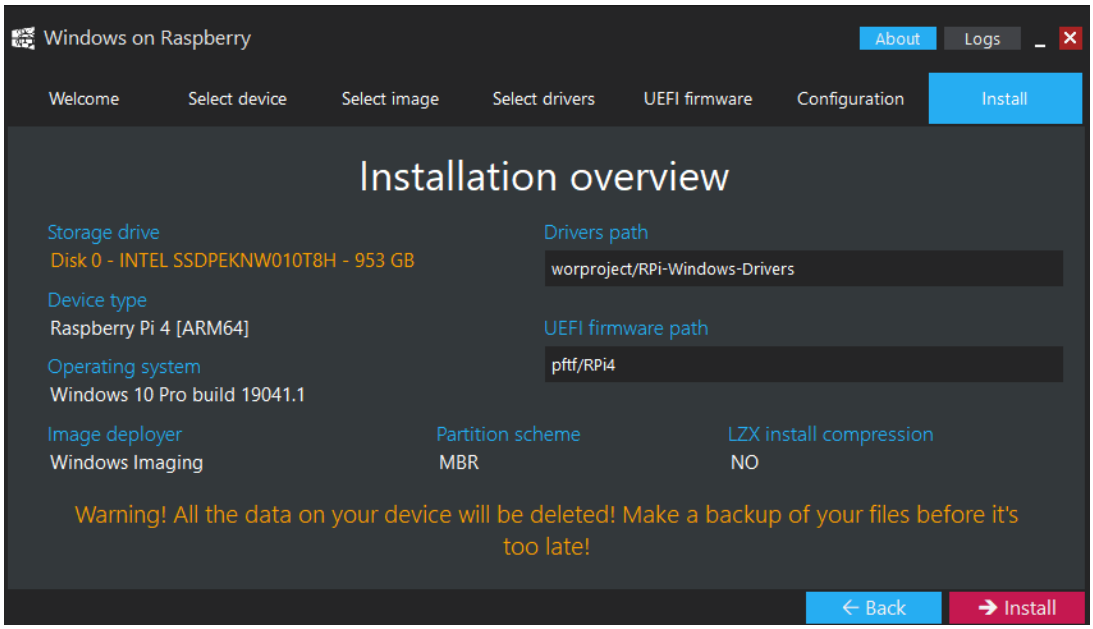
Il·lustració 17: Pantalla de selecció d'ubicació i de tipus de Raspberry

En la següent pantalla (Il·lustració 18) haurem de triar la imatge ISO, que ja hem descarregat prèviament amb la primera pàgina web. La seleccionem i també seleccionem quina versió de Windows té l'ordinador amb el qual estem fent la instal·lació.



Il·lustració 18: Pantalla de selecció d'imatge ISO i de versió de Windows

En les 3 següents pantalles, seleccionem la descàrrega de drivers, firmware... més nous amb les actualitzacions més recents i, un cop fet això, arribarem a la pantalla final (Il·lustració 19).



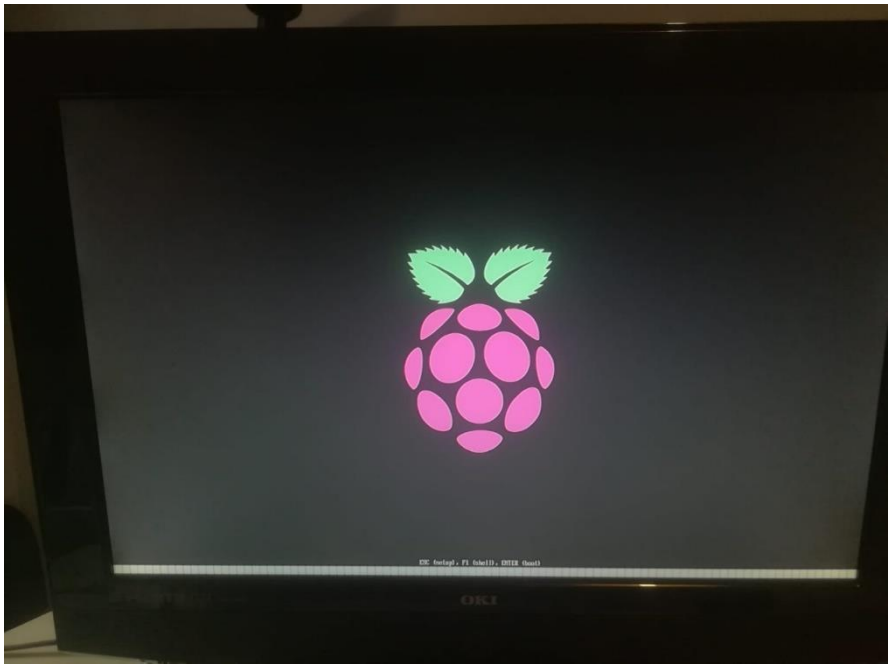
Il·lustració 19: Pantalla final del WoR

En la pantalla final (Il·lustració 19) apareix un resum de tota la informació que hem anat introduint en tots els passos anteriors. Mirem que tots els valors introduïts siguin correctes i cliquem el botó "Install".

Tot seguit començarà la instal·lació de la imatge ISO a la microSD.

Un cop acabada la instal·lació ja podem extreure la targeta microSD del nostre ordinador i ja la podem insertar a la Raspberry Pi.

A continuació, alimentem la Raspberry Pi connectant el cable d'alimentació a la placa i abans de començar la instal·lació del sistema operatiu a la Raspberry Pi ens apareixerà a la pantalla que hem connectat a la Raspberry la pantalla de càrrega/inicialització de la Raspberry (Il·lustració 20).

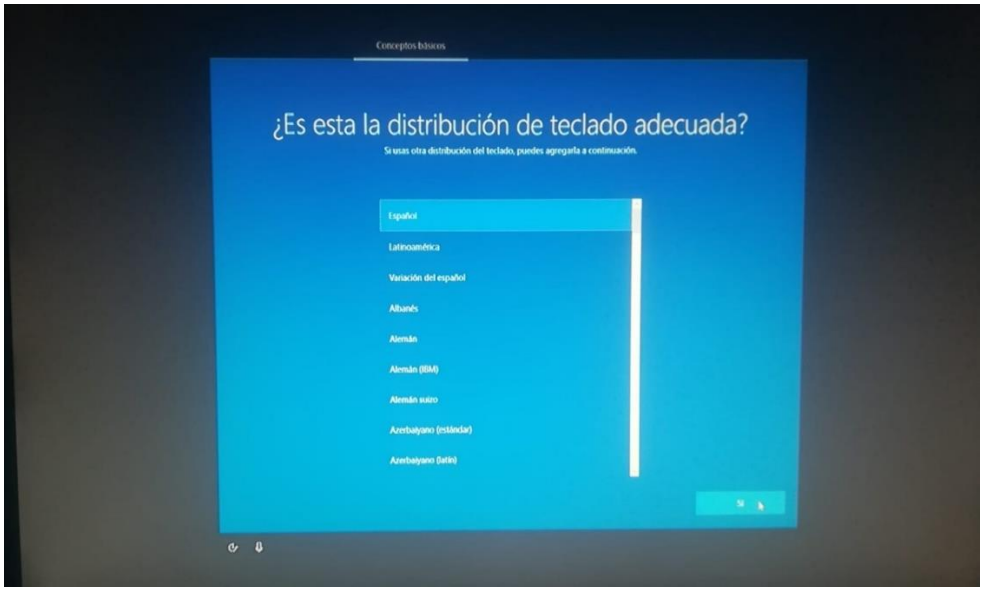


Il·lustració 20: Pantalla d'inicialització de la Raspberry Pi

Un cop s'hagi inicialitzat la Raspberry Pi, començarà el procés d'instal·lació del Windows IoT a la mateixa Raspberry Pi. Per realitzar el procés, ens aniran apareixent pantalles on se'ns demanarà diversa informació que haurem d'anar introduint.

Primer de tot ens apareix una pantalla on ens demana la regió del món on utilitzem el programa, en el nostre cas Espanya.

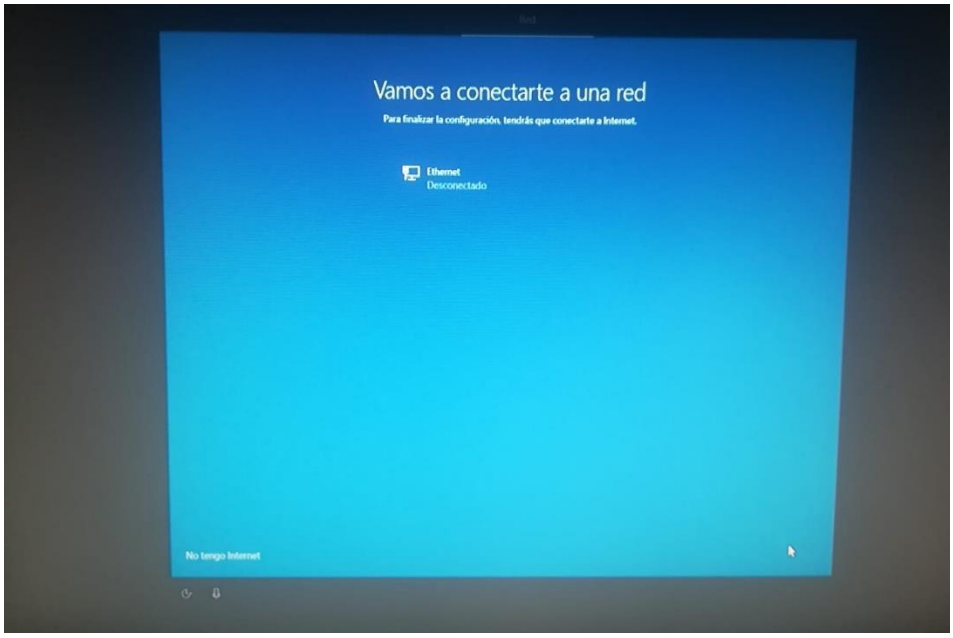
Seguidament (Il·lustració 21), ens demanarà quina distribució de teclat volem. Aquesta característica va molt lligada amb l'anterior ja que la distribució del teclat va molt lligada amb la regió on vivim. Per tant, seleccionem la distribució de teclat espanyola.



Il·lustració 21: Pantalla de selecció de la distribució del teclat

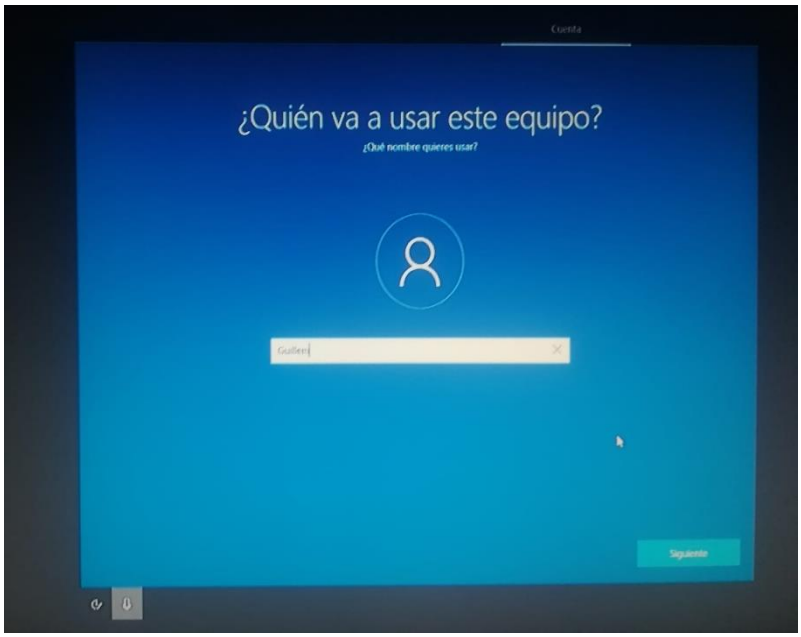
També se'ns demanarà si volem afegir una segona distribució del teclat. En el nostre cas, no ens cal i, per tant, posem que no.

La següent pantalla (Il·lustració 22) va amb relació amb la connectivitat que tindrem a la placa. Concretament ens demana si ens volem connectar a una xarxa mitjançant un cable Ethernet. És igual el què posem ja que un cop instal·lat el sistema operatiu podrem connectar-nos a qualsevol xarxa mitjançant un cable Ethernet.



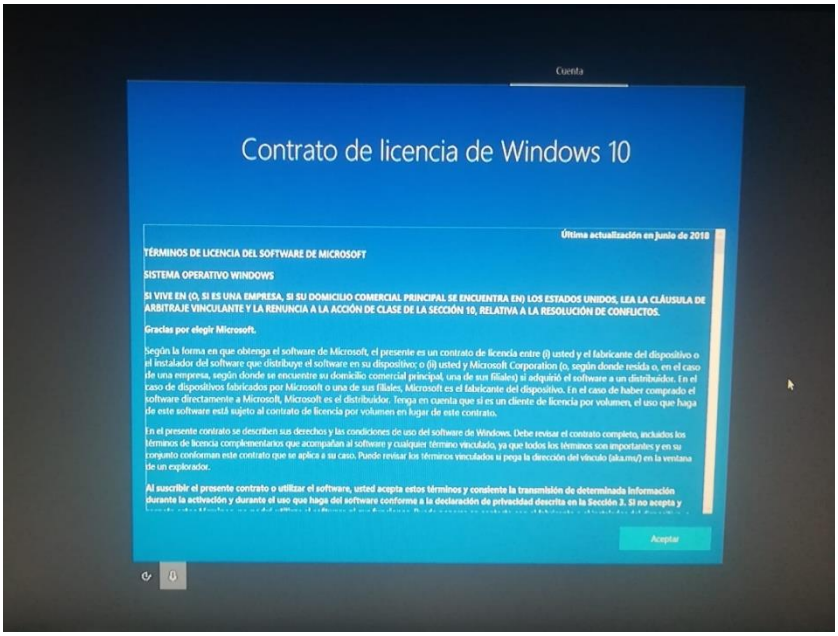
Il·lustració 22: Pantalla de sel·lecció d'una red

A continuació (Il·lustració 23) ens demana posar un nom d'usuari (en el meu cas, he posat el meu nom) i també ens demana una contrasenya per poder entrar dins del sistema (en el meu cas, no n'he posat cap).



Il·lustració 23: Pantalla d'introducció del nom d'usuari i contrasenya

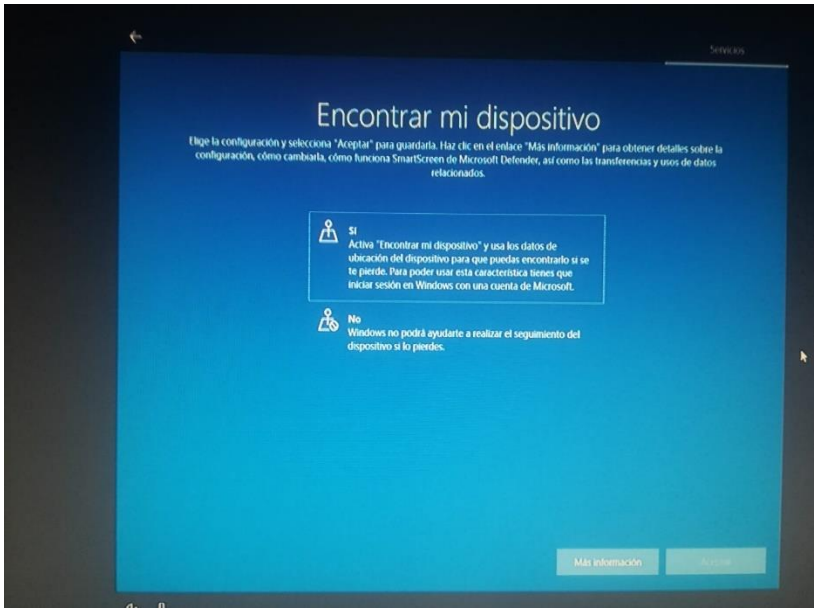
Després de la pantalla de selecció d'usuari, haurem d'acceptar el contracte de llicència de Windows (Il·lustració 24).



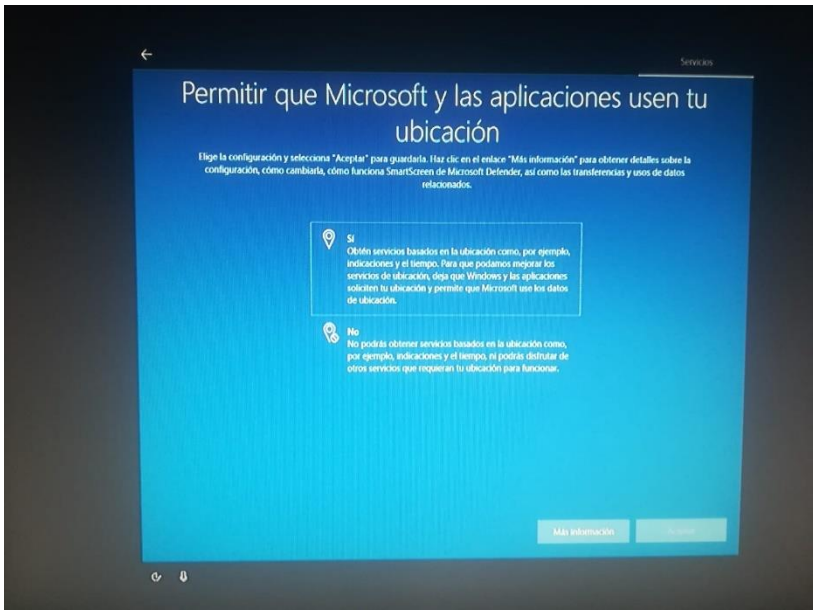
Il·lustració 24: Captura de pantalla del contracte de llicència de Windows

Finalment, i ja com a últims passos del procés d'instal·lació se'ns demana a veure si volem o no tot un seguit de serveis que ofereix Windows (un cop fet la instal·lació, també podrem habilitar/deshabilitar aquests serveis). Entre aquests serveis podem trobar els següents:

- Permetre que Windows conegui la teva ubicació (Il·lustració 25).
- Habilitar la funció "Encontrar mi dispositivo" (Il·lustració 26).
- Habilitar funcions de veu.
- Etc.

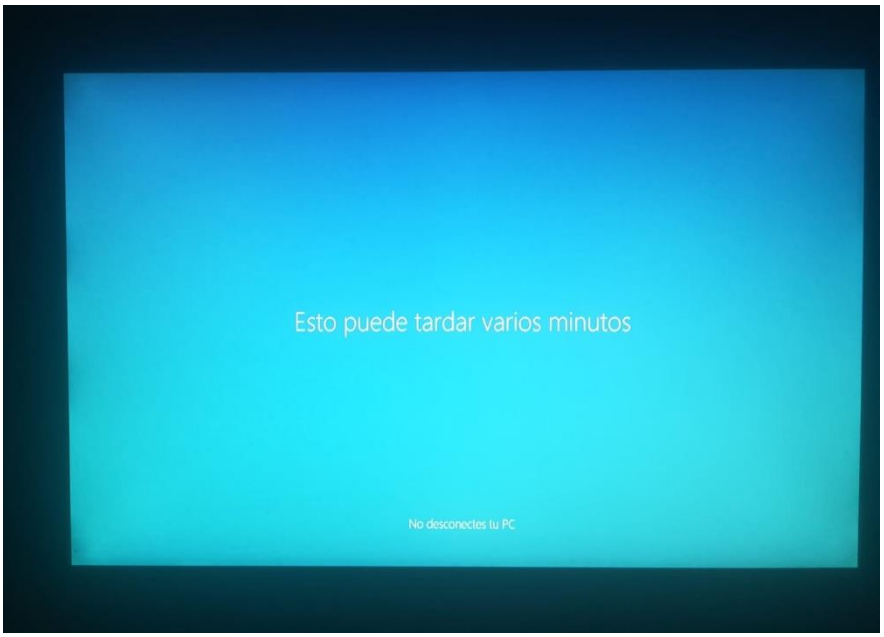


Il·lustració 25: Captura de pantalla de la funció de Windows "Encontrar mi dispositivo"



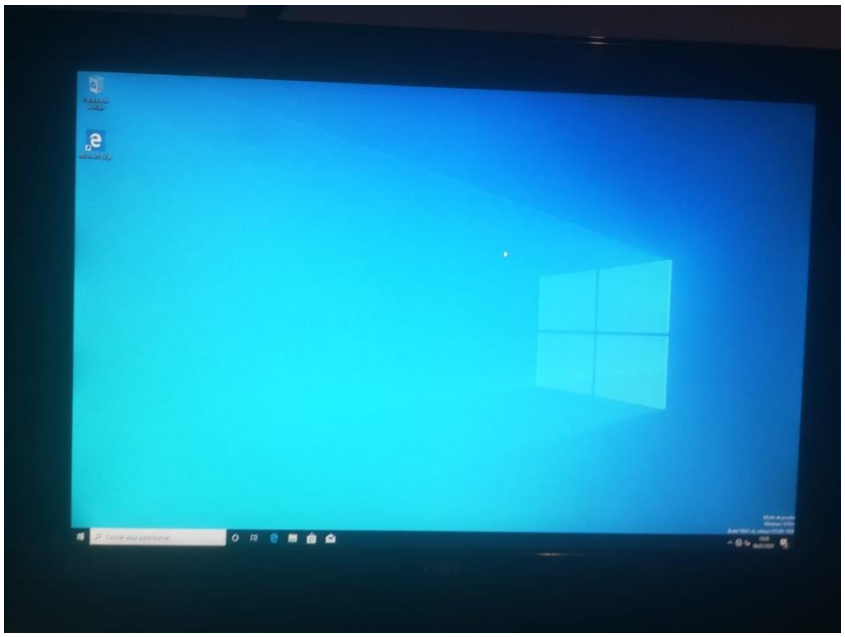
Il·lustració 26: Captura de pantalla de la funció de Windows "Permetre que s'usi la teva ubicació"

Finalment i, un cop acabades les pantalles de pregunta, es procedirà a la instal·lació final (Il·lustració 27), que pot durar varis minuts i que és la instal·lació del sistema operatiu a la Raspberry talment dita.



Il·lustració 27: Captura de pantalla del procés d'instal·lació final

Un cop s'acabi la instal·lació, es reiniciarà la Raspberry i un cop es torni a engegar ja ens apareixerà l'escriptori del Windows (Il·lustració 28) i el sistema operatiu ja estarà correctament instal·lat a la nostra Raspberry.



Il·lustració 28: Captura de pantalla de l'escriptori del Windows 10 IoT

4. VISUAL STUDIO 2019 I SOLUCIÓ ADOPTADA

4.1. INSTAL·LACIÓ DEL VISUAL STUDIO 2019

Per crear l'entorn gràfic del SCADA de la planta de biolixiviació utilitzarem el programa Visual Studio seguint amb el mateix programa utilitzat en els TFGs anteriors.

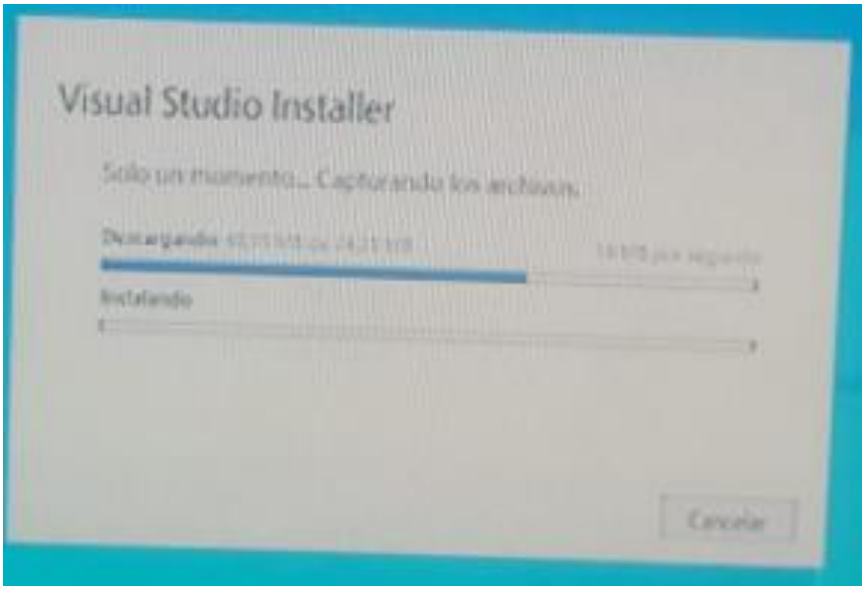
Visual Studio és un entorn de desenvolupament integrat o, més conegut com a IDE, creat per la companyia Microsoft i disponible pels sistemes operatius Windows, macOS i Linux. La versió actual és el Visual Studio 2019.

Un IDE o entorn de desenvolupament integrat és un software pel disseny d'aplicacions que combina o té disponible un conjunt d'eines de desenvolupament comunes en una sola interfície gràfica.

Per a obtenir el Visual Studio 2019, hem de descarregar des d'internet la versió gratuïta del Visual Studio des de la pàgina web oficial que equival a la Referència 4.

Un cop al web (Referència 4), ens baixem la versió gratuïta que s'anomena Visual Studio Community i se'ns descarregarà un paquet instal·lador anomenat: Visual Studio Installer.

Un cop tenim el paquet instal·lador a la carpeta de descàrregues del nostre PC, l'executem (Il·lustració 29).

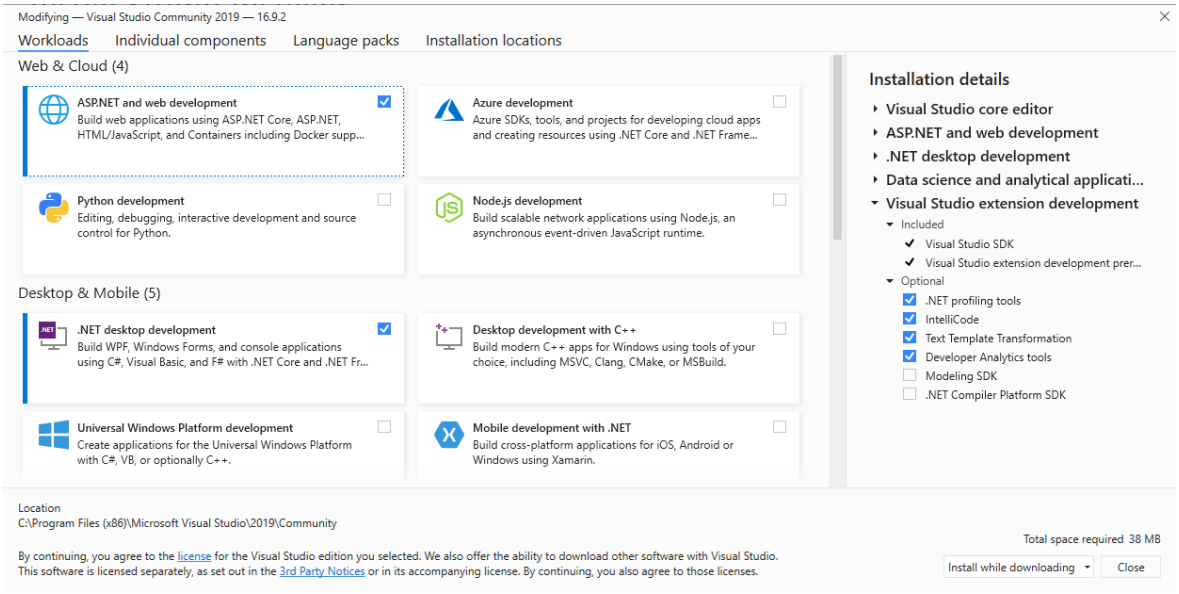


Il·lustració 29: Captura de pantalla de l'execució del paquet instal·lador de Visual Studio

Un cop executat el paquet instal·lador, ens apareixerà una pantalla on haurem d'escollir els paquets que volem descarregar per treballar amb el Visual Studio (Il·lustració 30).

Hem de tenir present que encara que no escollim un paquet i finalment el necessitem doncs el paquet instal·lador sempre estarà disponible en la nostra llista d'aplicacions del PC. Per tant, sempre tindrem la opció d'instal·lar paquets nous.

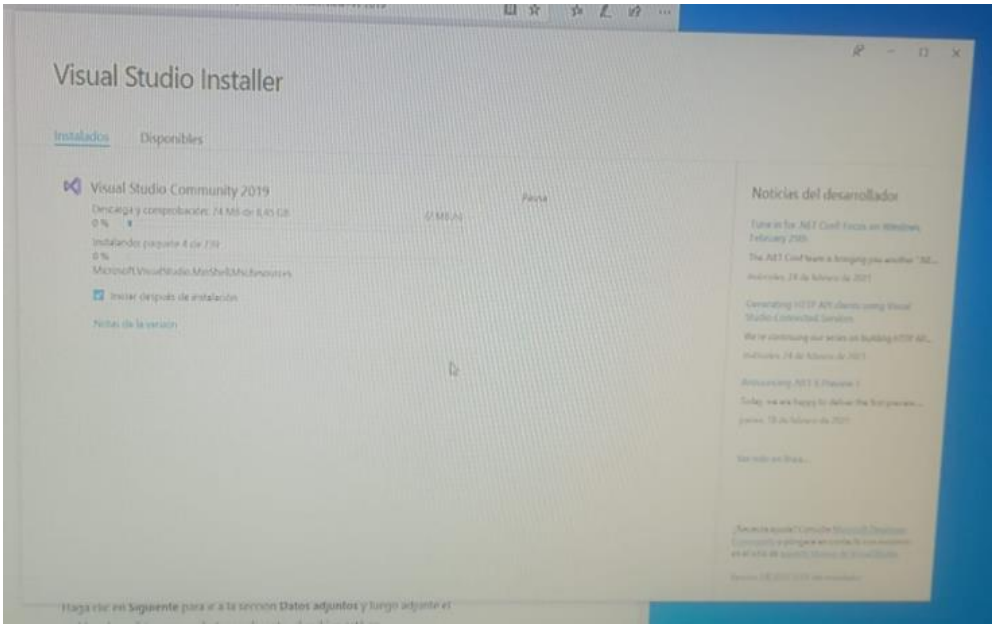
Dins dels paquets a instal·lar tenim des de paquets de llenguatge per poder treballar en el nostre idioma fins a paquets de llenguatge de programació o paquets per desenvolupar aplicacions de Gaming. Entre ells podem trobar: Python development, Universal Windows development, Game development with Unity...



Il·lustració 30: Captura de pantalla de selecció dels paquets a instal·lar a Visual Studio

A la dreta de la imatge ens aniran apareixent els paquets que hem seleccionat i els diversos subpaquets o fitxers que estan inclosos en un paquet seleccionat. D'aquest manera podrem gestionar exactament quins paquets i fitxers/subpaquets volem descarregar i quins no.

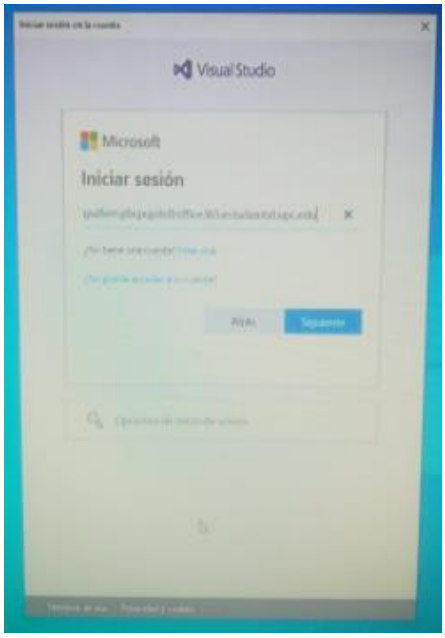
Escollim els paquets necessaris i procedim a realitzar la instal·lació (Il·lustració 31).



Il·lustració 31: Captura de pantalla de la instal·lació de Visual Studio

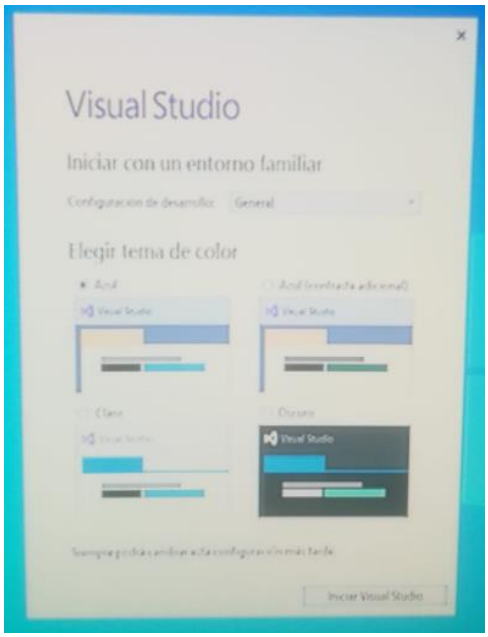
Un cop instal·lats els paquets i el programa principal de Visual Studio es farà un reinici de l'ordinador. Un cop reiniciat, al engegar el programa ens apareixeran un seguit de pantalles que haurem d'anar completant per finalitzar la instal·lació.

Primer ens demanarà si volem iniciar sessió en un compte Microsoft (Il·lustració 32), cosa necessària ja que la versió gratuïta sense compte de Microsoft de Visual Studio només dura un mes i en canvi si et registres amb el teu correu, la versió gratuïta és il·limitada.



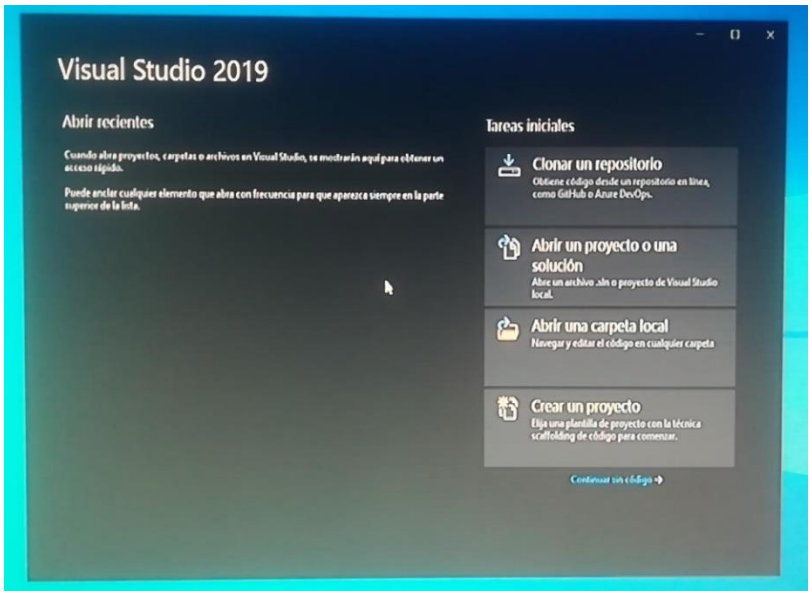
Il·lustració 32: Pantalla d'Inici de sessió al compte de Microsoft

Seguidament ens demana com volem que sigui el tema de color del Visual Studio (en el meu cas he triat el fons de color negre) (Il·lustració 33).



Il·lustració 33: Pantalla de tria del tema de color de l'IDE

Un cop fet això, ja accedim a la pantalla principal de Visual Studio (Il·lustració 34) on accedirem sempre en clicar la icona de la aplicació. En aquesta pantalla se'ns demana si volem crear un nou projecte, obrir-ne un, etc.

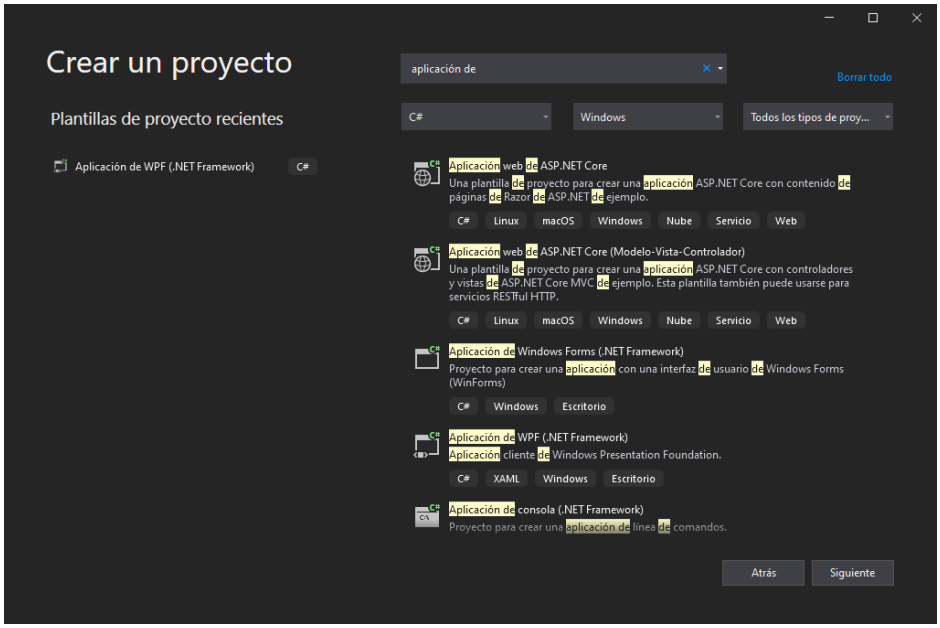


Il·lustració 34: Pantalla principal del Visual Studio

Escollim l'opció de crear un projecte nou i a partir d'aquí comencem l'edició de l'entorn gràfic de la planta de biolixiviació.

4.2. CREACIÓ D'UN PROJECTE AMB VISUAL STUDIO

Un cop hem clicat l'opció de crear un nou projecte, ens apareixerà una pantalla on podrem escollir el tipus de projecte que volem crear (Il·lustració 35). Podem filtrar els tipus de projecte per llenguatge de programació, per sistema operatiu, etc.



Il·lustració 35: Captura de pantalla de creació d'un nou projecte

L'aplicació HMI anteriorment creada estava realitzada amb la interfície d'usuari Windows Forms i la nova, creada per mi, es crearà amb la interfície d'usuari WPF (Windows Presentation Foundation). Aquest canvi es deu a diversos motius, els quals exposo a continuació:

- WPF és el framework més actual i, per tant, tant gràficament com a mode de programació ofereix unes solucions més adequades a l'actualitat.

- WPF et permet crear l'entorn gràfic HMI d'una manera interactiva, ja que disposa de 2 pantalles diferents per l'edició de la aplicació. Una per a la part gràfica en què el mateix usuari veu en directe tot el que està editant i una per la part de programació.

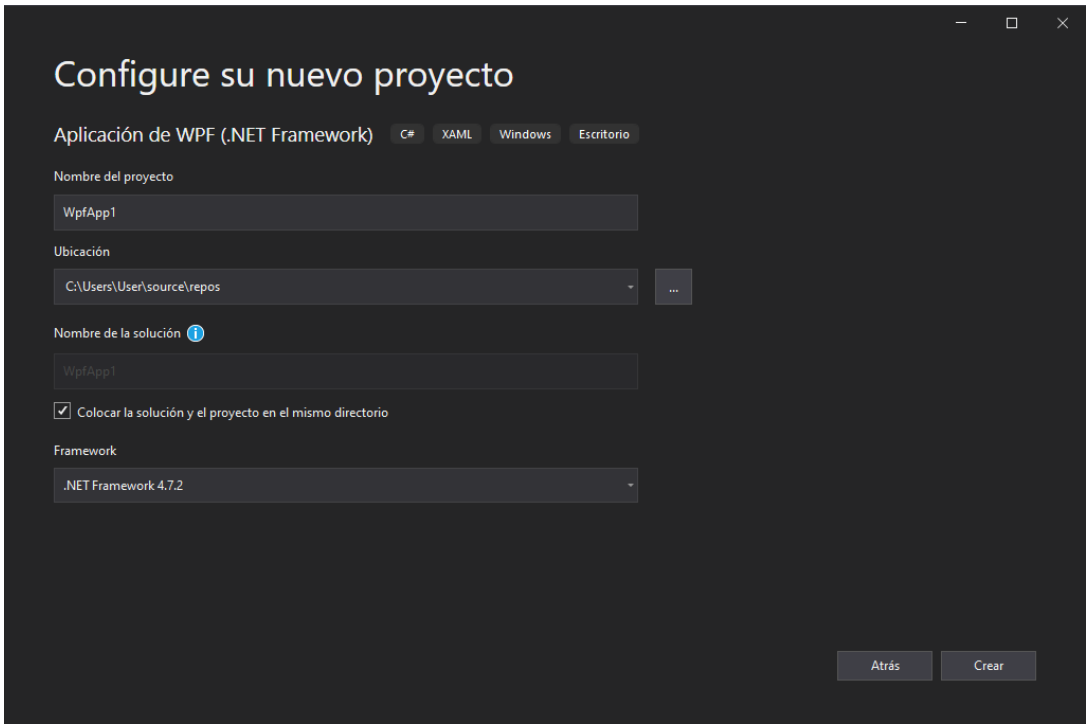
- Una última avantatge respecte del Windows Forms és que WPF combina diferents plataformes de desenvolupament per a l'edició de l'aplicació i no només es limita a les plataformes/opcions de Windows.

Per tant, per tot l'exposat anteriorment seleccionarem un projecte d'aplicació de WPF (.NET Framework). Tal com ja hem introduït abans el llenguatge de programació de WPF és C# i el llenguatge gràfic és XAML.

Tot seguit, ens demanarà que posem un nom al projecte, la ubicació on el volem guardar, el nom de la solució i la versió del framework (Il·lustració 36).

.NET Framework és un programa que té com a funció el desenvolupament de software sota un mateix llenguatge de codi (.NET).

Una solució és un contenidor de projectes de Visual Studio. És a dir, que podem tenir més d'un projecte dins d'una mateixa solució. Com que només ens interessa un projecte i prou, seleccionarem la opció "Colocar el nombre de la solución y el proyecto en el mismo directorio".

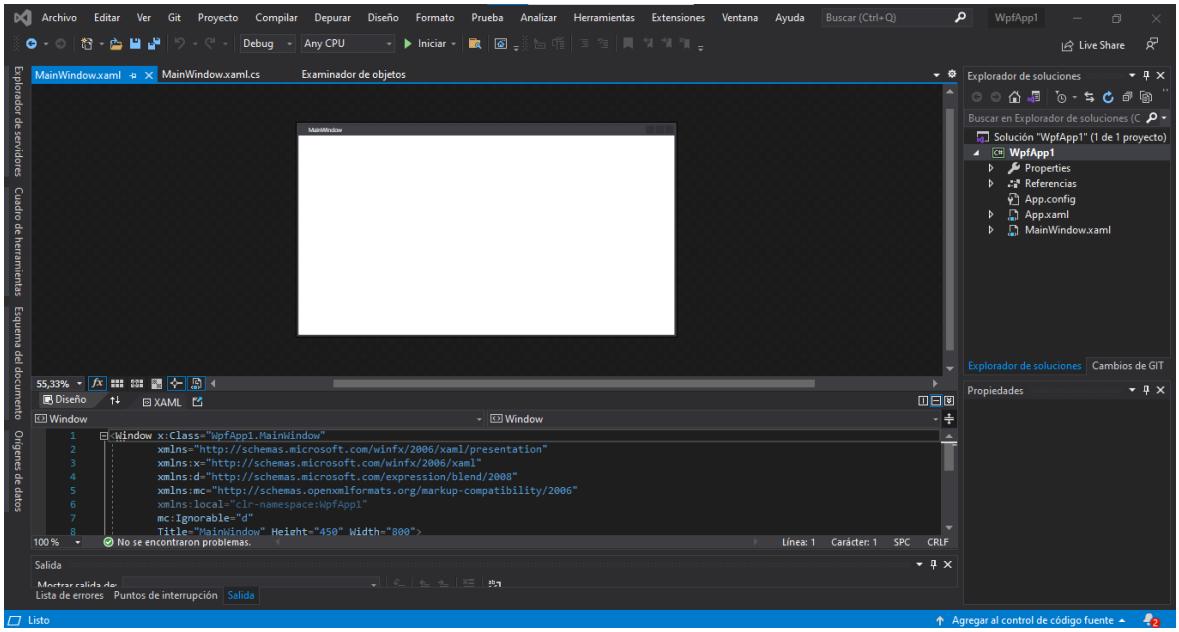


Il·lustració 36: Captura de pantalla de la configuració del nou projecte

El nostre projecte s'anomenarà "Planta_Biolixiviació".

Un cop haguem introduït el nom del projecte, la seva ubicació i haguem escollit el framework més actual, cliquem sobre de "Crear" i es crearà el nou projecte.

Un cop creat el nou projecte, ens apareixerà la pantalla principal del projecte per poder començar a editar-lo (Il·lustració 37).



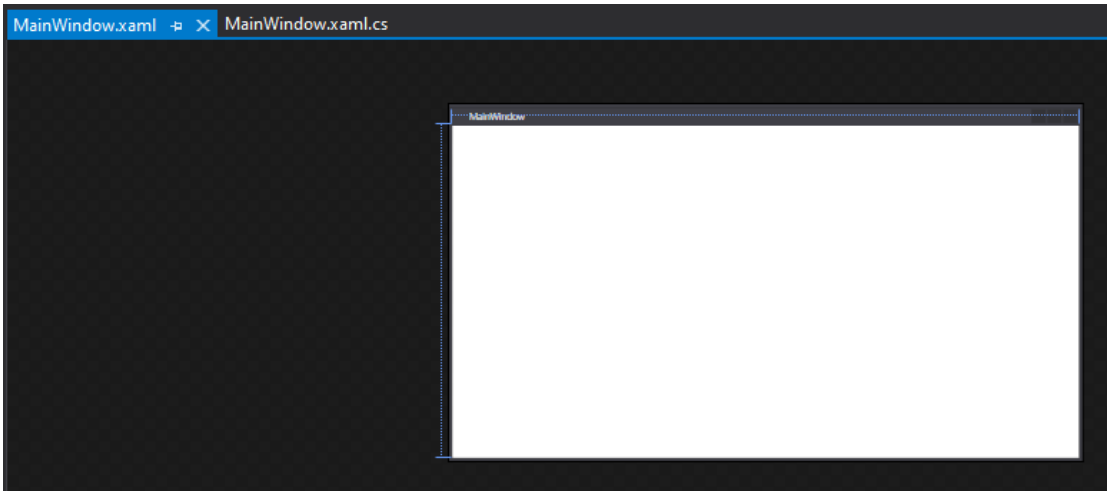
Il·lustració 37: Pantalla principal del nou projecte

4.3. PETITA INTRODUCCIÓ A VISUAL STUDIO

A continuació, explicarem breument com és l'entorn de treball de Visual Studio concretament per l'edició d'un projecte WPF.

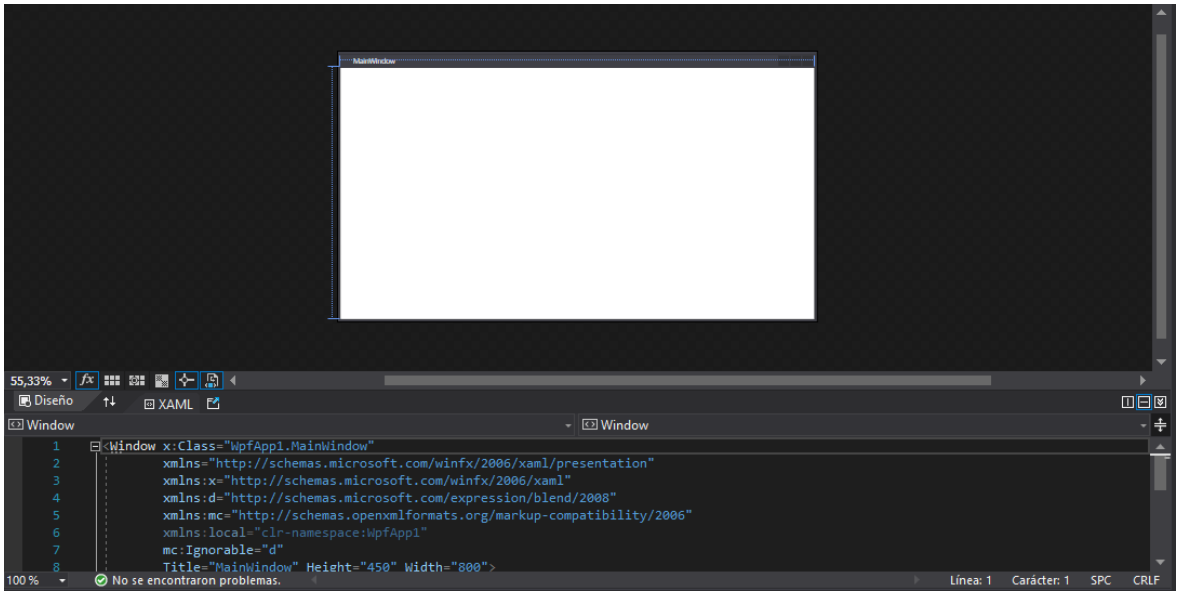
Primer de tot, com ja hem comentat, cal tenir en compte que tenim dos parts ben diferenciades: la part gràfica i la part de programació. La part gràfica, tal com indica el seu nom, és on editarem tota la part de botons, imatges, i veurem com va quedant estèticament la nostra aplicació. En canvi la part de programació és on hi haurà la connexió sèrie entre M-Duino i la nostra app i la part explícitament de programació.

Les dues parts anteriorment comentades tenen cadascuna d'elles una pestanya, que al clicar-la entrem a la pantalla d'edició d'una o l'altra part. La pestanya de la part gràfica s'anomena "MainWindow.xaml" i la de la part de programació "MainWindow.xaml.cs" (Il·lustració 38).



Il·lustració 38: Captura de pantalla de les pestanyes de Visual Studio

Dins de la part gràfica (MainWindow.xaml), tenim dues subparts (Il·lustració 39): La part de codi i la part talment gràfica. Per tant, tenim dues opcions per programar la part gràfica. O bé mitjançant codi, és a dir escrivint codi en llenguatge xaml; o bé, introduint directament botons, imatges, etc. dins de la pantalla que ens apareix i que simula el que serà la app un cop creada.



Il·lustració 39: Captura de pantalla de les dues subparts per editar la part gràfica

La part de programació (MainWindow.xaml.cs) està en llenguatge C# i també disposa de dues subparts: una part d'introducció de llibreries usant la comanda "using" i la part de programació talment dita (Il·lustració 40).

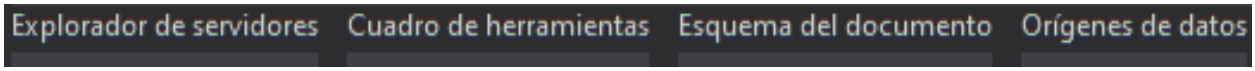
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace WpfApp1
{
    /// <summary>
    /// Lógica de interacción para MainWindow.xaml
    /// </summary>
    2 referencias
    public partial class MainWindow : Window
    {
        0 referencias
        public MainWindow()
        {
            InitializeComponent();
        }
    }
}
```

Il·lustració 40: Captura de pantalla de la part de programació

Parlant de l'entorn del Visual Studio, podem dir que al centre de la pantalla hi ha les pantalles comentades anteriorment.

A la part esquerra hi tenim una barra d'opcions (Il·lustració 41) on hi ha diverses pestanyes:

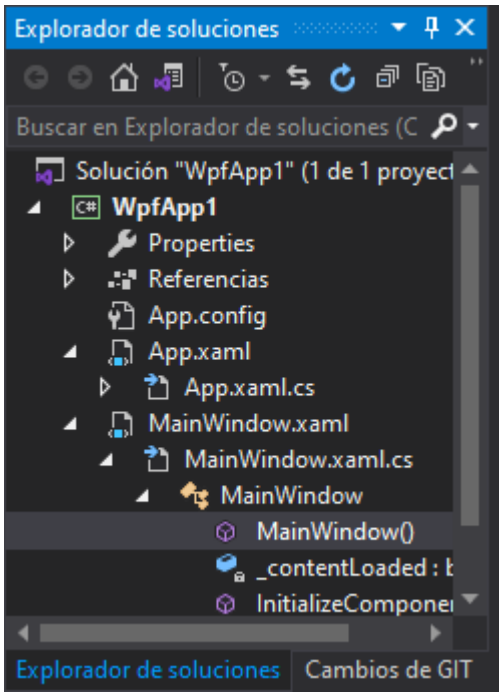
- **Explorador de servidors.** Serveix per connectar-se a servidors i bases de dades.
- **Quadre d'eines.** Hi ha tot un llistat d'eines que podem utilitzar per l'edició gràfica de l'aplicació HMI. Algunes d'aquestes eines són: Botons, Interruptors, LEDs, Gràfics, Quadres de Text, etc.
- **Esquema del document.** Esquema-resum de totes les eines introduïdes en la nostra app.
- **Origen de dades.** Serveix per agregar nous orígens de dades, que poden ser bases de dades, referències de serveis o objectes.



Il·lustració 41: Captura de pantalla de la barra d'opcions

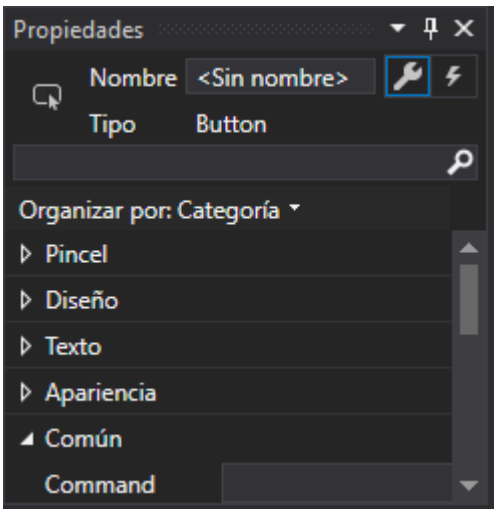
A la part dreta de la pantalla hi ha dues finestres: Una s'anomena explorador de solucions (Il·lustració 42) i l'altra propietats (Il·lustració 43).

La finestra anomenada "Explorador de Solucions" és un esquema desplegable de tots els arxius que componen la solució del nostre projecte. És a dir, tots els arxius que formen la nostra aplicació.



Il·lustració 42: Captura de pantalla de l'explorador de solucions

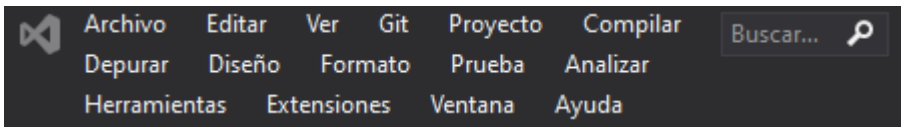
La finestra anomenada "Propietats" serveix per modificar les propietats de qualsevol element que seleccionem en la pantalla central, ja sigui un element gràfic, un mètode, un gràfic, etc.



Il·lustració 43: Captura de pantalla de la finestra "Propietats"

Per últim cal fer menció a la barra superior o menú (Il·lustració 44). En aquesta barra hi ha diverses pestanyes a seleccionar, les quals ens permeten escollir diverses funcionalitats de l'entorn de Visual Studio. Les opcions que hi ha són les següents: Arxiu, Editar, Veure, Git, Projecte, Compilar, Depurar, Disseny, Format, Prova, Analitzar, Eines, Extensions, Finestra i Ajuda.

En cadascuna de les opcions podem fer diferents accions com personalitzar els menús de l'IDE (Extensions), afegir finestres i pàgines de WPF (Projecte), connectar amb bases de dades (Eines) o compilar/recompilar la solució (Compilar).



Il·lustració 44: Captura de pantalla del menú de l'IDE de Visual Studio

5. PROCÉS DE CRACIÓ DE L'HMI

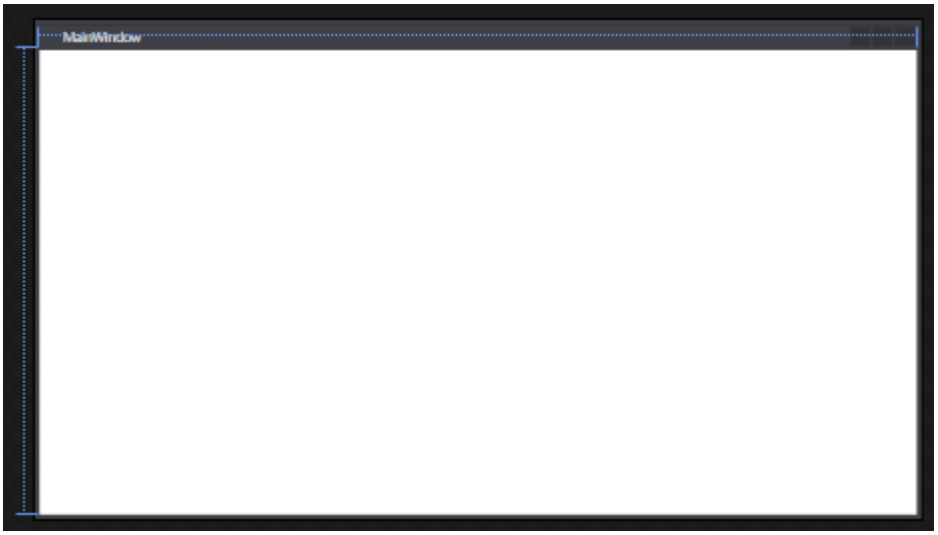
Com ja he comentat anteriorment, la aplicació que crearé serà en WPF i no pas en WinForms com estava en treballs anteriors. Per tant, partiré de l'aplicació creada pels treballs anteriors però hauré d'anar modificant moltes coses, ja que la programació canvia d'un entorn a l'altre. D'aquesta manera aniré explicant pas a pas, cadascuna de les etapes de creació de l'HMI.

5.1. ASPECTE I DISSENY GRÀFIC

Per començar a dissenyar la aplicació, començarem pels aspectes gràfics.

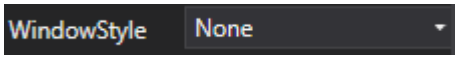
5.1.1. ASPECTES GRÀFICS GENERALS

El primer que farem és canviar l'estil de la finestra de l'aplicació. Un cop creem l'aplicació ens surt per defecte la Single Border Window (Il·lustració 45) i nosaltres volem una finestra buida.



Il·lustració 45: Finestra amb l'opció Single Border Window seleccionada

Per a canviar de Single Border Window a None, és a dir, a pantalla buida, hem d'accedir a les propietats de "Window" o finestra i anar a la secció "Aparence" o Aparença. Un cop allà seleccionem el desplegable on diu Window Style i seleccionem l'opció None (Il·lustració 46).



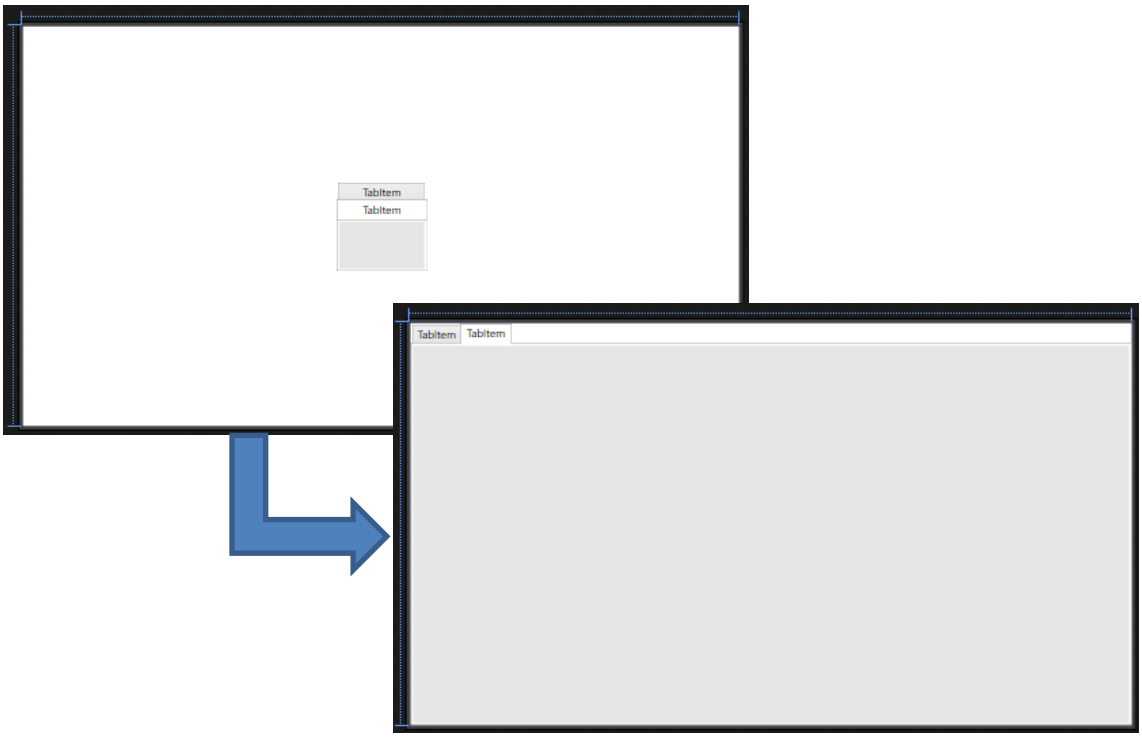
Il·lustració 46: Selecció de l'opció d'estil de finestra

A continuació, crearem un menú superior en format de pestanyes per a poder seleccionar les diferents pàgines que tindrà l'aplicació. Això ho farem a partir de l'eina anomenada "TabControl" (Il·lustració 47), que en anglès significa control de pestanyes.

Il·lustració 47: Símbol de l'eina "TabControl"

Per a introduir el control de pestanyes, anem al Quadre d'Eines, a la part dreta de l'entorn de Visual Studio. Un cop allà, busquem al cercador "TabControl" i un cop ens apareix el símbol, l'arrosseguem cap a la pantalla central.

Un cop hem arrossegat el "TabControl" dins de la pantalla central, posem un nom al Tab Control (nosaltres l'anomenem "TabControlPrincipal") i l'ampliem perquè ocupi tota la pantalla (Il·lustració 48), ja que l'eina a part del menú superior de control de pestanyes també inclou la pantalla per cada pestanya.

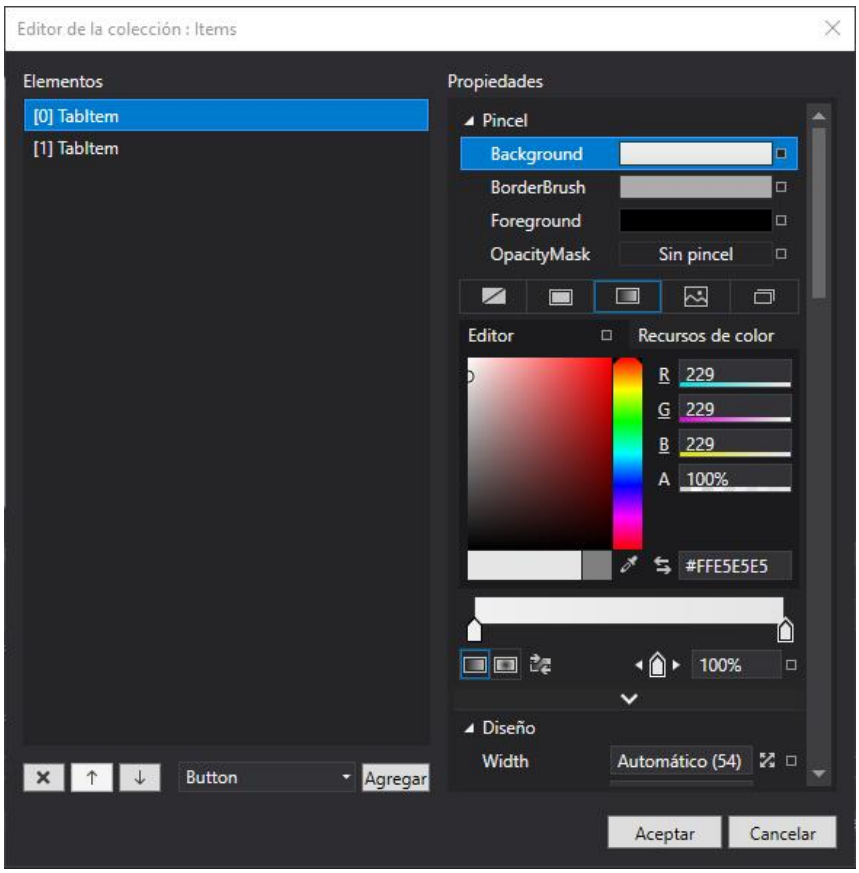


Il·lustració 48: Captura de pantalla amb el TabControl inicial i l'ampliat

El TabControl, per defecte es crea amb 2 TabItems, és a dir, 2 pestanyes. Per afegir més pestanyes, hem d'anar a la finestra de Propietats del TabControl. Un cop allà anem a l'apartat anomenat "Comú", cliquem sobre la pestanya Items i ens apareixerà una nova finestra emergent (Il·lustració 49).

En aquesta finestra emergent (Il·lustració 49) és on podrem editar cadascuna de les pestanyes i també on podem afegir-ne de noves o eliminar les ja creades.

En el nostre cas, volem 7 pestanyes en total: 5 per cadascuna de les etapes del procés de biolixiviació, 1 pantalla principal i 1 pantalla suplementària de visió de gràfics i dades.



Il·lustració 49: Finestra emergent sobre la edició dels items del TabControl

Un cop dins de la finestra emergent (Il·lustració 49), per crear noves TabItems o pestanyes, hem de seleccionar la opció de TabItem en el desplegable de baix a la dreta i seguidament clicar el botó d'"Agregar". Tal com hem comentat, nosaltres ho farem 5 cops, ja que per defecte ja hi ha 2 pestanyes creades i nosaltres en volem 7.

Dins de la finestra emergent (Il·lustració 49) i un cop seleccionat un dels TabItems prèviament creats, podem observar com a la part dreta de la finestra apareix un menú de propietats. Aquest menú és l'equivalent a la finestra de propietats de l'IDE del Visual però aplicat en cadascun dels Items del TabControl.

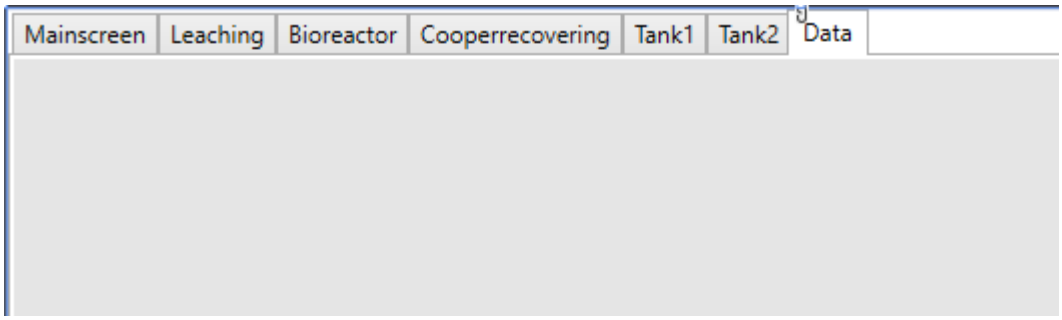
Per tant, per poder canviar o afegir propietats per cadascun dels Items haurem de fer servir el menú esmentat anteriorment. Però, cal dir que també hi ha una altra opció per a fer-ho, que és escriure directament amb codi XAML.

El primer que farem per cadascuna de les opcions és donar-les-hi un nom i donar-les-hi un Header name (el nom que apareixerà a la pestanya). Nosaltres posem per cada TabItem el mateix nom i Header name (Il·lustració 50).

Els anomenem Main_screen, Leaching, Bioreactor, Cooper_recovering, Tank 1, Tank 2 i Data (Il·lustració 51).

```
<TabItem Header="Main_screen" Name="Main_screen">
    <Grid Background="#FFE5E5E5"/>
</TabItem>
<TabItem Header="Leaching" Name="Leaching">
    <Grid Background="#FFE5E5E5"/>
</TabItem>
```

Il·lustració 50: Fragment del codi XAML referent a dos TabItems



Il·lustració 51: Captura de pantalla de l'aspecte gràfic dels Headers dels TabItems

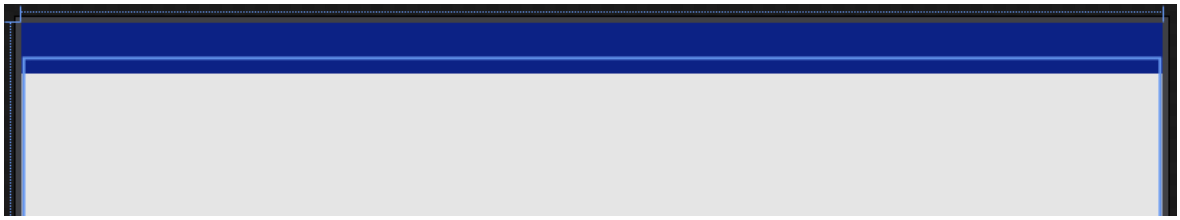
El TabItem ja està creat però per millorar l'aspecte gràfic de l'aplicació farem uns retocs gràfics mitjançant l'eina DockPanel i una sèrie de botons.

L'eina DockPanel té moltes aplicacions i es pot utilitzar per diferents motius, però nosaltres l'utilitzarem simplement per crear una barra de color blau, que serà on ubicarem també els botons i que ens permetrà millorar l'aspecte gràfic de la HMI.

L'eina DockPanel també està ubicada en el quadre d'eines. Un cop localitzem la eina, l'haurem d'arrossegar dins de la nostra pantalla central i un cop allà l'ampliarem perquè ocupi la part superior del TabControl així tapant-lo per poder fer una botonera més amena.

És important que el DockPanel ens quedi en el Grid i no dins del TabControl, ja que com ja hem comentat el que ens interessa és suplantar el TabControl.

Un cop col·locat el DockPanel i un cop ampliat, mitjançant la pantalla de propietats li posarem el color blau (Il·lustració 52 i Il·lustració 53).



Il·lustració 52: Captura de pantalla de l'aspecte gràfic del DockPanel

```
<DockPanel HorizontalAlignment="Left" Height="35" LastChildFill="False" VerticalAlignment="Top" Width="805" Background="#FF0C2285"/>
```

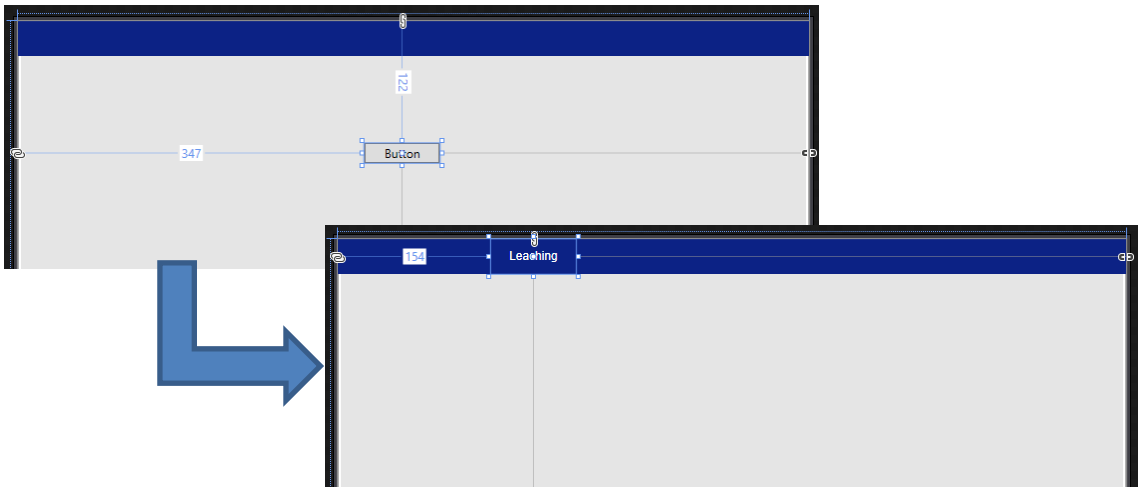
Il·lustració 53: Fragment del codi XAML referent al DockPanel

Seguidament, col·locarem 7 botons a sobre del DockPanel que acabaran de conformar el menú superior de la nostra aplicació.

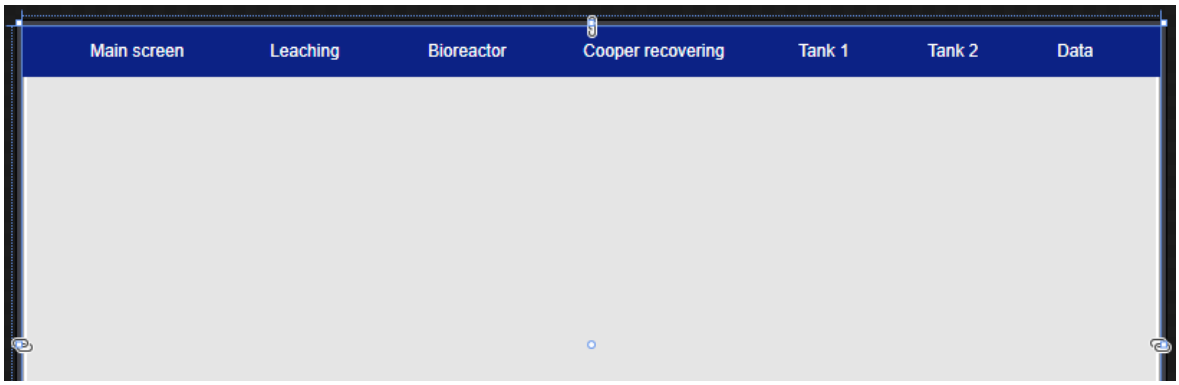
Els botons, com les altres eines abans citades també es troben a la part dreta de l'entorn de Visual Studio, concretament al quadre d'eines, Un cop localitzem l'eina "Button" l'arrosseguem cap al DockPanel i amb una separació constant anem col·locant els 7 botons (Il·lustració 54 i Il·lustració 55).

Per cada botó, o bé mitjançant la finestra de propietats o bé directament escrivint el codi XAML, n'haurem de definir les seves propietats:

- **Content:** Aquest propietat fa referència al contingut de dins del botó, és a dir el nom que portarà el botó. En el nostre cas s'anomenaràn: Main screen, Leaching, Bioreactor, Cooper recovering, Tank 1, Tank 2 i Data.
- **Click:** Nom de la funció que programarem en la part de programació quan es cliqui el botó. En el nostre cas s'anomenaràn: Mainscreen_Click, Leaching_Click, Bioreactor_Click, Cooperrecovering_Click, Tank1_Click, Tank2_Click i Data_Click.
- **Background**: És la propietat que fa referència al fons del botó. En el nostre cas el color de fons serà el mateix blau que el del DockPanel.
- **Foreground**: Fa referència al color de les lletres de dins del botó. En el nostre cas el posem de color blanc.
- **FontFamily:** Fa referència a la font o tipus de lletra que utilitzem per al text de dins del botó.
- **BorderBrush:** Fa referència al color de "Border" o vora/límit del botó, és a dir la línia fina que limita el botó. En el nostre cas, el posarem del mateix color blau de fons del botó i del Dock Panel perquè els botons quedin integrats dins del Dock Panel com si fos un sol conjunt.
- També hi ha les propietats de **posició** i **dimensions**, les quals es poden ajustar fàcilment mitjançant les fletxes de l'entorn gràfic.



Il·lustració 54: Captura de pantalla amb el botó per defecte i el retocat



Il·lustració 55: Captura de pantalla del menú superior acabat

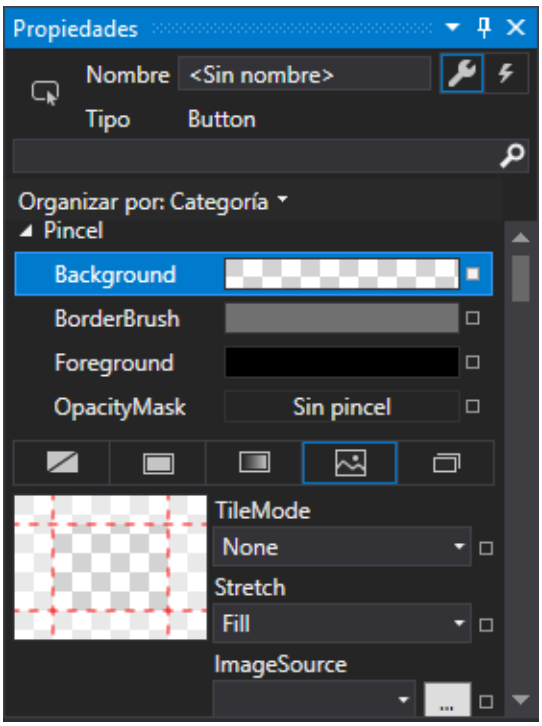
Per acabar la part gràfica més general de l'aplicació, és a dir la part que serà visible en qualsevol de les pestanyes de l'aplicació, tal com passa amb la barra o menú superior, introduirem un altre "Button". En aquest cas, l'introduirem a la part inferior esquerra de la pantalla i serà el botó d'apagada de l'aplicació (Il·lustració 57 i Il·lustració 58).

Per a posar aquest botó, seguirem els mateixos passos que per a introduir els botons anteriors, però l'única diferència serà que aquest botó en comptes de tenir un fons de color i lletres, tindrà com a fons una icona representativa del símbol d'apagat.

Les propietats que tindrà el botó seran les mateixes comentades anteriorment sumant-hi la propietat que s'anomena Button.Background, la qual ens permetrà introduir la icona seleccionada com a fons del botó.

Dins del Button.Background hi anirà un ImageBrush que no és res més que la pintada de l'àrea seleccionada, en aquest cas el fons del botó amb la imatge seleccionada.

Això es pot fer senzillament des de la finestra de propietats del botó (Il·lustració 56) i sense necessitat d'escriure el codi XAML.



Il·lustració 56: Finestra de propietats del Button

Simplement entrant a la categoria Pincel de les propietats i clicant sobre Background ens deixa escollir quin tipus de fons volem: Sense fons, Fons de color, Imatge... Si cliquem sobre "Pincel de diseño en mosaico" i seguidament sobre els 3 punts d'Image Source, podrem escollir la imatge que volem escollir de fons del botó.

```
<Button Name="Power_button" Click="Power_button_Click" Content="" HorizontalAlignment="Left" Margin="755,438,0,0"
    <Button.Background>
        <ImageBrush ImageSource="power-icon.png"/>
    </Button.Background>
</Button>
```

Il·lustració 57: Fragment del codi XAML referent al botó d'apagada de l'HMI

Il·lustració 58: Captura de pantalla del botó d'apagada

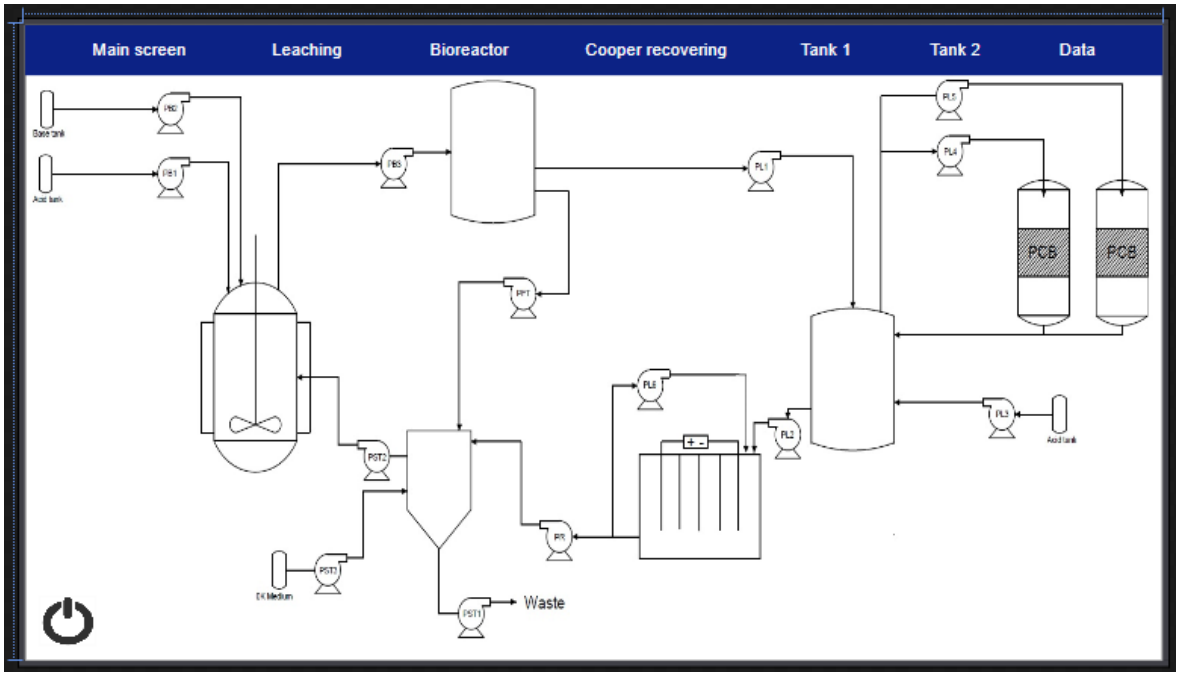
5.1.2. EDICIÓ GRÀFICA DE LA MAIN SCREEN

La pantalla principal o Main screen de la nostra aplicació estarà composta per un fons de pantalla on es veurà el diagrama de tota la planta de biolixiviació, un quadre amb diferents opcions de connexió/desconnexió de l'aplicació amb el port sèrie del PLC i un rellotge que ens mostrarà la data i l'hora actuals.

Comencem pel fons de pantalla (Il·lustració 59). Per a posar en el fons de pantalla una imatge del procés de biolixiviació hem d'utilitzar la propietat Background, que ja hem esmentat i utilitzat anteriorment.

Tal com ja sabem, ho podem fer des de la finestra de propietats o escrivint amb llenguatge XAML directament. El que sí que és important és seleccionar el TabItem corresponent a la Main_screen per posar el background. Si ho féssim directament, correríem el perill de que se'ns seleccionés la MainWindow i llavors les propietats de la finestra serien de la MainWindow i no de la Main_screen.

Un cop hem seleccionat el TabItem corresponent a la pantalla principal, ja podem seleccionar la propietat Background -> Pincel de diseño en mosaico -> ImageSource i seleccionem la imatge del procés de biolixiviació.



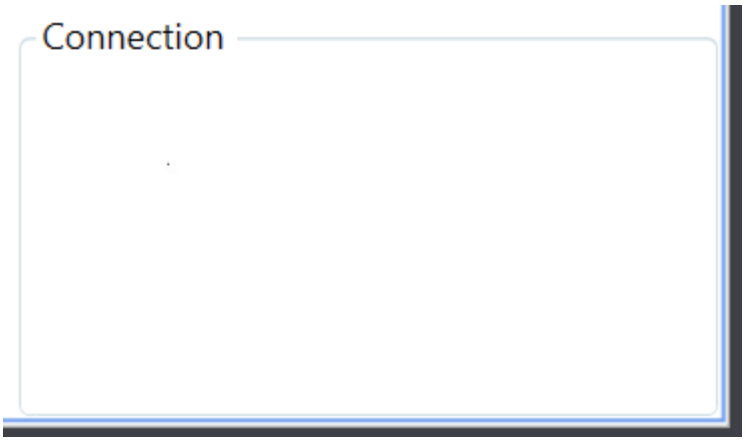
Il·lustració 59: Captura de pantalla de la Main screen amb el fons de pantalla

Tot seguit, hem de dissenyar el quadre de connexió/desconnexió del port sèrie del PLC amb el Visual Studio.

Per començar, crearem un GroupBox (Il·lustració 60) que és una eina que ens permet agrupar diferents botons i altres eines dins seu. Tal com hem fet anteriorment, per seleccionar-lo hem d'anar al quadre d'eines i buscar GroupBox per posteriorment arrossegar-lo cap a la pantalla.

La particularitat del GroupBox respecte d'altres eines d'agrupació és que té un contorn i un "Header" o títol característics i que gràficament fan millorar molt el disseny.

Un cop el GroupBox estigui a la pantalla central, li posem una mida amb la qual creiem que hi cabràn totes les eines que hi hem de posar dins. I com a títol li posem Connection, ja que serà el quadre de connexió al port sèrie.



Il·lustració 60: Captura de pantalla del GroupBox retocat

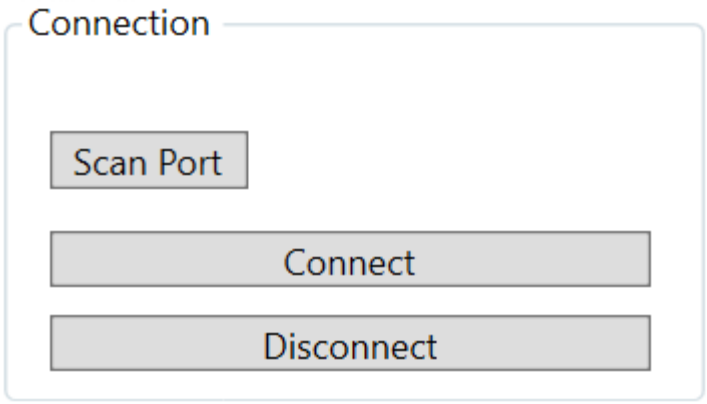
A dins del GrupBox hi introduirem les següents eines (Il·lustració 61): 1 botó de connexió al port sèrie, 1 botó de desconnexió del port sèrie, 1 ComboBox de selecció del Port d'entrada, 1 botó d'escaneig dels ports disponibles, 1 etiqueta de l'estat actual (Connected o Disconnected) i 2 Ellipses (rodones) de color vermell i verd que estaran sobreposades i indicaran l'estat actual (Connected o Disconnected).

Començarem per la creació dels botons, ja que ja n'hem parlat anteriorment. Per crear un botó hem d'anar a Quadre d'Eines -> Button i arrossegar-lo cap a la pantalla central.

Les propietats més importants i de les quals ja hem parlat anteriorment de cadascun dels botons que introduirem dins del GroupBox són les següents (Taula 4):

Taula 4. Propietats dels botons del GroupBox

Botó	Name	Content	Click
Connexió	Connect	Connect	Connect_Click
Desconnexió	Disconnect	Disconnect	Disconnect_Click
Escaneig de Ports	Scan_Port	Scan Port	ScanPort_Click

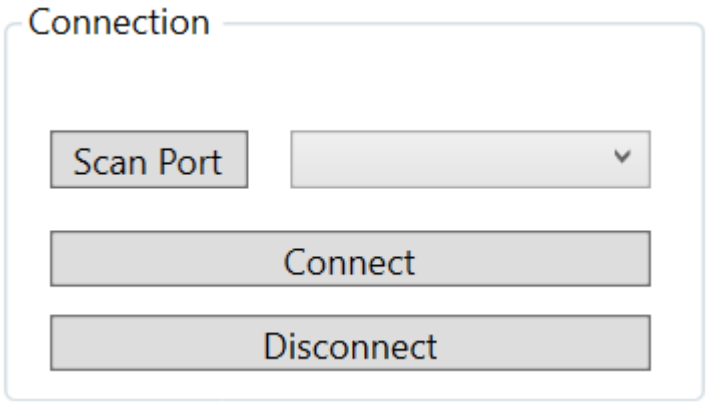


Il·lustració 61: Captura de pantalla del GroupBox amb els botons

Seguidament, introduirem el ComboBox (Il·lustració 62) anant al Quadre d'Eines i buscant ComboBox.

El ComboBox és una llista desplegable que serveix per poder escollir una opció d'entre les que apareixen dins de la llista desplegable.

El ComboBox ens servirà per seleccionar quin dels ports disponibles és el que té el PLC connectat. Com a propietat important només destaco el Name, que és ComboBoxPort.



Il·lustració 62: Captura de pantalla del GroupBox amb els botons i el ComboBox

Finalment, introduirem els dos indicadors d'estat de la connexió (Il·lustració 63 i Il·lustració 66).

El primer dels dos serà un Label o etiqueta, que tal i com les altres eines es troba en el Quadre d'Eines de la dreta de l'IDE de Visual Studio.

Les propietats més importants del Label són les següents (Taula 5):

Taula 5: Propietats del Label Status_Label

Name	Content
Status_Label	Status: Disconnected

El segon estarà format per dues Ellipses, que és una eina que no apareix per defecte dins de Visual Studio i que s'ha d'afegir mitjançant llibreries.

Una de les dues Ellipses serà de color verd i l'altra serà de color vermell i seran els indicadors lluminosos o de color que indicaran si el port sèrie està connectat o desconnectat.

El procés per crear una Ellipse és el mateix que en les altres eines: Quadre d'Eines -> Ellipse.

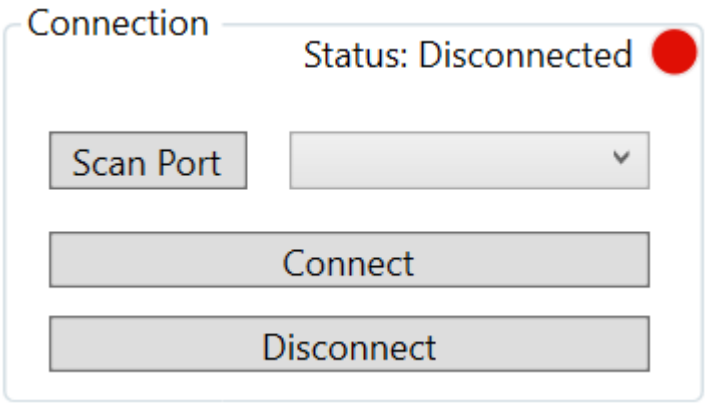
Les dues estaran superposades i mitjançant una propietat que ara esmentarem es produirà el canvi de verd cap a vermell o a l'inrevés.

Com ja hem comentat, a part de les propietats esmentades anteriorment, en aquesta eina entra en joc una propietat anomenada Visibility. Aquesta propietat ens permet invisibilitzar i/o fer visible qualsevol eina i nosaltres la utilitzarem per amagar la Ellipse verda i mostrar l'Ellipse vermella quan el port estigui desconnectat i a l'inrevés quan aquest estigui connectat.

Les propietats més importants de les Ellipses són les següents (Taula 6):

Taula 6: Propietats de les Ellipses d'estat de la connexió

Name	Fill	Visibility	Stroke
Red_disconnected	Vermell	Visible	Blanc
Green_connected	Verd	Collapsed	Blanc



Il·lustració 63: Captura de pantalla del GroupBox complet

Finalment, hem d'introduir el rellotge, és a dir, la data i hora actuals. Per fer-ho, crearem un Label amb les propietats següents:

Taula 7: Propietats del label Data_label

Name	Content
Data_Label	No data available

Com a contingut del label escrivim "No data available" perquè la funció per introduir la data i l'hora actuals la introduirem quan fem la programació de l'aplicació i mitjançant la propietat Content del label.

Per embellir l'etiqueta de la data i l'hora crearem un Frame, que és un objecte del quadre d'eines que de moment no hem utilitzat. Aquest objecte pot tenir diverses funcions, però nosaltres l'utilitzarem per fer un simple marc a l'entorn de la etiqueta de la data i la hora.

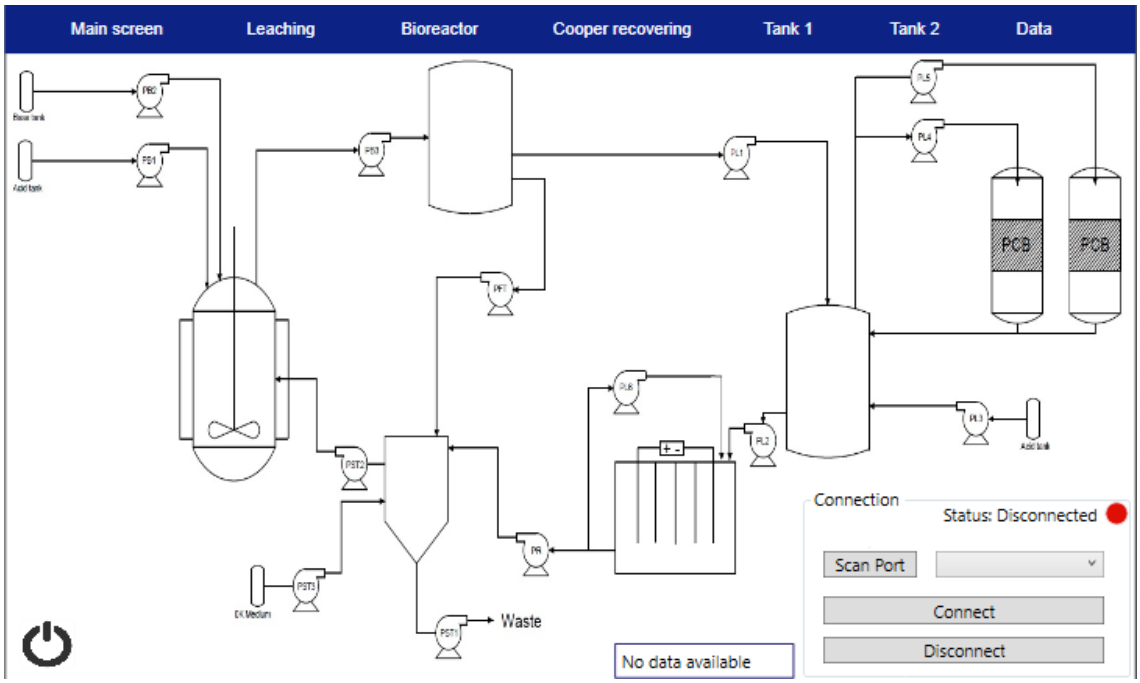
Com a propietats del Frame, destaquem:

- BorderBrush: És el color del marc. En el nostre cas el posarem de color blau.
- BorderThickness= És la gruixària del marc. En el nostre cas li posarem un 1.

En la següent imatge podem observar com queda el rellotge, abans i després d'engegar la aplicació. És a dir, sense mostrar la data i l'hora (abans d'engegar la aplicació, en el disseny WPF) (Il·lustració 64) i mostrant-la (un cop engegada la aplicació) (Il·lustració 65).

Il·lustració 64: Captura de pantalla del rellotge abans d'engegar la aplicació

Il·lustració 65: Captura de pantalla del rellotge un cop engegada la aplicació



Il·lustració 66: Captura de pantalla de la Main screen amb el fons de pantalla i el GroupBox

5.1.3. EDICIÓ GRÀFICA DE LES PANTALLES DE LES 5 ETAPES

Respecte les pantalles de cadascuna de les etapes del procés de biolixiviació, cadascuna tindrà un fons de pantalla diferent equivalent al diagrama corresponent per cada etapa, però l'estructura gràfica serà la mateixa per les 5 etapes.

L'estructura gràfica de cada pantalla constarà d'un fons de pantalla, de diferents indicadors lluminosos verds, que indicaran quan una bomba/sensor de nivell està engegat/parat i d'etiquetes que indicaran nivells de ph, mesura de color...

Els fons de pantalla que utilitzarem per cada etapa són els corresponents a la etapa en qüestió que ja hem observat en apartats anteriors. El procés per introduir aquests fons de pantalla també és el mateix que hem utilitzat anteriorment.

Cal recordar que és important seleccionar el TabItem corresponent a la etapa en concret per posar el fons de pantalla, sinó correm el risc de que el fons de pantalla se'ns seleccioni per la MainWindow.

Després només cal seleccionar la propietat Background -> Pincel de diseño en mosaico -> ImageSource i seleccionar la imatge de la etapa en concret que necessitem.

Pel que fa als indicadors de engegat/parat tant de bombes com de sensors de nivell, els crearem amb l'eina "Ellipse" tal com també hem fet anteriorment i també jugarem amb la propietat "Visibility" per fer aparèixer/desaparèixer aquests indicadors.

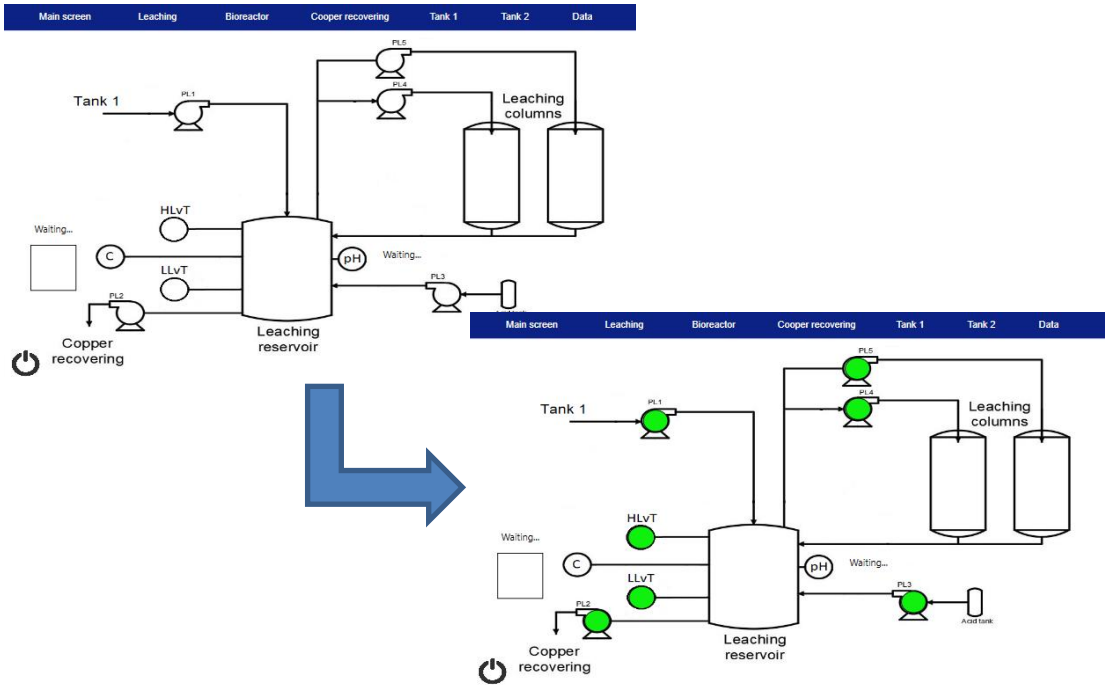
Per veure els codis XAML sencers de les pantalles de cadascuna de les etapes, podeu accedir als annexes del TFG on apareix tot el codi XAML.

Per cadascuna de les pantalles també tenim unes etiquetes, que crearem amb l'eina "Label". Aquestes etiquetes mostren valors de pH, de color, de redox, de temperatura i d'intensitat i, per tant, són valors que s'escanegen contínuament. El contingut o "Content" inicial de les etiquetes serà "Waiting..." i a mesura que es vagin escanejant valors, el contingut canviarà per el valor concret del pH, del color, del redox, de la temperatura o de la intensitat.

També crearem un visualitzador de color per cadascun dels sensors de color. Utilitzarem l'eina Rectangle, que és una forma geomètrica, és a dir una eina semblant a la Ellipse, i li posarem un marc de color negre (Stroke = Black) i un fons de color blanc (Fill = White). Mitjançant la programació, que més tard explicarem, farem que es mostri el color actual del sensor de color en el Rectangle mitjançant la propietat Fill.

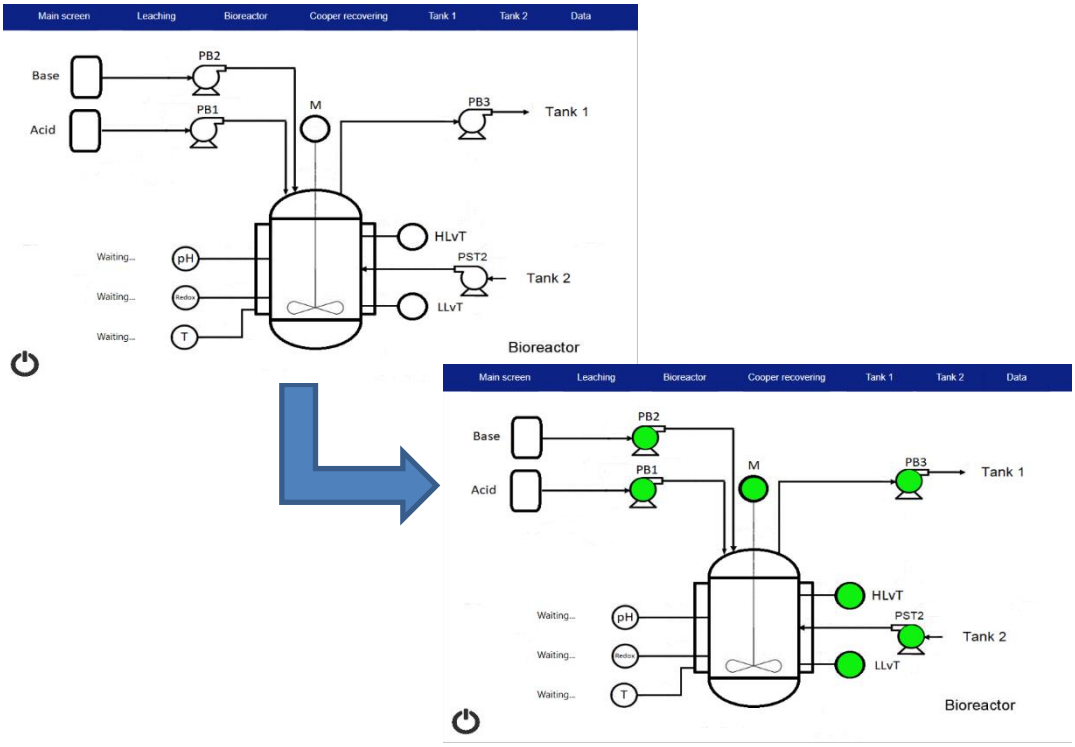
A continuació mostrem totes les pantalles de cadascuna de les etapes per veure com queden, amb i sense l'indicador verd lluminós:

LEACHING



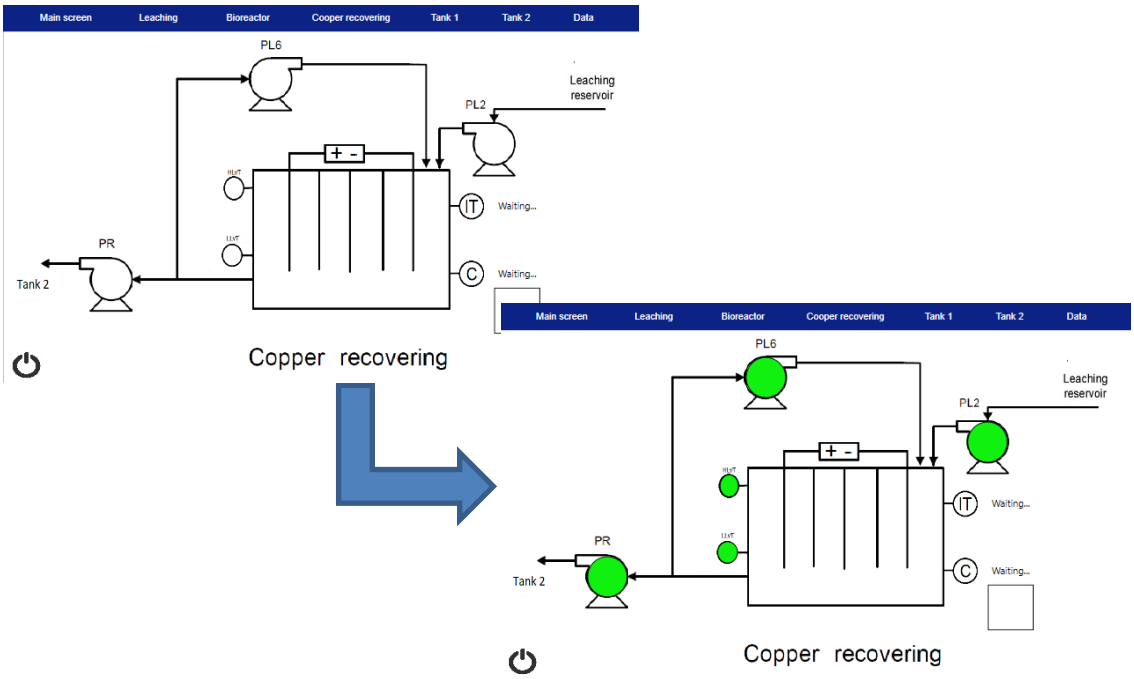
Il·lustració 67: Captura de pantalla de l'etapa de Lixiviació amb/sense indicadors

BIOREACTOR



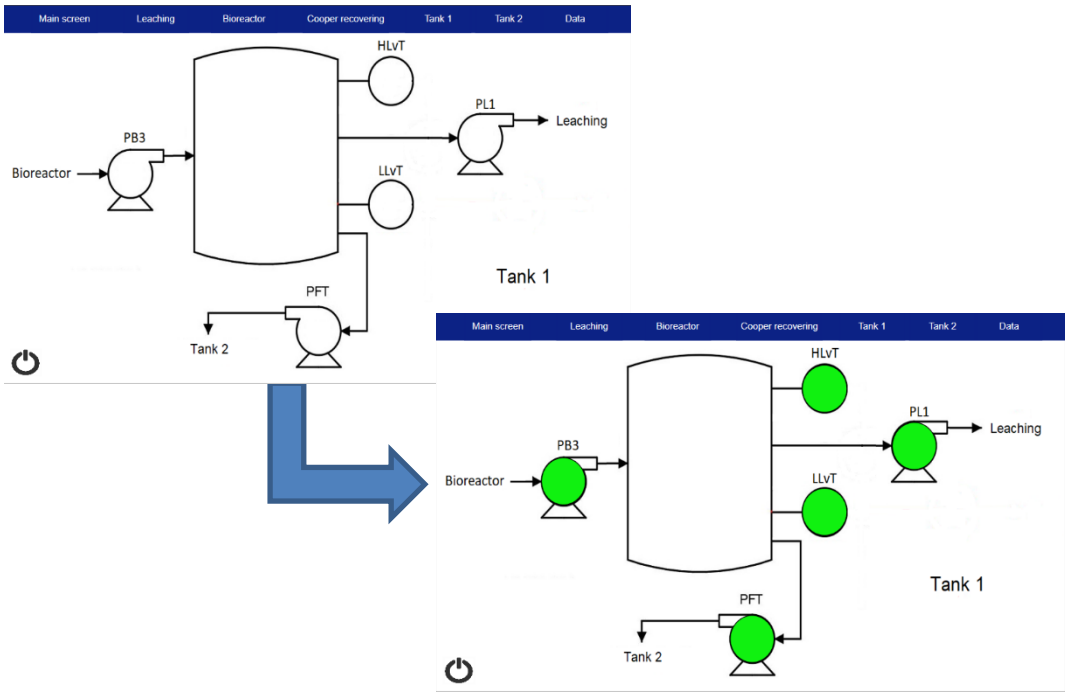
Il·lustració 68: Captura de pantalla de l'etapa del Bioreactor amb/sense indicadors

COOPER RECOVERING



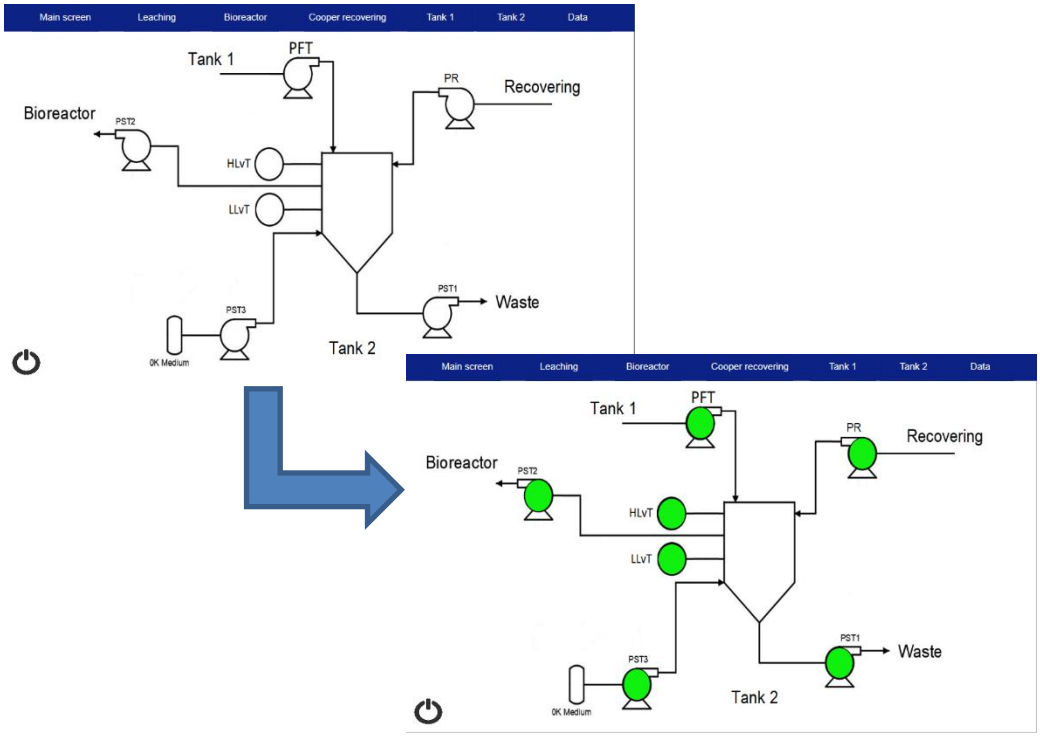
Il·lustració 69: Captura de pantalla de l'etapa de Recuperació del Coure amb/sense indicadors

TANK 1



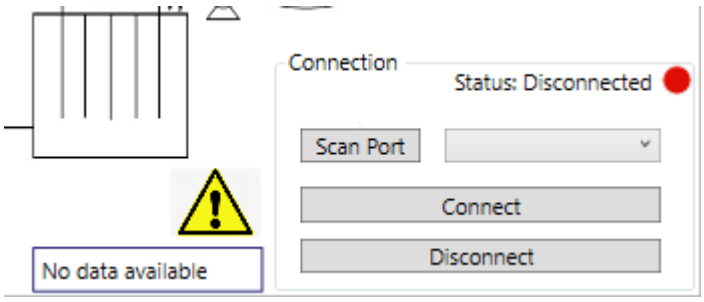
Il·lustració 70: Captura de pantalla de l'etapa Dipòsit 1 amb/sense indicadors

TANK 2



Il·lustració 71: Captura de pantalla de l'etapa del Dipòsit 2 amb/sense indicadors

Finalment, cal dir que en les pantalles de les cinc etapes i també en la pantalla principal s'inclou un TextBox amb un BackGround que mostra el logo d'advertència (Il·lustració 72). Aquest TextBox, mitjançant la propietat Visibility, està sempre ocult fins que salta una alarma. Quant salta, el TextBox amb el logo d'advertència es fa visible en totes les pantalles tot indicant-nos que dins el procés s'ha activat una alarma.



Il·lustració 72: Captura de pantalla del logo d'alarma en la pantalla principal

5.1.4. EDICIÓ GRÀFICA DE LA PANTALLA DE DADES

La pantalla de dades estarà composta per 3 subapartats ben diferenciats:

- El primer d'ells serà una pantalla d'alarmes on s'aniran mostrant totes les alarmes que afectin la planta de biolixiviació.
- El segon serà una llista que mostrarà els temps que han estat en funcionament les bombes PB1 i PB2 del Bioreactor i les bombes PL3, PL4, PL5 i PL6 de l'etapa de lixiviació des que s'engeguen fins que s'apaguen.
- I el tercer i últim serà una llista que mostrarà el valor dels sensors PH del Bioreactor, PH de l'etapa de lixiviació, Redox del Bioreactor, Intensitat de l'etapa de recuperació del coure i Temperatura del Bioreactor. Aquesta llista anirà mostrant el valor d'aquests sensors cada 20 segons des que s'engega l'aplicació i fins que s'apaga.

Començarem amb la explicació de la pantalla d'alarmes. Per a crear-la, utilitzarem l'eina TextBox del quadre d'eines. L'eina TextBox pot ser utilitzada per diversos motius, però nosaltres l'utilitzarem per mostrar un text que equivaldrà a la alarma vigent.

Les propietats més importants i que cal explicar d'aquest TextBox són les següents:

- Name: És el nom que li donem al TextBox. Nosaltres l'anomenarem Alarms_TextBox.
- Text: És el text que contindrà el TextBox dins seu. Nosaltres hem posat com a text inicial "NO ALARMS DECLARED". Més endavant, mitjançant la programació cridarem les alarmes i el Text canviarà pel text de l'alarma en qüestió.
- Background: És la propietat que defineix el color de fons del TextBox. Nosaltres el posarem de color vermell per mostrar senyal d'advertència ja que ensenya les alarmes.
- BorderBrush: És la propietat que defineix el color del marc del TextBox. Nosaltres el posarem de color negre.
- FontFamily: És la propietat que defineix la font del text. Nosaltres farem servir Arial, ja que és la font que hem utilitzat en la resta d'aplicació.
- FontSize: És la mida de la lletra.
- FontWeight: És la propietat que ens indica si la lletra està en negreta o no. En el nostre cas el posem en negreta perquè és destaqui més el text.
- IsReadOnly: És la propietat que ens indica si l'usuari nomes pot llegir el TextBox i, per tant, no el pot modificar ni escriure-hi. Nosaltres la posem a True ja que nomes volem que es mostri el text i que l'usuari no pugui escriure-hi ni modificar-lo.
- VerticalScrollBarVisibility: És la propietat que ens permet mostrar o no en el TextBox una barra vertical de desplaçament. Nosaltres la posem com a Visible, ja que, algunes de les alarmes son molt llargues i mitjançant la barra ens podem desplaçar verticalment per veure tot el missatge.

A més del TextBox explicat anteriorment, afegirem un altre TextBox que l'utilitzarem per mostrar el logo d'advertència, tot indicant que estem en l'apartat d'alarmes.

Per explicar aquest TextBox només ens fa falta una propietat, que és l'anomenada BackGround. El BackGround serà en forma d'ImageSource, ja que el fons de TextBox que volem serà una imatge, concretament el logo d'advertència.

En la imatge següent es mostra com queda l'apartat d'alarmes (Il·lustració 73):

Il·lustració 73: Captura de pantalla de l'apartat d'alarmes

Tot seguit, explicarem la creació de la llista que mostrarà el temps de funcionament de les bombes PB1, PB2, PL3, PL4, PL5 i PL6.

Per crear la llista utilitzarem l'eina ListView. Seguidament, explicarem les propietats més importants d'aquesta eina:

- Name: És el nom que li donem al ListView. Nosaltres l'anomenarem Llista_temps.
- VerticalScrollBarVisibility: És la propietat que ens permet mostrar o no en el ListView una barra vertical de desplaçament. Nosaltres la posem com a Visible, ja que si s'engeguen molt freqüentment les bombes, la llista pot arribar a ser molt llarga i mitjançant la barra ens podem desplaçar verticalment per veure totes les línies d'informació.
- FontFamily: És la propietat que defineix la font del text. Nosaltres farem servir Arial, ja que és la font que hem utilitzat en la resta d'aplicació.
- GridView: Aquesta propietat ens permet modificar l'aspecte de la ListView. Nosaltres, a partir d'aquesta propietat, crearem tres columnes:
    - La primera columna mostrarà la data i la hora en que s'ha engegat la bomba.
    - La segona columna indicarà la bomba en qüestió.
    - L'ultima columna mostrarà el temps que la bomba ha estat en funcionament.

  Per crear les columnes cal escriure el codi que ensenyem a continuació:

```
<GridView>
    <GridViewColumn Header="Date and Hour" Width="150" DisplayMemberBinding="{Binding Date_and_Hour}" />
    <GridViewColumn Header="Pump" Width="80" DisplayMemberBinding="{Binding Pump}" />
    <GridViewColumn Header="Time working" Width="125" DisplayMemberBinding="{Binding Time_working}" />
</GridView>
```

Il·lustració 74: Captura de pantalla del codi de creació de les columnes

A mes de la ListView, la llista de temps de funcionament de les bombes també tindrà un títol que el crearem mitjançant dues eines diferents: l'eina Rectangle i l'eina Label.

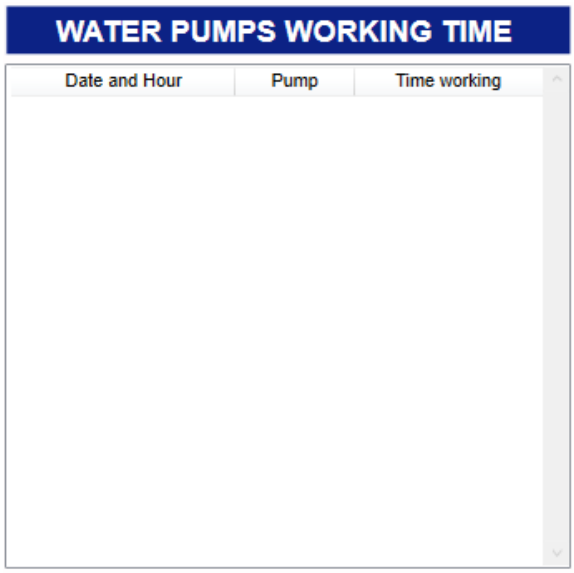
L'eina Rectangle és de la mateixa família que l'eina Ellipse, és a dir, són eines de formes geomètriques. Nosaltres l'utilitzarem per embellir el títol.

La propietat més important i gairebé la única que ens importa és la propietat Fill, que és la que ens permet pintar el rectangle del color que vulguem. Nosaltres li posem el color blau, ja que d'aquesta manera el color concorda amb el de la barra o menú principal.

L'eina Label l'utilitzem per escriure el text del títol. Les propietats més importants de l'eina Label són aquelles relacionades amb el text:

- Content: És el contingut de la Label. Nosaltres li posarem WATER PUMPS WORKING TIME. Això en català significa temps de funcionament de les bombes d'aigua.
- FontFamily: És la propietat que defineix la font del text. Nosaltres farem servir Arial, ja que és la font que hem utilitzat en la resta d'aplicació.
- FontSize: És la mida de la lletra.
- FontWeight: És la propietat que ens indica si la lletra està en negreta o no. En el nostre cas el posem en negreta perquè és destaqui més el text.
- ForeGround: És el color de les lletres. Nosaltres el posarem blanc per seguir amb el criteri de colors de la barra o menú principal.

En la imatge següent es mostra com queda la llista de temps de funcionament de les bombes d'aigua:



Il·lustració 75: Captura de pantalla de la llista del temps de funcionament de les bombes d'aigua

Per últim, explicarem la creació de la llista dels valors dels sensors.

Tal com vam fer a l'inici de l'aplicació en dividir-la en sis pantalles creant un TabControl , doncs ara farem el mateix per la llista de valors dels sensors. Crearem un TabControl amb sis TabItems diferents, un per cadascun dels sensors.

La manera com es crea un TabControl ja ha estat explicada anteriorment i, per tant, no parlarem de les propietats i ens limitarem a dir que cada TabItem tindrà el nom d'un sensor. Els noms dels sensors que posarem seran els següents:

- PH B: Phímetre del bioreactor.
- PH L: Phímetre de l'etapa de lixiviació.
- REDOX B: Sensor de redox del bioreactor.
- CURRENT CR: Sensor d'intensitat de l'etapa de recuperació del coure.
- TEMPERATURE B: Sensor de temperatura del bioreactor.

Dins de cada TabItem crearem una ListView amb dues columnes. En una columna es mostrarà la data i l'hora en que s'ha llegit el valor del sensor i en l'altre columna es mostrarà el valor del sensor. Tal com ja hem explicat, la lectura dels valors dels sensors es fara cada 20 segons.

No parlarem de les propietats del ListView ja que son pràcticament les mateixes que en el ListView del temps de funcionament de les bombes d'aigua, però ensenyarem tal com hem fet abans el codi de creació de les dues columnes:

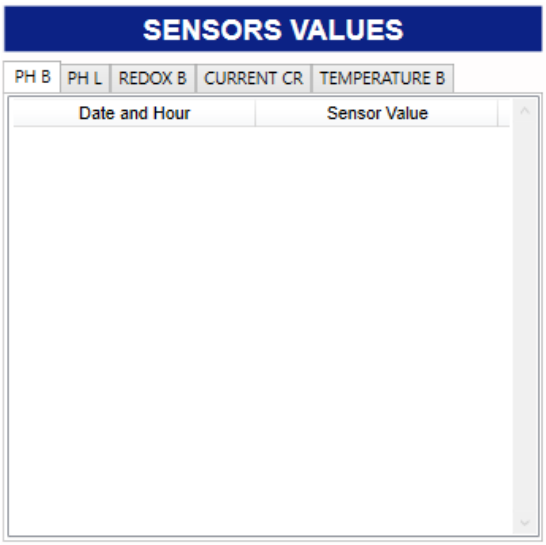
```
<GridView>
    <GridViewColumn Header="Date and Hour" Width="170" DisplayMemberBinding="{Binding Date_and_Hour}" />
    <GridViewColumn Header="Sensor Value" Width="170" DisplayMemberBinding="{Binding Sensor_Value}" />
</GridView>
```

Il·lustració 76: Captura de pantalla de la creació de les columnes

Per finalitzar la llista de valors dels sensors, també crearem un títol referent a la llista. Ho farem de la mateixa manera i amb les mateixes propietats que hem creat el títol de la llista de temps de funcionament de les bombes d'aigua.

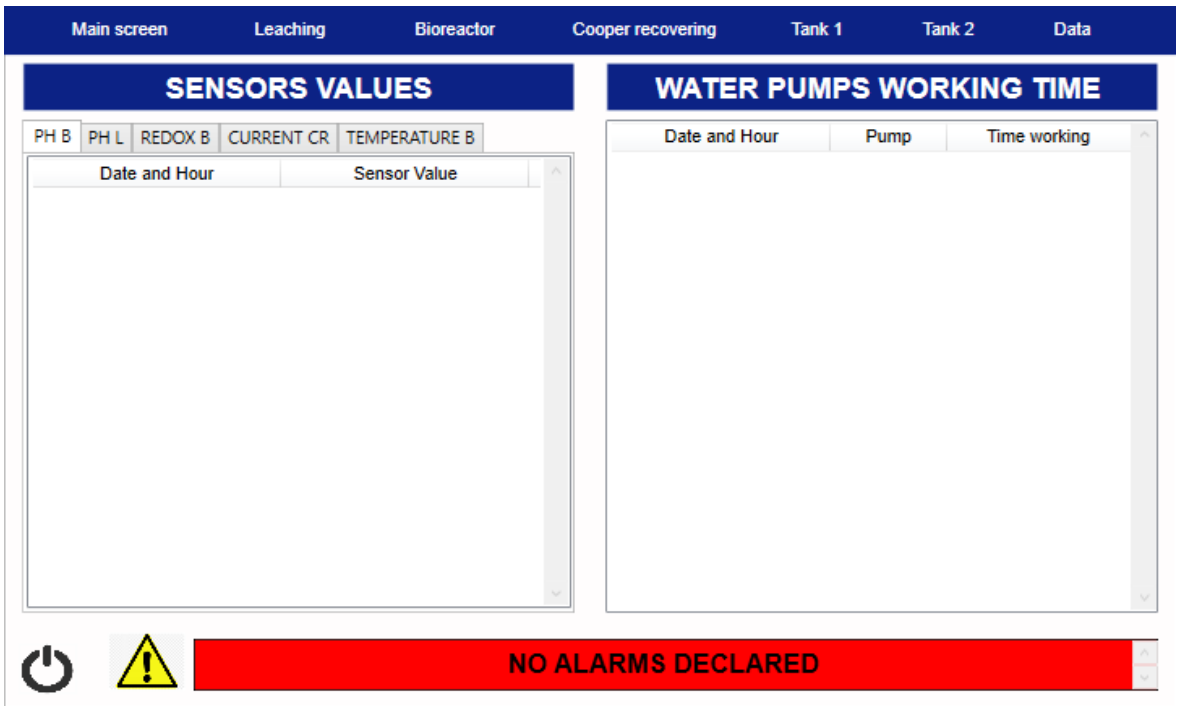
És a dir, utilitzarem la eina Rectangle per fer el fons del títol i utilitzarem l'eina Label per escriure el text del títol.

En la imatge següent es mostra com queda la llista de valors dels sensors (Il·lustració 77):



Il·lustració 77: Captura de pantalla de la llista de valors dels sensors

En la imatge següent i per acabar l'apartat d'edició gràfica de la pantalla de dades, mostrarem com ha quedat (Il·lustració 78):



Il·lustració 78: Captura de pantalla de la pantalla de dades

5.2. PROGRAMACIÓ DE L'APLICACIÓ

Per fer la explicació de la programació de l'aplicació ho farem dividint aquest apartat en 3 parts: en la primera part explicarem quines llibreries necessitem i què necessitem per començar a programar, en la segona part explicarem quines són les funcions més importants de la nostra aplicació i en l'últim apartat, ens centrarem en totes les funcions i llenguatge de comunicació entre el PLC i el Visual Studio.

Abans de començar, per això, cal dir que el llenguatge que utilitzarem per fer la programació de la nostra aplicació és el llenguatge C#. Aquest és un llenguatge de programació d'alt nivell sobretot orientat a objectes.

5.2.1. LLIBRERIES I INICIALITZACIÓ

Una llibreria és un conjunt de funcions que ens faciliten la tasca de programació.

La nostra aplicació tindrà les llibreries per defecte que ja venen en crear qualsevol aplicació WPF, però, a més a més n'hem afegit algunes que ens són útils pel desenvolupament de la nostra aplicació en particular.

El codi que hem d'introduir per afegir qualsevol llibreria és "using" i el nom de la llibreria en qüestió. Per exemple: "using System.Data".

Algunes de les llibreries que ja venen per defecte en crear qualsevol aplicació de WPF són, entre d'altres:

- System
- System.Windows
- System.Windows.Documents
- System.Windows.Media

Per exemple, la llibreria System.Windows.Media inclou funcions per introduir elements multimèdia en les nostres aplicacions.

De llibreries, n'hem afegit moltes, ja que com més llibreries, més disponibilitat de programació i d'eines per millorar la aplicació. Algunes de les que hem afegit i podem destacar o ressaltar, són les següents:

- System.Windows.Shapes: És una llibreria que conté diferents eines per introduir figures geomètriques, tals com el·lipses (les utilitzem en la nostra aplicació), polígons, rectangles...

- System.IO.Ports: És una llibreria que conté funcions per controlar ports sèrie.

- System.Threading.Tasks: És una llibreria que conté funcions per simplificar la escriptura de codis asíncrons.

Per començar, cal dir que tot el codi que crearem estarà contingut dins del namespace Planta_Biolixiviació (Il·lustració 79). Això significa que totes les funcions que creem quedaran contingudes dins d'aquest namespace.

```
namespace Planta_Biolixiviació
```

Il·lustració 79: Captura de pantalla del codi del namespace

La part inicial de la part de programació de la nostra aplicació, sobretot ens serveix per a definir i declarar variables, que és una part molt important, ja que les variables són fonamentals per a la programació.

Un exemple de codi de definició de variables és el següent (Il·lustració 80):

```
private int LDR_1L
```

Il·lustració 80: Captura de pantalla de la creació de la variable LDR_1L

A part d'això, en la part inicial, també hi trobem la definició d'un port sèrie (Il·lustració 81), que és el serialPort1, que és amb el qual treballarem tota l'estona per fer les connexions sèrie.

```
private System.IO.Ports.SerialPort serialPort1;
```

Il·lustració 81: Captura de pantalla de la creació del port sèrie serialPort1

També, apareix la creació d'un nou DispatcherTimer anomenat TimerSerial. Un DispatcherTimer és un control que executa un event o funció anomenada Tick durant un interval de temps determinat.

En el nostre cas, el creem perquè a partir d'aquesta funció i interval de temps anirem prenent dades del port sèrie.

La creació d'aquest DispatcherTimer es fa mitjançant el codi següent (Il·lustració 82):

```
DispatcherTimer TimerSerial = new DispatcherTimer();
```

Il·lustració 82: Captura de pantalla de la creació del DispatcherTimer

5.2.2. FUNCIONS

5.2.2.1. RELLOTGE INICIAL I VALORS DELS SENSORS

Dins de la funció public MainWindow que inicialitza l'aplicació, ens trobem dos Timers que, tot seguit, explicarem.

Els dos Timers s'han de crear de crear fora de la funció (Il·lustració 83).

```
DispatcherTimer Timer = new DispatcherTimer();
DispatcherTimer TimerVariables = new DispatcherTimer();
```

Il·lustració 83: Captura de pantalla de la creació dels Timers

Un dels dos Timers, l'anomenat Timer, és el Timer que servirà per crear el rellotge inicial i l'altre, l'anomenat TimerVariables és el que servirà per el procés de llegir els valors de les variables.

Dins de la funció public MainWindow (Il·lustració 84) es defineixen els dos Timers.

```
public MainWindow()
{
    InitializeComponent();
    Timer.Tick += new EventHandler(Timer_Click);
    TimerVariables.Tick += new EventHandler(Variables_Click);
    Timer.Interval = new TimeSpan(0, 0, 1);
    TimerVariables.Interval = new TimeSpan(0, 0, 20);
    Timer.Start();
    TimerVariables.Start();

}
```

Il·lustració 84: Captura de pantalla del codi de la funció MainWindow

El Timer del rellotge té un EventHandler anomenat Timer_Click i té un interval d'1 segon. També en el MainWindow li donem a Start perquè comenci a córrer.

L'EventHandler, simplement, és una funció que es realitza a cada interval del Timer. En el cas del Timer del rellotge, la funció Timer_Click s'executa cada segon.

La funció Timer_Click (Il·lustració 85) crea una variable DataTime anomenada d i li assigna el valor DateTime.Now, que és l'hora i data actuals. Finalment, mitjançant la propietat Content, envia la data actual cada segon a la etiqueta o Label de la pantalla principal.

```
private void Timer_Click(object sender, EventArgs e)
{
    DateTime d;
    d = DateTime.Now;
    Data_label.Content = d;
}
```

Il·lustració 85: Captura de pantalla del codi de la funció Timer_Click

El Timer TimerVariables té un EventHandler anomenat Variables_Click i té un interval de 20 segons. També en el MainWindow li donem a Start perquè comenci a córrer.

L'EventHandler, simplement, és una funció que es realitza a cada interval del Timer. En el cas del Timer TimerVariables, la funció Variables_Click s'executa cada 20 segons.

La funció Variables_Click (Il·lustració 86) crea 5 llistes amb la funció List, que anomenarem TempsPHB. TempsPHL, TempsREDOX, TempsCURRENT, TempsTEMPERATURE i tindrà uns ítems anomenats items_tempsPHB, items_tempsPHL, items_tempsREDOX, items_tempsCURRENT, items_tempsTEMPERATURE, respectivament.

Seguidament, declarem que aquest ítems que es creïn s'emmagatzemin a les llistes Llista_PHB, Llista_PHL, Llista_REDOX, Llista_CURRENT, Llista_TEMPERATURE, respectivament, que són els ListViews dels valors de cadascun dels sensors.

Per tant, cada 20 segons enviem un nou ítem a les llistes Llista_PHB, Llista_PHL, Llista_REDOX, Llista_CURRENT, Llista_TEMPERATURE, que inclourà la data i hora actuals, i el valor actual del sensor.

```
private void Variables_Click(object sender, EventArgs e)
{
    List<TempsPHB> items_tempsPHB = new List<TempsPHB>();
    Llista_PHB.ItemsSource = items_tempsPHB;

    List<TempsPHL> items_tempsPHL = new List<TempsPHL>();
    Llista_PHL.ItemsSource = items_tempsPHL;

    List<TempsREDOX> items_tempsREDOX = new List<TempsREDOX>();
    Llista_REDOX.ItemsSource = items_tempsREDOX;

    List<TempsCURRENT> items_tempsCURRENT = new List<TempsCURRENT>();
    Llista_CURRENT.ItemsSource = items_tempsCURRENT;

    List<TempsTEMPERATURE> items_tempsTEMPERATURE = new List<TempsTEMPERATURE>();
    Llista_TEMPERATURE.ItemsSource = items_tempsTEMPERATURE;

    items_tempsPHB.Add(new TempsPHB() { Date_and_Hour = DateTime.Now.ToString(), Sensor_Value = ValorPHB });
    items_tempsPHL.Add(new TempsPHL() { Date_and_Hour = DateTime.Now.ToString(), Sensor_Value = ValorPHL });
    items_tempsREDOX.Add(new TempsREDOX() { Date_and_Hour = DateTime.Now.ToString(), Sensor_Value = ValorRB });
    items_tempsCURRENT.Add(new TempsCURRENT() { Date_and_Hour = DateTime.Now.ToString(), Sensor_Value = ValorITR });
    items_tempsTEMPERATURE.Add(new TempsTEMPERATURE() { Date_and_Hour = DateTime.Now.ToString(), Sensor_Value = ValorTB });
}
```

Il·lustració 86: Captura de pantalla del codi de la funció Variables_Click

5.2.2.2. BOTONS DE LA BARRA PRINCIPAL

En la part d'edició gràfica, hem vist com per embellir el TabControl l'hem suplantat amb una barra blava horitzontal amb tants botons com TabItems té el TabControl.

Per poder fer funcionar tot això, necessitem, mitjançant la programació, associar els botons que clicarem amb les pestanyes o TabItems corresponents del TabControl.

Per fer això, necessitem crear una funció per cadascun dels botons mitjançant la propietat Click dels botons. Aquesta propietat (Click) ens permet crear una funció amb el nom que haguem definit per poder decidir què s'ha de fer quan cliquem tal botó.

En el cas que estem explicant, al clicar qualsevol dels botons de la barra blava, ens interessa accedir al TabItem seleccionat. És a dir, si cliquem el botó anomenat "Tank 1" ens interessa entrar a la pestanya/ TabItem del TabControl "Tank 1".

Per exemple, en el cas del botó Main Screen, haurem d'escriure el següent codi (Il·lustració 87):

```
private void Mainscreen_Click(object sender, RoutedEventArgs e)
{
    TabControlPrincipal.SelectedItem = Main_screen;
}
```

Il·lustració 87: Captura de pantalla del fragment de codi del botó Main Screen

Aquest codi, com ja hem explicat, ens permet que al clicar el botó "Main Screen", es seleccioni la pestanya "Main_Screen" del TabControlPrincipal.

Aquesta funció que hem vist en la imatge anterior és la mateixa per als altres botons de la barra blava però canviant els noms pels corresponents.

5.2.2.3. BOTÓ DE PARADA

El botó de parada de l'aplicació, o sigui el botó que apareix en tota la aplicació i que es simbolitza així  , també funciona com els botons explicats anteriorment. Posseeix la propietat Click, la qual ens permet crear una funció per gestionar que passa en clicar el botó.

Aquest botó funciona de la següent manera:

Si l'cliquem, és a dir volem aturar la aplicació, però encara estem connectats al port sèrie, ens apareixerà un "MessageBox". Aquest "MessageBox" o missatge de text, és tal com diu el nom, un text que ens apareixerà al centre de la pantalla avisant-nos que abans d'aturar l'aplicació, hem de desconnectar-la del port sèrie.

Per tant, no podem aturar la aplicació si el port sèrie no està desconnectat.

Si, en cas contrari, cliquem el botó i estem ja desconnectats del Serial Port, llavors l'aplicació s'aturarà immediatament.

La funció del botó de parada que ens permet fer l'exposat anteriorment és el següent (Il·lustració 88):

```
private void Power_button_Click(object sender, RoutedEventArgs e)
{
    if (isConnected)
    {
        MessageBox.Show("Please, click the Disconnect button before exit", "Information");
    }
    else
    {
        this.Close();
    }
}
```

Il·lustració 88: Captura de pantalla de la funció del botó de parada

5.2.2.4. BOTÓ D'ESCANEIG DELS PORTS

Aquest botó està ubicat a la Main Screen o a la pantalla principal i serveix per escanejar els ports sèrie disponibles per connectar-s'hi.

Per fer-ho, primer esborra tots els items que hi ha actualment dins del ComboBox dels Ports per a poder introduir els nous ports que trobarà.

Llavors, executa la funció getAvailableComPorts() (Il·lustració 89), que també és una funció creada per nosaltres, que agafa els noms dels ports disponibles actualment i els emmagatzema dins de la variable ports:

```
void getAvailableComPorts()
{
    ports = SerialPort.GetPortNames();
}
```

Il·lustració 89: Captura de pantalla de la funció getAvailableComPorts()

També cal esmentar, que la variable ports (Il·lustració 90), l'hem creada nosaltres al començament de l'aplicació donant-li forma de String[], és a dir una variable de text.

```
String[] ports;
```

Il·lustració 90: Captura de pantalla de creació de la variable ports com a string

Seguidament, per cadascun dels ports dins de la variable ports, s'afegirà un item amb el nom del port corresponent dins del ComboBox.

Un cop escanejats els ports, aquest ja estaran dins del ComboBox per ser seleccionats. El port que està en la posició [0] apareixerà directament com el predeterminat, però clicant sobre del ComboBox, podrem escollir el port que nosaltres vulguem.

També cal dir que un cop escanejats els ports, la variable portp (Il·lustració 91) passa de false a true. Aquesta variable també ha estat creada per nosaltres i té forma booleana.

```
bool portp = false;
```

Il·lustració 91: Captura de pantalla de la variable booleana portp

Aquesta variable ens servirà més tard per les funcions dels botons Connect i Desconnect.

La funció del botó d'escaneig de ports explicat anteriorment és la següent (Il·lustració 92):

```
private void ScanPort_Click(object sender, EventArgs e)
{
    {
        ComboBoxPort.Items.Clear();
        getAvailableComPorts();

        foreach (string port in ports)
        {
            ComboBoxPort.Items.Add(port);
            Console.WriteLine(port);
            if (ports[0] != null)
            {
                ComboBoxPort.SelectedItem = ports[0];
            }
        }
        portp = true;
    }
}
```

Il·lustració 92: Captura de pantalla de la funció Click del botó Scan Port

5.2.2.5. BOTÓ DE CONNEXIÓ

La funció corresponent que s'executa en clicar el botó de connexió és la següent:

Si la variable isConnected és en estat "false" i la variable portp és en estat "True", passen un seguit de coses que ara explicarem

Si això no passa, doncs al clicar el botó de "Connnect" no passa res a la nostra aplicació.

La variable isConnected ens indica si ja estem connectats o no a un port sèrie i la variable portp ens indica si ja em fet l'escaneig dels ports i per tant tenim algun port seleccionat.

Ens interessa que isConnected estigui "false" perquè significa que encara ens hem de connectar i ens interessa que portp estigui "true" perquè significa que ja tenim un port seleccionat per a connectar-nos.

Un cop tenim la variables isConnected a "false" i la variable portp a "true" passa el següent:

- Com que ja ens hem connectat al clicar el botó, doncs la variable isConnected passa ser "true".

- Seguidament creem el port sèrie serialPort1, seguidament definim un baud rate per el port sèrie, que en el nostre cas li assignem 115200 senyals per segon. També definim quin port sèrie assignem al serialPort1, que serà el que estigui seleccionat en el ComboBoxPort. Finalment, fem un serialPort1.Open() i així activem el port sèrie.

- Ara també hem de definir un interval pel DispatcherTimer, que en el nostre cas s'anomena TimerSerial. Li assignem un interval d'1 milisegon. També assignem la funció de Tick que s'haurà d'anar executant durant aquest interval, que s'anomenarà TimerSerial_Tick_1. Finalment, fem Un TimerSerial_Start() i es comença a executar la funció TimerSerial_Tick_1, que serà l'encarregada de rebre dades del port sèrie.

- Inhabilitem el ComboBoxPort, ja que ja no ens cal perquè ja estem connectats a un port.

- Inhabilitem el botó de Scan Port, ja que no ens cal perquè ja estem connectats a un port.

- Inhabilitem el botó de Connect, ja que no ens cal perquè ja estem connectats a un port.

- Habilitem el botó de Disconnect, ja que en qualsevol moment voldrem/podrem desconnectar-nos del port sèrie.

- Finalment, referent ja als aspectes gràfics de la aplicació, gràcies a la propietat Visibility, farem invisibile la el·lipse vermella i habilitarem la el·lipse verda de la pantalla d'inici, així indicant que ja estem connectats.

- També canviarem el contingut de la etiqueta de la pantalla d'inici. On abans hi posava "Status: Disconnected" ara hi posarà "Status: Connected".

Tot l'explicat anteriorment es pot traduir en llenguatge de programació C# mitjançant aquest codi (Il·lustració 93):

```
private void Connect_Click(object sender, EventArgs e)
{
    if (!isConnected && portp)
    {
        isConnected = true;
        serialPort1 = new System.IO.Ports.SerialPort();
        serialPort1.BaudRate = 115200;
        serialPort1.PortName = ComboBoxPort.SelectedItem.ToString();
        serialPort1.Open();
        TimerSerial.Interval = TimeSpan.FromMilliseconds(1);
        TimerSerial.Tick += TimerSerial_Tick_1;
        TimerSerial.Start();
        ComboBoxPort.IsEnabled = false;
        Scan_Port.IsEnabled = false;
        Connect.IsEnabled = false;
        Disconnect.IsEnabled = true;

        Red_disconnected.Visibility = Visibility.Collapsed;
        Green_connected.Visibility = Visibility.Visible;
        Status_label.Content = "Status: Connected";
    }
}
```

Il·lustració 93: Captura de pantalla de la funció del botó Connect

5.2.2.6. BOTÓ DE DESCONNEXIÓ

La funció corresponent que s'executa en clicar el botó de desconnexió és la següent:

Si la variable isConnected és en estat "true" i la variable portp és en estat "True", passen un seguit de coses que ara explicarem

Si això no passa, doncs al clicar el botó de "Disconnnect" no passa res a la nostra aplicació.

Ens interessa que isConnected estigui "true" perquè significa que estem connectats i ens podem desconnectar i ens interessa que portp estigui "true" perquè significa que ja tenim un port seleccionat per a connectar-nos.

Un cop tenim la variables isConnected a "true" i la variable portp a "true" passa el següent:

- Com que ja ens hem desconnectat al clicar el botó, doncs la variable isConnected passa ser "false".

- Fem un serialPort1.Close() i així desactivem el port sèrie.

- Fem Un TimerSerial_Stop() i es deixa d'executar la funció TimerSerial_Tick_1.

- Habilitem el ComboBoxPort, ja que al desconnectar-nos ja no necessitem cap port en concret i potser en volem escollir un de nou.

- Habilitem el botó de Scan Port, ja que al desconnectar-nos ja no necessitem cap port en concret i potser en volem escollir un de nou.

- Habilitem el botó de Connect, ja que actualment estem desconnectats i en qualsevol moment ens pot interessar connectar-nos de nou.

- Inhabilitem el botó de Disconnect, ja que ja estem desconnectats del port sèrie.

- Finalment, referent ja als aspectes gràfics de la aplicació, gràcies a la propietat Visibility, farem invisibile la el·lipse verda i habilitarem la el·lipse vermella de la pantalla d'inici, així indicant que ja estem desconnectats.

- També canviarem el contingut de la etiqueta de la pantalla d'inici. On abans hi posava "Status: Connected" ara hi posarà "Status: Disconnected".

Tot l'explicat anteriorment es pot traduir en llenguatge de programació C# mitjançant aquest codi (Il·lustració 94):

```
private void Disconnect_Click(object sender, EventArgs e)
{
    if (isConnected && portp)
    {
        isConnected = false;
        Red_disconnected.Visibility = Visibility.Visible;
        Green_connected.Visibility = Visibility.Collapsed;
        Status_label.Content = "Status : Disconnected";

        ComboBoxPort.IsEnabled = true;
        Scan_Port.IsEnabled = true;
        Connect.IsEnabled = true;
        Disconnect.IsEnabled = false;

        TimerSerial.Stop();

        serialPort1.Close();
    }
}
```

Il·lustració 94: Captura de pantalla de la funció del botó Disconnect

5.2.3. COMUNICACIÓ ENTRE EL PLC I VISUAL

Abans de començar la explicació de la comunicació entre el PLC i la nostra aplicació, cal dir que hem partit del programa d'Arduino creat en el primer TFG (Oscar Arrabal) per establir la comunicació sèrie.

És cert que s'han cambiat algunes coses, com per exemple que s'han introduït variables com els sensors de temperatura i intensitat que en el primer programa no hi eren. Però la idea de la comunicació sèrie és la mateixa que s'havia plantejat en aquell TFG.

També cal dir, que per la comunicació sèrie que es fa, tot el que arriba a la nostra aplicació és del tipus string. És a dir, tots els valor que ens arriben són del tipus string i segons quin tipus d'operacions vulguem fer en l'aplicació, doncs haurem de fer conversions de string a integer, a uint o el que ens faci falta.

El codi del PLC modificat està adjuntat als Annexes, concretament al 8.3. CODI M-DUINO.

Un cop feta aquesta petita introducció, comencem amb l'explicació de la comunicació PLC – Aplicació.

La comunicació entre el PLC i la nostra aplicació funciona de la següent manera:

El PLC envia cada interval de temps un codi de l'estil següent a la nostra aplicació:

L10
L20
L30
L40
L50

Els dos primer dígits equivalen a el nom de diferents actuadors o sensor, per exemple L4 equival al sensor de nivell alt del Bioreactor.

I l'últim dígit equival a si està activat/engegat o desactivat/parat. És a dir, si l'últim dígit és 0, vol dir que el sensor està desactivat i si aquest està a 1 doncs que aquest està activat.

La nostra aplicació emmagatzema aquest codi a la variable anomenada StrSerialIn.

A partir d'aquest codi es fan unes quantes transformacions fins que la variable StrSerialInLINES1 agafa una línia del codi, per exemple L40.

I la variable StrSerialInRam agafa el tros del codi dels dos dígits de davant, és a dir, en aquest cas, per exemple, L4.

Ara, per cada bomba o sensor tenim el mateix tros de codi però específic per cadascun. Per exemple, en el cas del sensor de nivell alt del bioreactor, que equival a L4, el codi és el següent (Il·lustració 95):

```
StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 1);
StrSerialInRam = StrSerialInLINES1.Substring(0, 2);
if (StrSerialInRam == "L4")
{
    LDR_4 = StrSerialInLINES(StrSerialInLine, 1);
    if (LDR_4.Substring(2, 1) == "1")
    {
        LEDHLvTB.Visibility = Visibility.Visible;
    }
    else
        LEDHLvTB.Visibility = Visibility.Collapsed;
}
else
    LDR_4 = LDR_4;
```

Il·lustració 95: Captura de pantalla del codi del port sèrie referent al sensor de nivell alt del bioreactor

En aquest codi, primer de tot, veiem que es defineixen StrSerialInLINES1 i StrSerialInRam, que en aquest cas equivaldrien a L40 i L4.

Seguidament, tenim un "if". Aquest "if" és aquí perquè el comportament de les variables és dinàmic i, per tant, poden tenir altres valors continguts a dins i fins que no apareix el valor concret que satisfà aquest "if" no es pot realitzar la funció.

Aquest "if" compara si StrSerialInRam equival a L4, en aquest cas, és a dir, mira si dins de les variables actualment hi ha el valor que ens interessa mirar.

Si StrSerialInRam equival a L4, llavors definim la següent variable:

- LDR_4: Aquesta variable equival a StrSerialInLINES1, és a dir que equival a L40, o sigui a tota la línia de codi de sensor.

Aquesta variable, evidentment, també és diferent per cadascun dels sensors/actuadors. Per exemple, pot ser LDR_3 per L3 o LDR_10 per L10.

A partir d'aquí es fa un LDR_4.Substring(), però aquest Substring serà diferent per cadascuna de les opcions que enumerarem a continuació:

- Si la variable és L3...L9, és a dir que només tenen un número i són variables que equivalen a sensors de nivell o bombes, es fa un LDR_4.Substring(2,1). Aquest fragment de codi serveix per aïllar l'últim dígit del codi de sensor per a poder-lo comparar. En aquest cas, aquest fragment de codi equivaldria a 0, perquè l'últim dígit de L40 és 0.

  Aquest fragment de codi es compara amb "1" i es fa un "if". Si el fragment de codi equival a "1", llavors vol dir que el sensor està activat i, per tant, mitjançant la propietat Visibility, s'il·lumina la casella de l'aplicació corresponent al sensor en qüestió.

  Si, per contrari, el fragment de codi no equival a "1", llavors, també mitjançant la propietat Visibility, s'invisibilitza la casella de l'aplicació corresponent al sensor en qüestió.

  D'aquesta manera tan simple podem observar a la nostra aplicació quan s'activa/desactiva un sensor o quan s'engega/apaga una bomba.

- Si la variable és L10...L27, és a dir que tenen dos números i són variables que equivalen a sensors de nivell o bombes, es fa un LDR_10.Substring(3,1), ja que l'aïllament de l'última xifra s'ha de fer retrocedir una posició perquè el nom de la variable té un número més.

  Un cop aïllat l'últim dígit del codi, passa el mateix que l'explicat pels casos anteriors.

- Si la variable és L28...L34, és a dir que tenen dos números i, a més a més són variables que equivalen a sensors analògics: pHímetre, sensor de color, sensor de redox, sensor de temperatura o sensor d'intensitat; llavors es fa un LDR_28.Substring(3). D'aquesta manera s'agafen tots els dígits que hi ha a darrere de L28.

  Per exemple, imaginem-nos que el sensor de pH ens dona un valor de 10, llavors el codi que arribarà serà L2810 i fent la comanda LDR_28.Substring(3), s'aïllaran els dígits a partir de la posició 3, per tant ens quedarà el 10 com a valor a retornar.

En aquests casos, a diferència dels anteriors, el valor que ens quedi un cop fet el Substring, l'emmagatzemem en una variable, que depenent del sensor en qüestió, s'anomenarà: ValorCR (Sensor de Color de l'etapa recuperació de coure), Valor CL (Sensor de Color de l'etapa de lixiviació), ValorPHL (Phímetre de l'etapa de lixiviació), PHB (Phímetre Del bioreactor), ValorITR (Sensor d'intensitat de l'etapa de recuperació del coure), ValorRB (Sensor de Redox del bioreactor), ValorTB (Sensor de temperatura del bioreactor).

```
StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 31);
StrSerialInRam = StrSerialInLINES1.Substring(0, 3);
if (StrSerialInRam == "L34")
{
    LDR_34 = StrSerialInLINES(StrSerialInLine, 31);
    ValorTB = LDR_34.Substring(3);

}
else
    LDR_34 = LDR_34;
```

Il·lustració 96: Captura de pantalla del codi del port sèrie referent al sensor de temperatura del bioreactor

En els casos de sensors analògics, un cop hem emmagatzemat el valor del sensor dins de la variable corresponent segons l'explicat anteriorment, doncs hem de mostrar aquest valor a les pantalles corresponents dels sensors. Per a fer-ho, utilitzarem la propietat Content dels diferents Labels.

```
labelCTR.Content = "CTR = " + ValorCR;
labelpHTL.Content = "PHTL = " + ValorPHL;
labelpHTB.Content = "PHTB = " + ValorPHB;
labelCTL.Content = "CTL = " + ValorCL;
labelIT.Content = "ITR = " + ValorITR + " mA";
labelREDOX.Content = "REDOX = " + ValorRB;
labelTEMP.Content = "TEMP = " + ValorTB + " °C";
```

Il·lustració 97: Captura de pantalla del codi que mostra el valor dels sensors als Labels corresponents

5.2.3.1. MOSTRADOR DE COLOR

Com ja hem explicat anteriorment, els sensors de color, a part de mostrar el valor que retornen en un Label, també mostraran el color actual del sensor en un mostrador creat a partir de l'eina Rectangle.

Per a fer-ho, primer de tot, convertim la variable ValorCL, de string cap a integer.

Seguidament, amb el codi que mostrarem a continuació (Il·lustració 98), convertim el valor del color que ens ha retornat el sensor en els tres components RGB.

Un cop tenim els tres components RGB, Red, Green i Blue, hem de crear un nou SolidColorBrush, que és una funció que ens permet crear un nou color. Nosaltres anomenem el nou color Brush_CL.

Finalment, mitjançant la propietat Fill de l'eina Rectangle pintem el mostrador amb el color corresponent.

Amb el sensor de color CR hem de seguir els mateixos passos.

```
ValorCLInt = int.Parse(ValorCL);
var blueValue_CL = ValorCLInt / 65536;
var greenValue_CL = (ValorCLInt - blueValue_CL * 65536) / 256;
var redValue_CL = ValorCLInt - blueValue_CL * 65536 - greenValue_CL * 256;
SolidColorBrush Brush_CL = new SolidColorBrush(Color.FromRgb(Convert.ToByte(redValue_CL), Convert.ToByte(greenValue_CL), Convert.ToByte(blueValue_CL)));
Color_Leaching.Fill = Brush_CL;
```

Il·lustració 98: Captura de pantalla del codi per mostrar el color del sensor

5.2.3.2. ALARMES

Les alarmes que crearem per la nostra aplicació són les següents:

- El pHímetre de l'etapa de lixiviació ha estat per sota d'1.6 o per sobre d'1.8 més de 10 mostres.

- El pHímetre del bioreactor ha estat per sota d'1.6 o per sobre d'1.8 més de 10 mostres.

- El sensor de nivell superior del bioreactor ha estat a ON més de 10 mostres.

- El sensor de nivell superior del tanc 2 està a ON.

- El sensor de nivell inferior del tanc 1 està a OFF.

Pel que fa a les dues últimes alarmes, el procediment per crear-les és molt senzill, però per dissenyar les primeres haurem de fer un petit codi que farà de comptador.

Per a dissenyar el comptador farem el següent:

- Nosaltres hem fet que cada mostra es faci al cap de 18 segons, d'aquesta manera arribem als 3 minuts. És a dir, l'alarma ens avisa si els pH estàn fora del rang 1.8 > Ph > 1.6 o el sensor de nivell està a ON més de 3 minuts.

- Pels casos del pH, farem un if que miri si el pH està fora del rang entre 1.6 i 1.8. Pel cas del sensor de nivell, farem un if que miri si està a ON.

- Dins del if crearem un loop for. Aquest loop s'esperarà 18 segons i després tornarà a fer un if. Si encara estem fora del rang del Ph o el sensor de nivell està a ON, la variable PHB_Comptador (PHL_Comptador o HLVTB_Comptador) sumarà 1. Si ja estem dins del rang o el sensor de nivell està a OFF, el comptador es posarà a 0 i s'aturarà el loop.

- Si el pH està fora del rang o el sensor està a ON, el loop s'anirà realitzant fins que arribem a 10. Un cop arribem a 10, el loop s'atura i el comptador és igual a 10.

- Quan tornem a estar en el rang de pH entre 1.6 i 1.8 o el sensor de nivell estigui a OFF, el comptador es posa automàticament a 0.

```
if ((ValorPHL_integer > 1.8)|(ValorPHL_integer < 1.6))
{
    for (int i = 0; i < 10; i++)
    {

        Thread.Sleep(18000);
        if ((ValorPHL_integer < 1.8) | (ValorPHL_integer > 1.6))
        {
            i = 0;
            PHL_Comptador = 0;
        }
        else
            PHL_Comptador += 1;

    }
}
else
    PHL_Comptador = 0;
```

Il·lustració 99: Captura de pantalla d'un dels codis dels comptadors per les alarmes

El codi de les alarmes funciona de la següent manera:

- Si qualsevol dels 3 comptadors està a 10, el sensor de nivell alt del tanc 2 està a ON o el sensor de nivell baix del tanc 1 està a OFF, es fan visibles tots els logos d'advertència de les pantalles, indicant que hi ha una alarma.

- Contràriament, si cap de les 5 coses està passant o deixen de passar, doncs es fan invisibles tots els logos d'advertència de les pantalles i en el mostrador d'alarmes apareix el missatge "NO ALARMS DECLARED".

```
if ((PHL_Comptador == 10) | (PHB_Comptador == 10) | (HLVTB_Comptador == 10) | (LEDHLvTST.Visibility == Visibility.Visible) | (LEDLLvTFT.Visibility == Visibility.Collapsed))
{
    Alarma_Main.Visibility = Visibility.Visible;
    Alarma_Leaching.Visibility = Visibility.Visible;
    Alarma_Bioreactor.Visibility = Visibility.Visible;
    Alarma_Recovering.Visibility = Visibility.Visible;
    Alarma_Tank1.Visibility = Visibility.Visible;
    Alarma_Tank2.Visibility = Visibility.Visible;
}
else
    Alarma_Main.Visibility = Visibility.Collapsed;
    Alarma_Leaching.Visibility = Visibility.Collapsed;
    Alarma_Bioreactor.Visibility = Visibility.Collapsed;
    Alarma_Recovering.Visibility = Visibility.Collapsed;
    Alarma_Tank1.Visibility = Visibility.Collapsed;
    Alarma_Tank2.Visibility = Visibility.Collapsed;
    Alarms_TextBox.Text = "NO ALARMS DECLARED";
```

Il·lustració 100: Captura de pantalla del primer fragment de codi per les alarmes

- Pel que fa al mostrador d'alarmes de la pantalla de dades, hem de fer un if per cada possibilitat que pugui passar, és a dir, si s'engega la alarma del PHL sola, doncs només s'ha de mostrar un missatge d'alarma. Però pot passar que hi hagi més d'una alarma alhora. Per aquest motiu hi ha diferents if: 5 if per cada alarma sola, 1 if per les 5 alarmes alhora i tota la resta de combinacions.

5.2.3.3. TEMPS DE FUNCIONAMENT DE LES BOMBES

Per calcular el temps de funcionament de les bombes, ho farem de la següent manera:

- Primer de tot, farem un if que quan qualsevol de les bombes a mesurar s'engegui, emmagatzemarà l'hora actual dins de la variable tempsInicial.

- Llavors dins del mateix if, hi haurà diferents if per assignar a la una variable diferent a cada bomba per saber quina és la que s'ha activat.

```
if ((PB1.Visibility == Visibility.Visible) | (PB2.Visibility == Visibility.Visible) | (PL3.Visibility == Visibility.Visible) | (PL4.Visibility == Visibility.Visible) | (PL5.Visibility == Visibility.Visible) | (PL6.Visibility == Visibility.Visible))
{
    tempsInicial = DateTime.Now;

    if (PB1.Visibility == Visibility.Visible)
    {
        s = "PB1";
    }

    if (PB2.Visibility == Visibility.Visible)
    {
        s1 = "PB2";
    }

    if (PL3.Visibility == Visibility.Visible)
    {
        s2 = "PL3";
    }

    if (PL4.Visibility == Visibility.Visible)
    {
        s3 = "PL4";
    }

    if (PL5.Visibility == Visibility.Visible)
    {
        s4 = "PL5";
    }

    if (PL6.Visibility == Visibility.Visible)
    {
        s5 = "PL6"; ;
    }
}
```

Il·lustració 101: Captura de pantalla del primer if per crear la llista del temps de funcionament de les bombes

- Seguidament, creem una llista amb la funció List, que anomenarem TempsBombes i tindrà uns ítems anomenats items_temps.

- Seguidament declarem que aquest ítems que es creïn s'emmagatzemin a la Llista_Temps, que és el ListView de temps de funcionament de les bombes de la pantalla de dades.

```
List<TempsBombes> items_temps = new List<TempsBombes>();
Llista_temps.ItemsSource = items_temps;
```

Il·lustració 102: Captura de pantalla de la creació de la llista

- El següent if mira quan les bombes s'apaguen i per diferenciar les que ja estan apagades de les que no, mira que la variable s sigui corresponent a la bomba que s'ha apagat.

- Un cop fet això, s'emmagatzema dins la variable temps_transcorregut el valor de la diferència entre l'hora actual i el tempsInicial. Posteriorment, aquest temps_transcorregut el convertim a string i l'emmagatzemem en una nova variable TempsDeLlista.

- Finalment, posem la variable definitòria a 0 i enviem un nou ítem a la Llista_Temps, que inclourà la data i hora actuals, el nom de la bomba en qüestió i el temps de funcionament de la bomba.

```
if (((PB1.Visibility == Visibility.Collapsed) & (s == "PB1")) | ((PB2.Visibility == Visibility.Collapsed) & (s1 == "PB2")) | ((PL3.Visibility == Visibility.Collapsed) & (s2 == "PL3"))
{
    if (s == "PB1")
    {
        r = "PB1";
    }

    if (s == "PB2")
    {
        r = "PB2";
    }

    if (s == "PL3")
    {
        r = "PL3";
    }

    if (s == "PL4")
    {
        r = "PL4";
    }

    if (s == "PL5")
    {
        r = "PL5";
    }

    if (s == "PL6")
    {
        r = "PL6";
    }

    temps_transcorregut = DateTime.Now - tempsInicial;
    TempsDeLlista = temps_transcorregut.ToString(@"hh\:mm\:ss");
    s = "";
    s1 = "";
    s2 = "";
    s3 = "";
    s4 = "";
    s5 = "";
    items_temps.Add(new TempsBombes() { Date_and_Hour = tempsInicial.ToString(), Pump = r, Time_working = TempsDeLlista });
}
```

Il·lustració 103: Captura de pantalla del segon if per crear la llista de temps de funcionament de les bombes

* La captura de pantalla anterior (Il·lustració 103) està tallada per dalt a la dreta, però cal dir que el codi que segueix és equiparable a la línia de codi que ja apareix però amb les bombes PL4, PL5 i PL6.

6. CONCLUSIONS

En aquest treball s'ha realitzat una aplicació de control de la planta de biolixiviació totalment nova, però amb el programa del PLC M-Duino semblant al dels TFGs anteriors. S'ha volgut fer una aplicació simple, entenedora i senzilla d'utilitzar. Per a fer-ho s'ha utilitzat el programa Visual Studio amb el framework WPF. A més d'això, en la Raspberry que allotjarà la aplicació s'ha instal·lat el sistema operatiu Windows IoT . També, més enllà de l'aplicació, dins del document, s'ha volgut explicar detalladament i de manera simple la realització de cada pas perquè de cara al futur pugui ser útil per a algú altre.

Durant la realització del treball he tingut diverses dificultats en la resolució de petits conflictes en concret, com per exemple el mal funcionament de la bomba d'aigua que tenia per fer proves o la mala connexió de l'USB en transmetre el port sèrie.

També cal dir que l'entorn de Visual Studio i més en concret WPF (Windows Presention Forms) és un món enorme amb molta informació i on cada vegada s'hi suma més gent. A l'hora de fer recerca i buscar informació, aquesta s'ha de desgranar i comparar molt bé per trobar la solució idònia.

Aquest treball té moltes línies obertes i molt de suc per treure, ja que es podrien implementar milers de funcions i eines que podrien fer servei a l'aplicació de control de la planta de biolixiviació.

Com a tall de conclusió, el treball pot ser infinit, ja que sempre es pot anar millorant i actualitzant a nous frameworks, eines de treball, etc.

7. BIBLIOGRAFIA

7.1. REFERÈNCIES

Referència 1: Raspberry Pi. Meet the Raspberry Pi 4 Model B. Element 14. Adreça URL: <https://www.element14.com/community/community/raspberry-pi>. [Consulta: abril 2021].

Referència 2: UUP. Unified Update Platform. Adreç URL: <https://uup.rg-adguard.net/>. [Consulta: abril 2021].

Referència 3: Windows on Raspberry. WoR Project. Adreça URL: <https://www.worproject.ml/downloads>. [Consulta: abril 2021].

Referència 4: Visual Studio. Las mejores herramientas en su categoria para cualquier desarrollador. Adreça URL: <https://visualstudio.microsoft.com/es>. [Consulta: abril 2021].

7.2. WEBGRAFIA

1. How to convert integer color code to Argb color code? https://www.c-sharpcorner.com/forums/how-to-convert-integer-color-code-to-argb-color-code

2. Stackoverflow (Portal on s'han fet diverses consultes) <https://stackoverflow.com/>

3. Microsoft Docs (Web on s'han fet diverses consultes) https://docs.microsoft.com/en-us/dotnet/api/?view=windowsdesktop-5.0

4. Como hacer un temporitzador en C# https://www.delftstack.com/es/howto/csharp/how-to-make-a-dealy-in-csharp/

5. Temporizador en C# https://www.delftstack.com/es/howto/csharp/timer-in-csharp/

6. Aprender a programar (Fòrum on s'han fet diverses consultes) https://aprenderaprogramar.com/foros/index.php?topic=5199.0

7. MSDN Forums (Fòrum on s'han fet diverses consultes) https://social.msdn.microsoft.com/Forums/en-US/home

8. Ciclos infinitos en C# https://www.kyocode.com/2018/10/ciclos-infinitos-en-c/

9. The Complete WPF Tutorial (Web on s'han fet diverses consultes) https://wpf-tutorial.com/

10. ¿Cómo agregar elementos de lista a ListView en C# winform? https://www.it-swarm-es.com/es/c%23/como-agregar-elementos-de-lista-listview-en-c-winform/832612546/

11. How to convert string to int in C#? < https://www.tutorialsteacher.com/articles/convert-string-to-int>

12. C# Timer exemple https://www.dotnetperls.com/timer

13. Medir tiempo de ejecución de un programa en C# https://giltesa.com/2013/04/24/medir-tiempo-de-ejecucion-de-un-programa-en-c

14. Mapping an integer to an RGB color in C# < https://www.py4u.net/discuss/738204>

8. ANNEXOS

8.1. CODI XAML (PART GRÀFICA)

```
<Window
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:local="clr-namespace:Planta_Biolixiviació"
      xmlns:ni="http://schemas.ni.com/controls/2009/xaml/presentation"
      xmlns:sys="clr-namespace:System;assembly=mscorlib"
      x:Class="Planta_Biolixiviació.MainWindow"
      mc:Ignorable="d"
      Title="MainWindow"       Height="491.338582677165"       Width="812.59842519685"
WindowStyle="None">
   <Grid ScrollViewer.VerticalScrollBarVisibility="Disabled">
      <TabControl x:Name="TabControlPrincipal" HorizontalAlignment="Left" Height="483"
VerticalAlignment="Top" Width="805">
         <TabItem Header="Main_screen" Name="Main_screen">
            <Grid>
               <Grid.Background>
                  <ImageBrush ImageSource="EsquemaBase.png"/>
               </Grid.Background>
               <GroupBox  Header="Connection"  HorizontalAlignment="Left"  Height="137"
Margin="564,318,0,0" VerticalAlignment="Top" Width="235">
                  <Button                Name="Disconnect"                Content="Disconnect"
HorizontalAlignment="Left"  Height="19"  Margin="10,90,0,0"  VerticalAlignment="Top"
Width="200" Click="Disconnect_Click"/>
               </GroupBox>
               <Button  Name="Connect"  Content="Connect"  HorizontalAlignment="Left"
Height="19"        Margin="580,397,0,0"        VerticalAlignment="Top"        Width="200"
Click="Connect_Click"/>
               <ComboBox Name="ComboBoxPort" HorizontalAlignment="Left" Height="19"
Margin="660,364,0,0" VerticalAlignment="Top" Width="120"/>
               <Button      Name="Scan_Port"      Content="Scan      Port"      Height="19"
HorizontalAlignment="Left"  Margin="580,364,0,0"  VerticalAlignment="Top"  Width="66"
Click="ScanPort_Click"/>
               <Label       Name="Status_label"       Content="Status:       Disconnected"
HorizontalAlignment="Left" Margin="660,325,0,0" VerticalAlignment="Top" Height="25"/>
```

```
            <Ellipse Name="Red_disconnected" Fill="#FFE00F05" Visibility="Visible"
HorizontalAlignment="Left" Height="17" Margin="780,329,0,0" Stroke="#FFF0EAEA"
VerticalAlignment="Top" Width="17"/>
            <Ellipse Name="Green_connected" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="17" Margin="780,329,0,0" Stroke="#FFF0EAEA"
VerticalAlignment="Top" Width="17"/>
            <Label Name="Data_label" Content="No data available"
HorizontalAlignment="Left" Margin="431,430,0,0" VerticalAlignment="Top" Height="25"
Width="133"/>
            <Frame Content="" HorizontalAlignment="Left" Height="25"
Margin="431,430,0,0" VerticalAlignment="Top" Width="128" BorderBrush="#FF0B0761"
BorderThickness="1"/>
            <TextBox Name="Alarma_Main" HorizontalAlignment="Left" Height="40"
Margin="506,385,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="53"
BorderBrush="White" Foreground="White" Visibility="Visible">
              <TextBox.Background>
                <ImageBrush ImageSource="warning.png"/>
              </TextBox.Background>
            </TextBox>
          </Grid>
        </TabItem>
        <TabItem Header="Leaching" Name="Leaching">
          <Grid Margin="33,12,-3,0">
            <Grid.Background>
              <ImageBrush ImageSource="Lixiviació_Boo.png"/>
            </Grid.Background>
            <Ellipse x:Name="LEDHLvTL" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="29" Margin="165,235,0,0" Stroke="Black"
VerticalAlignment="Top" Width="34" AutomationProperties.Name="HLvTL_Green"/>
            <Ellipse x:Name="LEDLLvTL" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="29" Margin="166,311,0,0" Stroke="Black"
VerticalAlignment="Top" Width="33" AutomationProperties.Name="LLvTL_Green"/>
            <Ellipse x:Name="PL1" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="28" Margin="184,88,0,0" Stroke="Black"
VerticalAlignment="Top" Width="32" AutomationProperties.Name="PL1"/>
            <Ellipse x:Name="PL2" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="28" Margin="110,341,0,0" Stroke="Black"
VerticalAlignment="Top" Width="32" AutomationProperties.Name="PL2"/>
            <Ellipse x:Name="PL3" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="28" Margin="510,317,0,0" Stroke="Black"
VerticalAlignment="Top" Width="32" AutomationProperties.Name="PL3"/>
```

```
            <Ellipse x:Name="PL4" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="28" Margin="440,73,0,0" Stroke="Black"
VerticalAlignment="Top" Width="32" AutomationProperties.Name="PL4"/>
            <Ellipse x:Name="PL5" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="28" Margin="439,22,0,0" Stroke="Black"
VerticalAlignment="Top" Width="32" AutomationProperties.Name="PL5"/>
            <Label Name="labelpHTL" Content="Waiting..." TabIndex="1"
HorizontalAlignment="Left" VerticalAlignment="Top" Margin="440,268,0,0" Height="Auto"
Width="Auto"/>
            <Label Name="labelCTL" Content="Waiting..." TabIndex="1"
HorizontalAlignment="Left" VerticalAlignment="Top" Margin="0,235,0,0" Height="Auto"
Width="Auto"/>
            <Rectangle Name="Color_Leaching" Fill="White" HorizontalAlignment="Left"
Height="58" Margin="0,269,0,0" Stroke="Black" VerticalAlignment="Top" Width="58"/>
            <TextBox Name ="Alarma_Leaching" HorizontalAlignment="Left" Height="40"
Margin="706,393,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="53"
BorderBrush="White" Foreground="White" Visibility="Collapsed">
              <TextBox.Background>
                <ImageBrush ImageSource="warning.png"/>
              </TextBox.Background>
            </TextBox>
          </Grid>
        </TabItem>
        <TabItem Header="Bioreactor" Name="Bioreactor">
          <Grid>
            <Grid.Background>
              <ImageBrush ImageSource="Biorreactor_Booo.png"/>
            </Grid.Background>
            <Ellipse x:Name="LEDHLvTB" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="33" Margin="500,254,0,0" Stroke="Black"
VerticalAlignment="Top" Width="34" AutomationProperties.Name="HLvTB_Green"/>
            <Ellipse x:Name="LEDLLvTB" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="33" Margin="500,342,0,0" Stroke="Black"
VerticalAlignment="Top" Width="34" AutomationProperties.Name="LLvTB_Green"/>
            <Ellipse x:Name="PB1" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="30" Margin="237,120,0,0" Stroke="Black"
VerticalAlignment="Top" Width="32" AutomationProperties.Name="PB1"/>
            <Ellipse x:Name="PB2" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="30" Margin="240,54,0,0" Stroke="Black"
VerticalAlignment="Top" Width="32" AutomationProperties.Name="PB2"/>
```

```
            <Ellipse x:Name="PB3" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="30" Margin="576,109,0,0" Stroke="Black"
VerticalAlignment="Top" Width="33" AutomationProperties.Name="PB3"/>
            <Ellipse x:Name="PST2_Bioreactor" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="30" Margin="579,307,0,0" Stroke="Black"
VerticalAlignment="Top" Width="33" AutomationProperties.Name="PST2_Bioreactor"/>
            <Ellipse x:Name="M" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="30" Margin="378,119,0,0" Stroke="Black"
VerticalAlignment="Top" Width="33" AutomationProperties.Name="M"/>
            <Label Name="labelpHTB" Content="Waiting..." TabIndex="1"
HorizontalAlignment="Left" VerticalAlignment="Top" Margin="75,282,0,0" Height="Auto"
Width="Auto"/>
            <Label Name="labelREDOX" Content="Waiting..." TabIndex="1"
HorizontalAlignment="Left" VerticalAlignment="Top" Margin="75,332,0,0" Height="Auto"
Width="Auto"/>
            <Label Name="labelTEMP" Content="Waiting..." TabIndex="1"
HorizontalAlignment="Left" VerticalAlignment="Top" Margin="75,383,0,0" Height="Auto"
Width="Auto"/>
            <TextBox Name="Alarma_Bioreactor" HorizontalAlignment="Left" Height="40"
Margin="736,405,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="53"
BorderBrush="White" Foreground="White" Visibility="Collapsed">
              <TextBox.Background>
                <ImageBrush ImageSource="warning.png"/>
              </TextBox.Background>
            </TextBox>
          </Grid>
        </TabItem>
        <TabItem Header="Cooper_recovering" Name="Cooper_recovering">
          <Grid>
            <Grid.Background>
              <ImageBrush ImageSource="CopperRecovering_Bo.png"/>
            </Grid.Background>
            <Ellipse x:Name="LEDHLvTR" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="29" Margin="277,193,0,0" Stroke="Black"
VerticalAlignment="Top" Width="24" AutomationProperties.Name="HLvTR_Green"/>
            <Ellipse x:Name="LEDLLvTR" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="28" Margin="274,278,0,0" Stroke="Black"
VerticalAlignment="Top" Width="26" AutomationProperties.Name="LLvTR_Green"/>
            <Ellipse x:Name="PR" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="52" Margin="109,297,0,0" Stroke="Black"
VerticalAlignment="Top" Width="51" AutomationProperties.Name="PR"/>
```

```
            <Ellipse x:Name="PL6" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="52" Margin="310,44,0,0" Stroke="Black"
VerticalAlignment="Top" Width="51" AutomationProperties.Name="PL6"/>
            <Ellipse x:Name="PL2_CR" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="52" Margin="591,126,0,0" Stroke="Black"
VerticalAlignment="Top" Width="51" AutomationProperties.Name="PL2_CR"/>
            <Label Name="labelCTR" Content="Waiting..." TabIndex="1"
HorizontalAlignment="Left" VerticalAlignment="Top" Margin="617,302,0,0" Height="Auto"
Width="Auto"/>
            <Label Name="labelIT" Content="Waiting..." TabIndex="1"
HorizontalAlignment="Left" VerticalAlignment="Top" Margin="617,216,0,0" Height="Auto"
Width="Auto"/>
            <Rectangle x:Name="Color_Recovering" Fill="White"
HorizontalAlignment="Left" Height="58" Margin="617,333,0,0" Stroke="Black"
VerticalAlignment="Top" Width="58"/>
            <TextBox Name="Alarma_Recovering" HorizontalAlignment="Left"
Height="40" Margin="736,405,0,0" TextWrapping="Wrap" VerticalAlignment="Top"
Width="53" BorderBrush="White" Foreground="White" Visibility="Collapsed">
              <TextBox.Background>
                <ImageBrush ImageSource="warning.png"/>
              </TextBox.Background>
            </TextBox>
          </Grid>
        </TabItem>
        <TabItem Header="Tank 1" Name="Tank1">
          <Grid>
            <Grid.Background>
              <ImageBrush ImageSource="Tank1_Boo.png"/>
            </Grid.Background>
            <Ellipse x:Name="LEDLLvTFT" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="61" Margin="465,198,0,0" Stroke="Black"
VerticalAlignment="Top" Width="58" AutomationProperties.Name="LLvTFT_Green"/>
            <Ellipse x:Name="LEDHLvTFT" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="61" Margin="465,41,0,0" Stroke="Black"
VerticalAlignment="Top" Width="58" AutomationProperties.Name="HLvTFT_Green"/>
            <Ellipse x:Name="PFT" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="61" Margin="373,360,0,0" Stroke="Black"
VerticalAlignment="Top" Width="57" AutomationProperties.Name="PFT"/>
            <Ellipse x:Name="PB3_T1" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="61" Margin="132,159,0,0" Stroke="Black"
VerticalAlignment="Top" Width="57" AutomationProperties.Name="PB3_T1"/>
```

```
            <Ellipse x:Name="PL1_T1" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="61" Margin="580,114,0,0" Stroke="Black"
VerticalAlignment="Top" Width="57" AutomationProperties.Name="PL1_T1"/>
            <TextBox Name="Alarma_Tank1" HorizontalAlignment="Left" Height="40"
Margin="736,405,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="53"
BorderBrush="White" Foreground="White" Visibility="Collapsed">
              <TextBox.Background>
                <ImageBrush ImageSource="warning.png"/>
              </TextBox.Background>
            </TextBox>
          </Grid>
        </TabItem>
        <TabItem Header="Tank 2" Name="Tank2">
          <Grid>
            <Grid.Background>
              <ImageBrush ImageSource="Tank2_Booo.png"/>
            </Grid.Background>
            <Ellipse x:Name="LEDLLvTST" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="40" Margin="319,217,0,0" Stroke="Black"
VerticalAlignment="Top" Width="36" AutomationProperties.Name="LLvTST_Green"/>
            <Ellipse x:Name="LEDHLvTST" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="40" Margin="319,159,0,0" Stroke="Black"
VerticalAlignment="Top" Width="36" AutomationProperties.Name="HLvTST_Green"/>
            <Ellipse x:Name="PST1" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="40" Margin="533,349,0,0" Stroke="Black"
VerticalAlignment="Top" Width="35" AutomationProperties.Name="PST1"/>
            <Ellipse x:Name="PST2" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="42" Margin="149,135,0,0" Stroke="Black"
VerticalAlignment="Top" Width="36" AutomationProperties.Name="PST2"/>
            <Ellipse x:Name="PST3" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="41" Margin="275,376,0,0" Stroke="Black"
VerticalAlignment="Top" Width="35" AutomationProperties.Name="PST3"/>
            <Ellipse x:Name="PFT_T2" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="41" Margin="356,43,0,0" Stroke="Black"
VerticalAlignment="Top" Width="35" AutomationProperties.Name="PFT_T2"/>
            <Ellipse x:Name="PR_T2" Fill="#FF10F110" Visibility="Collapsed"
HorizontalAlignment="Left" Height="41" Margin="562,82,0,0" Stroke="Black"
VerticalAlignment="Top" Width="35" AutomationProperties.Name="PR_T2"/>
            <TextBox Name="Alarma_Tank2" HorizontalAlignment="Left" Height="40"
Margin="736,405,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="53"
BorderBrush="White" Foreground="White" Visibility="Collapsed">
```

```
                <TextBox.Background>
                    <ImageBrush ImageSource="warning.png"/>
                </TextBox.Background>
            </TextBox>
        </Grid>
    </TabItem>
    <TabItem Header="Data" Name="Data">
        <Grid Background="#FFFFFDFD"
ScrollViewer.HorizontalScrollBarVisibility="Visible"
ScrollViewer.VerticalScrollBarVisibility="Disabled">
            <TextBox Name="Alarms_TextBox" HorizontalAlignment="Left" Height="35"
Margin="127,410,0,0" TextWrapping="Wrap" Text="NO ALARMS DECLARED"
VerticalAlignment="Top" Width="662" Background="Red" BorderBrush="Black"
TextAlignment="Center" HorizontalContentAlignment="Center"
VerticalContentAlignment="Center" FontFamily="Arial" FontSize="18" FontWeight="Bold"
MaxLines="1" IsReadOnly="True" VerticalScrollBarVisibility="Visible"/>
            <TextBox HorizontalAlignment="Left" Height="40" Margin="69,405,0,0"
TextWrapping="Wrap" VerticalAlignment="Top" Width="53" BorderBrush="White"
Foreground="White">
                <TextBox.Background>
                    <ImageBrush ImageSource="warning.png"/>
                </TextBox.Background>
            </TextBox>
            <ListView Name ="Llista_temps" HorizontalAlignment="Left" Height="338"
Margin="410,54,0,0" VerticalAlignment="Top" Width="379"
ScrollViewer.VerticalScrollBarVisibility="Visible" FontFamily="Arial"
ScrollViewer.HorizontalScrollBarVisibility="Disabled">
                <ListView.View>
                    <GridView>
                        <GridViewColumn Header="Date and Hour" Width="150"
DisplayMemberBinding="{Binding Date_and_Hour}" />
                        <GridViewColumn Header="Pump" Width="80"
DisplayMemberBinding="{Binding Pump}" />
                        <GridViewColumn Header="Time working" Width="125"
DisplayMemberBinding="{Binding Time_working}" />
                    </GridView>
                </ListView.View>
            </ListView>
            <Rectangle Fill="#FF0C2285" HorizontalAlignment="Left" Height="34"
Margin="410,15,0,0" Stroke="White" VerticalAlignment="Top" Width="379"/>
```

```
            <Rectangle Fill="#FF0C2285" HorizontalAlignment="Left" Height="34"
Margin="10,15,0,0" Stroke="White" VerticalAlignment="Top" Width="379"/>
            <Label Content="SENSORS VALUES" HorizontalAlignment="Left"
Margin="10,15,0,0" VerticalAlignment="Top" Height="34" Width="379"
Foreground="White" HorizontalContentAlignment="Center"
VerticalContentAlignment="Center" FontFamily="Arial" FontWeight="Bold"
FontSize="20"/>
            <Label Content="WATER PUMPS WORKING TIME "
HorizontalAlignment="Left" Margin="410,15,0,0" VerticalAlignment="Top" Height="34"
Width="379" Foreground="White" HorizontalContentAlignment="Center"
VerticalContentAlignment="Center" FontFamily="Arial" FontWeight="Bold"
FontSize="20"/>
            <TabControl HorizontalAlignment="Left" Height="338" Margin="10,54,0,0"
VerticalAlignment="Top" Width="379">
              <TabItem Header="PH B">
                <Grid Background="#FFE5E5E5">
                  <ListView x:Name ="Llista_PHB" HorizontalAlignment="Left"
Height="310" VerticalAlignment="Top" Width="373"
ScrollViewer.VerticalScrollBarVisibility="Visible" FontFamily="Arial"
ScrollViewer.HorizontalScrollBarVisibility="Disabled">
                    <ListView.View>
                      <GridView>
                        <GridViewColumn Header="Date and Hour" Width="170"
DisplayMemberBinding="{Binding Date_and_Hour}" />
                        <GridViewColumn Header="Sensor Value" Width="170"
DisplayMemberBinding="{Binding Sensor_Value}" />
                      </GridView>
                    </ListView.View>
                  </ListView>
                </Grid>
              </TabItem>
              <TabItem Header="PH L">
                <Grid Background="#FFE5E5E5">
                  <ListView x:Name ="Llista_PHL" HorizontalAlignment="Left"
Height="310" VerticalAlignment="Top" Width="373"
ScrollViewer.VerticalScrollBarVisibility="Visible" FontFamily="Arial"
ScrollViewer.HorizontalScrollBarVisibility="Disabled">
                    <ListView.View>
                      <GridView>
                        <GridViewColumn Header="Date and Hour" Width="170"
DisplayMemberBinding="{Binding Date_and_Hour}" />
```

```
                        <GridViewColumn Header="Sensor Value" Width="170"
DisplayMemberBinding="{Binding Sensor_Value}" />
                      </GridView>
                    </ListView.View>
                  </ListView>
                </Grid>
              </TabItem>
              <TabItem Header="REDOX B">
                <Grid Background="#FFE5E5E5">
                  <ListView x:Name ="Llista_REDOX" HorizontalAlignment="Left"
Height="310" VerticalAlignment="Top" Width="373"
ScrollViewer.VerticalScrollBarVisibility="Visible" FontFamily="Arial"
ScrollViewer.HorizontalScrollBarVisibility="Disabled">
                    <ListView.View>
                      <GridView>
                        <GridViewColumn Header="Date and Hour" Width="170"
DisplayMemberBinding="{Binding Date_and_Hour}" />
                        <GridViewColumn Header="Sensor Value" Width="170"
DisplayMemberBinding="{Binding Sensor_Value}" />
                      </GridView>
                    </ListView.View>
                  </ListView>
                </Grid>
              </TabItem>
              <TabItem Header="CURRENT CR">
                <Grid Background="#FFE5E5E5">
                  <ListView x:Name ="Llista_CURRENT" HorizontalAlignment="Left"
Height="310" VerticalAlignment="Top" Width="373"
ScrollViewer.VerticalScrollBarVisibility="Visible" FontFamily="Arial"
ScrollViewer.HorizontalScrollBarVisibility="Disabled">
                    <ListView.View>
                      <GridView>
                        <GridViewColumn Header="Date and Hour" Width="170"
DisplayMemberBinding="{Binding Date_and_Hour}" />
                        <GridViewColumn Header="Sensor Value [mA]" Width="170"
DisplayMemberBinding="{Binding Sensor_Value}" />
                      </GridView>
                    </ListView.View>
                  </ListView>
                </Grid>
              </TabItem>
```

```
                <TabItem Header="TEMPERATURE B">
                  <Grid Background="#FFE5E5E5">
                    <ListView x:Name ="Llista_TEMPERATURE" HorizontalAlignment="Left" Height="310" VerticalAlignment="Top" Width="373" ScrollViewer.VerticalScrollBarVisibility="Visible" FontFamily="Arial" ScrollViewer.HorizontalScrollBarVisibility="Disabled">
                      <ListView.View>
                        <GridView>
                          <GridViewColumn Header="Date and Hour" Width="170" DisplayMemberBinding="{Binding Date_and_Hour}" />
                          <GridViewColumn Header="Sensor Value [°C]" Width="170" DisplayMemberBinding="{Binding Sensor_Value}" />
                        </GridView>
                      </ListView.View>
                    </ListView>
                  </Grid>
                </TabItem>
              </TabControl>

            </Grid>
          </TabItem>
        </TabControl>
        <DockPanel HorizontalAlignment="Left" Height="35" LastChildFill="False" VerticalAlignment="Top" Width="805" Background="#FF0C2285"/>
        <Button Content="Leaching" Click="Leaching_Click" HorizontalAlignment="Left" Margin="154,0,0,0" VerticalAlignment="Top" Width="85" Height="35" Background="#FF0C2285" Foreground="White" FontFamily="Arial" BorderBrush="#FF0C2285"/>
        <Button Content="Data" Click="Data_Click" HorizontalAlignment="Left" Margin="690,0,0,0" VerticalAlignment="Top" Width="85" Height="35" Background="#FF0C2285" Foreground="White" FontFamily="Arial" BorderBrush="#FF0C2285"/>
        <Button Content="Tank 1" Click="Tank1_Click" HorizontalAlignment="Left" Margin="515,0,0,0" VerticalAlignment="Top" Width="85" Height="35" Background="#FF0C2285" Foreground="White" FontFamily="Arial" BorderBrush="#FF0C2285"/>
        <Button Content="Tank 2" Click="Tank2_Click" HorizontalAlignment="Left" Margin="605,0,0,0" VerticalAlignment="Top" Width="85" Height="35" Background="#FF0C2285" Foreground="White" FontFamily="Arial" BorderBrush="#FF0C2285"/>
```

```
    <Button Content="Main screen" Click="Mainscreen_Click" HorizontalAlignment="Left"
VerticalAlignment="Top" Width="85" Height="35" Background="#FF0C2285"
Foreground="White" FontFamily="Arial" BorderBrush="#FF0C2285" Margin="37,0,0,0"/>
    <Button Content="Bioreactor" Click="Bioreactor_Click" HorizontalAlignment="Left"
Margin="267,0,0,0" VerticalAlignment="Top" Width="85" Height="35"
Background="#FF0C2285" Foreground="White" FontFamily="Arial"
BorderBrush="#FF0C2285"/>
    <Button Content="Cooper recovering" Click="Cooperrecovering_Click"
HorizontalAlignment="Left" Margin="384,0,0,0" VerticalAlignment="Top" Width="110"
Height="35" Background="#FF0C2285" Foreground="White" FontFamily="Arial"
BorderBrush="#FF0C2285"/>
    <Button Name="Power_button" Click="Power_button_Click" Content=""
HorizontalAlignment="Left" Margin="10,438,0,0" VerticalAlignment="Top" Width="40"
Height="35" BorderBrush="#FFFDFDFD">
      <Button.Background>
        <ImageBrush ImageSource="power-icon.png"/>
      </Button.Background>
    </Button>
  </Grid>
</Window>
```

8.2. CODI C# (PART PROGRAMACIÓ)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Timers;
using System.Threading;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Threading;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Data.SqlClient;
using System.Data.OleDb;
using System.IO;
using System.Xml.XPath;
using System.Xml;
using Planta_Biolixiviació.Properties;
using System.IO.Ports;
using System.Diagnostics;
using System.Globalization;
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Security;
using Microsoft.VisualBasic;
```

```
namespace Planta_Biolixiviació
{
   /// <summary>
   /// Lógica de interacción para MainWindow.xaml
   /// </summary>
   public partial class MainWindow : Window
   {

      private System.IO.Ports.SerialPort serialPort1;
      DispatcherTimer TimerSerial = new DispatcherTimer();
      bool isConnected = false;
      bool portp = false;
      String[] ports;
      private string StrParse, LDR_3, LDR_4, LDR_5, LDR_6, LDR_7, LDR_8, LDR_9,
LDR_10, LDR_11, LDR_12, LDR_13, LDR_14, LDR_15, LDR_16, LDR_17, LDR_18,
LDR_19, LDR_20, LDR_21, LDR_22, LDR_23, LDR_24, LDR_25, LDR_26, LDR_27,
LDR_28, LDR_29, LDR_30, LDR_31, LDR_32, LDR_33, LDR_34;
      private string ValorPHL, ValorCL, ValorCR, ValorPHB, ValorITR, ValorRB, ValorTB;
      private int ValorPHL_integer, ValorPHB_integer;
      private int PHL_Comptador, PHB_Comptador, HLVTB_Comptador, r;
      private string TempsDeLlista, s;
      private Int32 ValorCLInt, ValorCRInt;
      TimeSpan temps_transcorregut;
      DateTime tempsInicial = DateTime.MinValue;
      private int Limit = 10;
      private string FilePathAndName;
      DispatcherTimer Timer = new DispatcherTimer();
      DispatcherTimer TimerVariables = new DispatcherTimer();

      public MainWindow()
      {
         InitializeComponent();
         Timer.Tick += new EventHandler(Timer_Click);
         TimerVariables.Tick += new EventHandler(Variables_Click);
         Timer.Interval = new TimeSpan(0, 0, 1);
         TimerVariables.Interval = new TimeSpan(0, 0, 20);
         Timer.Start();
         TimerVariables.Start();

      }
```

```
        private void Timer_Click(object sender, EventArgs e)
        {
            DateTime d;
            d = DateTime.Now;
            Data_label.Content = d;
        }

        private void Variables_Click(object sender, EventArgs e)
        {
            List<TempsPHB> items_tempsPHB = new List<TempsPHB>();
            Llista_PHB.ItemsSource = items_tempsPHB;

            List<TempsPHL> items_tempsPHL = new List<TempsPHL>();
            Llista_PHL.ItemsSource = items_tempsPHL;

            List<TempsREDOX> items_tempsREDOX = new List<TempsREDOX>();
            Llista_REDOX.ItemsSource = items_tempsREDOX;

            List<TempsCURRENT> items_tempsCURRENT = new List<TempsCURRENT>();
            Llista_CURRENT.ItemsSource = items_tempsCURRENT;

            List<TempsTEMPERATURE> items_tempsTEMPERATURE = new
List<TempsTEMPERATURE>();
            Llista_TEMPERATURE.ItemsSource = items_tempsTEMPERATURE;

            items_tempsPHB.Add(new TempsPHB() { Date_and_Hour =
DateTime.Now.ToString(), Sensor_Value = ValorPHB });
            items_tempsPHL.Add(new TempsPHL() { Date_and_Hour =
DateTime.Now.ToString(), Sensor_Value = ValorPHL });
            items_tempsREDOX.Add(new TempsREDOX() { Date_and_Hour =
DateTime.Now.ToString(), Sensor_Value = ValorRB });
            items_tempsCURRENT.Add(new TempsCURRENT() { Date_and_Hour =
DateTime.Now.ToString(), Sensor_Value = ValorITR });
            items_tempsTEMPERATURE.Add(new TempsTEMPERATURE() {
Date_and_Hour = DateTime.Now.ToString(), Sensor_Value = ValorTB });
        }

        void getAvailableComPorts()
        {
            ports = SerialPort.GetPortNames();
        }
```

```
public class TempsBombes
{
    public string Date_and_Hour { get; set; }

    public string Pump { get; set; }

    public string Time_working { get; set; }

}

public class TempsPHB
{
    public string Date_and_Hour { get; set; }

    public string Sensor_Value { get; set; }

}

public class TempsPHL
{
    public string Date_and_Hour { get; set; }

    public string Sensor_Value { get; set; }

}

public class TempsREDOX
{
    public string Date_and_Hour { get; set; }

    public string Sensor_Value { get; set; }

}
```

```
public class TempsCURRENT
    {
        public string Date_and_Hour { get; set; }

        public string Sensor_Value { get; set; }

    }

    public class TempsTEMPERATURE
    {
        public string Date_and_Hour { get; set; }

        public string Sensor_Value { get; set; }

    }

    private void Mainscreen_Click(object sender, RoutedEventArgs e)
    {
        TabControlPrincipal.SelectedItem = Main_screen;
    }

    private void Leaching_Click(object sender, RoutedEventArgs e)
    {
        TabControlPrincipal.SelectedItem = Leaching;

    }

    private void Bioreactor_Click(object sender, RoutedEventArgs e)
    {
        TabControlPrincipal.SelectedItem = Bioreactor;

    }

    private void Cooperrecovering_Click(object sender, RoutedEventArgs e)
    {
        TabControlPrincipal.SelectedItem = Cooper_recovering;
    }
```

```
        private void Tank1_Click(object sender, RoutedEventArgs e)
        {
            TabControlPrincipal.SelectedItem = Tank1;
        }

        private void Tank2_Click(object sender, RoutedEventArgs e)
        {
            TabControlPrincipal.SelectedItem = Tank2;

        }

        private void Data_Click(object sender, RoutedEventArgs e)
        {
            TabControlPrincipal.SelectedItem = Data;

        }

        private void Power_button_Click(object sender, RoutedEventArgs e)
        {
            if (isConnected)
            {
                MessageBox.Show("Please, click the Disconnect button before exit",
"Information");
            }
            else
            {
                this.Close();
            }
        }

        private void MainWindow_Load(object sender, EventArgs e)
        {
            WindowStartupLocation =
System.Windows.WindowStartupLocation.CenterScreen;
            Disconnect.IsEnabled = false;
            Connect.IsEnabled = false;
        }
```

```
private void ScanPort_Click(object sender, EventArgs e)
{
   {
      ComboBoxPort.Items.Clear();
      getAvailableComPorts();

      foreach (string port in ports)
      {
         ComboBoxPort.Items.Add(port);
         Console.WriteLine(port);
         if (ports[0] != null)
         {
            ComboBoxPort.SelectedItem = ports[0];
         }
      }
      portp = true;
   }
}

private void Connect_Click(object sender, EventArgs e)
{
   if (!isConnected && portp)
   {
      isConnected = true;
      serialPort1 = new System.IO.Ports.SerialPort();
      serialPort1.BaudRate = 115200;
      serialPort1.PortName = ComboBoxPort.SelectedItem.ToString();
      serialPort1.Open();
      TimerSerial.Interval = TimeSpan.FromMilliseconds(1);
      TimerSerial.Tick += TimerSerial_Tick_1;
      TimerSerial.Start();
      ComboBoxPort.IsEnabled = false;
      Scan_Port.IsEnabled = false;
      Connect.IsEnabled = false;
      Disconnect.IsEnabled = true;

      Red_disconnected.Visibility = Visibility.Collapsed;
      Green_connected.Visibility = Visibility.Visible;
      Status_label.Content = "Status: Connected";
   }
}
```

```
        private void Disconnect_Click(object sender, EventArgs e)
        {
            if (isConnected && portp)
            {
                isConnected = false;
                Red_disconnected.Visibility = Visibility.Visible;
                Green_connected.Visibility = Visibility.Collapsed;
                Status_label.Content = "Status : Disconnected";

                ComboBoxPort.IsEnabled = true;
                Scan_Port.IsEnabled = true;
                Connect.IsEnabled = true;
                Disconnect.IsEnabled = false;

                TimerSerial.Stop();

                serialPort1.Close();
            }
        }

        private static string StrSerialInLINES(string StrSerialInLine, int lineWanted)
        {
            var lines = StrSerialInLine.Split('\n');
            return lines[lineWanted];
        }

        private void TimerSerial_Tick_1(object sender, EventArgs e)
        {
            try
            {
                string StrSerialIn = serialPort1.ReadExisting();
                string StrSerialInLine = String.Join(Environment.NewLine, StrSerialIn.Split(new[]
{ ' ' }, StringSplitOptions.RemoveEmptyEntries));
                string StrSerialInRam;
                string StrSerialInLINES1;
```

```
StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 0);
StrSerialInRam = StrSerialInLINES1.Substring(0, 2);
if (StrSerialInRam == "L3")
{
   LDR_3 = StrSerialInLINES(StrSerialInLine, 0);
   if (LDR_3.Substring(2, 1) == "1")
   {
      LEDLLvTB.Visibility = Visibility.Visible;
   }
   else
      LEDLLvTB.Visibility = Visibility.Collapsed;
}
else
   LDR_3 = LDR_3;

StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 1);
StrSerialInRam = StrSerialInLINES1.Substring(0, 2);
if (StrSerialInRam == "L4")
{
   LDR_4 = StrSerialInLINES(StrSerialInLine, 1);
   if (LDR_4.Substring(2, 1) == "1")
   {
      LEDHLvTB.Visibility = Visibility.Visible;
   }
   else
      LEDHLvTB.Visibility = Visibility.Collapsed;
}
else
   LDR_4 = LDR_4;
```

```
StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 2);
StrSerialInRam = StrSerialInLINES1.Substring(0, 2);
if (StrSerialInRam == "L5")
{
   LDR_5 = StrSerialInLINES(StrSerialInLine, 2);
   if (LDR_5.Substring(2, 1) == "1")
   {

      LEDLLvTL.Visibility = Visibility.Visible;
   }
   else
      LEDLLvTL.Visibility = Visibility.Collapsed;
}
else
   LDR_5 = LDR_5;

StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 3);
StrSerialInRam = StrSerialInLINES1.Substring(0, 2);
if (StrSerialInRam == "L6")
{
   LDR_6 = StrSerialInLINES(StrSerialInLine, 3);
   if (LDR_6.Substring(2, 1) == "1")
   {
      LEDLLvTR.Visibility = Visibility.Visible;
   }
   else
      LEDLLvTR.Visibility = Visibility.Collapsed;

}
else
   LDR_6 = LDR_6;
```

```
StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 4);
StrSerialInRam = StrSerialInLINES1.Substring(0, 2);
if (StrSerialInRam == "L7")
{
   LDR_7 = StrSerialInLINES(StrSerialInLine, 4);
   if (LDR_7.Substring(2, 1) == "1")
   {
      LEDHLvTFT.Visibility = Visibility.Visible;
   }
   else
      LEDHLvTFT.Visibility = Visibility.Collapsed;
}
else
   LDR_7 = LDR_7;

StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 5);
StrSerialInRam = StrSerialInLINES1.Substring(0, 2);
if (StrSerialInRam == "L8")
{
   LDR_8 = StrSerialInLINES(StrSerialInLine, 5);
   if (LDR_8.Substring(2, 1) == "1")
   {
      LEDLLvTFT.Visibility = Visibility.Visible;
   }
   else
      LEDLLvTFT.Visibility = Visibility.Collapsed;
}
else
   LDR_8 = LDR_8;
```

```
StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 6);
StrSerialInRam = StrSerialInLINES1.Substring(0, 2);
if (StrSerialInRam == "L9")
{
   LDR_9 = StrSerialInLINES(StrSerialInLine, 6);
   if (LDR_9.Substring(2, 1) == "1")
   {
      LEDHLvTST.Visibility = Visibility.Visible;

   }
   else
      LEDHLvTST.Visibility = Visibility.Collapsed;

}
else
   LDR_9 = LDR_9;

StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 7);
StrSerialInRam = StrSerialInLINES1.Substring(0, 3);
if (StrSerialInRam == "L10")
{
   LDR_10 = StrSerialInLINES(StrSerialInLine, 7);
   if (LDR_10.Substring(3, 1) == "1")
   {
      LEDLLvTST.Visibility = Visibility.Visible;
   }
   else
      LEDLLvTST.Visibility = Visibility.Collapsed;

}
else
   LDR_10 = LDR_10;
```

```
StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 8);
StrSerialInRam = StrSerialInLINES1.Substring(0, 3);
if (StrSerialInRam == "L11")
{
   LDR_11 = StrSerialInLINES(StrSerialInLine, 8);
   if (LDR_11.Substring(3, 1) == "1")
   {
      LEDHLvTL.Visibility = Visibility.Visible;
   }
   else
      LEDHLvTL.Visibility = Visibility.Collapsed;

}
else
   LDR_11 = LDR_11;

StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 9);
StrSerialInRam = StrSerialInLINES1.Substring(0, 3);
if (StrSerialInRam == "L12")
{
   LDR_12 = StrSerialInLINES(StrSerialInLine, 9);
   if (LDR_12.Substring(3, 1) == "1")
   {
      LEDHLvTR.Visibility = Visibility.Visible;
   }
   else
      LEDHLvTR.Visibility = Visibility.Collapsed;

}
else
   LDR_12 = LDR_12;
```

```
StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 10);
StrSerialInRam = StrSerialInLINES1.Substring(0, 3);
if (StrSerialInRam == "L13")
{
   LDR_13 = StrSerialInLINES(StrSerialInLine, 10);
   if (LDR_13.Substring(3, 1) == "1")
   {
      PB1.Visibility = Visibility.Visible;
   }
   else
      PB1.Visibility = Visibility.Collapsed;


}
else
   LDR_13 = LDR_13;

StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 11);
StrSerialInRam = StrSerialInLINES1.Substring(0, 3);
if (StrSerialInRam == "L14")
{
   LDR_14 = StrSerialInLINES(StrSerialInLine, 11);
   if (LDR_14.Substring(3, 1) == "1")
   {
      PB2.Visibility = Visibility.Visible;
   }
   else
      PB2.Visibility = Visibility.Collapsed;


}
else
   LDR_14 = LDR_14;
```

```
StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 12);
StrSerialInRam = StrSerialInLINES1.Substring(0, 3);
if (StrSerialInRam == "L15")
{
   LDR_15 = StrSerialInLINES(StrSerialInLine, 12);
   if (LDR_15.Substring(3, 1) == "1")
   {
      PB3.Visibility = Visibility.Visible;
      PB3_T1.Visibility = Visibility.Visible;
   }
   else
      PB3.Visibility = Visibility.Collapsed;
      PB3_T1.Visibility = Visibility.Collapsed;

}
else
   LDR_15 = LDR_15;

StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 13);
StrSerialInRam = StrSerialInLINES1.Substring(0, 3);
if (StrSerialInRam == "L16")
{
   LDR_16 = StrSerialInLINES(StrSerialInLine, 13);
   if (LDR_16.Substring(3, 1) == "1")
   {
      PL1.Visibility = Visibility.Visible;
      PL1_T1.Visibility = Visibility.Visible;
   }
   else
      PL1.Visibility = Visibility.Collapsed;
      PL1_T1.Visibility = Visibility.Collapsed;


}
else
   LDR_16 = LDR_16;
```

```
StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 14);
StrSerialInRam = StrSerialInLINES1.Substring(0, 3);
if (StrSerialInRam == "L17")
{
   LDR_17 = StrSerialInLINES(StrSerialInLine, 14);
   if (LDR_17.Substring(3, 1) == "1")
   {
      PL2.Visibility = Visibility.Visible;
      PL2_CR.Visibility = Visibility.Visible;
   }
   else
      PL2.Visibility = Visibility.Collapsed;
      PL2_CR.Visibility = Visibility.Collapsed;


}
else
   LDR_17 = LDR_17;

StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 15);
StrSerialInRam = StrSerialInLINES1.Substring(0, 3);
if (StrSerialInRam == "L18")
{
   LDR_18 = StrSerialInLINES(StrSerialInLine, 15);
   if (LDR_18.Substring(3, 1) == "1")
   {
      PL3.Visibility = Visibility.Visible;
   }
   else
      PL3.Visibility = Visibility.Collapsed;

}
else
   LDR_18 = LDR_18;
```

```
StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 16);
StrSerialInRam = StrSerialInLINES1.Substring(0, 3);
if (StrSerialInRam == "L19")
{
   LDR_19 = StrSerialInLINES(StrSerialInLine, 16);
   if (LDR_19.Substring(3, 1) == "1")
   {
      PL4.Visibility = Visibility.Visible;
   }
   else
      PL4.Visibility = Visibility.Collapsed;

}
else
   LDR_19 = LDR_19;

StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 17);
StrSerialInRam = StrSerialInLINES1.Substring(0, 3);
if (StrSerialInRam == "L20")
{
   LDR_20 = StrSerialInLINES(StrSerialInLine, 17);
   if (LDR_20.Substring(3, 1) == "1")
   {
      PL5.Visibility = Visibility.Visible;
   }
   else
      PL5.Visibility = Visibility.Collapsed;


}
else
   LDR_20 = LDR_20;
```

```
StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 18);
StrSerialInRam = StrSerialInLINES1.Substring(0, 3);
if (StrSerialInRam == "L21")
{
   LDR_21 = StrSerialInLINES(StrSerialInLine, 18);
   if (LDR_21.Substring(3, 1) == "1")
   {
      PR.Visibility = Visibility.Visible;
      PR_T2.Visibility = Visibility.Visible;
   }
   else
      PR.Visibility = Visibility.Collapsed;
      PR_T2.Visibility = Visibility.Collapsed;


}
else
   LDR_21 = LDR_21;

StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 19);
StrSerialInRam = StrSerialInLINES1.Substring(0, 3);
if (StrSerialInRam == "L22")
{
   LDR_22 = StrSerialInLINES(StrSerialInLine, 19);
   if (LDR_22.Substring(3, 1) == "1")
   {
      PFT.Visibility = Visibility.Visible;
      PFT_T2.Visibility = Visibility.Visible;
   }
   else
      PFT.Visibility = Visibility.Collapsed;
      PFT_T2.Visibility = Visibility.Collapsed;
}
else
   LDR_22 = LDR_22;
```

```
StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 20);
StrSerialInRam = StrSerialInLINES1.Substring(0, 3);
if (StrSerialInRam == "L23")
{
   LDR_23 = StrSerialInLINES(StrSerialInLine, 20);
   if (LDR_23.Substring(3, 1) == "1")
   {
      PST1.Visibility = Visibility.Visible;
   }
   else
      PST1.Visibility = Visibility.Collapsed;

}
else
   LDR_23 = LDR_23;

StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 21);
StrSerialInRam = StrSerialInLINES1.Substring(0, 3);
if (StrSerialInRam == "L24")
{
   LDR_24 = StrSerialInLINES(StrSerialInLine, 21);
   if (LDR_24.Substring(3, 1) == "1")
   {
      PST2.Visibility = Visibility.Visible;
      PST2_Bioreactor.Visibility = Visibility.Visible;
   }
   else
      PST2.Visibility = Visibility.Collapsed;
      PST2_Bioreactor.Visibility = Visibility.Collapsed;


}
else
   LDR_24 = LDR_24;
```

```
StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 22);
StrSerialInRam = StrSerialInLINES1.Substring(0, 3);
if (StrSerialInRam == "L25")
{
   LDR_25 = StrSerialInLINES(StrSerialInLine, 22);
   if (LDR_25.Substring(3, 1) == "1")
   {

      PST3.Visibility = Visibility.Visible;
   }
   else
      PST3.Visibility = Visibility.Collapsed;

}
else
   LDR_25 = LDR_25;

StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 23);
StrSerialInRam = StrSerialInLINES1.Substring(0, 3);
if (StrSerialInRam == "L26")
{
   LDR_26 = StrSerialInLINES(StrSerialInLine, 23);
   if (LDR_26.Substring(3, 1) == "1")
   {

      PL6.Visibility = Visibility.Visible;
   }
   else
      PL6.Visibility = Visibility.Collapsed;

}
else
   LDR_26 = LDR_26;
```

```
StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 24);
StrSerialInRam = StrSerialInLINES1.Substring(0, 3);
if (StrSerialInRam == "L27")
{
   LDR_27 = StrSerialInLINES(StrSerialInLine, 24);
   if (LDR_27.Substring(3, 1) == "1")
   {

      M.Visibility = Visibility.Visible;
   }
   else
      M.Visibility = Visibility.Collapsed;

}
else
   LDR_27 = LDR_27;

StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 25);
StrSerialInRam = StrSerialInLINES1.Substring(0, 3);
if (StrSerialInRam == "L28")
{
   LDR_28 = StrSerialInLINES(StrSerialInLine, 25);
   ValorPHL = LDR_28.Substring(3);

}
else
   LDR_28 = LDR_28;

StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 26);
StrSerialInRam = StrSerialInLINES1.Substring(0, 3);
if (StrSerialInRam == "L29")
{
   LDR_29 = StrSerialInLINES(StrSerialInLine, 26);
   ValorPHB = LDR_29.Substring(3);

}
else
   LDR_29 = LDR_29;
```

```
StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 27);
StrSerialInRam = StrSerialInLINES1.Substring(0, 3);
if (StrSerialInRam == "L30")
{
   LDR_30 = StrSerialInLINES(StrSerialInLine, 27);
   ValorCL = LDR_30.Substring(3);

}
else
   LDR_30 = LDR_30;

StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 28);
StrSerialInRam = StrSerialInLINES1.Substring(0, 3);
if (StrSerialInRam == "L31")
{
   LDR_31 = StrSerialInLINES(StrSerialInLine, 28);
   ValorCR = LDR_31.Substring(3);

}
else
   LDR_31 = LDR_31;

StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 29);
StrSerialInRam = StrSerialInLINES1.Substring(0, 3);
if (StrSerialInRam == "L32")
{
   LDR_32 = StrSerialInLINES(StrSerialInLine, 29);
   ValorITR = LDR_32.Substring(3);

}
else
   LDR_32 = LDR_32;
```

```
            StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 30);
            StrSerialInRam = StrSerialInLINES1.Substring(0, 3);
            if (StrSerialInRam == "L33")
            {
               LDR_33 = StrSerialInLINES(StrSerialInLine, 30);
               ValorRB = LDR_33.Substring(3);

            }
            else
               LDR_33 = LDR_33;

            StrSerialInLINES1 = StrSerialInLINES(StrSerialInLine, 31);
            StrSerialInRam = StrSerialInLINES1.Substring(0, 3);
            if (StrSerialInRam == "L34")
            {
               LDR_34 = StrSerialInLINES(StrSerialInLine, 31);
               ValorTB = LDR_34.Substring(3);

            }
            else
               LDR_34 = LDR_34;

            labelCTR.Content = "CTR = " + ValorCR;
            labelpHTL.Content = "PHTL = " + ValorPHL;
            labelpHTB.Content = "PHTB = " + ValorPHB;
            labelCTL.Content = "CTL = " + ValorCL;
            labelIT.Content = "ITR = " + ValorITR + " mA";
            labelREDOX.Content = "REDOX = " + ValorRB;
            labelTEMP.Content = "TEMP = " + ValorTB + " °C";

            ValorCLInt = int.Parse(ValorCL);
            var blueValue_CL = ValorCLInt / 65536;
            var greenValue_CL = (ValorCLInt - blueValue_CL * 65536) / 256;
            var redValue_CL = ValorCLInt - blueValue_CL * 65536 - greenValue_CL * 256;
            SolidColorBrush Brush_CL = new
SolidColorBrush(Color.FromRgb(Convert.ToByte(redValue_CL),
Convert.ToByte(greenValue_CL), Convert.ToByte(blueValue_CL)));
            Color_Leaching.Fill = Brush_CL;
```

```
            ValorCRInt = int.Parse(ValorCR);
            var blueValue_CR = ValorCRInt / 65536;
            var greenValue_CR = (ValorCRInt - blueValue_CR * 65536) / 256;
            var redValue_CR = ValorCRInt - blueValue_CR * 65536 - greenValue_CR * 256;
            SolidColorBrush Brush_CR = new
SolidColorBrush(Color.FromRgb(Convert.ToByte(redValue_CR),
Convert.ToByte(greenValue_CR), Convert.ToByte(blueValue_CR)));
            Color_Recovering.Fill = Brush_CR;

            ValorPHL_integer = int.Parse(ValorPHL);
            ValorPHB_integer = int.Parse(ValorPHB);

if ((ValorPHL_integer > 1.8)|(ValorPHL_integer < 1.6))
            {
               for (int i = 0; i < 10; i++)
               {

                  Thread.Sleep(18000);
                  if ((ValorPHL_integer < 1.8) | (ValorPHL_integer > 1.6))
                  {
                     i = 0;
                     PHL_Comptador = 0;
                  }
                  else
                     PHL_Comptador += 1;

               }
            }
            else
               PHL_Comptador = 0;
```

```
if ((ValorPHB_integer > 1.8) | (ValorPHB_integer < 1.6))
{
    for (int i = 0; i < 10; i++)
    {

        Thread.Sleep(18000);
        if ((ValorPHB_integer < 1.8) | (ValorPHB_integer > 1.6))
        {
            i = 0;
            PHB_Comptador = 0;
        }
        else
            PHB_Comptador += 1;

    }
}
else
    PHB_Comptador = 0;

if (LEDHLvTB.Visibility == Visibility.Visible)
{
    for (int i = 0; i < 10; i++)
    {

        Thread.Sleep(18000);
        if (LEDHLvTB.Visibility == Visibility.Collapsed)
        {
            i = 0;
            HLVTB_Comptador = 0;
        }
        else
            HLVTB_Comptador += 1;

    }
}
else
    HLVTB_Comptador = 0;
```

```
            if ((PHL_Comptador == 10) | (PHB_Comptador == 10) | (HLVTB_Comptador ==
10) | (LEDHLvTST.Visibility == Visibility.Visible) | (LEDLLvTFT.Visibility ==
Visibility.Collapsed))
            {
                Alarma_Main.Visibility = Visibility.Visible;
                Alarma_Leaching.Visibility = Visibility.Visible;
                Alarma_Bioreactor.Visibility = Visibility.Visible;
                Alarma_Recovering.Visibility = Visibility.Visible;
                Alarma_Tank1.Visibility = Visibility.Visible;
                Alarma_Tank2.Visibility = Visibility.Visible;
            }
            else
                Alarma_Main.Visibility = Visibility.Collapsed;
                Alarma_Leaching.Visibility = Visibility.Collapsed;
                Alarma_Bioreactor.Visibility = Visibility.Collapsed;
                Alarma_Recovering.Visibility = Visibility.Collapsed;
                Alarma_Tank1.Visibility = Visibility.Collapsed;
                Alarma_Tank2.Visibility = Visibility.Collapsed;
                Alarms_TextBox.Text = "NO ALARMS DECLARED";

            if (PHL_Comptador == 10)
            {
                Alarms_TextBox.Text = "LEACHING PH SENSOR HAS BEEN BELOW 1.6 OR
ABOVE 1.8 FOR MORE THAN 10 SAMPLES";
            }

            if (PHB_Comptador == 10)
            {
                Alarms_TextBox.Text = "BIOREACTOR PH SENSOR HAS BEEN BELOW 1.6
OR ABOVE 1.8 FOR MORE THAN 10 SAMPLES";
            }

            if (HLVTB_Comptador == 10)
            {
                Alarms_TextBox.Text = "BIOREACTOR HIGH LEVEL SENSOR HAS BEEN
ON MORE THAN 10 SAMPLES";
            }
```

```
            if (LEDHLvTST.Visibility == Visibility.Visible)
            {
                Alarms_TextBox.Text = "TANK 2 HIGH LEVEL SENSOR IS ON";
            }

            if (LEDLLvTFT.Visibility == Visibility.Collapsed)
            {
                Alarms_TextBox.Text = "TANK 1 LOW LEVEL SENSOR IS OFF";
            }

            if ((PHL_Comptador == 10) & (PHB_Comptador == 10))
            {
                Alarms_TextBox.Text = "LEACHING PH SENSOR AND BIOREACTOR PH
SENSOR HAVE BEEN BELOW 1.6 OR ABOVE 1.8 FOR MORE THAN 10 SAMPLES";
            }

            if ((PHL_Comptador == 10) & (HLVTB_Comptador == 10))
            {
                Alarms_TextBox.Text = "LEACHING PH SENSOR HAS BEEN BELOW 1.6 OR
ABOVE 1.8 FOR MORE THAN 10 SAMPLES AND BIOREACTOR HIGH LEVEL SENSOR
HAS BEEN ON MORE THAN 10 SAMPLES";
            }

            if ((PHL_Comptador == 10) & (LEDHLvTST.Visibility == Visibility.Visible))
            {
                Alarms_TextBox.Text = "LEACHING PH SENSOR HAS BEEN BELOW 1.6 OR
ABOVE 1.8 FOR MORE THAN 10 SAMPLES AND TANK 2 HIGH LEVEL SENSOR IS ON";
            }

            if ((PHL_Comptador == 10) & (LEDLLvTFT.Visibility == Visibility.Collapsed))
            {
                Alarms_TextBox.Text = "LEACHING PH SENSOR HAS BEEN BELOW 1.6 OR
ABOVE 1.8 FOR MORE THAN 10 SAMPLES AND TANK 1 LOW LEVEL SENSOR IS
OFF";
            }
```

```
            if ((PHB_Comptador == 10) & (HLVTB_Comptador == 10))
            {
                Alarms_TextBox.Text = "BIOREACTOR PH SENSOR HAS BEEN BELOW 1.6
OR ABOVE 1.8 FOR MORE THAN 10 SAMPLES AND BIOREACTOR HIGH LEVEL
SENSOR HAS BEEN ON MORE THAN 10 SAMPLES";
            }

            if ((PHB_Comptador == 10) & (LEDHLvTST.Visibility == Visibility.Visible))
            {
                Alarms_TextBox.Text = "BIOREACTOR PH SENSOR HAS BEEN BELOW 1.6
OR ABOVE 1.8 FOR MORE THAN 10 SAMPLES AND TANK 2 HIGH LEVEL SENSOR IS
ON";
            }

            if ((PHB_Comptador == 10) & (LEDLLvTFT.Visibility == Visibility.Collapsed))
            {
                Alarms_TextBox.Text = "BIOREACTOR PH SENSOR HAS BEEN BELOW 1.6
OR ABOVE 1.8 FOR MORE THAN 10 SAMPLES AND TANK 1 LOW LEVEL SENSOR IS
OFF";
            }

            if ((HLVTB_Comptador == 10) & (LEDHLvTST.Visibility == Visibility.Visible))
            {
                Alarms_TextBox.Text =  "BIOREACTOR HIGH LEVEL SENSOR HAS BEEN
ON MORE THAN 10 SAMPLES AND TANK 2 HIGH LEVEL SENSOR IS ON";
            }

            if ((HLVTB_Comptador == 10) & (LEDLLvTFT.Visibility == Visibility.Collapsed))
            {
                Alarms_TextBox.Text =  "BIOREACTOR HIGH LEVEL SENSOR HAS BEEN
ON MORE THAN 10 SAMPLES AND TANK 1 LOW LEVEL SENSOR IS OFF";
            }

            if ((LEDHLvTST.Visibility == Visibility.Visible) & (LEDLLvTFT.Visibility ==
Visibility.Collapsed))
            {
                Alarms_TextBox.Text = "TANK 2 HIGH LEVEL SENSOR IS ON AND TANK 1
LOW LEVEL SENSOR IS OFF";
            }
```

```
            if ((PHL_Comptador == 10) & (PHB_Comptador == 10) & (HLVTB_Comptador
== 10))
            {
                Alarms_TextBox.Text = "LEACHING PH SENSOR AND BIOREACTOR PH
SENSOR HAVE BEEN BELOW 1.6 OR ABOVE 1.8 FOR MORE THAN 10 SAMPLES AND
BIOREACTOR HIGH LEVEL SENSOR HAS BEEN ON MORE THAN 10 SAMPLES";
            }

            if ((PHL_Comptador == 10) & (PHB_Comptador == 10) & (LEDHLvTST.Visibility
== Visibility.Visible))
            {
                Alarms_TextBox.Text = "LEACHING PH SENSOR AND BIOREACTOR PH
SENSOR HAVE BEEN BELOW 1.6 OR ABOVE 1.8 FOR MORE THAN 10 SAMPLES AND
TANK 2 HIGH LEVEL SENSOR IS ON";
            }

            if ((PHL_Comptador == 10) & (PHB_Comptador == 10) & (LEDLLvTFT.Visibility
== Visibility.Collapsed))
            {
                Alarms_TextBox.Text = "LEACHING PH SENSOR AND BIOREACTOR PH
SENSOR HAVE BEEN BELOW 1.6 OR ABOVE 1.8 FOR MORE THAN 10 SAMPLES AND
TANK 1 LOW LEVEL SENSOR IS OFF";
            }

            if ((PHL_Comptador == 10) & (HLVTB_Comptador == 10) &
(LEDHLvTST.Visibility == Visibility.Visible))
            {
                Alarms_TextBox.Text = "LEACHING PH SENSOR HAS BEEN BELOW 1.6 OR
ABOVE 1.8 FOR MORE THAN 10 SAMPLES, BIOREACTOR HIGH LEVEL SENSOR HAS
BEEN ON MORE THAN 10 SAMPLES AND TANK 2 HIGH LEVEL SENSOR IS ON";
            }

            if ((PHL_Comptador == 10) & (HLVTB_Comptador == 10) &
(LEDLLvTFT.Visibility == Visibility.Collapsed))
            {
                Alarms_TextBox.Text = "LEACHING PH SENSOR HAS BEEN BELOW 1.6 OR
ABOVE 1.8 FOR MORE THAN 10 SAMPLES, BIOREACTOR HIGH LEVEL SENSOR HAS
BEEN ON MORE THAN 10 SAMPLES AND TANK 1 LOW LEVEL SENSOR IS OFF";
            }
```

```
        if ((PHL_Comptador == 10) & (LEDHLvTST.Visibility == Visibility.Visible) &
(LEDLLvTFT.Visibility == Visibility.Collapsed))
        {
            Alarms_TextBox.Text = "LEACHING PH SENSOR HAS BEEN BELOW 1.6 OR
ABOVE 1.8 FOR MORE THAN 10 SAMPLES, TANK 2 HIGH LEVEL SENSOR IS ON AND
TANK 1 LOW LEVEL SENSOR IS OFF";
        }

        if ((PHB_Comptador == 10) & (HLVTB_Comptador == 10) &
(LEDHLvTST.Visibility == Visibility.Visible))
        {
            Alarms_TextBox.Text = "BIOREACTOR PH SENSOR HAS BEEN BELOW 1.6
OR ABOVE 1.8 FOR MORE THAN 10 SAMPLES, BIOREACTOR HIGH LEVEL SENSOR
HAS BEEN ON MORE THAN 10 SAMPLES AND TANK 2 HIGH LEVEL SENSOR IS ON";
        }

        if ((PHB_Comptador == 10) & (HLVTB_Comptador == 10) &
(LEDLLvTFT.Visibility == Visibility.Collapsed))
        {
            Alarms_TextBox.Text = "BIOREACTOR PH SENSOR HAS BEEN BELOW 1.6
OR ABOVE 1.8 FOR MORE THAN 10 SAMPLES, BIOREACTOR HIGH LEVEL SENSOR
HAS BEEN ON MORE THAN 10 SAMPLES AND TANK 1 LOW LEVEL SENSOR IS OFF";
        }

        if ((PHB_Comptador == 10) & (LEDHLvTST.Visibility == Visibility.Visible) &
(LEDLLvTFT.Visibility == Visibility.Collapsed))
        {
            Alarms_TextBox.Text = "BIOREACTOR PH SENSOR HAS BEEN BELOW 1.6
OR ABOVE 1.8 FOR MORE THAN 10 SAMPLES, TANK 2 HIGH LEVEL SENSOR IS ON
AND TANK 1 LOW LEVEL SENSOR IS OFF";
        }

        if ((HLVTB_Comptador == 10) & (LEDHLvTST.Visibility == Visibility.Visible) &
(LEDLLvTFT.Visibility == Visibility.Collapsed))
        {
            Alarms_TextBox.Text = "BIOREACTOR HIGH LEVEL SENSOR HAS BEEN
ON MORE THAN 10 SAMPLES, TANK 2 HIGH LEVEL SENSOR IS ON AND TANK 1 LOW
LEVEL SENSOR IS OFF";
        }
```

```
            if ((PHL_Comptador == 10) & (PHB_Comptador == 10) & (HLVTB_Comptador
== 10) & (LEDHLvTST.Visibility == Visibility.Visible))
            {
                Alarms_TextBox.Text = "LEACHING PH SENSOR AND BIOREACTOR PH
SENSOR HAVE BEEN BELOW 1.6 OR ABOVE 1.8 FOR MORE THAN 10 SAMPLES,
BIOREACTOR HIGH LEVEL SENSOR HAS BEEN ON MORE THAN 10 SAMPLES AND
TANK 2 HIGH LEVEL SENSOR IS ON";
            }

            if ((PHL_Comptador == 10) & (PHB_Comptador == 10) & (HLVTB_Comptador
== 10) & (LEDLLvTFT.Visibility == Visibility.Collapsed))
            {
                Alarms_TextBox.Text = "LEACHING PH SENSOR AND BIOREACTOR PH
SENSOR HAVE BEEN BELOW 1.6 OR ABOVE 1.8 FOR MORE THAN 10 SAMPLES,
BIOREACTOR HIGH LEVEL SENSOR HAS BEEN ON MORE THAN 10 SAMPLES AND
TANK 1 LOW LEVEL SENSOR IS OFF";
            }

            if ((PHB_Comptador == 10) & (HLVTB_Comptador == 10) &
(LEDHLvTST.Visibility == Visibility.Visible) & (LEDLLvTFT.Visibility == Visibility.Collapsed))
            {
                Alarms_TextBox.Text = "BIOREACTOR PH SENSOR HAS BEEN BELOW 1.6
OR ABOVE 1.8 FOR MORE THAN 10 SAMPLES, BIOREACTOR HIGH LEVEL SENSOR
HAS BEEN ON MORE THAN 10 SAMPLES, TANK 2 HIGH LEVEL SENSOR IS ON AND
TANK 1 LOW LEVEL SENSOR IS OFF";
            }

            if ((PHL_Comptador == 10) & (HLVTB_Comptador == 10) &
(LEDHLvTST.Visibility == Visibility.Visible) & (LEDLLvTFT.Visibility == Visibility.Collapsed))
            {
                Alarms_TextBox.Text = "LEACHING PH SENSOR HAS BEEN BELOW 1.6 OR
ABOVE 1.8 FOR MORE THAN 10 SAMPLES, BIOREACTOR HIGH LEVEL SENSOR HAS
BEEN ON MORE THAN 10 SAMPLES, TANK 2 HIGH LEVEL SENSOR IS ON AND TANK
1 LOW LEVEL SENSOR IS OFF";
            }
```

```
            if ((PHL_Comptador == 10) & (PHB_Comptador == 10) & (LEDHLvTST.Visibility
== Visibility.Visible) & (LEDLLvTFT.Visibility == Visibility.Collapsed))
            {
               Alarms_TextBox.Text = "LEACHING PH SENSOR AND BIOREACTOR PH
SENSOR HAVE BEEN BELOW 1.6 OR ABOVE 1.8 FOR MORE THAN 10 SAMPLES,
TANK 2 HIGH LEVEL SENSOR IS ON AND TANK 1 LOW LEVEL SENSOR IS OFF";
            }

            if ((PHL_Comptador == 10) & (PHB_Comptador == 10) & (HLVTB_Comptador
== 10) & (LEDHLvTST.Visibility == Visibility.Visible) & (LEDLLvTFT.Visibility ==
Visibility.Collapsed))
            {
               Alarms_TextBox.Text = "LEACHING PH SENSOR AND BIOREACTOR PH
SENSOR HAVE BEEN BELOW 1.6 OR ABOVE 1.8 FOR MORE THAN 10 SAMPLES,
BIOREACTOR HIGH LEVEL SENSOR HAS BEEN ON MORE THAN 10 SAMPLES, TANK
2 HIGH LEVEL SENSOR IS ON AND TANK 1 LOW LEVEL SENSOR IS OFF";
            }


if ((PB1.Visibility == Visibility.Visible) | (PB2.Visibility == Visibility.Visible) | (PL3.Visibility ==
Visibility.Visible) | (PL4.Visibility == Visibility.Visible) | (PL5.Visibility == Visibility.Visible) |
(PL6.Visibility == Visibility.Visible))
            {
               tempsInicial = DateTime.Now;

               if (PB1.Visibility == Visibility.Visible)
               {
                  s = "PB1";
               }

               if (PB2.Visibility == Visibility.Visible)
               {
                  s1 = "PB2";
               }

               if (PL3.Visibility == Visibility.Visible)
               {
                  s2 = "PL3";
               }
```

```
            if (PL4.Visibility == Visibility.Visible)
            {
                s3 = "PL4";
            }

            if (PL5.Visibility == Visibility.Visible)
            {
                s4 = "PL5";
            }

            if (PL6.Visibility == Visibility.Visible)
            {
                s5 = "PL6"; ;
            }
        }

        List<TempsBombes> items_temps = new List<TempsBombes>();
        Llista_temps.ItemsSource = items_temps;

        if (((PB1.Visibility == Visibility.Collapsed) & (s == "PB1")) | ((PB2.Visibility ==
Visibility.Collapsed) & (s1 == "PB2")) | ((PL3.Visibility == Visibility.Collapsed) & (s2 ==
"PL3")) | ((PL4.Visibility == Visibility.Collapsed) & (s3 == "PL4")) | ((PL5.Visibility ==
Visibility.Collapsed) & (s4 == "PL5")) | ((PL6.Visibility == Visibility.Collapsed) & (s5 ==
"PL6")))
        {
            if (s == "PB1")
            {
                r = "PB1";
            }

            if (s == "PB2")
            {
                r = "PB2";
            }

            if (s == "PL3")
            {
                r = "PL3";
            }
```

```
                if (s == "PL4")
                {
                    r = "PL4";
                }

                if (s == "PL5")
                {
                    r = "PL5";
                }

                if (s == "PL6")
                {
                    r = "PL6";
                }

                temps_transcorregut = DateTime.Now - tempsInicial;
                TempsDeLlista = temps_transcorregut.ToString(@"hh\:mm\:ss");
                s = "";
                s1 = "";
                s2 = "";
                s3 = "";
                s4 = "";
                s5 = "";
                items_temps.Add(new TempsBombes() { Date_and_Hour =
tempsInicial.ToString(), Pump = r, Time_working = TempsDeLlista });
            }

        }
        catch (Exception ex)
        {
        }
    }

  }
}
```

8.3. CODI M-DUINO

```
#include <EEPROM.h>
#include "Pins.h"
#include "Wire.h"
volatile byte state = LOW;
float llindar_h=1.8;
float llindar_l=1.6;
int value, value0, value2, value3, value4, value5, value6, value7, value8, value9;
int v0, v1, v2, v3, v4, v5, v6, v7, v8, v9, v11, v12, v13, v14, v15;
int val0, val1, val2, val3, val4, val5, val6;
int
w=0,LVYSTT=0,LVCL2=0,LVYL1=0,p=0,b=0,o=0,d=0,j=0,r=0,aa=0,LVYL3=0,LVY3=0,LV
CL1=0,LVLCFT2=0;

void setup()
{
  Serial.begin(115200);
  Wire.begin();
  attachInterrupt(digitalPinToInterrupt(18),blink,RISING);
  rst();
  state=HIGH;
}
void loop()
{
  if (state=HIGH){
      llegeix();                                    // LLEGIM
      static unsigned long timepoint = millis();
      if(millis()-timepoint>1000U)  //time interval: 1s
      {
        timepoint = millis();
        biore();                                    //EXECUTEM
        dipi();
        dipo();
        lix();
        recu();
        }
    actu();                                         // ACTUALITZEM VALORS
    impri();                                        // PART GRAFICA
  }
}
```

```
void blink(){
  state = HIGH;
  rst();
  }
```

```
const int PB  = 36; // Bomba PB1
const int PBO = 37; // Bomba PB2
const int PBI = 48; // Bomba PB3
const int PL  = 49; // Bomba PL1
const int PLO = 45; // Bomba PL2
const int PLI = 44; // Bomba PL3
const int PLE = 43; // Bomba PL4
const int PLU = 42; // Bomba PL5
const int PR = 41;  // Bomba PR
const int PFT = 40; // Bomba PFT
const int PST = 39; // Bomba PST1
const int PS = 38;  // Bomba PST2
const int PT = 47;  // Bomba PST3
const int PLA = 53; // Bomba PL6
const int MB = 4; // Motor Agitador

const int HLVTB  = 22; // nivell HLvTB
const int HLVTL  = 23; // nivell HLvTL
const int LLVTL  = 24; // nivell LlvTL
const int HLVTR  = 25; // nivell HLvTR
const int LLVTR  = 26; // nivell LlvTR
const int HLVTFT = 27; // nivell HLvTFT
const int LLVTFT = 28; // nivell LLvTFT
const int HLVTST = 29; // nivell HLvTST
const int LLVTST = 30; // nivell LLvTST
const int LLVTB = 31; // nivell LLvTB

const int PHL = 54; // phímetre pHL
const int PHB = 55; // phímetre pHB
const int CL = 56; // Sensor color CL
const int CR = 57; // Sensor color CR
const int ITR = 58; // Sensor intensitat ITR
const int RB = 59; // Sensor REDOX RB
const int TB =60; // Sensor Temperatura TB
```

```
void actu(){
  digitalWrite(PB,v0);
  digitalWrite(PBO,v1);
  digitalWrite(PBI,v2);
  digitalWrite(PL,v3);
  digitalWrite(PLO,v4);
  digitalWrite(PLI,v5);
  digitalWrite(PLE,v6);
  digitalWrite(PLU,v7);
  digitalWrite(PR,v8);
  digitalWrite(PFT,v9);
  digitalWrite(PST,v11);
  digitalWrite(PS,v12);
  digitalWrite(PT,v13);
  digitalWrite(PLA,v14);
  digitalWrite(MB,v15); //TOTES LES ACTUACIONS
   }
```

```
//**********************CONTROL DEL BIOREACTOR*********************************//
void biore(){

     if (val1>=llindar_h)
      {
        v0=1;

      }
      else if (val1<=llindar_l)
      {
        v1=1;

      }
      else{
        v0=0;
        v1=0;

      }

   if (value==HIGH){
    digitalWrite(PBI,HIGH);
    b=b+1;
      if (b>=10){
        LVYSTT=1;
        }
    }
    else if (value==LOW){
     b=0;
     LVYSTT=0;
    }
}
```

```
//*******************CONTROL DEL DIPÓSIT PULMÓ 1*******************************//
void dipi(){
   if (value9==HIGH){
    digitalWrite(PFT,HIGH);

   }

   if (value0==HIGH){
    digitalWrite(PBI,HIGH);
    o=o+1;
      if (o>=10){
        LVYL1=1;
        }
   }
    else if (value0==LOW){
    o=0;
   }
}
```

```
//********************CONTROL DEL DIPÓSIT PULMÓ 2*******************************//
void dipo(){
   if (value2==HIGH || LVYSTT==1){
     digitalWrite(PS,LOW);
     digitalWrite(PT,HIGH);
    }

    else if (value2==LOW){
     digitalWrite(PS,HIGH);
     digitalWrite(PT,LOW);

    }

   if (value3==HIGH){
     rst();
     state=LOW;
      }
}
```

```
char incomingByte;
void llegeix(){
   value=digitalRead(HLVTB);
   value0=digitalRead(LLVTFT);
   value2=digitalRead(LLVTST);
   value3=digitalRead(HLVTST);
   value4=digitalRead(LLVTL);
   value5=digitalRead(HLVTR);
   value6=digitalRead(HLVTL);
   value7=digitalRead(LLVTR);
   value8=digitalRead(LLVTB);
   value9=digitalRead(HLVTFT);
   v0=digitalRead(PB);
   v1=digitalRead(PBO);
   v2=digitalRead(PBI);
   v3=digitalRead(PL);
   v4=digitalRead(PLO);
   v5=digitalRead(PLI);
   v6=digitalRead(PLE);
   v7=digitalRead(PLU);
   v8=digitalRead(PR);
   v9=digitalRead(PFT);
   v11=digitalRead(PST);
   v12=digitalRead(PS);
   v13=digitalRead(PT);
   v14=digitalRead(PLA);
   v15=digitalRead(MB);
   val0=analogRead(PHL);
   val1=analogRead(PHB);
   val2=analogRead(CL);
   val3=analogRead(CR);
   val4=analogRead(ITR);
   val5=analogRead(RB);
   val6=analogRead(TB);
}
```

```
void impri(){
  Serial.print("L3");
  Serial.println(value8);
  Serial.print("L4");
  Serial.println(value);
  Serial.print("L5");
  Serial.println(value4);
  Serial.print("L6");
  Serial.println(value7);
  Serial.print("L7");
  Serial.println(value9);
  Serial.print("L8");
  Serial.println(value0);
  Serial.print("L9");
  Serial.println(value3);
  Serial.print("L10");
  Serial.println(value2);
  Serial.print("L11");
  Serial.println(value6);
  Serial.print("L12");
  Serial.println(value5);
  Serial.print("L13");
  Serial.println(v0);
  Serial.print("L14");
  Serial.println(v1);
  Serial.print("L15");
  Serial.println(v2);
  Serial.print("L16");
  Serial.println(v3);
  Serial.print("L17");
  Serial.println(v4);
  Serial.print("L18");
  Serial.println(v5);
  Serial.print("L19");
  Serial.println(v6);
  Serial.print("L20");
  Serial.println(v7);
  Serial.print("L21");
  Serial.println(v8);
  Serial.print("L22");
  Serial.println(v9);
```

```
 Serial.print("L23");
 Serial.println(v11);
 Serial.print("L24");
 Serial.println(v12);
 Serial.print("L25");
 Serial.println(v13);
 Serial.print("L26");
 Serial.println(v14);
 Serial.print("L27");
 Serial.println(v15);
 Serial.print("L28");
 Serial.println(val0);
 Serial.print("L29");
 Serial.println(val1);
 Serial.print("L30");
 Serial.println(val2);
 Serial.print("L31");
 Serial.println(val3);
 Serial.print("L32");
 Serial.println(val4);
 Serial.print("L33");
 Serial.println(val5);
 Serial.print("L34");
 Serial.println(val6);
 }
```

```
//**********************CONTROL DE LA LIXIVIACIÓ*********************************//
String inString = ""; // COM port incoming data buffer
void lix(){
   if (value4==HIGH && LVCL2==0){
     digitalWrite(PLO,HIGH);
   }

   else if (value4==LOW || LVCL2==1){
       digitalWrite(PLO,LOW);
    }
   if (LVCL2==1 && LVLCFT2==0){
     digitalWrite(PL,HIGH);
   }
   if (LVCL1==HIGH){
     digitalWrite(PL,LOW);
   }
    else if (LVCL1==LOW){
     LVY3=1;
     digitalWrite(PLI,LOW);
     digitalWrite(PLE,LOW);
     digitalWrite(PLU,LOW);

     if (val0>=llindar_h && LVYL3==0)
       {
         digitalWrite(PLE,HIGH);

       }
       else if ( LVYL3==1){

       }
    }
}
```

```
//*****************CONTROL DE LA RECUPERACIÓ DE COURE*************************//
void recu(){
   if (value7==HIGH){
    digitalWrite(PR,LOW);
    }

   if (value5==HIGH){
    digitalWrite(PBI,HIGH);
    r=r+1;
      if (r>=10){
        LVYL1=1;
        }
    }
    else if (value5==LOW){
     r=0;
    }
}
```

```
void rst(){
  digitalWrite(PB,LOW);
  digitalWrite(PBO,LOW);
  digitalWrite(PS,LOW);
  digitalWrite(PST,LOW);
  digitalWrite(PFT,LOW);
  digitalWrite(PR,LOW);
  digitalWrite(PLU,LOW);
  digitalWrite(PLE,LOW);
  digitalWrite(PLI,LOW);
  digitalWrite(PLO,LOW);
  digitalWrite(PT,LOW);
  digitalWrite(PBI,LOW);
  digitalWrite(PL,LOW);
  digitalWrite(PLA,LOW);
}
```