UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
Departamento de Informática y Automática

TRABAJO FIN DE MÁSTER
MÁSTER EN INGENIERÍA DE SISTEMAS Y CONTROL

PhotoBioLib: una librería de Modelica para el modelado y simulación de fotobiorreactores

Agustín Pérez Castro

Dirigida por: Dr. José Sánchez Moreno (UNED)
Dr. Francisco Gabriel Acién Fernández (UAL)

Junio 2014

UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
Departamento de Informática y Automática

TRABAJO FIN DE MÁSTER
MÁSTER EN INGENIERÍA DE SISTEMAS Y CONTROL

PhotoBioLib: una librería de Modelica para el modelado y simulación de fotobiorreactores

Tipo B: Proyecto específico propuesto por el alumno

Agustín Pérez Castro

Dirigida por: Dr. José Sánchez Moreno (UNED)
Dr. Francisco Gabriel Acién Fernández (UAL)

A mis padres.

Agradecimientos

Deseo agradecer a todas aquellas personas que, con su ayuda, han contribuido a la realización de este trabajo fin de máster, mencionando especialmente:

- A los profesores José Sánchez Moreno y Francisco Gabriel Acién Fernández, como directores del trabajo fin de máster, por haberme guiado y solucionado siempre mis dudas.
- Al resto de profesores del Máster en Ingeniería de Sistemas y Control que han contribuido a aumentar mi conocimiento en el área.
- A mis compañeros del Departamento de Informática y Automática de la UNED, mencionando especialmente a Sebastián Dormido y José Sánchez por darme la oportunidad de seguir formándome en estos temas y por acogerme tan bien en Madrid.
- A todos los miembros del grupo de investigación en Automática, Robótica y Mecatrónica de la Universidad de Almería, mencionando especialmente a Manolo Berenguel y José Luis Guzmán, por iniciarme en el mundo de la investigación y confiar en mí.
- A mis amigos de siempre, compañeros de carrera y amistades de Brasil.
- A María del Mar por su ayuda, apoyo y comprensión.
- Y por último, a toda mi familia, que han estado conmigo en todas las decisiones que he tomado. Mis padres, Agustín y Amparo, mi hermano Álvaro y mi hermana Ampi.

A todos ellos, gracias

Índice

Índice de figuras

Índice de tablas

PhotoBioLib: una librería de *Modelica* para el modelado y simulación de fotobiorreactores

Agustín Pérez Castro
Máster en Ingeniería de Sistemas y Control
Escuela Técnica Superior de Ingeniería Informática
Universidad Nacional de Educación a Distancia

Resumen

Este trabajo presenta la librería *PhotoBioLib*. Dicha librería, implementada en *Modelica*, permite el modelado y simulación de fotobiorreactores construyéndolos como si de sistemas reales se trataran, sin necesidad de conocimientos de programación ni de las dinámicas involucradas. Así, se obtienen como resultado modelos con gran parecido con la realidad, con gran potencial de configuración, con la posibilidad de interconexión con otras librerías y con grandes velocidades de simulación.

Palabras clave: Fotobiorreactores, Modelado, Simulación, Modelica, PhotoBioLib.

1. Introducción

Pese al creciente interés en fotobiorreactores, no existen ni métodos ni software específico para su modelado [16]. La complejidad de modelado e identificación de parámetros, las dificultades en la sensorización de sistemas biológicos y la variedad de configuraciones de fotobiorreactores son, en parte, los causantes de que no existan herramientas específicas para el modelado y simulación de estos sistemas, y que tampoco, dentro de los entornos de modelado y simulación genéricos, existan librerías que hagan referencia específica a los fotobiorreactores.

Es por lo anterior que se ha diseñado la librería *PhotoBioLib* en el lenguaje *Modelica*.

En la presente sección se hace referencia al estado del arte en el modelado y simulación de fotobiorreactores, y se enumeran las principales características del lenguaje de modelado *Modelica* y del entorno *Dymola*. En la Sección 2 se exponen brevemente los fenómenos modelados y las ecuaciones que los describen. Más adelante, en la Sección 3, se habla del diseño de la librería desde la perspectiva de los dominios considerados, su esquema de composición y su arquitectura de paquetes. Los resultados obtenidos con el uso de la librería son presentados en la Sección 5. Por último, las virtudes de la librería implementada junto a las posibles mejoras a implementar se enumeran en la Sección 6.

1.1. Estado del arte de la simulación de fotobiorreactores

En la bibliografía referida a fotobiorreactores, solo se pueden encontrar aplicaciones desarrolladas desde cero, sin el apoyo de librerías especificas; o el uso de programas de modelado

y simulación, más o menos genéricos, para modelar y simular algunos aspectos parciales, como pueden ser dinámicas de fluidos o transferencias de calor. Se enumeran a continuación, algunas de las aplicaciones utilizadas en la bibliografía, explicando brevemente en qué tipo de trabajos se utilizan y con qué objetivo específico.

Aspen Plus. Es un entorno de modelado de procesos para el diseño conceptual, optimización y supervisión del rendimiento para las industrias químicas, de polímeros, de productos químicos especializados, de metales y minerales, e industrias energéticas basadas en carbón [1]. Esta aplicación se usa por ejemplo en [11], donde se hace un estudio de la viabilidad económica de un fotobiorreactor abierto y otro cerrado. En este artículo se utiliza Aspen Plus para modelar un balance de masa y energía.

COMSOL Multiphysics. Es un entorno de simulación que facilita todo los pasos en el proceso de modelado - definiendo su geometría, mallado, especificando su física, resolviendo, y visualizando los resultados [2]. En [17], donde se presenta un fotobiorreactor vertical diseñado para la producción de oxigeno a escala de laboratorio, se utiliza *COMSOL Multiphysics* para examinar las propiedades de transferencia de calor.

Labview. Haciendo uso de *Labview* [3], en [15] con el objetivo de evaluar el potencial de generación de biomasa y la captura de CO_2, teniendo en cuenta distintos tipos de especies de algas, tipos de fotobiorreactores, y localizaciones de instalación, se desarrolla la herramienta *EvAlgae*.

ANSYS Fluent. Este software posee las capacidades de modelado físico necesarias para modelar flujos, turbulencias, transferencias de calor y reacciones [4]. En [19] se utiliza esta herramienta para realizar una simulación numérica del rendimiento de la agitación, el historial de la intensidad de luz recibida por las algas y los patrones de fluido dentro de distintos tipos de diseños de fotobiorreactores.

Matlab/Simulink. *Matlab* es un lenguaje de alto nivel y un entorno interactivo para el cálculo numérico, la visualización y la programación, mientras que *Simulink* es un entorno de diagramas de bloque para la simulación multidominio y el diseño basado en modelos [5]. Haciendo uso de las herramientas anteriores como un simple entorno de programación, modelado y simulación, en [12], se implementa, simula, calibra y valida un modelo dinámico para el cultivo de microalgas, el cual tiene en cuenta la transferencia de materia y energía, y los fenómenos biológicos.

MeteoNorm. A través de la interpolación de datos de distintas bases meteorológicas proporciona cálculos de radiación solar, temperatura y otros parámetros meteorológicos en cualquier lugar del mundo [6]. Así esta herramienta es utilizada para generar los datos meteorológicos necesarios para en [13] realizar simulaciones y desarrollar un modelo de temperatura de un fotobiorreactor plano cerrado que permita investigar los parámetros relevantes que influyen en el comportamiento térmico del mismo.

SuperPro Designer. Este software facilita el modelado, evaluación y optimización de procesos integrados en una amplia gama de industrias (farmacéutica, biotecnología, tratamiento de aguas residuales, control de la polución del aire, etc.) [7]. En [14] se utiliza *SuperPro Designer* para modelar un proceso de captura de CO_2, considerando el crecimiento y la separación de la biomasa, además de la separación del aceite. Todo lo anterior con el objetivo de desarrollar una tecnología sostenible a escala piloto para la captura, a través de microalgas, de los gases de combustión de una central eléctrica para la producción de biodiesel.

Tracepro. Es una herramienta para el modelado de la propagación de la luz en sistemas optomecánicos [8]. De esta forma este software se utiliza para diseñar un placa de luz optima donde, basandose en una patente de colectores solares, se utilizan cables de fibra óptica para distribuir la luz a través del fotobiorreactor [18].

También se pueden encontrar en la bibliografía artículos donde se usan combinaciones de las herramientas anteriores. Por ejemplo, en [9] se hace uso de *COMSOL Multiphysics* y de *Aspen Plus* para estudiar la dinámica de fluidos y el consumo de CO_2 dentro de un fotobiorreactor helicoidal.

1.2. *Modelica*

Modelica es un lenguaje de modelado orientado a objetos que facilita el modelado y simulación de sistemas físicos. *Dymola* es una herramienta que implementa dicho lenguaje, quizás la más importante de las que usan el lenguaje *Modelica*. *Dymola* tiene licencia comercial, pero también se puede hacer uso de herramientas libres y gratuitas como son el caso de *OpenModelica* o *JModelica*.

Las características más importantes de *Modelica* son:

- Descripción acausal de modelos, es decir, basada en ecuaciones. La asignación de causalidad queda en manos del entorno.
- Posibilidad del uso de algoritmos. Estos permiten la descripción de modelos con asignación manual de causalidad así como el uso de funciones.
- Características inherentes a un lenguaje orientado a objetos, como son: abstracción, encapsulamiento, modularidad, principio de ocultación, polimorfismo y herencia múltiple.
- Uso de anotaciones para la descripción gráfica de los distintos elementos así como la posibilidad de documentación en formato html.
- Posibilidad de implementación de modelos híbridos, es decir, con parte continua y partes discretas, así como el uso de eventos, ya sean de tiempo o de estado.

En lo que respecta al entorno de simulación *Dymola*, se pueden destacar las siguientes características:

- Simulación eficiente de modelos. Además de las manipulaciones simbólicas de los modelos estos son traducidos y compilados en C, lo cual permite una mayor eficiencia que otros entornos que usan lenguajes interpretados.
- Mejoras en los algoritmos de manipulación simbólica y de integración sobre otras herramientas.
- Representación de resultados, tanto con la representación en gráficas de datos como con una herramienta de representación de simulaciones tridimensional.

2. Hipótesis de modelado

En esta sección se describen los fenómenos modelados así como las ecuaciones utilizadas. Las ecuaciones que se utilizan en la librería están inspiradas en las que se presentan en [10] y [12]. El modelado está basado en balances de materia y energía, fenómenos biológicos y en ciertas condiciones de equilibrio que se han de cumplir. Pese a que los modelos propuestos en la bibliografía utilizada, [10, 12], son continuos, con el objeto de hacer modelos más fieles a la realidad y de evitar problemas numéricos, tales como indeterminaciones, se van a plantear pequeños cambios que consideren eventos discretos, por lo que se plantea un modelado híbrido.

2.1. Tasa fotosintética

Las ecuaciones de esta subsección describen la radiación media que alcanza el cultivo y la producción de biomasa, O_2 disuelto y CO_2 disuelto en el líquido, como consecuencia de la actividad de las algas.

Tal y como se muestra en la ecuación (1), la radiación media en un punto de un fotobiorreactor, $I_{av}(t)$, vendrá determinada por la radiación solar en la superficie horizontal sin obstáculos, $I_0(t)$; la trayectoria de la luz, d_t; la concentración de biomasa, $C_b(t)$; el coeficiente de extinción de biomasa, K_a; y el factor de distribución, α, que representa la fracción de radiación disponible en un área particular del reactor [12].

$$I_{av}(t) = \frac{\alpha I_0(t)}{K_a d_t C_b(t)}(1 - exp(K_a d_t C_b(t))) \tag{1}$$

Para el modelado de la producción de O_2 y CO_2 por las microalgas, se seguirán las ecuaciones y parámetros definidos por [10].

En la ecuación (2) se describe como afecta la radiación, $I_{av}(t)$, a la producción de O_2. Dicha producción de O_2, $P_{O_2}[I_{av}](t)$, vendrá determinada por la tasa máxima de fotosíntesis, $P_{O_{2,max}}$; los parámetros de forma m y K_i; y por el exponente de forma n.

$$P_{O_2}[I_{av}](t) = \frac{P_{O_{2,max}} I_{av}(t)^n}{K_i exp(I_{av}(t)m) + I_{av}(t)^n} \tag{2}$$

Basandose en el modelo de Arrhenius, la ecuación (3) modela la tasa de fotosíntesis normalizada en función de la temperatura. Donde, A_1 y A_2 son factores preexponenciales; Ea_1 y Ea_2 son las energías de activación; la temperatura es representada por $T(t)$; y R es la constante universal de los gases ideales.

$$\overline{P_{O_2}[T](t)} = \left(A_1 exp\left(\frac{-Ea_1}{R\ T(t)}\right) - A_2 exp\left(\frac{-Ea_2}{R\ T(t)}\right)\right) \tag{3}$$

Tal y como exponen los autores de [10] la influencia del pH en la tasa de fotosíntesis es similar a la observada para la temperatura. Es por esto que para el modelado de la tasa fotosintética normalizada en función del pH se usa un modelo similar al de Arrhenius. En la ecuación (4), B_1 y

B_2 son factores preexponenciales; C_1 y C_2 son las energías de activación; y el pH es representado por $pH(t)$

$$\overline{P_{O_2}[pH](t)} = \left(B_1 exp\left(\frac{-C_1}{pH(t)}\right) - B_2 exp\left(\frac{-C_2}{pH(t)}\right)\right) \tag{4}$$

Una alta concentración de oxigeno reduce la tasa de fotosíntesis y favorece la fotorespiración de los cultivos [10]. Para el modelado de la influencia del O_2 disuelto en el cultivo, $P_{O_2}[O_2](t)$, se propone la ecuación de potencial (5), donde; K_{O_2} es la constante de inhibición del oxigeno, z es un parámetro de forma, y la concentración de O_2 disuelto es representada por $O_2(t)$.

$$\overline{P_{O_2}[O_2](t)} = \left(1 - \left(\frac{O_2(t)}{K_{O_2}}\right)^z\right) \tag{5}$$

En los experimentos realizados para la obtención de las ecuaciones (1)-(5) y su calibración se ha observado que cada variable es independiente del resto [10]. Por tanto, la tasa fotosintética puede ser modelada como la combinación de las ecuaciones anteriormente descritas, ecuaciones (6) y (7).

$$P_{O_2}[I_{av}, T, pH, O_2](t) = P_{O_2}[I_{av}](t)\ \overline{P_{O_2}[T](t)}\ \overline{P_{O_2}[pH](t)}\ \overline{P_{O_2}[O_2](t)} \tag{6}$$

$$\begin{aligned} P_{O_2}(t) \quad = \quad & \frac{P_{O_{2,max}} I_{av}(t)^n}{K_i exp(I_{av}(t)m) + I_{av}(t)^n} \left(A_1 exp\left(\frac{-Ea_1}{R\ T(t)}\right) - A_2 exp\left(\frac{-Ea_2}{R\ T(t)}\right)\right) \\ & \left(B_1 exp\left(\frac{-C_1}{pH(t)}\right) - B_2 exp\left(\frac{-C_2}{pH(t)}\right)\right)\left(1 - \left(\frac{O_2(t)}{K_{O_2}}\right)^z\right) \end{aligned} \tag{7}$$

En [12] se modela la respiración del cultivo en función de la tasa máxima de fotosíntesis y de la tasa de respiración r. Con esto, el modelo completo de la producción de O_2 queda representado por la ecuación (8). Además, se considera que la producción de CO_2 será la inversa de la de O_2, ecuación (9).

$$\begin{aligned} P_{O_2}(t) \quad = \quad & \frac{P_{O_{2,max}} I_{av}(t)^n}{K_i exp(I_{av}(t)m) + I_{av}(t)^n} \left(A_1 exp\left(\frac{-Ea_1}{R\ T(t)}\right) - A_2 exp\left(\frac{-Ea_2}{R\ T(t)}\right)\right) \\ & \left(B_1 exp\left(\frac{-C_1}{pH(t)}\right) - B_2 exp\left(\frac{-C_2}{pH(t)}\right)\right)\left(1 - \left(\frac{O_2(t)}{K_{O_2}}\right)^z\right) - r P_{O_{2,max}} \end{aligned} \tag{8}$$

$$P_{CO_2}(t) = -P_{O_2}(t) \tag{9}$$

Hay que considerar en el modelado aquellos eventos discretos que produzcan discontinuidades, cambios en la dinámica o problemas numéricos. En el caso de la tasa fotosintética hay que tener en cuenta la existencia o no de radiación, es decir, si es de noche o de día. Por este motivo se incluye la variable booleana $eRad(t)$, la cual tomará el valor *true* en el día y el valor *false* durante la noche, ecuación (10). Así, el cálculo de $P_{O_2}(t)$ queda como se muestra en la ecuación (11).

$$eRad(t) = \begin{cases} true & si\ I_0(t) > 0 \\ false & si\ I_0(t) \leq 0 \end{cases} \tag{10}$$

$$P_{O_2}(t) = \begin{cases} Eq.(8) & si\ eRad(t) = true \\ -r P_{O_{2,max}} & si\ eRad(t) = false \end{cases} \tag{11}$$

2.2. Generación y absorción de materia por las microalgas

En función de la tasa fotosintética modelada en la sección anterior, las microalgas generaran nueva biomasa a partir de las moléculas de CO_2 disueltas en el medio, liberando a su vez moléculas de O_2. Dicha producción de biomasa, así como el flujo de O_2 y CO_2, vendrá determinada por las ecuaciones (12), (13) y (14) respectivamente. Donde, Y_{bo} es el coeficiente de rendimiento de biomasa, M_{O_2} es el peso molecular del O_2, y M_{CO_2} es el peso molecular del CO_2.

$$fC_{b,f}(t) = V_l(t) P_{O_2}(t) Y_{bo} \tag{12}$$

$$fO_{2,f}(t) = \frac{P_{O_2}(t)}{M_{O_2}} C_b(t) \tag{13}$$

$$fCO_{2,f}(t) = \frac{P_{CO_2}(t)}{M_{CO_2}} C_b(t) \tag{14}$$

2.3. Relación entre el CO_2, C_T y H^+

El carbono inorgánico total, C_T, es la suma de las distintas especies de carbono inorgánico, ecuación (15). Por tanto, considerando las constantes de equilibrio K_w, K_1 y K_2, descritas en las ecuaciones (16), (17) y (18), se puede establecer una relación entre la variación de la concentración de $C_T(t)$ y la concentración de $CO_2(t)$, tal y como se muestra en la ecuación (19).

$$C_T(t) = CO_2(t) + HCO_3^-(t) + CO_2^=(t) \tag{15}$$

$$K_w = OH^-(t)\ H^+(t) \tag{16}$$

$$K_1 = \frac{HCO_3^-(t)\ H^+(t)}{CO_2(t)} \tag{17}$$

$$K_2 = \frac{CO_3^=(t)\ H^+(t)}{HCO_3^-(t)} \tag{18}$$

$$\frac{dC_T(t)}{dt} = \left(1 + \frac{K_1}{H^+(t)} + \frac{K_1 K_2}{H^+(t)^2}\right) \frac{dCO_2(t)}{dt} - CO_2(t) \left(\frac{K_1}{H^+(t)^2} + \frac{2K_1K_2}{H^+(t)^3}\right) \frac{dH^+(t)}{dt} \tag{19}$$

En lo que se refiere a la relación entre la concentración de $CO_2(t)$ y la concentración de $H^+(t)$, se ha de cumplir la restricción de electroneutralidad expresada en la ecuación (20). Si se consideran concentraciones constantes de cationes y aniones, se llega a la relación que se muestra en la ecuación (21). En la ecuación (22) se muestra el cálculo del $pH(t)$

$$\frac{dH^+(t)}{dt} + \frac{dCat^+(t)}{dt} = \frac{dOH^-(t)}{dt} + \frac{dHCO_3^-(t)}{dt} + 2\frac{dCO_3^=(t)}{dt} + \frac{dAn^-(t)}{dt} \tag{20}$$

$$\frac{dH^+(t)}{dt} = \frac{\frac{K_1}{H^+(t)} + \frac{2K_1K_2}{H^+(t)^2}}{1 + \frac{K_w}{H^+(t)^2} + \frac{K_1CO_2(t)}{H^+(t)^2} + 4\frac{2K_1K_2}{H^+(t)^3}} \frac{dCO_2(t)}{dt} \tag{21}$$

$$pH(t) = -log_{10}(H^+(t)) \tag{22}$$

2.4. Transferencia de materia entre la fase gaseosa y la fase líquida

En el cultivo se produce una transferencia de materia entre la fase gaseosa y la fase líquida. No obstante, esta transferencia solo se modela para el O_2 y al CO_2, la transferencia de N_2 no se considera al no ser soluble en agua. Así, en las ecuaciones (23) y (24) se muestran dichas transferencias en función del volumen de líquido, $V_l(t)$; de los coeficientes de transferencia de masa volumétrico para el O_2 y el CO_2, $K_{laO2}(t)$ y $K_{laCO2}(t)$; y de la concentración en equilibrio de O_2 y CO_2 en la fase gaseosa, $O_2^*(t)$ y $CO_2^*(t)$.

$$fO_{2,gl}(t) = V_l(t)K_{laO_2}(t)(O_2^*(t) - O_2(t)) \tag{23}$$

$$fCO_{2,gl}(t) = V_l(t)K_{laCO_2}(t)(CO_2^*(t) - CO_2(t)) \tag{24}$$

Las concentraciones en equilibrio, ecuaciones (25) y (26), van en función de la constante de Henry para el O_2 y el CO_2, H_{O_2} y H_{CO_2}; de la presión en el líquido, P_t; y de las fracciones molares del O_2 y el CO_2, $yO_2(t)$ y $yCO_2(t)$. Dichas fracciones molares, representadas en las ecuaciones (27)-(30), son calculadas a partir de la concentración de los distintos componentes del gas; $O_{2g}(t)$, $CO_{2g}(t)$ y $N_{2g}(t)$.

$$O_2^*(t) = H_{O_2}P_t\ yO_2(t) \tag{25}$$

$$CO_2^*(t) = H_{CO_2}P_t\ yCO_2(t) \tag{26}$$

$$yO_2(t) = \frac{O_{2g}(t)}{O_{2g}(t) + CO_{2g}(t) + N_{2g}(t)} \tag{27}$$

$$yCO_2(t) = \frac{CO_{2g}(t)}{O_{2g}(t) + CO_{2g}(t) + N_{2g}(t)} \tag{28}$$

$$yN_2(t) = \frac{N_{2g}(t)}{O_{2g}(t) + CO_{2g}(t) + N_{2g}(t)} \tag{29}$$

$$yO_2(t) + yCO_2(t) + yN_2(t) = 1 \tag{30}$$

En las ecuaciones (31) y (32) se representa el calculo de los coeficientes de transferencia de masa volumétrico para el O_2 y el CO_2, $K_{laO2}(t)$ y $K_{laCO2}(t)$, en función de los parámetros de forma a y b; del coeficiente de transferencia para el CO_2, KCO_2; y del *hold-up*, $\epsilon(t)$.

$$K_{laO_2}(t) = a\ \epsilon(t)^b \tag{31}$$

$$K_{laCO_2}(t) = K_{laO_2}(t)KCO_2 \tag{32}$$

El cálculo del $\epsilon(t)$ será diferente en función del tipo de componente que se este modelando. Esto es debido a que no en todos los fotobiorreactores ni en todas sus partes el gas interactua de la misma forma con el líquido. Por ejemplo, mientras que en la columna de burbujeo de un fotobiorreactor cerrado el gas y el líquido tienen un sentido de desplazamiento opuesto, en el lazo del mismo fotobiorreactor el sentido de movimiento es el mismo. Las dos formas que se usarán para el cálculo del $\epsilon(t)$ en este trabajo son las que se muestran en las ecuaciones (33) y (34). En las ecuaciones anteriormente descritas $U_g(t)$ y $U_l(t)$ son las velocidades superficiales del gas y el líquido, C_0 es un parámetro del modelo de deriva de flujo, y U_∞ es la tasa de ascensión de las burbujas. En el caso concreto de un fotobiorreactor cerrado, la ecuación (33) sería la utilizada en el lazo y la ecuación (34) la utilizada en la columna de burbujeo.

$$\epsilon(t) = \frac{V_g(t)}{V_l(t)V_g(t)} \tag{33}$$

$$\epsilon(t) = \frac{U_g(t)}{C_0 U_g(t) + U_l(t) + U_\infty} \tag{34}$$

La transferencia de materia dependerá de la existencia de gas, representada por la variable booleana $eGas(t)$, ecuación (35). Por tanto, cuando $eGas(t) = false$, no se producirá transferencia de materia y las ecuaciones anteriores de la presente sección serán nulas.

$$eGas(t) = \begin{cases} true & si\ V_g(t) > 0 \\ false & si\ V_g(t) \leq 0 \end{cases} \tag{35}$$

2.5. Flujos de calor

En cualquier fotobiorreactor se producirán flujos de calor desde/hacia distintos orígenes/destinos y mediante distintos mecanismos. Los mecanismos considerados serán tres: convección, radiación y transferencia de calor por medio de transferencia de materia. En lo que respecta a los orígenes/destinos estos serán la radiación solar, la temperatura ambiente, intercambiadores de calor y la adición/substracción de medio de cultivo. Este último mecanismo dependerá del flujo de volumen, $Q_l(t)$; de la temperatura del flujo, $T(t)$; y de su capacidad calorífica volumétrica, C_v; tal y como se muestra en la ecuación (36).

$$fQ_m(t) = Q_l(t)C_v T(t) \tag{36}$$

Las transferencias de calor por convección vendrán determinas por ecuaciones de la forma de la ecuación (37), donde: h es el coeficiente de transmisión, S_{conv} es la superficie de contacto, y las temperaturas de los elementos que entran en contacto son $T_1(y)$ y $T_2(t)$.

$$fQ_c(t) = hS_{conv}(T_1(t) - T_2(t)) \tag{37}$$

En lo que refiere a la transferencia por radiación, ésta vendrá marcada por la ecuación (38). En dicha ecuación se considera la radiación solar, $I_c(t)$; la superficie que recibe dicha radiación, S_{rad}; y la absortividad de la radiación solar aR; así como el factor de distribución, α.

$$fQ_r(t) = \alpha\ I_c(t)S_{rad}\ aR \tag{38}$$

2.6. Balances de masa en la fase gaseosa

Como ya se ha comentado, la fase gaseosa considera tres concentraciones $O_{2g}(t)$, $CO_{2g}(t)$ y $N_{2g}(t)$, dichas concentraciones son calculadas como el cociente de los moles totales de cada componente, $O_{2gt}(t)$, $CO_{2gt}(t)$ y $N_{2gt}(t)$, y el volumen total de gas $V_g(t)$, ver ecuaciones (39), (40) y (41).

$$O_{2g}(t) = \begin{cases} \dfrac{O_{2gt}(t)}{V_g(t)} & si\ eGas(t) = true \\ 0 & si\ eGas(t) = false \end{cases} \tag{39}$$

$$CO_{2g}(t) = \begin{cases} \dfrac{CO_{2gt}(t)}{V_g(t)} & si\ eGas(t) = true \\ 0 & si\ eGas(t) = false \end{cases} \tag{40}$$

$$N_{2g}(t) = \begin{cases} \dfrac{N_{2gt}(t)}{V_g(t)} & si\ eGas(t) = true \\ 0 & si\ eGas(t) = false \end{cases} \tag{41}$$

Por otro lado, el flujo de materia que provoca un transporte de gas con un caudal $Q_g(t)$, se describe en las ecuaciones (42), (43) y (44).

$$fO_{2g,m}(t) = Q_g(t)O_{2g}(t) \tag{42}$$

$$fCO_{2g,m}(t) = Q_g(t)CO_{2g}(t) \tag{43}$$

$$fN_{2g,m}(t) = Q_g(t)N_{2g}(t) \tag{44}$$

Por tanto, el balance de masa en la fase gaseosa quedará como se plantea en las ecuaciones (45)-(48). Hay que destacar, que en las ecuaciones mencionadas únicamente se muestran el tipo de elementos, pero en función de la configuración del fotobiorreactor y de la pieza que se este modelando, dichos elementos podrán estar ausentes o bien pueden aparecer más de una vez.

$$\frac{dV_g(t)}{dt} = Q_g(t) \tag{45}$$

$$\frac{dO_{2gt}(t)}{dt} = fO_{2g,m}(t) - fO_{2,gl}(t) \tag{46}$$

$$\frac{dCO_{2gt}(t)}{dt} = fCO_{2g,m}(t) - fCO_{2,gl}(t) \tag{47}$$

$$\frac{dN_{2gt}(t)}{dt} = fN_{2g,m}(t) \tag{48}$$

2.7. Balances de masa y energía en la fase líquida

La fase líquida considera las concentraciones $O_2(t)$, $C_T(t)$, $C_b(t)$, $CO_2(t)$ y $H^+(t)$. Las dos últimas son calculadas a partir de las ecuaciones (19) y (21), el resto se calculan a partir de las ecuaciones (50), (51) y (52) como el cociente entre los moles o kilogramos de cada componente; $O_{2t}(t)$, $C_{Tt}(t)$ y $C_{bt}(t)$; y el volumen total de líquido, $V_l(t)$. La temperatura, $T(t)$, se calcula de forma similar a partir de la cantidad total de calor, $Q_t(t)$, en la ecuación (53). Al igual que ocurre en la fase gaseosa el cálculo de las concentraciones depende de la existencia de líquido, ver ecuación (49).

$$eLiq(t) = \begin{cases} true & si\ V_l(t) > 0 \\ false & si\ V_l(t) \leq 0 \end{cases} \tag{49}$$

$$O_2(t) = \begin{cases} \dfrac{O_{2t}(t)}{V_l(t)} & si\ eLiq(t) = true \\ 0 & si\ eLiq(t) = false \end{cases} \tag{50}$$

$$C_T(t) = \begin{cases} \dfrac{C_{Tt}(t)}{V_l(t)} & si\ eLiq(t) = true \\ 0 & si\ eLiq(t) = false \end{cases} \tag{51}$$

$$C_b(t) = \begin{cases} \dfrac{C_{bt}(t)}{V_l(t)} & si\ eLiq(t) = true \\ 0 & si\ eLiq(t) = false \end{cases} \tag{52}$$

$$T(t) = \begin{cases} \dfrac{Q_t(t)}{V_l(t)C_v} & si\ eLiq(t) = true \\ 0 & si\ eLiq(t) = false \end{cases} \tag{53}$$

Los flujos de materia que se producen como consecuencia del transporte de líquido con un caudal $Q_l(t)$ se muestran en las ecuaciones (54), (55) y (56). El flujo de energía calorífica como consecuencia del flujo de materia, líquido, ya fue expuesto anteriormente en la ecuación (36)

$$fO_{2,m}(t) = Q_l(t)O_2(t) \tag{54}$$

$$fC_{T,m}(t) = Q_l(t)C_T(t) \tag{55}$$

$$fC_{b,m}(t) = Q_l(t)C_b(t) \tag{56}$$

Así, el balance de masa y energía en la fase líquida quedará como se plantea en las ecuaciones (57)-(61). Igual que ocurría en la fase gaseosa, los elementos de las ecuaciones podrán estar ausentes o bien pueden aparecer más de una vez en función de la parte del fotobiorreactor que se este modelando y de la configuración del mismo.

$$\frac{dV_l(t)}{dt} = Q_l(t) \tag{57}$$

$$\frac{dO_{2t}(t)}{dt} = fO_{2,m}(t) + fO_{2,gl}(t) + fO_{2,f}(t) \tag{58}$$

$$\frac{dC_{Tt}(t)}{dt} = fC_{T,m}(t) + fCO_{2,gl}(t) + fCO_{2,f}(t) \tag{59}$$

$$\frac{dC_{bt}(t)}{dt} = fC_{b,m}(t) + fC_{b,gl}(t) + fC_{b,f}(t) \tag{60}$$

$$\frac{dQ_t(t)}{dt} = fQ_m(t) + fQ_c(t) + fQ_r(t) \tag{61}$$

3. Diseño de la librería

El diseño de la librería se puede describir desde tres perspectivas: los dominios implementados, el esquema de composición de clases y la arquitectura de paquetes.

3.1. Dominios

Se han considerado cuatro dominios, gas, *Gas*; liquído, *Liq*; calor, *Heat*; y líquido con biomasa, *Bio*. Por tanto, los conectores, las interfaces, las clases básicas y algunas de las partes son descritas en función de estos. Además, unos dominios engloban a otros, lo cual se traduce en relaciones de herencia que hacen que las variables de unos estén presentes en otros. Así, el dominio *Heat* está incluido en el dominio *Liq* y éste está incluido en el dominio *Bio*. Esto queda bien representado en los conectores diseñados, cuyas parejas de variables de potencial y de flujo, varibles *accross* y *through*, se muestran en la Tabla 1.

Tabla 1: Variables de los conectores

Conector	**Variable *across***	**Variable *through***
GasCon	Volumen de gas - V_g $[m^3]$	Caudal de gas - fV_g $[m^3\ s^{-1}]$
	Concentración de O_2 - O_{2g} $[mol\ m^{-3}]$	Flujo de O_2 - fO_{2g} $[mol\ s^{-1}]$
	Concentración de CO_2 - CO_{2g} $[mol\ m^{-3}]$	Flujo de CO_2 - fCO_{2g} $[mol\ s^{-1}]$
	Concentración de N_2 - N_{2g} $[mol\ m^{-3}]$	Flujo de N_2 - fN_{2g} $[mol\ s^{-1}]$
HeatCon	Temperatura - T $[K]$	Flujo de calor - fQ $[J\ s^{-1}]$
LiqCon		
BioCon		
LiqCon	Volumen de líquido - V_l $[m^3]$	Caudal de líquido - fV_l $[m^3\ s^{-1}]$
BioCon		
BioCon	Concentración de biomasa - Cb $[Kg\ m^{-3}]$	Flujo de biomasa - fCb $[Kg\ s^{-1}]$
	Concentración de O_2 - O_2 $[mol\ m^{-3}]$	Flujo de O_2 - fO_2 $[mol\ s^{-1}]$
	Concentración de CO_2 - CO_2 $[mol\ m^{-3}]$	Flujo de CO_2 - fCO_2 $[mol\ s^{-1}]$
	Concentración de C_T - C_T $[mol\ m^{-3}]$	Flujo de C_T - fC_T $[mol\ s^{-1}]$
	Concentración de H^+ - H^+ $[mol\ m^{-3}]$	Flujo de H^+ - fH^+ $[mol\ s^{-1}]$

3.2. Esquema de composición

El esquema de composición de clases, representado en la Figura 1, se divide en cuatro niveles.

En el nivel 0 encontramos los *conectores* y las *interfaces* creadas a través de los mismos; estas interfaces serán las que describan las conexiones con el exterior de cada una de las clases. Además, en este nivel se sitúan también las representaciones gráficas, *iconos*, y las *ecuaciones* que describen la dinámica de cada clase.

A través de la composición de un conjunto de ecuaciones, una interfaz para la transferencia de información y un icono para su descripción gráfica, se crean cada una de las clases *básicas*.

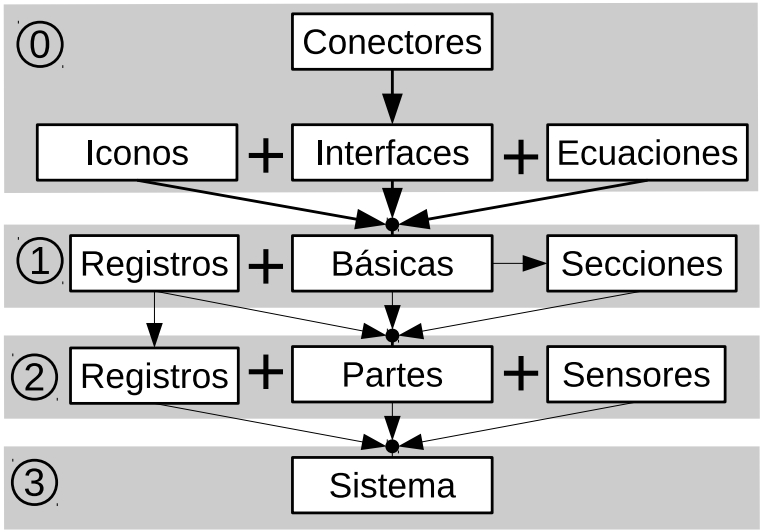


Figura 1: Esquema de composición de clases

Estas clases serán las que marquen los volúmenes de control y los fenómenos de transporte, lo cual queda reflejado en sus iconos, como se puede ver en la Sección 3.6. Dichas clases básicas se encuentran en el nivel 1 junto a las *secciones*, clases compuestas a partir de clases básicas pero que no tienen entidad por si mismas para considerarse partes; y los *registros*, conjuntos de parámetros y constantes. En la Sección 3.6.4 se muestra la composición de una sección.

Para la composición de las *partes* se hará uso de clases básicas, secciones y otras partes, como se muestra en la Sección 3.8. Además, para la configuración de los parámetros se hará uso de conjuntos de registros, los cuales a su vez pueden estar compuestos por conjuntos de registros del nivel anterior. Todo esto, junto a los *sensores*, formará el nivel 2 del esquema de composición. En la Figura 2 se muestran algunas de las partes implementadas.

(a) HeatExchanger (b) Column (c) Loop

Figura 2: Algunas de las *partes*

Por último, en el nivel 3 del esquema de composición se encuentran las clases que, a partir de la unión de distintas partes, el uso de sensores para la medición de potenciales y flujos, y su configuración con conjuntos de registros, forman sistemas completos equiparables a fotobiorreactores reales. Como ejemplo de sistema completo se ha construido el fotobiorreactor cerrado que se muestra en la Sección 4.

3.3. Arquitectura de la librería

A continuación se expone, de forma resumida, el contenido de cada uno de los paquetes principales, ver Figura 3.

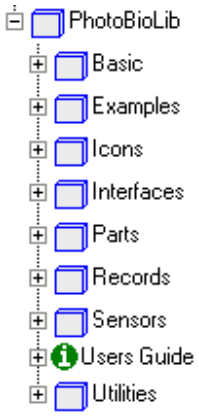


Figura 3: Arquitectura de paquetes de la librería

- *Basic*: Este paquete contiene los elementos básicos.
- *Examples*: Será el paquete objeto a contener los ejemplos de fotobiorrectores que se diseñen haciendo uso de los componentes de la librería, es decir, los sistemas completos como el de la Figura 46.
- *Icons*: Contiene las descripciones gráficas de cada uno de los modelos.
- *Interfaces*: Aquí se encuentran los conectores diseñados, así como el conjunto de modelos parciales que describen las interfaces.
- *Parts*: Dentro de este paquete se encuentran los modelos que más se asimilan a los componentes físicos reales, es decir, la columna, el lazo, las bombas, los depósitos, etc.
- *Records*: Es donde se almacenan los registros, es decir, clases tipo *record* que contienen las constantes, parámetros y valores iniciales de las clases básicas y las partes.
- *Sensors*: En este paquete se almacenan los componentes que tienen un comportamiento similar al que tendrían los sensores físicos. Se separan del paquete *Parts* porque, a diferencia de estos, no van a modificar el comportamiento dinámico del sistema que se modele, sino que serán una herramienta para extraer información, ya sea para un registro de datos más claro o para el uso en los posibles bucles de control.
- *Users Guide*: Paquete de documentación que contendrá la información más relevante sobre la librería.
- *Utilities*: Clases que servirán como herramientas para la simulación, como por ejemplo para la lectura de ficheros o tratamiento de datos de entrada.

En las próximas secciones se habla de las clases más relevantes que forman la librería.

3.4. Conectores

Se han definido una serie de conectores que permitirán la unión de los distintos elementos del sistema. Como ya se ha comentado, dichos conectores se encuentran en el paquete *Interfaces*. En la Tabla 1 se muestran sus variables separadas en variables de potencial y de flujo. Las variables de potencial, variables *across*, serán igualadas en el punto de conexión. Las variables de flujo, variables *through*, sumaran cero en el punto de conexión. En la Figura 4 se muestra la representación gráfica de los distintos conectores implementados

(a) GasCon (b) LiqCon (c) HeatCon (d) BioCon

Figura 4: Representación de los conectores

3.5. Interfaces

Como se comentó en la Sección 3.2, con el fin de aprovechar las características que ofrece el modelado orientado a objetos, cada una de las clases estará compuesta a su vez por tres componentes diferenciados. Así, sobre la base de las ecuaciones se compone, a través de herencia, con otras dos clases: una que almacena lo referente a la representación gráfica, como pueden ser los iconos; y otra que marcará la interfaz, es decir, el conjunto de conectores que servirán para unir los distintos componentes entre sí.

De esta forma, cualquier conexión entre clases será definida por las interfaces, esto, además de facilitar la implementación y mantenimiento del código, permitirá la reutilización de interfaces para clases que tengan la misma conexión con el exterior. En la Figura 5 se muestran algunos ejemplos de las mismas, para una referencia completa de la misma se puede consultar el Anexo B. Dentro de la librería, las interfaces están en el paquete *Partials*, que a su vez, se encuentra en el paquete *Interfaces*. Las interfaces se construyen a partir de la composición de distintos conectores.

(a) (b) (c) (d)

Figura 5: Ejemplos de interfaces

3.6. Elementos atómicos, paquete *Basic*

Las clases que se describen en esta sección son las que definiren los volúmenes de control y los fenómenos de transporte. A partir de la composición de estos elementos básicos se crearán el resto

de los componentes, tal y como se muestra más adelante. Como ya se comentó anteriormente, estos elementos se ubican en el paquete *Basic*. Además, dentro de dicho paquete hay otros cuatro paquetes que describen los balances de gas, paquete *Gas*; de líquido, paquete *Liquid*; de temperatura, paquete *Heat*; y de cultivo, paquete *Biomass*.

3.6.1. Elementos atómicos de calor, paquete *Basic.Heat*

En este paquete se encuentran los elementos atómicos que hacen referencia al calor. Sin embargo, la transferencia de calor debida a la transferencia de materia será desplazada al paquete *Basic.Liquid*, ya que dicho flujo de calor tiene lugar a través de los conectores de líquido *LiqCon*.

Conexión con el entorno, modelo *Environment*. Esta clase se encarga de tratar la temperatura del entorno para hacerla accesible a un conector del tipo *HeatCon* y de convertir la radiación fotosintéticamente activa, $I_0(t)$, medida en $[\mu E\ m^{-2}s^{-1}]$ a su equivalente en $[Wm^{-2}]$, $I_c(t)$. Su interfaz está compuesta por dos entradas, una de radiación y otra de temperatura; una salida, de radiación; y un conector de temperatura, *HeatCon*. El icono de este modelo se puede ver en la Figura 6.

Figura 6: Icono de la clase *Environment*

Transferencia de calor por convección, modelo *HeatConvection*. Haciendo uso de la ecuación (37) esta clase calcula la transferencia de calor por convección. Las conexiones con el exterior son a través de dos conectores de calor *HeatCon*. Su representación gráfica se muestra en la Figura 7.

Figura 7: Icono de la clase *HeatConvection*

Transferencia de calor por radiación, modelo *HeatRadiation*. La transferencia de calor por radiación que en esta clase se modela viene descrita por la ecuación (38). Tiene una entrada de radiación y un conector *HeatCon*. En la Figura 8 se muestra su representación gráfica.

Transferencia de calor por convección y radiación, modelo *HeatConvRad*. Mediante la composición de los elementos atómicos de calor ya presentados, ver Figura 9, se crea este

Figura 8: Icono de la clase *HeatRadiation*

modelo. El mismo permite, con la información recibida a través de las entradas de la temperatura exterior y la radiación solar, calcular las transferencias de calor que se producen por radiación y convección con el entorno y dirigir dicho flujo a través de un único conector *HeatCon*. Su icono se muestra en la Figura 10

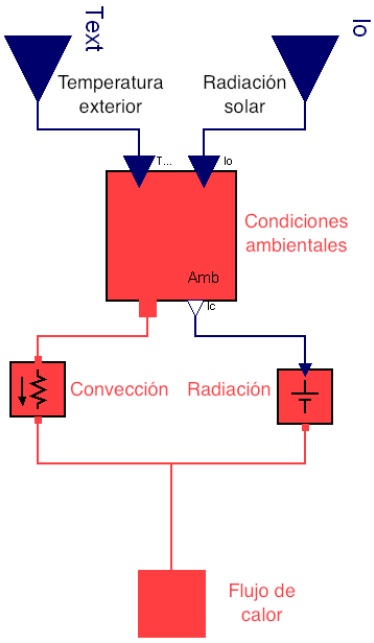


Figura 9: Diagrama de composición de *HeatConvRad*

Figura 10: Icono de la clase *HeatConvRad*

3.6.2. Elementos atómicos de líquido, paquete *Basic.Liquid*

La fase líquida está dividida en dos capas, una primera que considera la transferencia y acumulación de líquido y calor, y una segunda que, además de lo anterior, tendrá en cuenta los flujos y concentraciones de: biomasa, O_2, CO_2, C_T y H^+. La primera de las dos capas es la que se modela en el presente paquete. Esta división, aparte de facilitar la comprensión y mantenimiento del código, permitirá que en aquellos casos en los que solo es necesario modelar el balance de masa y energía del líquido y su temperatura, como es el caso del intercambiador de calor, tener menos ecuaciones y, por tanto, un modelo más eficiente.

Volumen de control de líquido y energía calorífica, modelo *LiqCapacitor*. Esta clase es la que actúa como volumen de control, acumulador, de líquido y de energía calorífica. Para conectar ambos elementos con el exterior dispone de un conector de líquido, *LiqCon*, y de un conector de calor, *HeatCon*. Es importante recordar, que la transferencia de calor como consecuencia de la transferencia de materia se producirá a través del conector de líquido, y no a través del conector de calor. En lo que se refiere a su formulación, esta clase hace uso de las ecuaciones (53), (57) y (61). Su icono se muestra en la Figura 11.

Figura 11: Icono de la clase *LiqCapacitor*

Transferencia de líquido y calor por transferencia de materia, modelo *LiqResistor*. El caudal de transferencia de líquido será marcado por la entrada que posee la interfaz de este modelo, dicho caudal entrará por el conector de líquido izquierdo y saldrá en la misma cantidad por el derecho, con el correspondiente flujo de energía. Este fenómeno responde a la ecuación (36). En la Figura 12 se puede ver su representación gráfica.

Figura 12: Icono de la clase *LiqResistor*

3.6.3. Elementos atómicos de gas, paquete *Basic.Gas*

En este paquete se encuentran los elementos atómicos en los que únicamente interfiere el gas. Aquellos elementos en los que se relaciona la fase líquida con la fase gaseosa se encuentran en el paquete *Basic.Biomass*.

Volumen de control de la fase gaseosa, modelo *GasCapacitor*. Define el volumen de control, acumulador, de la fase gaseosa por completo. En esta clase se calculan las concentraciones de los distintos elementos en función de las ecuaciones (39), (40) y (41). Además, los flujos se integran para obtener el volumen o cantidad de moles de cada elemento tal y como muestran las ecuaciones (45)-(48). La representación gráfica de esta clase es la que se muestra en la Figura 13.

Figura 13: Icono de la clase *GasCapacitor*

Transferencia de gas, modelo *GasResistor*. Tal y como ocurre en la transferencia de líquido, la transferencia de gas está marcada por la entrada que posee la interfaz de este modelo. Dicho caudal entrará por el conector de gas izquierdo y saldrá en la misma cantidad por el derecho, desplazando de esta forma las cantidades de moles de cada elemento que se calculan en las ecuaciones (42), (43) y (44). En la Figura 14 se puede ver su representación gráfica.

Figura 14: Icono de la clase *GasResistor*

3.6.4. Elementos atómicos con biomasa, paquete *Basic.Biomass*

Todos aquellos elementos atómicos que hagan uso de conectores del tipo *BioCon*, y afecten por tanto a la dinámica de la biomasa y concentraciones en la fase líquida, están incluidos en el presente paquete. De esta forma, en este paquete se encuentran las clases que modelan la fase líquida y aquellas que marcan la interacción entre la fase líquida y la gaseosa. Además, se incluyen clases que modelan secciones de partes más complejas, las cuales por sí solas no tienen entidad para poder considerarse dentro del paquete *Parts*, como, por ejemplo, una sección de tubería con gas y líquido en su interior,

Producción de biomasa, O_2 y CO_2; modelo *AlgaeSource*. En el interior de este modelo se calcula la tasa fotosintética, ecuaciones (1), (9) y (11); y la generación de biomasa, O_2 y CO_2, ecuaciones (12), (13) y (14). Su interfaz está compuesta por una entrada de radiación y por un conector *BioCon*, siendo su representación gráfica la que se muestra en la Figura 15.

Volumen de control de la fase líquida completa, modelo *BioCapacitor*. Como ya se ha comentado, esta clase es hija de la clase *LiqCapacitor*, por lo que hereda de la misma las funciones de volumen de control de líquido y de energía calorífica. Además, a lo anterior añade

Figura 15: Icono de la clase *AlgaeSource*

las concentraciones de biomasa, O_2, CO_2, C_T y H^+, las cuales calcula en función de las ecuaciones (50), (51), (52), (19) y (21). Previo al calculo de las concentraciones descritas por las ecuaciones anteriores se han de calcular las cantidades totales de cada uno de los componentes, esto se lleva a cabo conforme a las ecuaciones (58), (59) y (60). La comunicación con el exterior se realiza a través del conector *HeatCon* heredado de su clase padre y de un conector *BioCon* que reemplaza al conector *LiqCon* de la clase que hereda. En la Figura 16 se puede ver su icono.

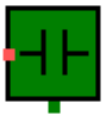


Figura 16: Icono de la clase *BioCapacitor*

Transferencia de materia en la fase líquida, modelo *BioResistor*. De la misma forma que la clase *BioCapacitor* hereda de la clase *LiqCapacitor*, esta clase hereda de la clase *LiqResistor*. Por tanto, a las ecuaciones utilizadas en *LiqResistor* hay que añadir las ecuaciones (54), (55) y (56), que definen el flujo de O_2, C_T y biomasa como consecuencia del desplazamiento de líquido. En este caso, los dos conectores de la clase padre son reemplazados por dos conectores *BioCon*, manteniendo la entrada que marca el caudal. La representación gráfica se muestra en la Figura 17.

Figura 17: Icono de la clase *BioResistor*

Clase parcial para la transferencia de materia, modelo parcial *LiqGasResistor_Partial*. En la Sección 2.4 se expone que la transferencia de materia entre la fase gaseosa y la fase líquida tiene pequeñas diferencias en función de qué parte del fotobiorreactor se modele. No obstante, la mayoría de las ecuaciones son comunes, independientemente de como se modele la transferencia de materia. Es por esto que esta clase parcial, la cual no es posible instanciar, servirá como base,

es decir, como padre, para las clases que modelan los diferentes tipos de transferencia de materia. Las ecuaciones que incorpora esta clase son las que van de la (23) a la (32) , ambas incluidas. En la Figura 18 se muestra su icono.

Figura 18: Icono de la clase *LiqGasResistor_ Partial*

Transferencia de materia en un lazo, modelo *LiqGasResistor_ Loop.* Esta clase hereda de la clase *LiqGasResistor_ Partial* y añade la ecuación (33). En la versión actual de la librería este modelo servirá para el cálculo de la transferencia de materia en el interior de una sección del lazo de un fotobiorreactor cerrado. No obstante en un futuro se espera reutilizar en otros componentes. Su representación gráfica es también heredada de su clase padre.

Transferencia de materia en un columna de burbujeo, modelo *LiqGasResistor_ Column.* Al igual que ocurre con *LiqGasResistor_ Loop* esta clase también hereda de la clase *LiqGasResistor_ Partial* y añade en este caso la ecuación (34). En la versión actual de la librería, este modelo servirá para el cálculo de la transferencia de materia en el interior de una columna de burbujeo de un fotobiorreactor cerrado y, al igual que ocurría con *LiqGasResistor_ Loop*, se espera reutilizar en otro tipo de componentes como, por ejemplo, en el foso de un fotobiorreactor de tipo *raceway.* A la interfaz de su clase padre se añaden dos entradas que marcan las velocidades superficiales del líquido y del gas. Su icono es el que se muestra en la Figura 19.

Figura 19: Icono de la clase *LiqGasResistor_ Column*

Regulación del flujo de gas, con volumen de líquido constante; modelo *FlowRegulator_ LiqFCons.* Este modelo es utilizado para el cálculo de la velocidad de desplazamiento del gas en función de la velocidad de desplazamiento del líquido en el interior del lazo de un fotobiorreactor cerrado. Para la implementación de esta clase se considera la simplificación de que el volumen de gas que entra en una sección del lazo no desplaza volumen de líquido. Dicha simplificación es válida debido a la pequeña parte que el volumen de gas representa sobre el volumen total. La interfaz de este modelo está compuesta por un conector de gas, una entrada con la velocidad del líquido a la entrada y dos salidas para marcar las velocidades, caudales, del líquido y del gas. Las ecuaciones (62) y (63) calculan los flujos de líquido y gas, donde: $Q_{l,in}(t)$ y $Q_{l,out}(t)$ son los flujos de líquido a la entrada y salida del modelo, $Q_{g,out}(t)$ es el flujo de gas, A_t

es el área de la sección transversal de una sección de lazo y dX es la longitud de dicha sección. El icono de esta clase es el mostrado en la Figura 20

$$Q_{l,out}(t) = Q_{l,in}(t) \tag{62}$$

$$Q_{g,out}(t) = \frac{Q_{l,in}(t)}{A_t}\frac{V_g(t)}{dX} \tag{63}$$

Figura 20: Icono de la clase *FlowRegulator_ LiqFCons*

Regulación del flujo de gas, con volumen de líquido variable; modelo *FlowRegulator_ LiqFVar.* En versiones anteriores de la librería se implementó una versión más compleja que sí consideraba el desplazamiento de líquido. Esta clase no se utilizada en la versión actual ya que tras realizar varios experimentos se vio que las diferencias en los resultados al usar o no la simplificación eran prácticamente inapreciables, un pequeño ruido de sensorización tendría más efecto, mientras que el tiempo de simulación, en el mejor de los casos, se duplicaba. No obstante, por si fuera útil en futuras implementaciones, se mantiene el modelo. En este caso se utilizan las ecuaciones (64) y (65) . El icono es el mismo que el de la clase *FlowRegulator_ LiqFCons.*

$$Q_{l,out}(t) = Q_{l,in}(t)\frac{dV_g(t)}{dt} \tag{64}$$

$$Q_{g,out}(t) = \frac{Q_{l,out}(t)V_l(t)}{A_t dX - V_g(t)} \tag{65}$$

Sección de una columna de burbujeo; modelo *ColumnSection.* En la Figura 21 se muestra cómo, a partir de elementos atómicos, se crea una sección de columna de burbujeo. La interfaz del modelo está compuesta por: una entrada que marca la temperatura exterior, otra que marca la radiación solar, un conector de calor y dos pares de conectores de fase líquida y de fase gaseosa. Hay que notar que mientras que el gas seguirá un sentido ascendente, el sentido de circulación del líquido será descendente. En lo que se refiere a su representación gráfica ésta se muestra en la Figura 22.

Sección del lazo de un fotobiorreactor cerrado; modelo *LoopSection.* Esta clase define una sección de lazo a partir de la composición de elementos atómicos, ver Figura 23. La interfaz consta de dos pares de conectores de fase líquida y gaseosa, y de dos entradas, una para la temperatura y otra para la radiación solar. En la Figura 24 se puede ver su representación gráfica.

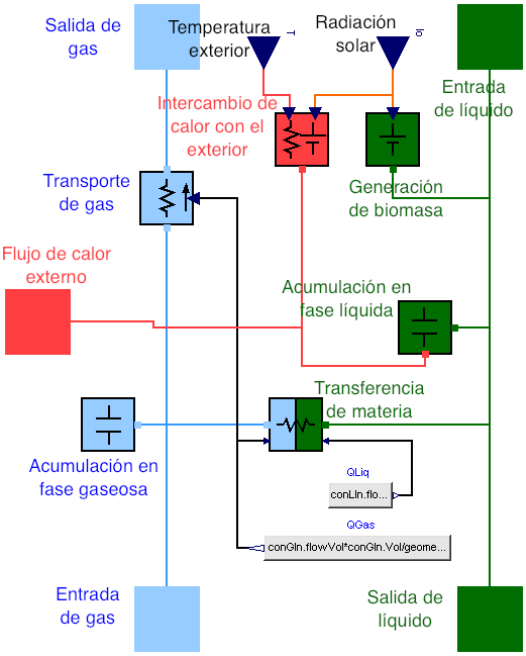


Figura 21: Diagrama de composición de *ColumnSection*

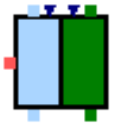


Figura 22: Icono de la clase *ColumnSection*

3.7. Colecciones de parámetros, paquete *Records*

Cada uno de los componentes básicos presentados en la sección anterior, así como modelos más complejos como los que contiene el paquete *Parts*, tienen una serie de parámetros, constantes y valores iniciales que han de definirse. Con el fin de tener una implementación más limpia, además de facilitar la introducción de dichos parámetros, y simplificar la transmisión de los mismos cuando se produce herencia o composición, estos se agrupan en colecciones en función de su tipo o su dominio de aplicación. De esta forma se definen los *Records* y los paquetes del mismo tipo de clases que se exponen a continuación:

- *Algae*: Colección con los parámetros que definen el comportamiento de una microalga. En futuras versiones de la librería se espera incorporar un paquete que tenga predefinidas las propiedades de distintos tipos de algas para que el usar una u otra alga en los experimentos sea tan sencillo como arrastrar el tipo de alga que se desea utilizar. En la Figura 25 se muestra la representación gráfica de este *Record*.
- *Constants*: En esta clase se almacenan las constante físicas utilizadas en el modelado.
- *Geometry*: Este paquete contiene colecciones que marcar los parámetros geométricos de los modelos.

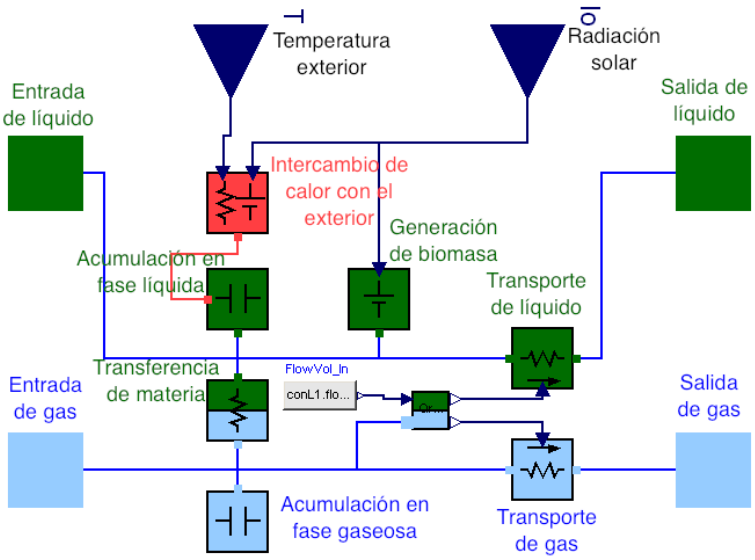


Figura 23: Diagrama de composición de *LoopSection*

Figura 24: Icono de la clase *LoopSection*

- *Heat*: Aquí se almacenan los parámetros que afectan a la transferencia de calor y al cálculo de temperaturas.
- *Initial*: Tanto las modelos atómicos como los modelos compuestos necesitan de valores de inicialización para sus estados. Estos se proporcionan en este paquete, ya sea para los distintos dominios en general o para las partes físicas diferenciadas en particular.
- *LocationAndDate*: En esta versión de la librería esta colección de registros está vacía. En próximas versiones será aquí donde se almacenen los datos de fecha y localización que permitan la simulación de un fotobiorreactor en distintos lugares del mundo y en distintas épocas del año, a través del cálculo de la radiación solar y la temperatura ambiente.
- *MassTransfer*: Paquete con las colecciones de datos que marcarán la transferencia de materia entre fases.
- *Operation*: Aquí se marcan todos los parámetros de operación, que en la versión actual incluyen los valores iniciales de todos los componentes y algunas entradas fijas durante toda la simulación. En futuras versiones en este *Record* se incluirán también aquellos parámetros referentes a la configuración de los controladores.
- *PhysicalPar_ ClosedPhotobioreactor*: En este *Record* se marcan todos los parámetros físicos que necesita un fotobiorreactor cerrado para ser simulado, es decir, los parámetros geométricos, los referentes a la transferencia de calor y los que caracterizan la transferencia de

materia en cada uno de sus componentes. No se incluyen aquí los parámetros del microalga al considerarla una entidad diferenciada.

Figura 25: Icono de la clase *Algae*

Para una descripción más detallada de los parámetros que están incluidos en cada *Record*, así como de sus valores por defecto y de la composición y herencia de estas clases, se puede consultar el Anexo B.

3.8. Componentes físicos, paquete *Parts*

Aquellos modelos que representan componentes físicos de los distintos fotobiorreactores son los que se almacenan en este paquete. Esto incluye desde las distintas bombas o depósitos, hasta intercambiadores de calor o las columnas de burbujeo. Sin embargo, como se comentó anteriormente, los sensores, pese a poder ser considerados también componentes físicos de un fotobiorreactor, se incluyen en un paquete diferenciado ya que estos son utilizados únicamente para la extracción de información y no afectan a la dinámica de los sistemas modelados.

3.8.1. Componentes utilizados en el dominio gaseoso, paquete *Parts.Gas*

En este paquete se incluyen aquellos componentes físicos en los que se almacena o por los que se transporta gas. Estos van desde una bomba de gas, hasta un depósito, pasando por una tubería para unir distintas fuentes de gas. En la Figura 26 se muestran los distintos componentes modelados.

(a) *GasPump* (b) *GasSource* (c) *GasTank* (d) *Tpipe*

Figura 26: Componentes en el paquete *Parts.Gas*

3.8.2. Componentes utilizados en el dominio líquido, paquete *Parts.Liquid*

Aquellos componentes que almacenan o transportan líquido sin biomasa son los que se incluyen en este paquete. En la versión actual de la librería estos componentes servirán únicamente para el modelado del flujo de líquido que pasa a través del intercambiador de calor. En la Figura 27 se muestran los distintos componentes modelados.

(a) *LiqPump* (b) *LiqSource* (c) *LiqTank*

Figura 27: Componentes en el paquete *Parts.Liquid*

3.8.3. Componentes relacionados con la biomasa, paquete *Parts.Biomass*

Aquí están los componentes equivalentes a los modelados en *Parts.Liquid* cuando el líquido incluye biomasa. Sus representaciones gráficas se muestran en la Figura 28.

(a) *BioPump* (b) *BioSource* (c) *BioTank*

Figura 28: Componentes en el paquete *Parts.Biomass*

3.8.4. Intercambiador de calor, clase *HeatExchanger*

Esta clase es la que modela un intercambiador de calor. Igual que ocurre con otros componentes se ha modelado a partir de la composición de distintos elementos atómicos, ver Figura 29. La transferencia de calor producida por el flujo del líquido en el interior del intercambiados tiene lugar a través de un conector de calor *HeatCon*, en lo que se refiera a dicho flujo de líquido este entrará por el conector *LiqCon* inferior y saldrá por el superior del mismo tipo. El icono que lo representa se puede ver en la Figura 30.

3.8.5. Columna de burbujeo, clase *Column*

La columna de burbujeo es modelada como una mezcla perfecta, por esto mismo se basa en la utilización de una única instancia de la clase *ColumnSection*.

A través de la composición que se muestra en la Figura 31 se construye una columna de burbujeo. Su interfaz consta de los siguientes componentes: dos conectores *GasCon*, el inferior servirá para la entrada de gas y el superior para la salida del mismo; dos conectores *BioCon* situados a la derecha de la columna, el flujo procedente del lazo entrara por el superior y a través del inferior volverá al mismo; dos conectores *BioCon* situados a la izquierda de la columna, a través del inferior se producirá la entrada de nuevo medio de cultivo y por el superior se producirá el cosechado mediante el desbordamiento de la columna; por ultimo hay dos entradas que sirven para marcar la temperatura exterior y la radiación solar. En la Figura 32 se muestra su icono.

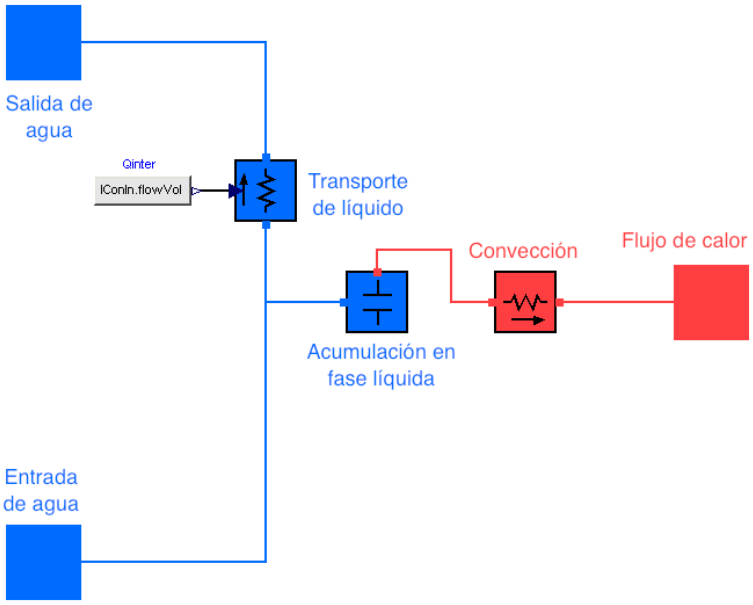


Figura 29: Diagrama de composición de *HeatExchanger*

Figura 30: Icono de la clase *HeatExchanger*

3.8.6. Columna con intercambiador de calor, clase *Column_HeatExchanger*

Esta clase es muy similar a la clase *Column*, la única diferencia, como se puede apreciar en la Figura 33, es la conexión de un intercambiador de calor. Dicho intercambiador hace necesario la inclusión a la interfaz de dos conectores *LiqCon* para la entrada y salida de líquido al intercambiador. En la Figura 34 se muestra su icono.

3.8.7. Lazo de un fotobiorreactor cerrado, clase *Loop*

Este lazo es modelado como un flujo pistón, donde el número de secciones en las que se parte el mismo es marcado por el parámetro nE el cual se incluye en la descripción de su geometría.

La composición de esta clase se muestra en la Figura 35. Su interfaz, aparte de contar con las entradas de temperatura y radiación, tiene también tres parejas de conectores *GasCon* y *BioCon*. La pareja que se encuentra en la parte inferior izquierda es la entrada en el lazo, la pareja superior izquierda es la salida del lazo y la pareja situada a la derecha tiene como finalidad la conexión de sensores de potencial. Su icono es el que se muestra en la Figura 36. En la Figura 35 no se observan conexiones porque éstas se generan de forma dinámica en función del número de secciones en las que se parta el lazo.

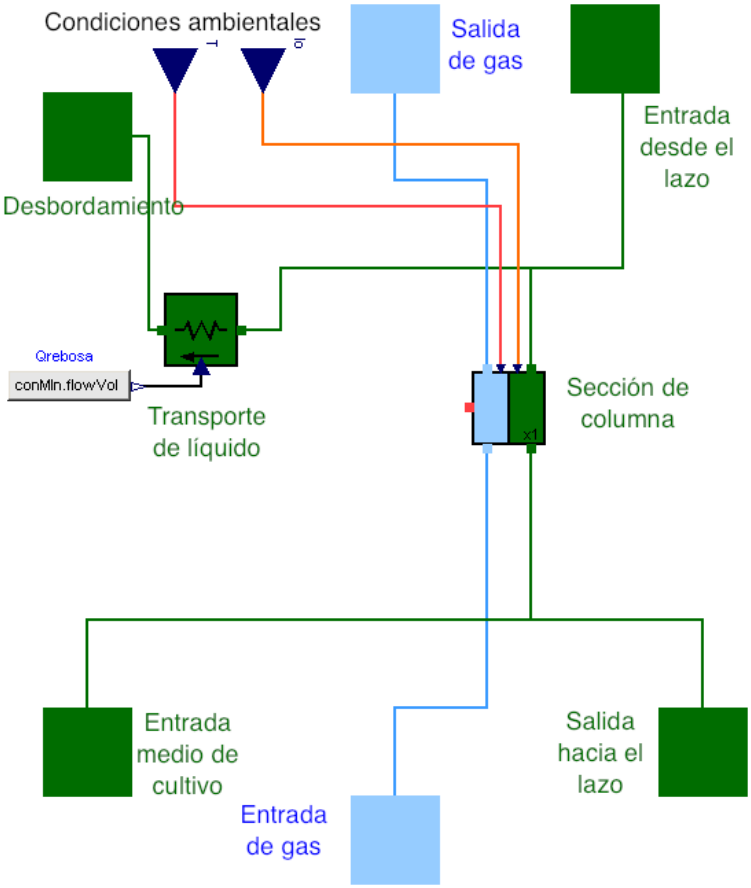


Figura 31: Diagrama de composición de *Column*

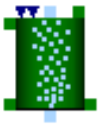


Figura 32: Icono de la clase *Column*

3.9. Sensores, paquete *Sensors*

Cuando se analizan los resultados de una simulación solo se requieren los valores de ciertas variables para observar su evolución o para compararlas con otros datos ya almacenados, es decir, se actúa de forma similar a cuando se incorporan sensores a sistemas físicos reales. *Modelica* permite almacenar cada una de las variables que se calculan durante la simulación y, aunque esto, puede ser de utilidad, también se puede convertir en un problema a la hora de buscar la variables que se desea analizar o incluso a la hora de guardar los datos. Esto se hace patente principalmente cuando los modelos implementados tienen la magnitud de los que se desarrollan con la librería que se presenta en este trabajo.

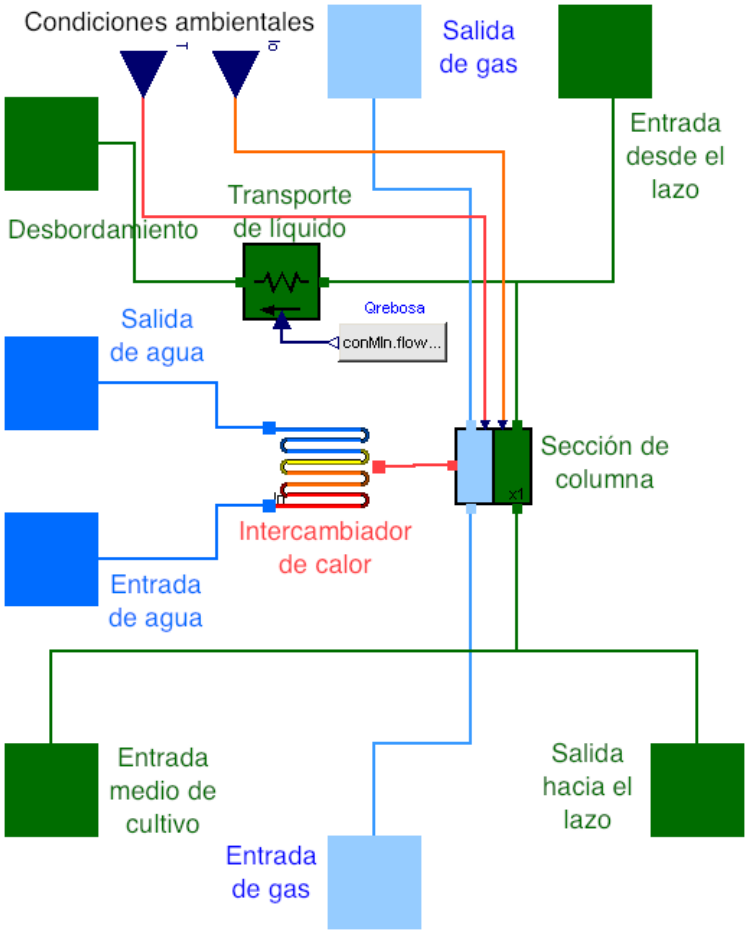


Figura 33: Diagrama de composición de *Column_HeatExchanger*

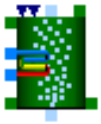


Figura 34: Icono de la clase *Column_HeatExchanger*

Así, con el uso de sensores se limitan y clasifican las variables que se quieren muestrear, al mismo tiempo que se obtiene un modelo final más fiel a la realidad y se facilita la inclusión de bucles de control que se alimenten de las medidas tomadas por dichos sensores. El procedimiento a seguir para obtener solo la información que se desea de una simulación es tan sencillo como marcar aquellas clases que no son sensores como *protected* y desactivar el registro de dicho tipo de datos. Si por cualquier motivo se quiere acceder a variables protegidas será tan sencillo como volver a activar el registro de las mismas.

Igual que en el resto de paquetes de la librería, los sensores están diferenciados por el dominio en el que realizan las medidas. Dentro de cada uno de los dominios se distingue entre sensores

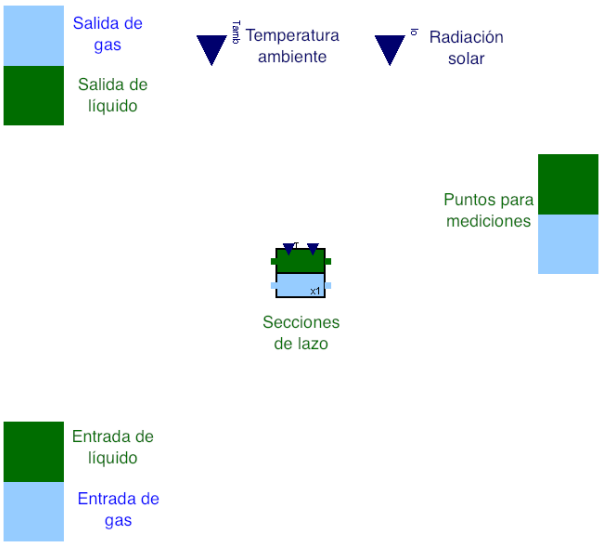


Figura 35: Diagrama de composición de *Loop*.

Figura 36: Icono de la clase *Loop*

de potencial y sensores de flujo. Igual que ocurre en el mundo real, estos sensores se tendrán que conectar en paralelo o en serie en función de si miden potencial o si miden flujo.

Sensores de gas. En las Figuras 37 y 38 se muestrann los distintos tipos de sensores utilizados para medir flujos y concentraciones en el gas. En la Tabla 2 se describe que medida proporciona cada una de las salidas.

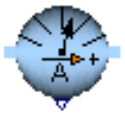


Figura 37: Icono de la clase *GasFlowSensor*

Sensores de líquido. En las Figuras 39 y 40 se muestran los distintos tipos de sensores para el líquido y en la Tabla 3 se describen sus salidas.

Tabla 2: Salidas de los sensores de gas

Sensor	Salida	Medición
GasFlowSensor	1	Flujo de gas - $f_g(t)[m^3 s^{-1}]$
	2	Flujo de O_2 - $f_{O_2}(t)[mol\ s^{-1}]$
	3	Flujo de CO_2 - $f_{CO_2}(t)[mol\ s^{-1}]$
	4	Flujo de N_2 - $f_{N_2}(t)[mol\ s^{-1}]$
GasPotentialSensor	1	Volumen de gas - $V_g(t)[m^3]$
	2	Concentración de O_2 - $O_{2g}(t)[mol\ m^{-3}]$
	3	Concentración de CO_2 - $CO_{2g}(t)[mol\ m^{-3}]$
	4	Concentración de N_2 - $N_{2g}(t)[mol\ m^{-3}]$
	5	Fracción molar de O_2 - $yO_2(t)[-]$
	6	Fracción molar de CO_2 - $yCO_2(t)[-]$
	7	Fracción molar de N_2 - $yN_2(t)[-]$

Tabla 3: Salidas de los sensores de líquido

Sensor	Salida	Medición
LiqFlowSensor	1	Flujo de líquido - $f_g(t)[m^3 s^{-1}]$
	2	Flujo de calor - $f_Q(t)[J\ s^{-1}]$
LiqPotentialSensor	1	Volumen de líquido - $V_l(t)[m^3]$
	2	Temperatura - $T(t)[K]$
	3	Temperatura - $T(t)[C]$

Figura 38: Icono de la clase *GasPotentialSensor*

Figura 39: Icono de la clase *LiqFlowSensor*

Figura 40: Icono de la clase *LiqPotentialSensor*

Sensores de líquido y biomasa. En las Figuras 41 y 42 se muestran los distintos tipos de sensores para el líquido y en la Tabla 4 se describen sus salidas.

Figura 41: Icono de la clase *BioFlowSensor*

Figura 42: Icono de la clase *BioPotentialSensor*

Tabla 4: Salidas de los sensores de líquido y biomasa

Sensor	Salida	Medición
BioFlowSensor	1	Flujo de líquido - $f_g(t)[m^3 s^{-1}]$
	2	Flujo de calor - $f_Q(t)[J\ s^{-1}]$
	3	Flujo de biomasa - $f_{C_b}(t)[Kg\ s^{-1}]$
	4	Flujo de O_2 - $f_{O_2}(t)[mol\ s^{-1}]$
	5	Flujo de CO_2 - $f_{CO_2}(t)[mol\ s^{-1}]$
	6	Flujo de C_T - $f_{C_T}(t)[mol\ s^{-1}]$
	7	Flujo de H^+ - $f_{H^+}(t)[mol\ s^{-1}]$
BioPotentialSensor	1	Volumen de líquido - $V_l(t)[m^3]$
	2	Temperatura - $T(t)[K]$
	3	Temperatura - $T(t)[C]$
	4	Concentración de biomasa - $C_b(t)[Kg\ m^{-3}]$
	5	Concentración de O_2 - $O_2(t)[mol\ m^{-3}]$
	6	Concentración de CO_2 - $CO_2(t)[mol\ m^{-3}]$
	7	Concentración de C_T - $C_T(t)[mol\ m^{-3}]$
	8	Concentración de H^+ - $H^+(t)[mol\ m^{-3}]$
	9	Potencial de pH - $pH(t)[-]$

4. Ejemplo de uso

Uno de los objetivos de la librería es que pueda ser usada por cualquier usuario sin necesidad de conocimientos de programación, dando la posibilidad de construir los modelos como si se tratara de sistemas físicos reales a partir de sus distintos componentes. Además, se incluye en el paquete *Examples* un fotobiorreactor cerrado completamente operativo y en futuras versiones de la librería se incluirán las partes necesarias para construir otro tipo de fotobiorreactores.

A la hora de modificar los parámetros de los modelos diseñados estos se pueden modificar directamente, pulsando sobre cada uno de los elementos e introduciendo aquellos parámetros que se desea que sean distintos de los valores por defecto. No obstante se recomienda seguir los pasos que a continuación se describen.

El primer paso consistiría en construir el sistema a partir de los componentes deseados, añadiendo las distintas partes y orígenes de datos en el diagrama de composición, tal y como se muestra en la Figura 43. En este punto sería posible establecer los parámetros simplemente pulsando sobre cada elemento y modificando aquellos que se desea que sean diferentes a los valores por defecto, ver Figura 44, aunque no se recomienda esta forma de proceder.

Tras construir el sistema en lugar de marcar directamente los parámetros se añadirá al diagrama de composición aquellas clases *Record* que engloban los parámetros necesarios, por ejemplo la clase *Algae*, ver Figura 45a. Tras incluir el conjunto de parámetros y darle un nombre se incluirá dicho nombre dentro de los parámetros de los componentes, como se puede ver en la Figura 45b. Esta forma de proceder supondrá grandes ventajas, las más destacables son:

- Mayor claridad en la organización de parámetros.
- La posibilidad de reutilizar un mismo conjunto de parámetros para distintas partes del modelo. En este caso tanto el lazo como la columna necesitan los parámetros de la microalga, y estos han de ser, además, iguales. Usando este mecanismo se evita el tener que introducir los datos por duplicado y, además, se asegura la ausencia de errores.
- La facilidad para utilizar conjuntos de datos previamente definidos. Aprovechando esta ventaja se puede crear un paquete con las propiedades de distintas algas en función de sus especies y luego, simplemente, arrastrar la especie deseada al modelo.
- El poder mantener varios conjuntos de datos en un modelo y poder seleccionar de forma ágil y rápida qué conjunto se desea utilizar para la simulación. Esto puede ser útil, por ejemplo, para tener dos conjuntos de valores iniciales y alternar entre los mismos, dejando ambos guardados en el modelo para futuros usos.

Siguiendo las recomendaciones anteriormente expuestas se ha creado el modelo *ClosedPhotobioreactor*, situado en el paquete *Examples* y cuyo diagrama de composición se puede ver en la Figura 46

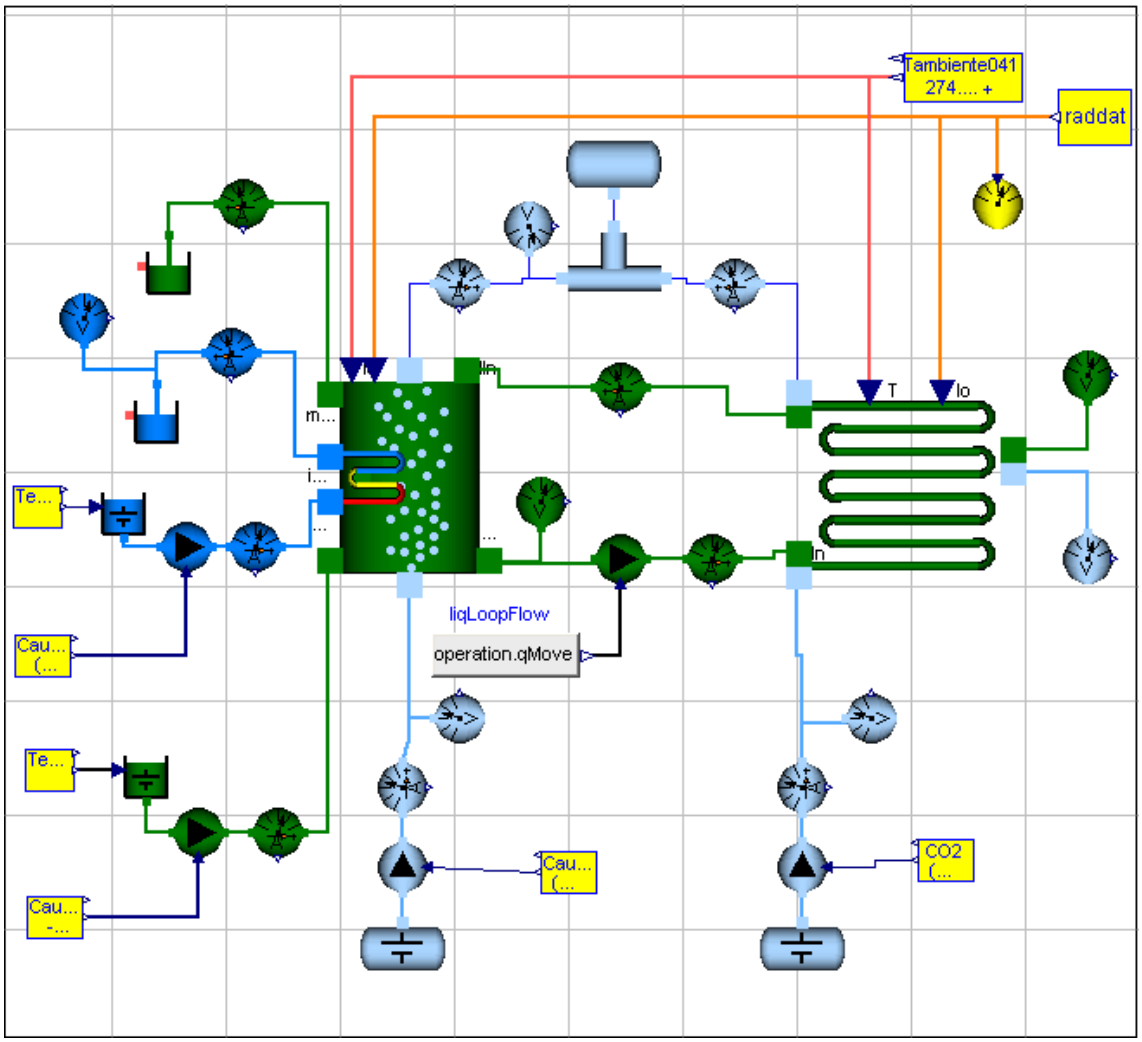


Figura 43: Diagrama de composición de un fotobiorreactor cerrado

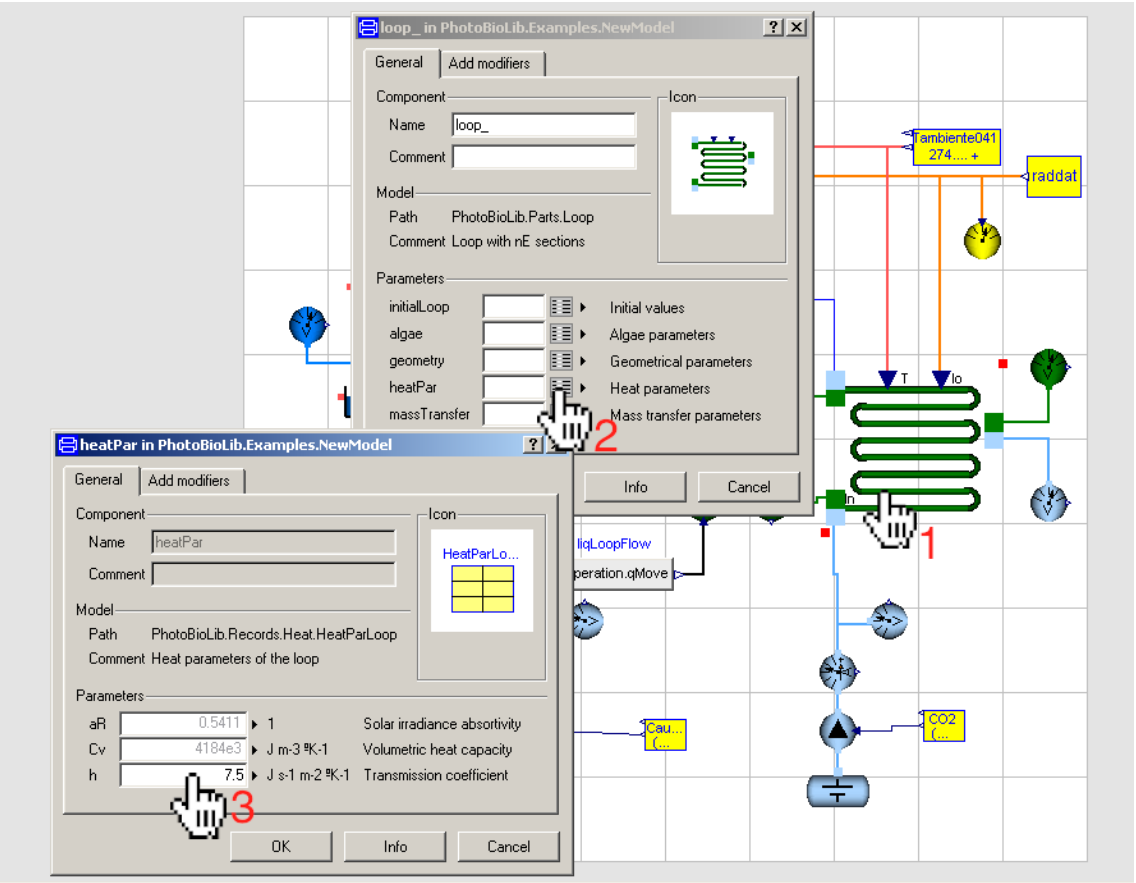
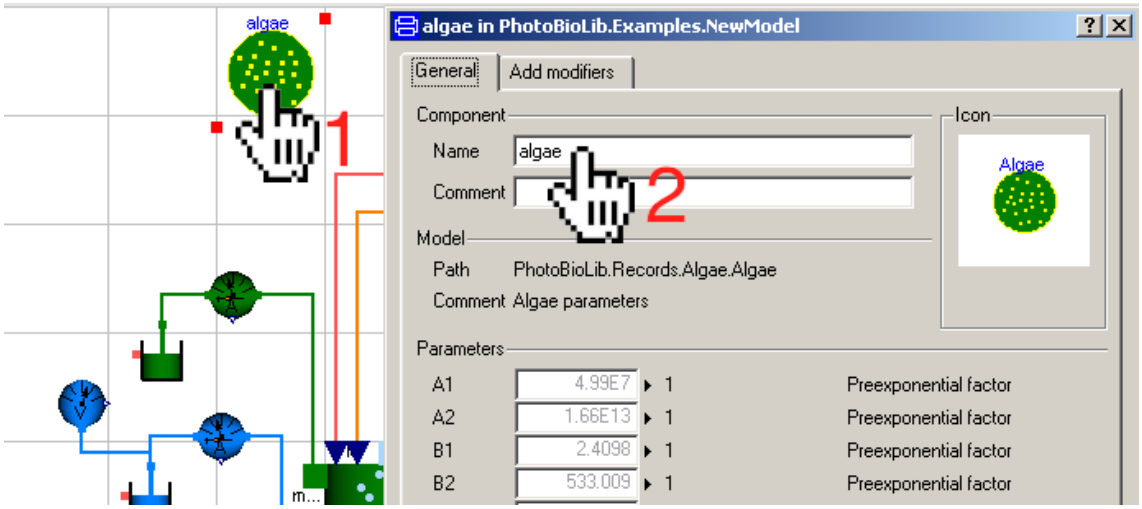
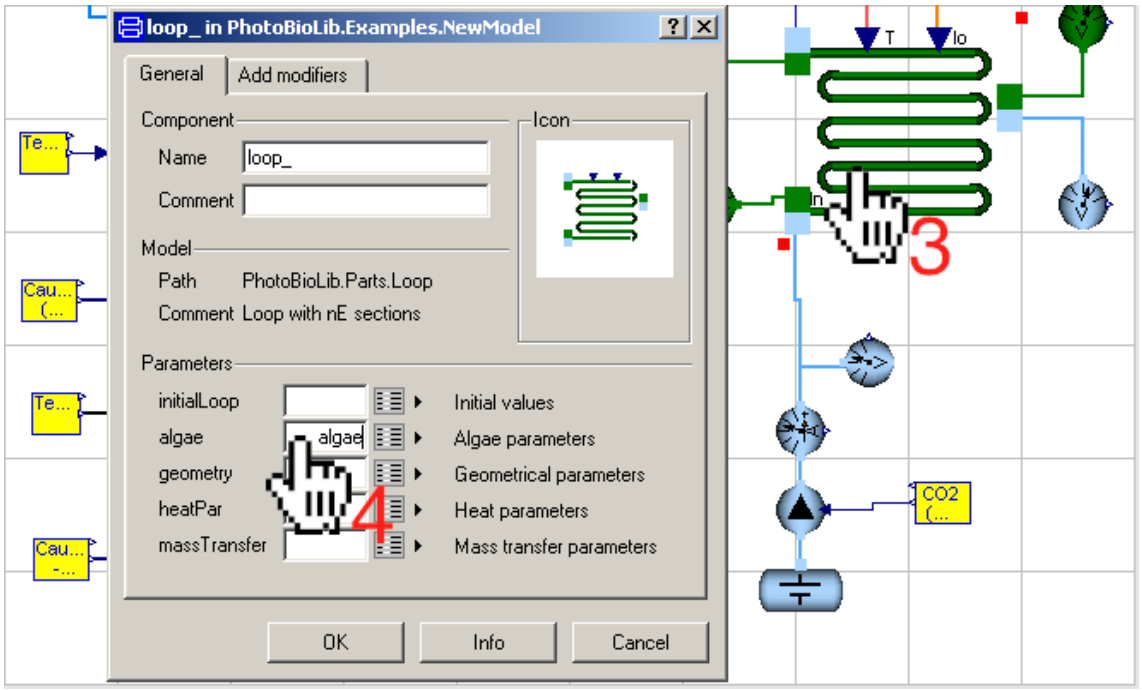


Figura 44: Cambio de un parámetro del modelo



(a) Pasos 1 y 2



(b) Pasos 3 y 4

Figura 45: Pasos para la modificación de parámetros

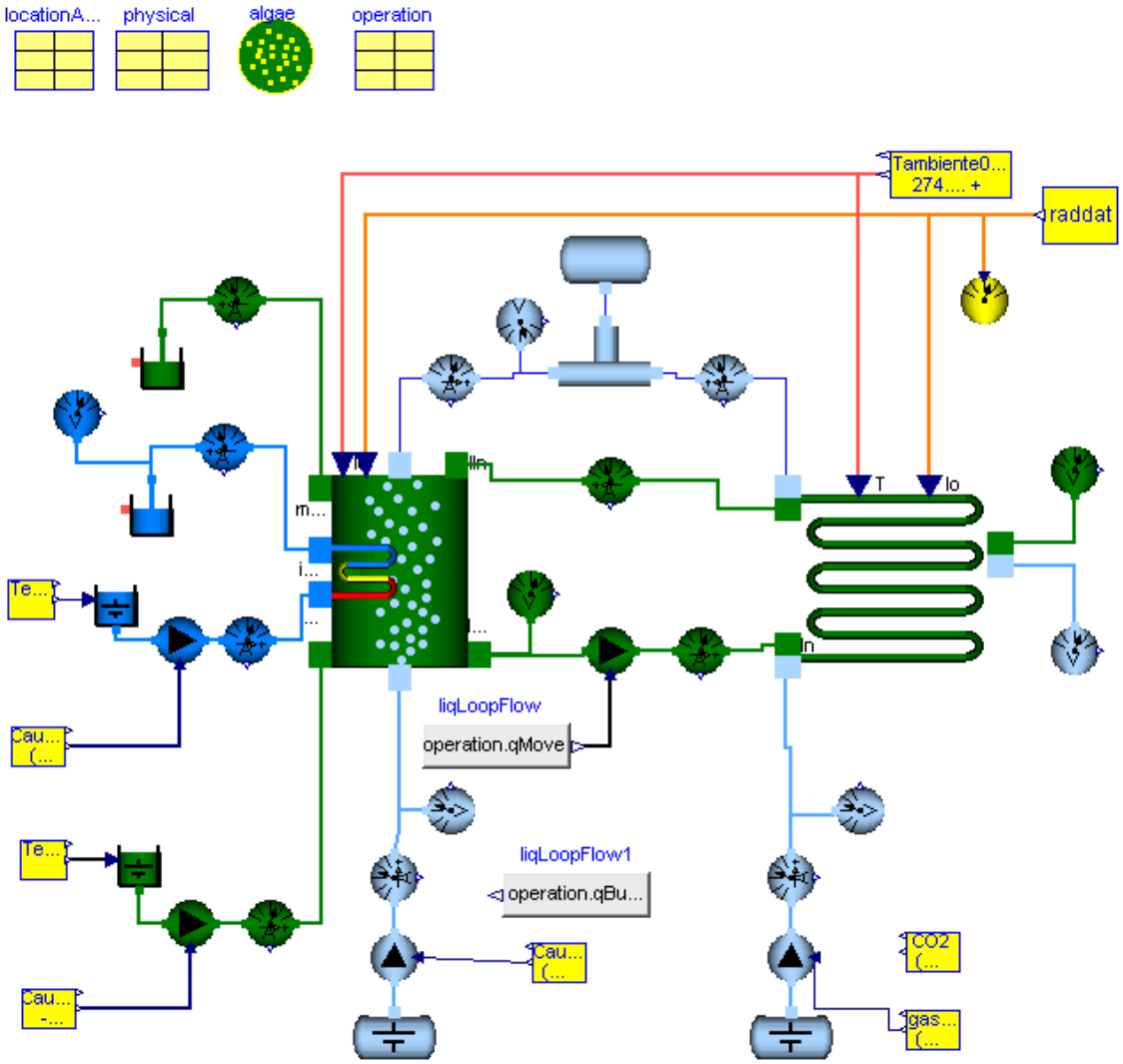


Figura 46: Diagrama de composición de *ClosedPhotobioreactor*

5. Resultados

En la Figura 46 se muestra la composición de un fotobiorreactor cerrado, el cual se corresponde con el descrito en [12] y representa a un sistema real que se encuentra en la *Estación Experimental de las Palmerillas*, en Almería.

La planta anteriormente mencionada está formada por una columna de burbujeo, un intercambiador de calor, un lazo, distintas bombas, inyectores de gas y sensores. La columna de burbujeo, además de mezclar el cultivo y extraer parte del oxígeno del mismo, a través de la inyección de aire en su base, es el punto en donde se inyecta nuevo medio de cultivo y de donde se extrae la biomasa. La inyección de medio de cultivo se realiza en la base y el cosechado tiene lugar por desbordamiento en la parte superior. Dentro la columna se encuentra el intercambiador de calor, encargado de calentar o enfriar el cultivo en función del caudal y temperatura del agua que pase por su interior. A la entrada y salida de la columna está conectado el lazo, formado por una tubería serpenteante en cuyo inicio de la misma se produce una inyección de CO_2 que alimenta el proceso de fotosíntesis que da como resultado la generación de nueva biomasa y O_2.

5.1. Tratamiento de datos

Tanto para las entradas al modelo como para la comprobación de las salidas se van a utilizar datos reales de sensores. Todas las señales han sido filtradas con un filtro paso bajo con una frecuencia de corte de $1/2\pi 60Hz$, además de eliminar aquellos valores sin sentido, como pueden ser los de radiación negativa. Posteriormente se han interpolado para cambiar el tiempo de muestro de $1s$ a $60s$. No obstante, no se han considerado problemas como posibles errores de sensorización o *offset* en las medidas, ya que no era el objetivo de este trabajo. Sin embargo, como se verá a continuación, esto no es inconveniente para comprobar que el modelo sigue de forma correcta la dinámica del sistema.

5.2. Entradas del sistema y perturbaciones

Las entradas y perturbaciones del sistema se pueden dividir en tres grupos: radiación, caudales y temperaturas. La diferenciación entre entradas o perturbaciones irá en función de si se puede o no actuar sobre las mismas, lo cual dependerá a su vez de los actuadores y controladores que se incorporen al sistema, por tanto para simplificar se consideraran todas por igual.

En la Figura 47 se muestra la evolución de la radiación solar, medida en $[\mu E\ m^{-2}\ s^{-1}]$, a lo largo de un día completo. Los datos corresponden con la radiación registrada un 4 de Febrero. Como ya se ha comentado anteriormente, los datos por debajo de 0 han sido eliminados al no tener sentido físico. Dicha radiación solar, entre otros factores, provocará la variación de la temperatura ambiente, tal y como se ve en la Figura 48.

Para la recepción de radiación solar en el mayor área posible por parte de las microalgas, al mismo tiempo reciben una inyección de CO_2, se hace circular el cultivo a través del lazo. El caudal de circulación a través del lazo en este caso se ha marcado por un valor fijo igual a $5{,}553\ 10^{-3}[m^3\ s^{-1}]$.

Para mantener la temperatura de cultivo en niveles óptimos para las microalgas, la columna de burbujeo dispone en su interior de un intercambiador de calor. El flujo de agua que pasa por

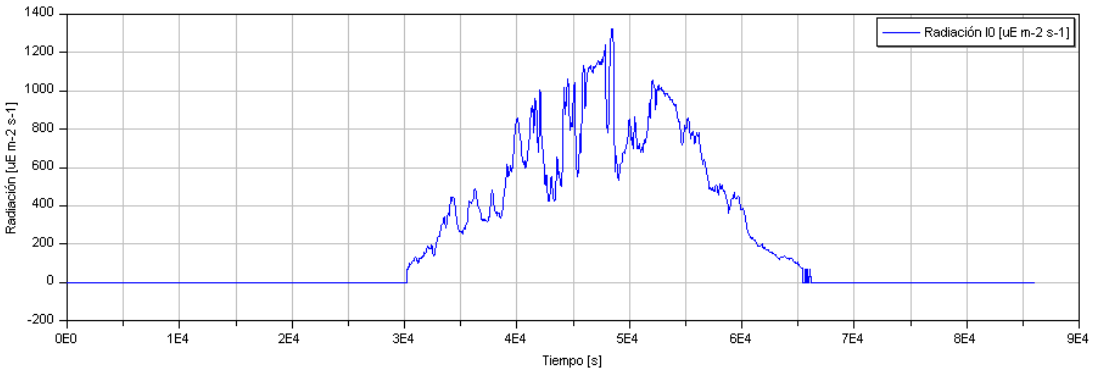


Figura 47: Evolución de la radiación solar

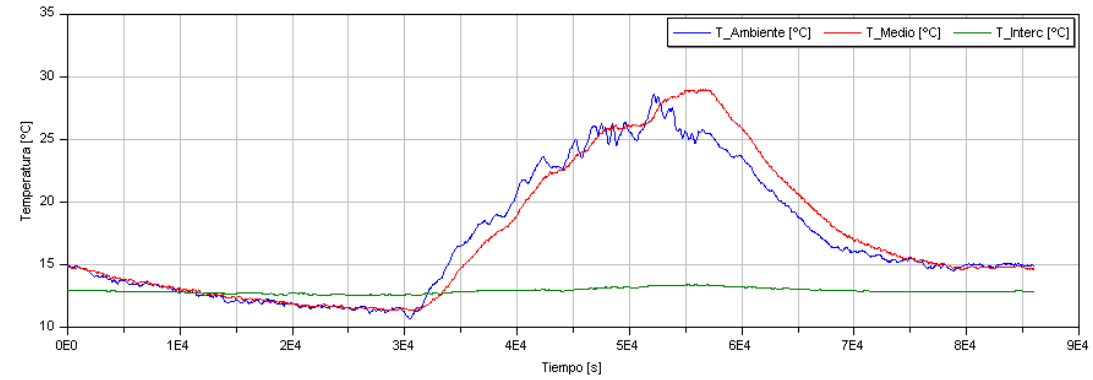


Figura 48: Temperaturas de entrada

su interior se muestra en la Figura 49. Por otro lado, la temperatura del agua a la entrada del intercambiador se describe, junto a otras temperaturas, en la Figura 48.

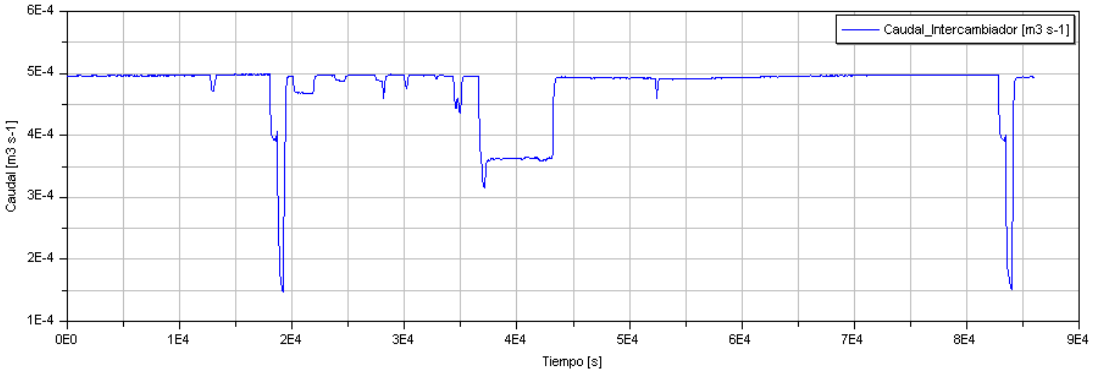


Figura 49: Caudal de líquido en el interior del intercambiador de calor

A través de la base de la columna de burbujeo se inyecta nuevo medio de cultivo sin biomasa, a su vez dicho volumen adicional provoca un desbordamiento en la parte superior de la columna que sirve como medio para realizar el cosechado. Por tanto, ambos volúmenes, el de medio de cultivo y de cosechado serán equivalentes. Se puede ver su evolución en la Figura 50. En la Figura 48 se muestra la temperatura a la que se encuentra el medio de cultivo a la hora de inyectarse en la columna.

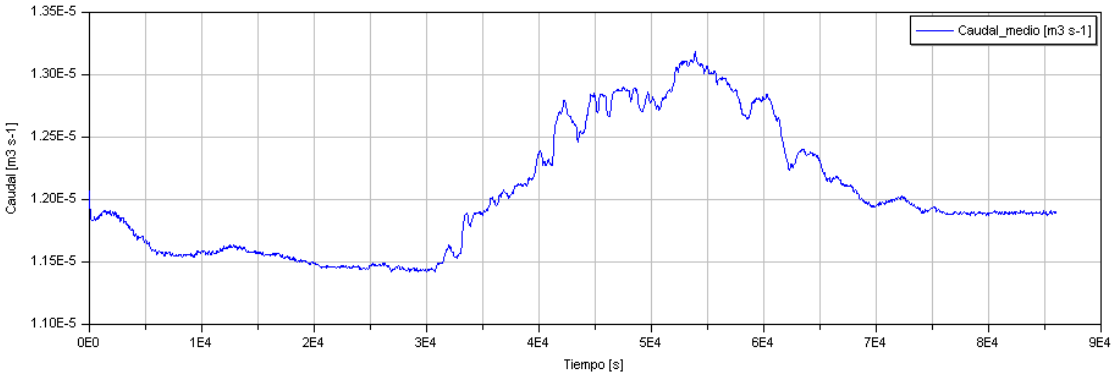


Figura 50: Caudal del medio de cultivo

Con el fin de modificar las concentraciones de O_2 y CO_2 disueltos en el medio de cultivo y a su vez mejorar el mezclado del mismo se producen dos inyecciones de gas en el fotobiorreactor. La primera de ellas se produce en la columna, donde se inyecta aire con concentraciones de gases iguales a las encontradas en el ambiente, ver Figura 51. La segunda se produce al inicio del lazo, el gas que aquí se inyecta tiene una alta concentración de CO_2 y su caudal se puede ver en la Figura 52.

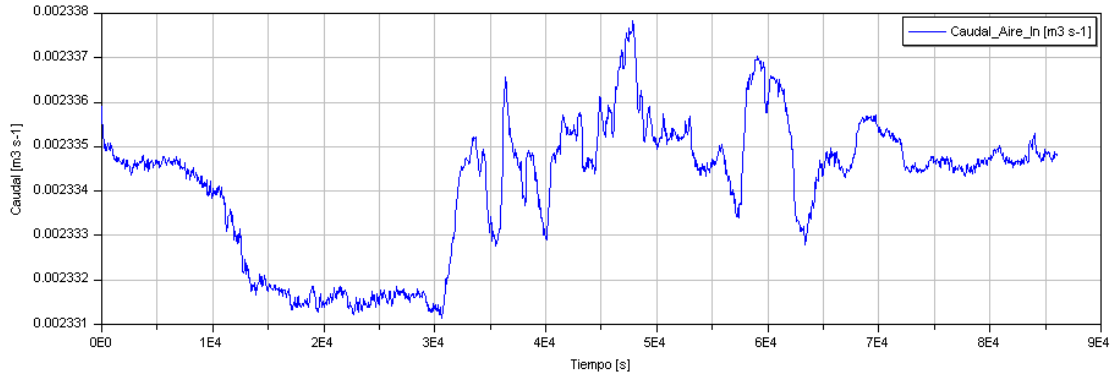


Figura 51: Caudal de inyección de aire en la columna de burbujeo

5.3. Salidas medibles y validación

Para la validación del modelo construido en la Sección 4 se disponen de medidas de sensores de temperatura, concentración de O_2 disuelto y de pH. Pese a que el seguimiento de la dinámica parece bastante bueno, se pueden observar errores achacables a tres motivos: ruidos en las

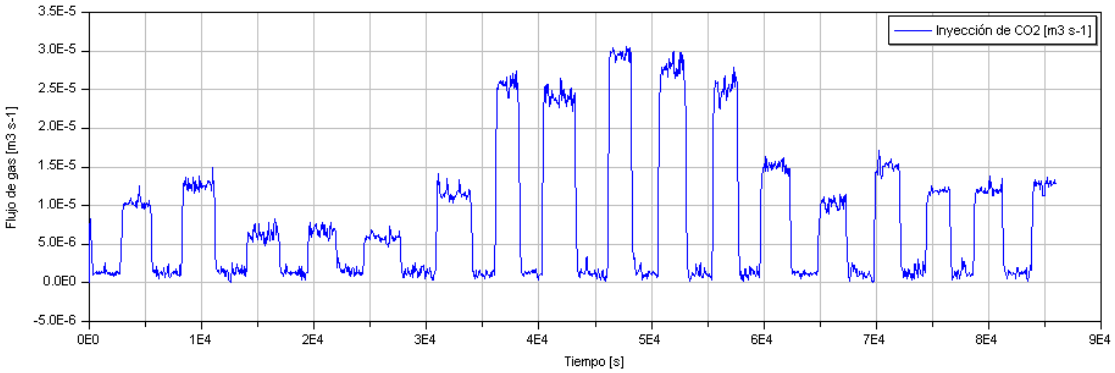


Figura 52: Caudal de inyección de CO_2 en el lazo

medidas, errores en la calibración de los sensores y errores en la calibración del modelo. Este último se pretende resolver en futuras versiones de la librería incluyendo un mecanismo para la calibración de los modelos diseñados a partir de medidas de sus equivalente reales. El resto de errores no se ha considerado su resolución ya que no era el objetivo del trabajo que se presenta. Para esta simulación se han utilizado valores extraídos de la bibliografía utilizada con pequeñas modificaciones introducidas de forma manual.

En las Figuras 53 y 54 se muestra la variación de la concentración de O_2 disuelto en el medio a la entrada y a la salida del lazo. Se puede observar como, a pesar del *offset*, el seguimiento de la dinámica es bastante bueno y refleja el efecto de la radiación solar en la producción de O_2 por parte de las microalgas.

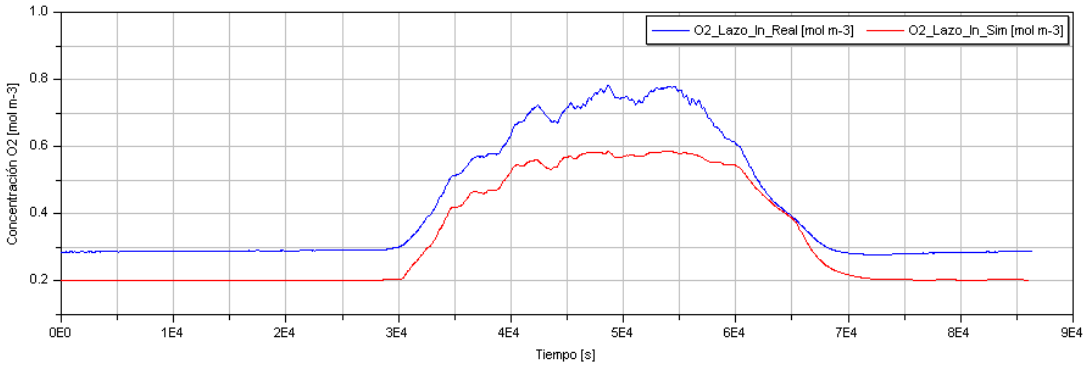


Figura 53: Concentración de O_2 disuelto a la entrada del lazo

En lo que se refiere al pH se dispone también de medidas del mismo a la entrada y a la salida del lazo, tal y como se muestra en las Figuras 55 y 56

Para la validación de la transferencia de calor se muestran las temperaturas medidas y simuladas a la entrada y salida del lazo en las Figuras 57 y 58. Además, en la Figura 59 se muestra la temperatura del agua a la salida del intercambiador, la cual refleja la absorción de calor en las horas de sol.

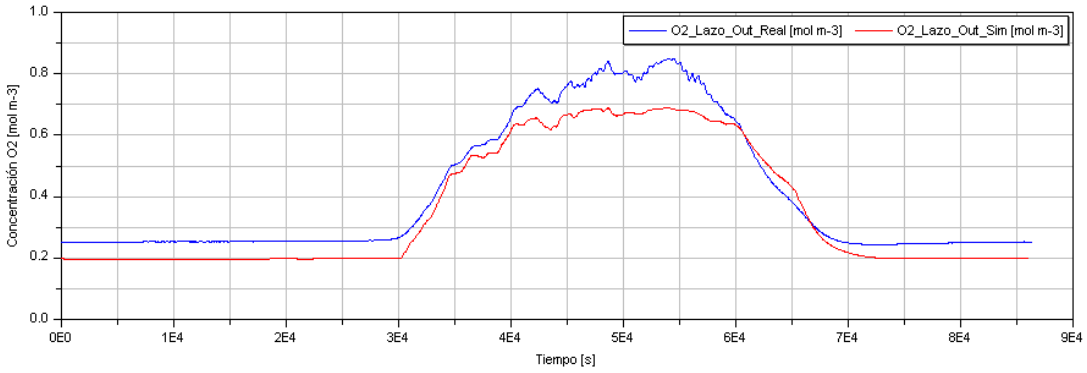


Figura 54: Concentración de O_2 disuelto a la salida del lazo

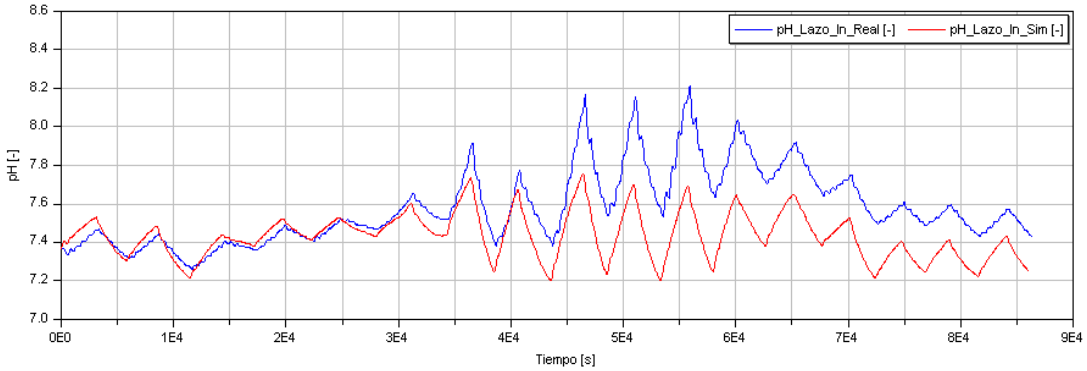


Figura 55: pH a la entrada del lazo

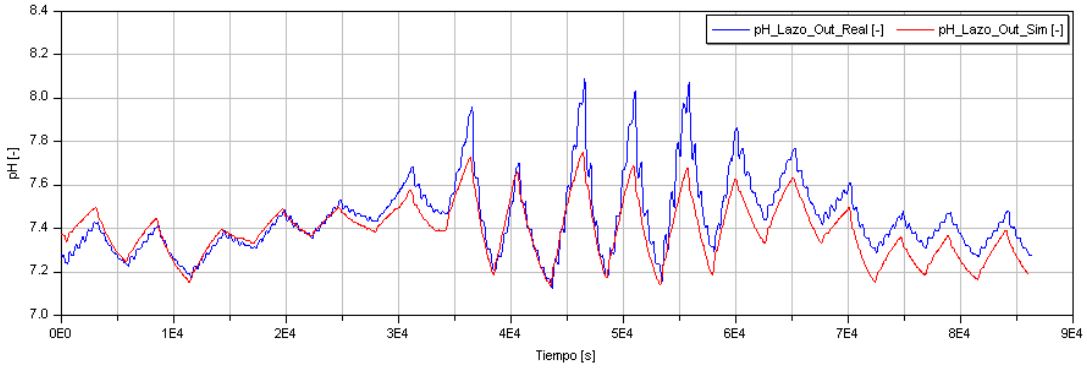


Figura 56: pH a la salida del lazo

5.4. Salidas no medibles

Uno de los grandes potenciales de la librería diseñada es que puede proporcionar medidas sobre variables que no se han considerado o no se pueden medir en el sistema real, ya sea en lo referente a las magnitudes o a las posiciones donde se realizan dichas mediciones. Además, el uso de los sensores virtuales diseñados permite simplificar el registro de los datos y colocar sensores

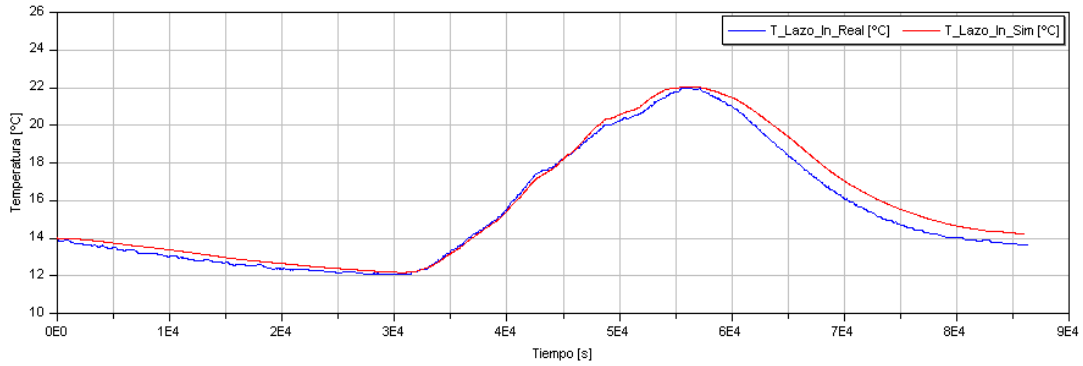


Figura 57: Temperatura del cultivo a la entrada del lazo

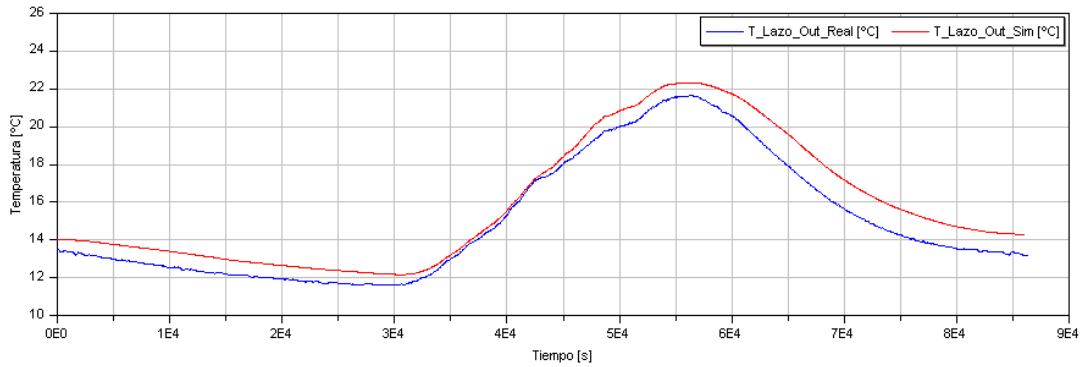


Figura 58: Temperatura del cultivo a la salida del lazo

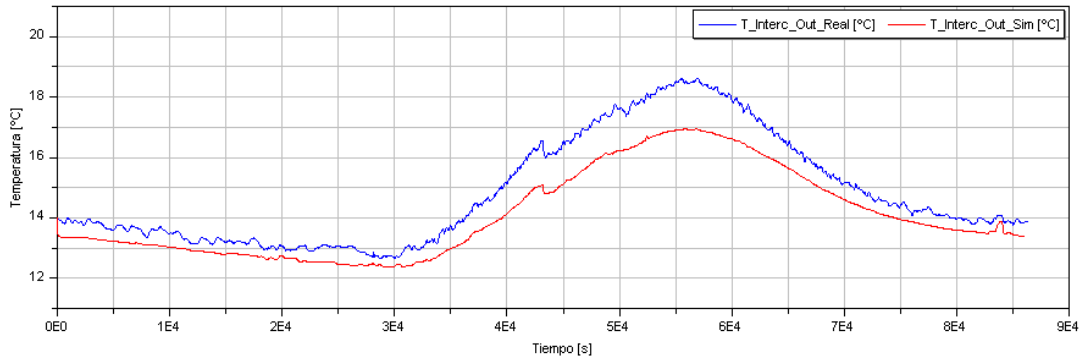


Figura 59: Temperatura del líquido a la salida del intercambiador

como se colocarían en un sistema real. Permitiendo además facilitar las decisiones sobre en que puntos es mejor colocar nuevos sensores en el sistema real.

Como ejemplo de lo anteriormente expuesto se muestran las medidas de la concentración de biomasa a la entrada y a la salida del lazo. En la Figura 60 se observa una tendencia de general en la caída de la concentración, motivada por el cosechado y la inyección de nuevo medio de

cultivo de biomasa, amortiguada con la generación de biomasa que se produce en las horas de sol.

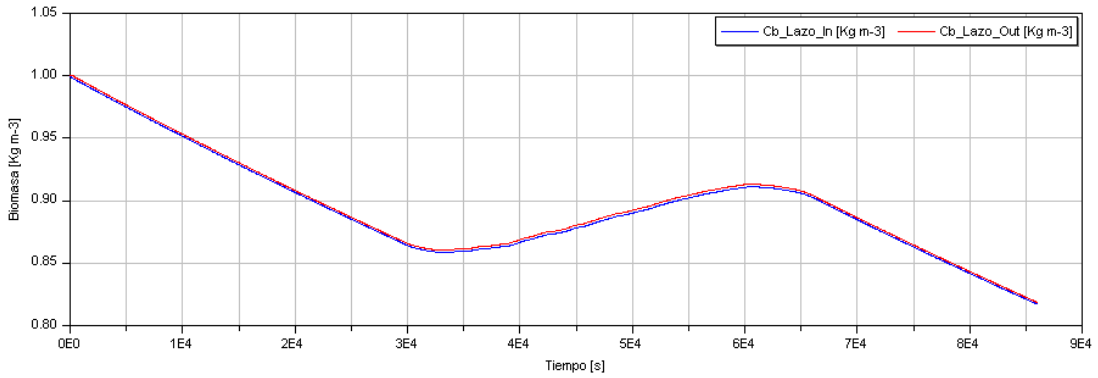


Figura 60: Evolución de la concentración de biomasa

Por último, en las Figuras 61 y 62 se muestran las fracciones molares de O_2 y de CO_2 en el gas que es liberado por el fotobiorreactor. Como era de esperar, la fracción molar de O_2, $yO_2(t)$, aumenta en las horas de sol coincidiendo con la fotosíntesis de las microalgas, por otra lado también se observa como, pese a las inyecciones de CO_2 en fase gaseosa en el lazo, éste es prácticamente absorbido por el cultivo y, por tanto, su fracción molar a la salida, $yCO_2(t)$, es mínima.

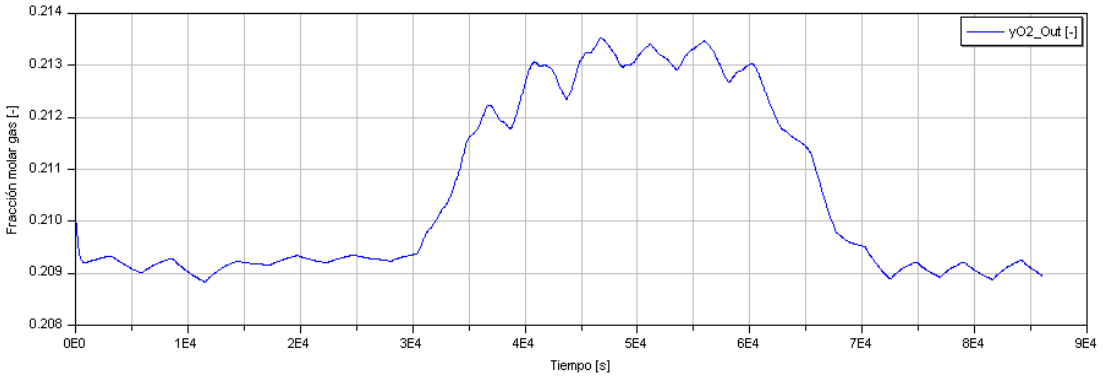


Figura 61: Fracción molar de O_2 en el gas de salida

5.5. Eficiencia de la librería

La librería presentada, además de permitir una construcción y configuración de modelos muy sencilla y potente, permitiendo además la ampliación con otras librerías, y tener una gran capacidad para la medición de cualquier variable en cualquier punto, permite realizar simulaciones con una cantidad considerable de datos de entrada en tiempos bastante reducidos.

En la Tabla 5 se muestran los tiempos que se han obtenido al simular el ejemplo presentado en la Sección 4 haciendo uso de distintos algoritmos. Todas las simulaciones se han realizado en

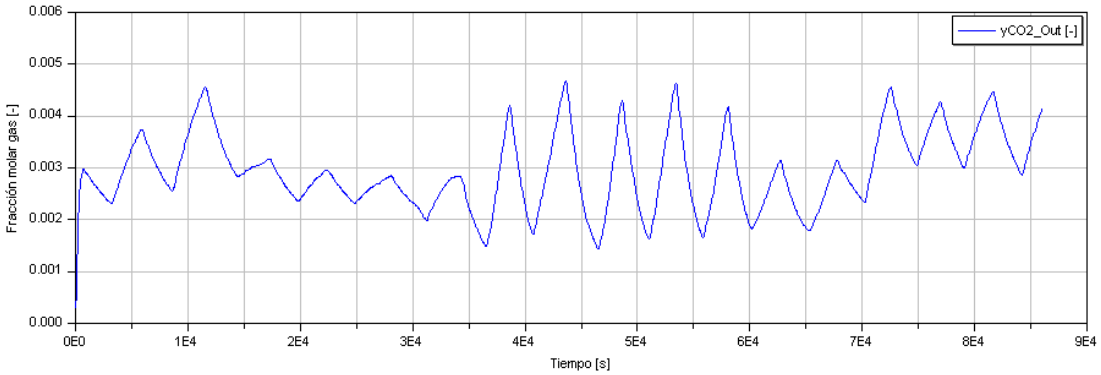


Figura 62: Fracción molar de CO_2 en el gas de salida

la versión 6.1 de *Dymola*, sobre una máquina virtual de *Windows XP*, a la cual solo se le permite el acceso a un único núcleo del procesador. La maquina virtual corre sobre un ordenador portatil *MacBook Pro* con un procesador *i7* de primera generación.

Tabla 5: Tiempos de simulación en función del algoritmo y la sensibilidad

Algoritmo	Sensibilidad	Tiempo de simulación
Dassl	10^{-5}	125,00 s
	10^{-10}	300,00 s
Dopri45 - order 5	10^{-5}	2,61 s
	10^{-10}	10,09 s
Dopri853 - order 8	10^{-5}	3,76 s
	10^{-10}	10,23 s
Esdirk45a - order 5 stiff	10^{-5}	48,34 s
	10^{-10}	155,00 s
Lsodar	10^{-5}	3,19 s
	10^{-10}	9,42 s
Radau lla - order 5 stiff	10^{-5}	39,76 s
	10^{-10}	119,14 s

Todos los resultados de la Tabla 5 han obtenido, al menos con una simple visualización de su evolución, las mismas salidas, además de detectar el mismo número de eventos de estado. Las gráficas de simulación que se muestran en esta sección se han obtenido utilizando el algoritmo *Dopri45* con una sensibilidad de 10^{-10}.

6. Conclusiones y líneas futuras de trabajo

En las secciones anteriores se ha presentado la librería diseñada para el modelado y simulación de fotobiorreactores *PhotoBioLib*, las principales ventajas que aporta el uso de esta librería sobre otras opciones son:

- Modelos basados en primeros principios. Lo cual aporta ventajas sobre modelos basados en funciones de transferencia, cajas negras u otras estrategias.
- Gran potencial de configuración. A partir de las partes preconstruidas y las clases básicas se podrá construir, sencilla y rápidamente, cualquier tipo de fotobiorreactor.
- Facilidad de uso. Puede ser utilizado por cualquier usuario sin necesidad de tener conocimientos previos de programación, arrastrando y uniendo las distintas partes.
- Gran parecido de los modelos con la realidad. Se pueden construir modelos añadiendo partes y sensores como si se tratara de un sistema real. Esto por un lado permite reproducir fácilmente plantes reales ya existentes y, por otro, el crear plantas o probar modificaciones de las mismas en simulación antes de implementarlo en la realidad.
- Velocidad de simulación. El uso del lenguaje *Modelica*, y más concretamente del entorno *Dymola*, permiten que modelos con miles de ecuaciones y tiempos largos de simulación, con pequeños muestreos, como el diseñado en la Figura 46, puedan simular días completos en menos de 3 segundos. Esto es posible debido a que la herramienta *Dymola* realiza un pretratamiento de las ecuaciones diferenciales para posteriormente traducir el modelo a lenguaje *C*, siendo esto transparente para el usuario.
- Gran facilidad para la interconexión con otros modelos o librerías. A los modelos creados con la librería *PhotoBioLib* se pueden incluir otros componentes ya disponibles en *Modelica* como bloques para el tratamiento de señales o controladores, o otras librerías como, por ejemplo, alguna que modele la temperatura dentro de un invernadero para simular el comportamiento de un fotobiorreactor dentro del mismo.

Respecto a las posibles mejoras que se pueden incluir en la librería se destacan las siguientes:

- Creación de componentes y ejemplos para la simulación de otros tipos de fotobiorreactores. En la actualidad ya está en desarrollo la ampliación de la librería para la simulación de fotobiorreactores abiertos del tipo *raceway*.
- Inclusión de modelos para la simulación de la radiación solar y la temperatura de cualquier parte del mundo en cualquier fecha deseada. En la próxima versión de la librería ya se va a incluir un modelo para la simulación de la radiación en función de la fecha, la longitud y la latitud.
- Implementación de un paquete de controladores. Especialmente de control predictivo basado en modelo.
- Posibilidad de sintonizar los parámetros y validar los modelos a partir de datos registrados de sistemas reales y de una forma más o menos automática.

Agradecimientos

El autor agradece el apoyo financiero del Ministerio de Economía y Competitividad en virtud de los proyectos DPI2011-27818-C02-01 y DPI2011-27818-C02-02, de la beca FPI enmarcada en este último y del proyecto CENIT VIDA financiado por el Ministerio de Economía y Competitividad y el CDTI.

Referencias

[1] http://www.aspentech.com/products/aspen-plus.aspx.

[2] http://www.comsol.com/comsol-multiphysics.

[3] http://www.ni.com/labview/esa/.

[4] http://www.ansys.com/Products/Simulation+Technology/Fluid+Dynamics/Fluid+Dynamics+Products/ANSYS+Fluent.

[5] http://www.mathworks.es/products/.

[6] http://meteonorm.com/products/meteonorm-software/.

[7] http://www.intelligen.com/superpro_overview.html.

[8] http://www.lambdares.com/tracepro.

[9] K. Cheenkachorn, N. Choosri, A. Chutrapukdeekul, and T. Kangsadan. Computational Modeling of Microalgae Culture Using a Helical Photobioreactor. In *Recent Advances in Fluid Mechanics and Heat & Mass Transfer*, volume 2, pages 300–305, 2011.

[10] T. A. Costache, F. G. Acién Fernández, M. M. Morales, J. M. Fernández-Sevilla, I. Stamatin, and E. Molina. Comprehensive model of microalgae photosynthesis rate as a function of culture conditions in photobioreactors. *Applied microbiology and biotechnology*, 97(17):7627–37, Sept. 2013.

[11] R. Davis, A. Aden, and P. T. Pienkos. Techno-economic analysis of autotrophic microalgae for fuel production. *Applied Energy*, 88(10):3524–3531, Oct. 2011.

[12] I. Fernández, F. G. Acién, J. M. Fernández, J. L. Guzmán, J. J. Magán, and M. Berenguel. Dynamic model of microalgal production in tubular photobioreactors. *Bioresource technology*, 126:172–81, Dec. 2012.

[13] V. Goetz, F. Le Borgne, J. Pruvost, G. Plantard, and J. Legrand. A generic temperature model for solar photobioreactors. *Chemical Engineering Journal*, 175:443–449, Nov. 2011.

[14] P. Iancu, V. Ple, and S. Velea. Flue Gas CO 2 Capture by Microalgae in Photobioreactor : a Sustainable Technology. *Chemical Engineering Transactions*, 29(2006):799–804, 2012.

[15] J. Miñón Martínez. *Modelo de biomasa algal para la captura de dióxido de carbono y su desarrollo en un software de evaluación*. PhD thesis, Universidad de Valladolid, 2012.

[16] V. Papáček, V. Štumbauer, D. Štys, K. Petera, and C. Matonoha. Growth impact of hydrodynamic dispersion in a Couette–Taylor bioreactor. *Mathematical and Computer Modelling*, 54(7-8):1791–1795, Oct. 2011.

[17] K. J. Sink. *Design and Construction of a Lighting System to Illuminate a Photobioreactor*. PhD thesis, Ohio University, 2011.

[18] B. Tamburic, F. W. Zemichael, P. Crudge, G. C. Maitland, and K. Hellgardt. Design of a novel flat-plate photobioreactor system for green algal hydrogen production. *International Journal of Hydrogen Energy*, 36(11):6578–6591, June 2011.

[19] L. B. Wu, Z. Li, and Y. Z. Song. Hydrodynamic conditions in designed spiral photobioreactors. *Bioresource technology*, 101(1):298–303, Jan. 2010.

A. Parámetros y variables

Tabla 6: Parámetros y variables

Parámetro / Variable	**Descripción**	**Valor y unidades**
α	Factor de distribución	$[-]$
A_1	Factor preexponencial	$[-]$
A_2	Factor preexponencial	$[-]$
B_1	Factor preexponencial	$[-]$
B_2	Factor preexponencial	$[-]$
aR	Absortividad de radiación solar	$[-]$
C_1	Energía de activación	$[-]$
C_2	Energía de activación	$[-]$
$C_b(t)$	Concentración de biomasa	$[Kg\ m^{-3}]$
$C_{bt}(t)$	Cantidad total de biomasa	$[Kg]$
$CO_2(t)$	Concentración de CO_2 en la fase líquida	$[mol\ m^{-3}]$
$CO_{2t}(t)$	Moles totales de CO_2 en la fase líquida	$[mol]$
$CO_{2g}(t)$	Concentración de CO_2 en la fase gaseosa	$[mol\ m^{-3}]$
$CO_{2gt}(t)$	Moles totales de CO_2 en la fase gaseosa	$[mol]$
$CO_2^*(t)$	Concentración en equilibrio de CO_2 con la fase gaseosa	$[mol\ m^{-3}]$
$C_T(t)$	Concentración de carbono inorgánico total en la fase líquida	$[mol\ m^{-3}]$
$C_{Tt}(t)$	Moles totales de carbono inorgánico total en la fase líquida	$[mol]$
C_v	Capacidad calorífica volumétrica	$[Jm^{-3}K^{-1}]$
d_t	Trayectoria de la luz	$[m]$
Ea_1	Energía de activación	$[mol\ J^{-1}]$
Ea_2	Energía de activación	$[mol\ J^{-1}]$
$eRad(t)$	Existencia de radiación solar	$[-]$
$eGas(t)$	Existencia de gas	$[-]$
$eLiq(t)$	Existencia de líquido	$[-]$
$\epsilon(t)$	Hold-up	$[-]$
$fC_{b,f}(t)$	Flujo de biomasa por la fotosíntesis	$[Kg\ s^{-1}]$
$fC_{b,m}(t)$	Flujo de biomasa por un flujo de líquido	$[Kg\ s^{-1}]$
$fO_{2,f}(t)$	Flujo de O_2 por la fotosíntesis	$[mol\ s^{-1}]$
$fO_{2,m}(t)$	Flujo de O_2 por un flujo de líquido	$[mol\ s^{-1}]$
$fO_{2,gl}(t)$	Flujo de O_2 por la transferencia de materia entre fases	$[mol\ s^{-1}]$
$fO_{2g,m}(t)$	Flujo de O_2 por un flujo de gas	$[mol\ s^{-1}]$

Parámetro / Variable	**Descripción**	**Valor y unidades**
$fCO_{2,f}(t)$	Flujo de CO_2 por la fotosíntesis	$[mol\ s^{-1}]$
$fCO_{2,m}(t)$	Flujo de CO_2 por un flujo de líquido	$[mol\ s^{-1}]$
$fCO_{2,gl}(t)$	Flujo de O_2 por la transferencia de materia entre fases	$[mol\ s^{-1}]$
$fCO_{2g,m}(t)$	Flujo de CO_2 por un flujo de gas	$[mol\ s^{-1}]$
$fN_{2g,m}(t)$	Flujo de N_2 por un flujo de gas	$[mol\ s^{-1}]$
h	Coeficiente de transmisión calorífica	$[Js^{-1}m^{-2}K^{-1}]$
H_{O_2}	Constante de Henry para el O_2 en el agua a 20	1,07 $[mol\ atm^{-1}m^{-3}]$
H_{CO_2}	Constante de Henry para el CO_2 en el agua a 20º.	38,36 $[mol\ atm^{-1}m^{-3}]$
$I_{av}(t)$	Radiación media en cada punto del reactor	$[\mu E\ m^{-2}\ s^{-1}]$
$I_c(t)$	Radiación solar	$[W\ m^{-2}]$
$I_0(t)$	Radiación solar en la superficie horizontal sin obstáculos	$[\mu E\ m^{-2}\ s^{-1}]$
K_a	Coeficiente de extinción de biomasa	$[m^2Kg^{-1}]$
K_i	Parametro de forma	$[uE\ m^{-2}s^{-1}]$
$K_{la,O_2}(t)$	Coeficiente de transferencia de masa volumetrico para el O_2	$[s^{-1}]$
$K_{la,CO_2}(t)$	Coeficiente de transferencia de masa volumetrico para el CO_2	$[s^{-1}]$
K_{O_2}	Constante de inhibición de oxigeno	0,7202 $[mol\ m^{-3}]$
K_w	Constante de equilibrio	$10^{-14}[-]$
K_1	Constante de equilibrio	$10^{-6,381}[-]$
K_2	Constante de equilibrio	$10^{-10,377}[-]$
m	Parámetro de forma	$[-]$
M_{O_2}	Peso molar del O_2	$3{,}2 \cdot 10^{-2}\ [Kg\ mol^{-1}]$
$M_{CO_2}]$	Peso molar del CO_2	$4{,}4 \cdot 10^{-2}\ [Kg\ mol^{-1}$
n	Exponente de forma	$[-]$
$N_{2g}(t)$	Concentración de N_2 en la fase gaseosa	$[mol\ m^{-3}]$
$N_{2gt}(t)$	Moles totales de N_2 en la fase gaseosa	$[mol]$
$O_2(t)$	Concentración de O_2 en la fase líquida	$[mol\ m^{-3}]$
$O_{2t}(t)$	Moles totales de O_2 en la fase líquida	$[mol]$
$O_{2g}(t)$	Concentración de O_2 en la fase gaseosa	$[mol\ m^{-3}]$
$O_{2gt}(t)$	Moles totales de O_2 en la fase gaseosa	$[mol]$
$O_2^*(t)$	Concentración en equilibrio de O_2 con la fase gaseosa	$[mol\ m^{-3}]$
$P_{O_2}(t)$	Producción de O_2, en función de la biomasa	$[s^{-1}]$
$P_{O_{2,max}}$	Tasa máxima de fotosíntesis	$[s^{-1}]$
$P_{CO_2}(t)$	Producción de CO_2, en función de la biomasa	$[s^{-1}]$

Parámetro / Variable	**Descripción**	**Valor y unidades**
$pH(t)$	Potencial de pH	$[-]$
P_T	Presión total	$[atm]$
$Q(t)$	Flujo de calor	$[Js^{-1}]$
$Q_t(t)$	Calor total	$[J]$
$Q_g(t)$	Flujo de gas	$[m^3s^{-1}]$
$Q_l(t)$	Flujo de líquido	$[m^3s^{-1}]$
r	Tasa de respiración	$[-]$
R	Constante universal de los gases ideales	8,314472 $[JK^{-1}mol^{-1}]$
S_{cond}	Superficie de contacto en transferencia de calor	$[m^2]$
S_{rad}	Superficie donde incide la radiación solar	$[m^2]$
$T(t)$	Temperatura del líquido	$[K]$
$V_g(t)$	Volumen de gas	$[m^{-3}]$
$V_l(t)$	Volumen de líquido	$[m^3]$
Y_{bo}	Coeficiente de rendimiento de biomasa	$[-]$
$yO_2(t)$	Fracción molar de O_2 en la fase gaseosa	$[-]$
$yCO_2(t)$	Fracción molar de CO_2 en fase gaseosa	$[-]$
$yN_2(t)$	Fracción molar de N_2 en fase gaseosa	$[-]$
$y_T(t)$	Suma de las fracciones molares	$[-]$
z	Parámetro de forma	$[-]$

B. Documentación

En las páginas a continuación de ésta se incluye la documentación de referencia de la librería *PhotoBioLib*. Hay que tener en consideración que es una librería en pleno desarrollo, la librería será objeto de una tésis doctoral del mismo autor que este trabajo. Por esto, en la documentación se pueden encontrar comentarios en el código, así como clases y paquetes a los que no se hacen referencia y forman parte de futuras ampliaciones de la librería.

PhotoBioLib

Package for modeling photobioreactors. TFM version

Package Content

Name	Description
Basic	Basic elements
Examples	Examples
Icons	Icons package
Interfaces	Interfaces package
Parts	Parts package
Records	Records package
Sensors	Sensors package
UsersGuide	Users Guide
Utilities	Utilities package

PhotoBioLib.Basic

Basic elements

Package Content

Name	Description
Biomass	Basic biomass elements
Gas	Basic gas elements
Heat	Basic heat elements
Liquid	Basic liquid elements

PhotoBioLib.Basic.Biomass

Basic biomass elements

Package Content

Name	Description
AlgaeSource	Algae activity
BioCapacitor	Liquiq capacitor
BioResistor	Liquid resistor
ColumnSection	Column Section
FlowRegulator_LiqFCons	Flow regulator with constant liquid volume
FlowRegulator_LiqFVar	Flow regulator with variable liquid volume
LiqGasResistor_Column	Mass transfer in the column
LiqGasResistor_Loop	Resistor de fase líquida a gasesosa, lazo
LiqGasResistor_Partial	Mass transfer
LoopSection	Loop section

PhotoBioLib.Basic.Biomass.AlgaeSource

Algae activity

Parameters

Type	Name	Default	Description
Geometry	geometry		Geometrical parameters
Real	alpha	geometry.alpha	Distribution solar factor [1]
Real	dt	geometry.dt	Total diameter [m]
Algae	algae		Algae parameters
Real	A1	algae.A1	Preexponential factor [1]
Real	A2	algae.A2	Preexponential factor [1]
Real	B1	algae.B1	Preexponential factor [1]
Real	B2	algae.B2	Preexponential factor [1]
Real	C1	algae.B1	Activation energie [1]
Real	C2	algae.C2	Activation energie [1]
Real	Ea1	algae.Ea1	Activation energie [mol J-1]
Real	Ea2	algae.Ea2	Activation energie [mol J-1]
Real	Ka	algae.Ka	Extinction coefficient [m2 Kg-1]
Real	Ki	algae.Ki	Form parameter [uE m-2 s-1]
Real	KO2	algae.KO2	Oxygen inhibition constant [mol m-3]
Real	m	algae.m	Form parameter [1]
Real	n	algae.n	Form exponent [1]
Real	nO2	algae.nO2	Form parameter [1]
Real	PO2max	algae.PO2max	Maximum photosynthesis rate [KgO2 KgB-1 s-1)=1/s]
Real	r	algae.r	Respiration factor [1]
Real	R	algae.R	Gas constant [J K-1 mol-1]
Real	Ybo	algae.Ybo	Biomass yield coefficient [KgB KgO2-1]
Real	z	algae.z	Form parameter [1]
Real	epsAlgae	15	

Connectors

Type	Name	Description
BioCon	lCon	Liquid connector
input RealInput	Io	Input radiation [uE m-2 s-1]

Modelica definition

```
model AlgaeSource "Algae activity"
  // Icon and connectors
  extends PhotoBioLib.Interfaces.Partials.Biomass.BiolRad1;
  extends PhotoBioLib.Icons.Basic.Biomass.AlgaeSourceIcon;

  // Constants and math functions
  import Modelica.Math.log10;
  import Modelica.Math.exp;
  import PhotoBioLib.Records.Constants "Constants";
  constant Real MO2(unit="Kg mol-1") = Constants.MO2
    "Molecular weight of oxygen";
  constant Real MCO2(unit="Kg mol-1") = Constants.MCO2
    "Molecular weight of carbon dyoxide";

  //
  // Parameters
  //

  // Geometrical parameters - Cambiar
  parameter PhotoBioLib.Records.Geometry.Geometry geometry
    "Geometrical parameters";
  parameter Real alpha(unit="1") = geometry.alpha "Distribution solar factor";
  parameter Real dt(unit="m") = geometry.dt "Total diameter";

  // Algae parameters
  parameter PhotoBioLib.Records.Algae.Algae algae "Algae parameters";
  parameter Real A1(unit="1") = algae.A1 "Preexponential factor";
  parameter Real A2(unit="1") = algae.A2 "Preexponential factor";
  parameter Real B1(unit="1") = algae.B1 "Preexponential factor";
  parameter Real B2(unit="1") = algae.B2 "Preexponential factor";
  parameter Real C1(unit="1") = algae.B1 "Activation energie";
  parameter Real C2(unit="1") = algae.C2 "Activation energie";
  parameter Real Ea1(unit="mol J-1") = algae.Ea1 "Activation energie";
  parameter Real Ea2(unit="mol J-1") = algae.Ea2 "Activation energie";
  parameter Real Ka(unit="m2 Kg-1") = algae.Ka "Extinction coefficient";
  parameter Real Ki(unit="uE m-2 s-1") = algae.Ki "Form parameter";
  parameter Real KO2(unit="mol m-3") = algae.KO2 "Oxygen inhibition constant";
  parameter Real m(unit="1") = algae.m "Form parameter";
  parameter Real n(unit="1") = algae.n "Form exponent";
  parameter Real nO2(unit="1") = algae.nO2 "Form parameter";
  parameter Real PO2max(unit="KgO2 KgB-1 s-1)=1/s") = algae.PO2max
    "Maximum photosynthesis rate";
  parameter Real r(unit="1") = algae.r "Respiration factor";
  parameter Real R(unit="J K-1 mol-1") = algae.R "Gas constant";
  parameter Real Ybo(unit="KgB KgO2-1") = algae.Ybo "Biomass yield coefficient";
  parameter Real z(unit="1") = algae.z "Form parameter";

  //
  // Variables
  //
  Real Iav(unit="uE m-2 s-1") "Average solar irradiance";
  Real PO2(unit="KgO2 KgB-1 s-1") "Photosysthesis rate";
  Real PCO2(unit="KgCO2 KgB-1 s-1") "Carbon dioxide consuption rate";
  Real pH(unit="1") "pH";

  // Event variables
  Boolean eRad "Availability of radiation";
  Boolean eO2 "Availability of oxygen";

  Real aux;
```

```
  parameter Real epsAlgae = 15;//15;

initial equation
  eRad = Io>epsAlgae;
  eO2 = lCon.O2>0;

equation
  when {pre(eRad) and Io < epsAlgae, not pre(eRad) and Io>epsAlgae} then //< <=
    eRad = Io>epsAlgae;
  end when;

  when {pre(eO2) and lCon.O2 <= 0, not pre(eO2) and lCon.O2>0} then
    eO2 = lCon.O2 >0;
  end when;

  lCon.flowQ = 0;

  if eRad then
    Iav = (Io*alpha)/(Ka*lCon.Cb*dt)*(1 - exp(-Ka*dt*lCon.Cb));
    pH = -log10(lCon.H);
    aux = noEvent(if lCon.O2>0 then (1-(lCon.O2/KO2)^z) else 1);
    //aux = 1-(lCon.O2/KO2)^z;
    PO2 = (PO2max*Iav^n)/(Ki*exp(Iav*m) + Iav^n) * aux *(B1*exp(-C1/pH) -B2*exp(-C2/pH))
        *(A1*exp(-Ea1/(R*(lCon.T))) -A2*exp(-Ea2/(R*(lCon.T)))) -r*PO2max;//*(1-exp(-lCon.O2*nO2));

    PCO2 = -PO2;
  else
    Iav = 0;
    pH =0;
    aux=0;
    //PO2 = noEvent(if lCon.O2 >0 then -r*PO2max else 0);
    //PO2 = -r*PO2max;
    PO2 = -r*PO2max;//*(1-exp(-lCon.O2*nO2));//(1-(1E-10)^lCon.O2);
    //PO2 = if eO2 then -r*PO2max else 0;
    PCO2 = -PO2;
  end if;

  -lCon.flowCb = lCon.Vol*PO2*lCon.Cb*Ybo;
  -lCon.flowO2 = lCon.Vol*PO2*lCon.Cb/MO2;
  -lCon.flowCt = lCon.Vol*PCO2*lCon.Cb/MCO2;

  lCon.flowVol = 0;
  lCon.flowCO2 = 0;
  lCon.flowH = 0;
end AlgaeSource;
```

PhotoBioLib.Basic.Biomass.BioCapacitor

Liquiq capacitor

Parameters

Type	Name	Default	Description
replaceable class	con	Liq1Heat1	
replaceable class	icon	LiqCapacitorIcon	Icon
replaceable class	initialValues	InitialLiq	
InitialBio	initialLiq	extends PhotoBioLib.Records....	Initial liquid values
HeatPar	heatPar		Heat parameters
Real	Cv	heatPar.Cv	Volumetric heat capacity [J m-3 ºK-1)]
Real	Volini	initialLiq.Volini	Initial liquid volume [m3]
Real	Tini	initialLiq.Tini	Inititat temperature [ºK]
Real	Cbini	initialLiq.Cbini	Initial biomass concentration [Kg m-3]
Real	O2ini	initialLiq.O2ini	Initial oxygen concentration [mol m-3]
Real	CO2ini	initialLiq.CO2ini	Initial carbon dioxide concetration [mol m-3]
Real	Ctini	initialLiq.Ctini	Initial total inorganic carbon concentration [mol m-3]
Real	Hini	initialLiq.Hini	Initial hidrogen concentration [mol m3-]

Connectors

Type	Name	Description
replaceable class	con	
BioCon	lCon	Liquid connector
HeatCon	hCon	Heat connector
replaceable class	icon	Icon
replaceable class	initialValues	

Modelica definition

```
model BioCapacitor "Liquiq capacitor"
/* Pendiente
    No se considera que el volumen puede ser 0 o bajar. Útil para tanques vacios
*/
    extends PhotoBioLib.Basic.Liquid.LiqCapacitor(
    redeclare class icon = PhotoBioLib.Icons.Basic.Biomass.BioCapacitorIcon,
    redeclare class con = PhotoBioLib.Interfaces.Partials.Biomass.BioHeat1,
    redeclare class initialValues = PhotoBioLib.Records.Initial.InitialBio);

  import Modelica.Math.log10;
  import PhotoBioLib.Records.Constants;

  // Equilibrium constants
  constant Real Kw(unit="1") = Constants.Kw "Equilibrium constant";
  constant Real K1(unit="1") = Constants.K1 "Equilibrium constant";
  constant Real K2(unit="1") = Constants.K2 "Equilibrium constant";

  // Parámetros para marcar potenciales iniciales
  parameter Real Cbini(unit="Kg m-3") = initialLiq.Cbini
    "Initial biomass concentration";
  parameter Real O2ini(unit="mol m-3") = initialLiq.O2ini
    "Initial oxygen concentration";
  parameter Real CO2ini(unit="mol m-3") = initialLiq.CO2ini
    "Initial carbon dioxide concetration";
  parameter Real Ctini(unit="mol m-3") = initialLiq.Ctini
    "Initial total inorganic carbon concentration";
  parameter Real Hini(unit="mol m3-") = initialLiq.Hini
    "Initial hidrogen concentration";

  // Variables
  Real Cbt(unit="Kg",start=Cbini*Volini,fixed=false)
    "Total kilograms of biomass";
  Real O2t(unit="mol",start=O2ini*Volini,fixed=false) "Total moles of oxygen";
  Real Ctt(unit="mol",start=Ctini*Volini,fixed=false)
    "Total moles of inorganic carbon";
  Real pH(unit="1") "pH";

initial equation
  lCon.H =Hini;
  lCon.CO2 =CO2ini;

equation

  when O2t <0 then
    reinit(O2t,0);
  end when;

  when Ctt<0 then
    reinit(Ctt,0);
  end when;

  // Integración de componentes
  der(Cbt) = lCon.flowCb;
  der(O2t) = lCon.flowO2;
  der(Ctt) = lCon.flowCt;

  if eVol then
    lCon.Cb = Cbt/lCon.Vol;
    lCon.O2 = O2t/lCon.Vol;
    lCon.Ct = Ctt/lCon.Vol;
  else
    lCon.Cb = 0;
    lCon.O2 = 0;
    lCon.Ct = 0;
  end if;

  pH = -log10(lCon.H);
  // Cálculo de CO2 e H
  der(lCon.Ct) = (1 + K1/lCon.H + K1*K2/lCon.H^2)*der(lCon.CO2) - lCon.CO2*(K1/lCon.H^2
    + 2*K1*K2/lCon.H^3)*der(lCon.H);
  der(lCon.H) = (K1/lCon.H + 2*K1*K2/lCon.H^2)/(1 + Kw/lCon.H^2 + K1*lCon.CO2/lCon.H^2
    + 4*2*K1*K2/lCon.H^3)*der(lCon.CO2);// + lCon.flowH;

end BioCapacitor;
```

PhotoBioLib.Basic.Biomass.BioResistor

Liquid resistor

Parameters

Type	Name	Default	Description
replaceable class	con	Liq2Flow1	
replaceable class	icon	LiqResistorIcon	
HeatPar	heatPar		Heat parameters
Real	Cv	heatPar.Cv	Volumetric heat capacity [J m-3 ºK-1)]

Connectors

Type	Name	Description
replaceable class	con	
BioCon	con1	Liquid connector on the left

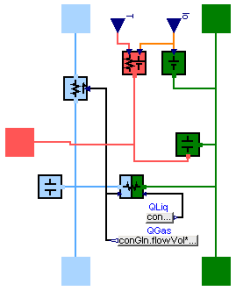
BioCon	con2	Liquid connector on the rigth
input RealInput	qLiq	Input liquid flow [m3 s-1]
replaceable class icon		

Modelica definition

```
model BioResistor "Liquid resistor"

  extends PhotoBioLib.Basic.Liquid.LiqResistor(redeclare class icon =
        PhotoBioLib.Icons.Basic.Biomass.BioResistorIcon, redeclare class con =
        PhotoBioLib.Interfaces.Partials.Biomass.Bio2Flow1);

equation
  con1.flowO2 = con1.O2*con1.flowVol;
  con2.flowO2 = -con1.flowO2;
  con1.flowCt = con1.Ct*con1.flowVol;
  con2.flowCt = -con1.flowCt;
  con1.flowCb = con1.Cb*con1.flowVol;
  con2.flowCb = -con1.flowCb;

  con1.flowH = 0;
  //con1.flowH = con1.H*con1.flowVol;
  con2.flowH = -con1.flowH;

  con2.flowCO2 = -con1.flowCO2;
  con2.flowCO2 = 0;

end BioResistor;
```

PhotoBioLib.Basic.Biomass.ColumnSection

Column Section



Parameters

Type	Name	Default	Description
GeometryColumn	geometry		Geometry parameters
HeatParColumn	heatPar		Heat parameters
MassTransferColumn	massTransfer		Mass transfer parameters
InitialGas	initialGas		Initial gas values
InitialBio	initialLiq		Initial liq values
Algae	algae		Algae parameters

Connectors

Type	Name	Description
BioCon	conLIn	Liquid connector on the left
BioCon	conLOut	Liquid connector on the rigth
GasCon	conGOut	Gas connector on the left
GasCon	conGIn	Gas connector on the rigth
input RealInput	Io	Input radiation [uE m-2 s-1]
input RealInput	T	Input temperature [°K]
HeatCon	hCon	Heat connector

Modelica definition

```
model ColumnSection "Column Section"
  extends PhotoBioLib.Interfaces.Partials.Biomass.Bio2Gas2RadlTlHeatl_V;
  extends PhotoBioLib.Icons.Basic.Biomass.ColumnSectionIcon;

  parameter PhotoBioLib.Records.Geometry.GeometryColumn geometry
    "Geometry parameters";
  parameter PhotoBioLib.Records.Heat.HeatParColumn heatPar "Heat parameters";
  parameter PhotoBioLib.Records.MassTransfer.MassTransferColumn massTransfer
    "Mass transfer parameters";

  parameter PhotoBioLib.Records.Initial.InitialGas initialGas
    "Initial gas values";
  parameter PhotoBioLib.Records.Initial.InitialBio initialLiq
    "Initial liq values";
  parameter PhotoBioLib.Records.Algae.Algae algae "Algae parameters";

  PhotoBioLib.Basic.Biomass.AlgaeSource algaeSource(algae=algae, geometry=
        geometry);
  PhotoBioLib.Basic.Gas.GasCapacitor gasCapacitor(initialGas=initialGas);
  PhotoBioLib.Basic.Gas.GasResistor gasResistor;
  PhotoBioLib.Basic.Biomass.BioCapacitor liqCapacitor(
    Volini=geometry.Vol,
    initialLiq=initialLiq,
    heatPar=heatPar);
  LiqGasResistor_Column liqGasResistor_Column(
      geometry=geometry,
      massTransfer=massTransfer);
  Modelica.Blocks.Sources.RealExpression QLiq(
    y=conLIn.flowVol);
  Modelica.Blocks.Sources.RealExpression QGas(
    y=conGIn.flowVol*conGIn.Vol/geometry.Vol);
  PhotoBioLib.Basic.Heat.HeatConvRad heatConvRad(geometry=geometry, heatPar=
        heatPar);

equation
  connect(conGOut, gasResistor.con2);
  connect(gasResistor.con1, conGIn);
  connect(gasCapacitor.con, gasResistor.con1);
  connect(liqGasResistor_Column.conG, gasCapacitor.con);
  connect(QGas.y, liqGasResistor_Column.Qgas);
  connect(QGas.y, gasResistor.qGas);
  connect(QLiq.y, liqGasResistor_Column.Qliq);
  connect(conLOut, liqGasResistor_Column.conL);
  connect(Io, algaeSource.Io);
  connect(Io, heatConvRad.Io);
  connect(T, heatConvRad.Text);
  connect(liqCapacitor.lCon, conLIn);
  connect(liqCapacitor.lCon, conLOut);
  connect(algaeSource.lCon, liqCapacitor.lCon);
  connect(heatConvRad.hCon, liqCapacitor.hCon);
  connect(hCon, liqCapacitor.hCon);
end ColumnSection;
```

PhotoBioLib.Basic.Biomass.FlowRegulator_LiqFCons

Flow regulator with constant liquid volume

Parameters

Type	Name	Default	Description
Geometry	geometry		Geometrical parameters
Real	At	geometry.At	Section area [m2]
Real	dX	geometry.dX	Length [m]

Connectors

Type	Name	Description
input RealInput	Qlin	Input liquid flow [m3 s-1]
output RealOutput	Qlout	Output liquid flow [m3 s-1]
output RealOutput	Qgout	Output gas flow [m3 s-1]
GasCon	gasCon	Gas connector

Modelica definition

```
model FlowRegulator_LiqFCons
  "Flow regulator with constant liquid volume"
  extends PhotoBioLib.Interfaces.Partials.Biomass.BiolGasIn_Vel_InOut;
  extends PhotoBioLib.Icons.Basic.Biomass.FlowRegulatorIcon;

  parameter PhotoBioLib.Records.Geometry.Geometry geometry
    "Geometrical parameters";
  parameter Real At(unit="m2")= geometry.At "Section area";
  parameter Real dX(unit="m")=geometry.dX "Length";

 /* Boolean eVol;

initial equation
  eVol = gasCon.Vol>0;*/

equation
  /*when {pre(eVol) and gasCon.Vol <=0, not pre(eVol) and gasCon.Vol >0} then
    eVol = gasCon.Vol>0;
  end when;*/
```

```
  Qlout = Qlin;
  Qgout = Qlin/At * gasCon.Vol/dX;
  //Qgout = if eVol then Qlin/At * gasCon.Vol/dX else 0;

  gasCon.flowO2 = 0;
  gasCon.flowCO2 = 0;
  gasCon.flowN2 = 0;
  gasCon.flowVol = 0;

end FlowRegulator_LiqFCons;
```

PhotoBioLib.Basic.Biomass.FlowRegulator_LiqFVar

Flow regulator with variable liquid volume

Parameters

Type	Name	Default	Description
Geometry	geometry		Geometrical parameters
Real	At	geometry.At	Section area [m2]
Real	dX	geometry.dX	Length [m]

Connectors

Type	Name	Description
input RealInput	Qlin	Input liquid flow [m3 s-1]
output RealOutput	Qlout	Output liquid flow [m3 s-1]
output RealOutput	Qgout	Output gas flow [m3 s-1]
GasCon	gasCon	Gas connector

Modelica definition

```
model FlowRegulator_LiqFVar
  "Flow regulator with variable liquid volume"
  extends PhotoBioLib.Interfaces.Partials.Biomass.BiolGasIn_Vel_InOut;
  extends PhotoBioLib.Icons.Basic.Biomass.FlowRegulatorIcon;

  parameter PhotoBioLib.Records.Geometry.Geometry geometry
    "Geometrical parameters";
  parameter Real At(unit="m2")= geometry.At "Section area";
  parameter Real dX(unit="m")=geometry.dX "Length";
equation
  //Qlout = Qlin;
  //Qgout = Qlin/sArea * gasCon.Vol/dX;
  Qlout = Qlin+der(gasCon.Vol);
  Qgout = Qlout*gasCon.Vol/(At*dX-gasCon.Vol);

  // No hay flujo, solo calculos de velocidades
  gasCon.flowO2 = 0;
  gasCon.flowCO2 = 0;
  gasCon.flowN2 = 0;
  gasCon.flowVol = 0;

end FlowRegulator_LiqFVar;
```

PhotoBioLib.Basic.Biomass.LiqGasResistor_Column

Mass transfer in the column

Parameters

Type	Name	Default	Description
Geometry	geometry		Geometrical parameters
Real	At	geometry.At	Section area [m2]
MassTransfer	massTransfer		Mass transfer parameters
Real	a	massTransfer.a	Form parameter [1]
Real	b	massTransfer.b	Form parameter [1]
Real	KCO2	massTransfer.KCO2	Transfer coefficient for CO2 [1]
Real	eps	0	
Real	limInfGas	eps	
Real	limSupGas	eps	
Real	Vinf	massTransfer.Vinf	Bubble accesion rate [m s-1]
Real	C0	massTransfer.C0	Drift flux model parameter [1]

Connectors

Type	Name	Description
BioCon	conL	Liquid connector
GasCon	conG	Gas connector
input RealInput	Qliq	Input liquid flow
input RealInput	Qgas	Input gas flow

Modelica definition

```
model LiqGasResistor_Column "Mass transfer in the column"

  extends LiqGasResistor_Partial;
  extends PhotoBioLib.Interfaces.Partials.Others.Flow2;

  // En un futuro ponerle entradas
  Real Vgas(unit="m s-1") "Superficial velocity of gas";// = 0.0186;
  Real Vliq(unit="m s-1") "Superficial velocity of liquid";// = 0.0441;

  parameter Real Vinf(unit="m s-1") = massTransfer.Vinf "Bubble accesion rate";//0.651;
  parameter Real C0(unit="1") = massTransfer.C0 "Drift flux model parameter";//0.996;
equation

  if gas then
    Vgas = Qgas/At;
    Vliq = Qliq/At;
    epsilon = Vgas/((C0*Vgas+Vliq) + Vinf);
  else
    Vgas = 0;
    Vliq = 0;
    epsilon = 0;
  end if;
end LiqGasResistor_Column;
```

PhotoBioLib.Basic.Biomass.LiqGasResistor_Loop

Resistor de fase líquida a gasesosa, lazo

Parameters

Type	Name	Default	Description
Geometry	geometry		Geometrical parameters
Real	At	geometry.At	Section area [m2]
MassTransfer	massTransfer		Mass transfer parameters
Real	a	massTransfer.a	Form parameter [1]
Real	b	massTransfer.b	Form parameter [1]
Real	KCO2	massTransfer.KCO2	Transfer coefficient for CO2 [1]
Real	eps	0	
Real	limInfGas	eps	
Real	limSupGas	eps	

Connectors

Type	Name	Description
BioCon	conL	Liquid connector
GasCon	conG	Gas connector

Modelica definition

```
model LiqGasResistor_Loop "Resistor de fase líquida a gasesosa, lazo"
  extends LiqGasResistor_Partial;

equation
  if gas then
    epsilon = if conG.Vol>0 then conG.Vol/(conG.Vol + conL.Vol) else 0;//*(1-exp(-conG.Vol*10000000000));
  else
    epsilon = 0;
  end if;

end LiqGasResistor_Loop;
```

PhotoBioLib.Basic.Biomass.LiqGasResistor_Partial

Mass transfer

Parameters

Type	Name	Default	Description
Geometry	geometry		Geometrical parameters
Real	At	geometry.At	Section area [m2]
MassTransfer	massTransfer		Mass transfer parameters
Real	a	massTransfer.a	Form parameter [1]
Real	b	massTransfer.b	Form parameter [1]
Real	KCO2	massTransfer.KCO2	Transfer coefficient for CO2 [1]
Real	eps	0	
Real	limInfGas	eps	
Real	limSupGas	eps	

Connectors

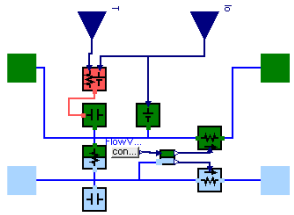
Type	Name	Description
BioCon	conL	Liquid connector
GasCon	conG	Gas connector

Modelica definition

```
partial model LiqGasResistor_Partial "Mass transfer"
  extends PhotoBioLib.Interfaces.Partials.Biomass.BiolGas1;
  extends PhotoBioLib.Icons.Basic.Biomass.LiqGasResistorIcon;

  // Constants
  import PhotoBioLib.Records.Constants;
  constant Real HO2=Constants.HO2;
  constant Real HCO2=Constants.HCO2;
  constant Real Pt=Constants.Pt;
  constant Real Vmol=Constants.Vmol;

  // Parameters
  parameter PhotoBioLib.Records.Geometry.Geometry geometry
    "Geometrical parameters";
  parameter Real At(unit="m2")= geometry.At "Section area";

  parameter PhotoBioLib.Records.MassTransfer.MassTransfer massTransfer
    "Mass transfer parameters";
  //parameter Photobioreactor14Abril.Records.InitialsOld initials;
  parameter Real a(unit="1") = massTransfer.a "Form parameter";
  parameter Real b(unit="1") = massTransfer.b "Form parameter";
  parameter Real KCO2(unit="1") = massTransfer.KCO2
    "Transfer coefficient for CO2";

  // Variables
  Real O2sat(unit="mol m-3")
    "Equilibrium concentration with gas phase for oxygen";
  Real CO2sat(unit="mol m-3")
    "Equilibrium concentration with gas phase for carbon dioxide";
  Real KlaO2(unit="s-1")
    "Volumetric gas-liquid mass transfer coefficient for oxygen";
  Real KlaCO2(unit="s-1")
    "Volumetric gas-liquid mass transfer coefficient for carbon dioxide";
  Real yO2(unit="1") "Oxygen molar fraction";
  Real yCO2(unit="s-1") "Carbon dioxide molar fraction";
  Real epsilon(unit="1") "Hold-up";

  Boolean gas "Gas availability";

  // Parametros para limites
  parameter Real eps= 0;//1E-7;
  parameter Real limInfGas = eps;// 1E-10;
  parameter Real limSupGas = eps;//1E-10;*10

  //Real restaO2;
  //Real restaCO2;

initial equation
  gas = conG.Vol>limSupGas;

equation
  when {pre(gas) and conG.Vol<=limInfGas, not pre(gas) and conG.Vol>limSupGas} then
    gas = conG.Vol>limSupGas;//if gas then false else true;//conG.Vol>limSupGas;
  end when;

  conL.flowQ = 0;
  //restaO2 = (O2sat - conL.O2);
  //restaCO2 = (CO2sat - conL.CO2);
  if gas then
    yO2 = conG.O2/(conG.O2 + conG.CO2 + conG.N2);
    yCO2 = conG.CO2/(conG.O2 + conG.CO2 + conG.N2);

    conG.flowO2 = conL.Vol*KlaO2*(O2sat - conL.O2);//*(1-exp(-conG.O2*10));
    conG.flowCO2 = conL.Vol*KlaCO2*(CO2sat - conL.CO2);//*(1-exp(-conG.CO2*10));

    KlaO2 = noEvent(if epsilon>0 then a*epsilon^b else 0);
    //KlaO2 = a*epsilon^b;
    //KlaO2 = if epsilon>0 then a*epsilon^b else 0;
    //KlaO2 = a* (epsilon*(1-exp(-epsilon))) ^b;
    KlaCO2 = KCO2*KlaO2;
    O2sat = HO2*Pt*yO2;
    CO2sat = HCO2*Pt*yCO2;
    conL.flowO2 = -conG.flowO2;
    conL.flowCt = -conG.flowCO2;
    conG.flowVol = (conG.flowO2 + conG.flowCO2)*Vmol;
  else
    yO2 = 0;
    yCO2 = 0;
    conG.flowO2 = 0;
    conG.flowCO2 = 0;

    KlaO2 = 0;
    KlaCO2 = 0;
    O2sat = 0;
    CO2sat = 0;
    conL.flowO2 = 0;
    conL.flowCt = 0;
    conG.flowVol = 0;
  end if;

  conG.flowN2 = 0;
  conL.flowH = 0;
  conL.flowCO2 = 0;
  conL.flowCb = 0;
  conL.flowVol = 0;
end LiqGasResistor_Partial;
```

PhotoBioLib.Basic.Biomass.LoopSection

Loop section



Parameters

Type	Name	Default	Description
Algae	algae		Algae parameters
InitialGas	initialGas		Initial gas values
InitialBio	initialLiq		Initial liquid values
Geometry	geometry		Geometrical parameters
HeatPar	heatPar		Heat parameters
MassTransfer	massTransfer		Mass transfer parameters

Connectors

Type	Name	Description
BioCon	conL1	Liquid connector on the left
BioCon	conL2	Liquid connector on the rigth
GasCon	conG1	Gas connector on the left
GasCon	conG2	Gas connector on the rigth
input RealInput	Io	Input radiation [uE m-2 s-1]
input RealInput	T	Input temperatura [ºK]

Modelica definition

```
model LoopSection "Loop section"
  extends PhotoBioLib.Interfaces.Partials.Biomass.Bio2Gas2Rad1T1;
  extends PhotoBioLib.Icons.Basic.Biomass.LoopSectionIcon;
  parameter PhotoBioLib.Records.Algae.Algae algae "Algae parameters";

  parameter PhotoBioLib.Records.Initial.InitialGas initialGas
    "Initial gas values";
  parameter PhotoBioLib.Records.Initial.InitialBio initialLiq
    "Initial liquid values";
```

```
  parameter PhotoBioLib.Records.Geometry.Geometry geometry
    "Geometrical parameters";
  parameter PhotoBioLib.Records.Heat.HeatPar heatPar "Heat parameters";
  parameter PhotoBioLib.Records.MassTransfer.MassTransfer massTransfer
    "Mass transfer parameters";

  PhotoBioLib.Basic.Biomass.BioCapacitor liqCapacitor(
    Volini=geometry.Vol,
    initialLiq=initialLiq,
    heatPar=heatPar);
  PhotoBioLib.Basic.Biomass.BioResistor liqResistor(heatPar=heatPar);
  PhotoBioLib.Basic.Gas.GasCapacitor gasCapacitor(initialGas=initialGas);
  PhotoBioLib.Basic.Gas.GasResistor gasResistor;
  PhotoBioLib.Basic.Biomass.AlgaeSource algaeSourceRad(algae=algae);
  PhotoBioLib.Basic.Biomass.FlowRegulator_LiqFCons equilibrioCaudalCons(
      geometry=geometry);
  Modelica.Blocks.Sources.RealExpression FlowVol_In(
    y=conL1.flowVol);
  LiqGasResistor_Loop liqGasResistor_Loop(
    geometry=geometry,
    massTransfer=massTransfer);
  PhotoBioLib.Basic.Heat.HeatConvRad heatConvRad(geometry=geometry, heatPar=
        heatPar);

equation
  connect(liqResistor.con1, liqCapacitor.lCon);
  connect(gasResistor.con1, gasCapacitor.con);
  connect(conL1, liqCapacitor.lCon);
  connect(conG1, gasCapacitor.con);
  connect(liqResistor.con2, conL2);
  connect(gasResistor.con2, conG2);
  connect(equilibrioCaudalCons.Qlout, liqResistor.qLiq);
  connect(equilibrioCaudalCons.Qgout, gasResistor.qGas);
  connect(equilibrioCaudalCons.gasCon, gasCapacitor.con);
  connect(FlowVol_In.y, equilibrioCaudalCons.Qlin);
  connect(liqCapacitor.lCon, liqGasResistor_Loop.conL);
  connect(liqGasResistor_Loop.conG, gasCapacitor.con);
  connect(algaeSourceRad.lCon, liqCapacitor.lCon);
  connect(heatConvRad.hCon, liqCapacitor.hCon);
  connect(heatConvRad.Io, Io);
  connect(T, heatConvRad.Text);
  connect(algaeSourceRad.Io, Io);
end LoopSection;
```

PhotoBioLib.Basic.Gas

Basic gas elements

Package Content

Name	Description
GasCapacitor	Gas capacitor
GasResistor	Gas resistor

PhotoBioLib.Basic.Gas.GasCapacitor

Gas capacitor

Parameters

Type	Name	Default	Description
InitialGas	initialGas		Initial gas values
Real	O2ini	initialGas.O2ini	Initial oxygen concentration [mol m-3]
Real	CO2ini	initialGas.CO2ini	Initial carbon dioxide concentration [mol m-3]
Real	N2ini	initialGas.N2ini	Initital nitrogen concentration [mol m-3]
Real	Volini	initialGas.Volini	Initial gas volume [m3]
Real	eps	0	
Real	eps2	10*eps	

Connectors

Type	Name	Description
GasCon	con	Gas connector

Modelica definition

```
model GasCapacitor "Gas capacitor"
  // Icon and connectors
  extends PhotoBioLib.Interfaces.Partials.Gas.Gas1;
  replaceable class icon =
      PhotoBioLib.Icons.Basic.Gas.GasCapacitorIcon "Icon";
  extends icon;

  // Constants and math functions
  import PhotoBioLib.Records.Constants "Constants";
  constant Real Vmol(unit="m3 mol-1") = Constants.Vmol "Molar volumen";

  // Parameters
  parameter PhotoBioLib.Records.Initial.InitialGas initialGas
    "Initial gas values";
  parameter Real O2ini(unit="mol m-3") = initialGas.O2ini
    "Initial oxygen concentration";
  parameter Real CO2ini(unit="mol m-3") = initialGas.CO2ini
    "Initial carbon dioxide concentration";
  parameter Real N2ini(unit="mol m-3") = initialGas.N2ini
    "Initital nitrogen concentration";
  parameter Real Volini(unit="m3") = initialGas.Volini "Initial gas volume";

  Real yO2p;

  // Variables
  Real O2t(unit="mol",start=O2ini*Volini,fixed=false) "Total moles of oxygen";
  Real CO2t(unit="mol",start=CO2ini*Volini,fixed=false)
    "Total moles of carbon dioxide";
  Real N2t(unit="mol",start=N2ini*Volini,fixed=false) "Total moles of nitrogen";

  // Event Variables
  //Boolean eO2 "Availability of oxygen";
  //Boolean eCO2 "Availability of carbon dioxide";
  //Boolean eN2 "Availability of nitrogen";
  Boolean eVol "Availability of gas";

  parameter Real eps=0;//1E-30;
  parameter Real eps2=10*eps;//1E-5;

initial equation
  eVol = con.Vol>0;

equation
  when {pre(eVol) and con.Vol <=0, not pre(eVol) and con.Vol >0} then
    eVol = con.Vol>0;
  end when;

  der(O2t) = con.flowO2;
  der(CO2t)= con.flowCO2;
  der(N2t) = con.flowN2;

  con.Vol = (O2t+CO2t+N2t)*Constants.Vmol;
  if eVol then
    con.O2 =  O2t/con.Vol;
    con.CO2 = CO2t/con.Vol;
    con.N2 =  N2t/con.Vol;
    //con.Vol = (O2t+CO2t+N2t)*Constants.Vmol;
    yO2p = O2t/(O2t+CO2t+N2t);
  else
    con.O2 = 0;//con.O2;
    con.CO2 = 0;//con.CO2;
    con.N2 = 0;//con.N2;
    //con.Vol = 0;
    yO2p=0;
  end if;

end GasCapacitor;
```

PhotoBioLib.Basic.Gas.GasResistor

Gas resistor

Connectors

Type	Name	Description
GasCon	con1	Gas connector on the left
GasCon	con2	Gas connector on the rigth
input RealInput	qGas	Input gas flow [m3 s-1]

Modelica definition

```
model GasResistor "Gas resistor"
  import Modelica.Math.exp;
  extends PhotoBioLib.Interfaces.Partials.Gas.Gas2Flow1;
  replaceable class icon =
      PhotoBioLib.Icons.Basic.Gas.GasResistorIcon;
  extends icon;


  //parameter Real eps = 1e-8;
   //Real nFlow;
equation
  //con1.flowVol = if con1.Vol>eps then qGas else 0;
  //nFlow = (1-exp(-con1.Vol*1/100000000000000));
  //con1.flowVol = qGas*nFlow;
  con1.flowVol = qGas;
  con2.flowVol = -con1.flowVol;

  con1.flowO2 = con1.O2*con1.flowVol;
  con2.flowO2 = -con1.flowO2;
```

```
  con1.flowCO2 = con1.CO2*con1.flowVol;
  con2.flowCO2 = -con1.flowCO2;
  con1.flowN2 = con1.N2*con1.flowVol;
  con2.flowN2 = -con1.flowN2;

/*
  con1.flowO2 = if con1.Vol*con1.O2 > eps then con1.O2*con1.flowVol else 0;
  con2.flowO2 = -con1.flowO2;
  con1.flowCO2 = if con1.Vol*con1.CO2 > eps then con1.CO2*con1.flowVol else 0;
  con2.flowCO2 = -con1.flowCO2;
  con1.flowN2 = if con1.Vol*con1.N2 > eps then con1.N2*con1.flowVol else 0;
  con2.flowN2 = -con1.flowN2;
*/
end GasResistor;
```

PhotoBioLib.Basic.Heat

Basic heat elements

Package Content

Name	Description
Environment	Environment
HeatConvection	Convection heat transfer
HeatConvRad	Convection and radiation heat transfers
HeatRadiation	Radiation heat transfer

PhotoBioLib.Basic.Heat.Environment

Environment

Connectors

Type	Name	Description
HeatCon	hCon	Heat connector
input RealInput	Io	Input radiation [uE m-2 s-1]
input RealInput	Text	Temperature [°K]
output RealOutput	Ic	Output radiation [W m-2]

Modelica definition

```
model Environment "Environment"
  extends PhotoBioLib.Interfaces.Partials.Heat.Heat1Rad2T1;
  extends PhotoBioLib.Icons.Basic.Heat.EnvironmentIcon;

equation
  Ic = Io/2;
  hCon.T = Text;

end Environment;
```

PhotoBioLib.Basic.Heat.HeatConvection

Convection heat transfer

Parameters

Type	Name	Default	Description
Geometry	geometry		Geometry parameters
HeatPar	heatPar		Heat parameters
Real	h	heatPar.h	Transmission coefficient [J s-1 m-2 °K-1]
Real	S	geometry.S	Surface area [m2]

Connectors

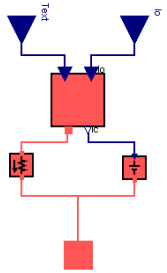
Type	Name	Description
HeatCon	hCon1	Heat connector on the left
HeatCon	hCon2	Heat connector on the right

Modelica definition

```
model HeatConvection "Convection heat transfer"
  import Modelica.Constants.pi;
  extends PhotoBioLib.Interfaces.Partials.Heat.Heat2;
  extends PhotoBioLib.Icons.Basic.Heat.HeatConvectionIcon;

  parameter PhotoBioLib.Records.Geometry.Geometry geometry
    "Geometry parameters";
  parameter PhotoBioLib.Records.Heat.HeatPar heatPar "Heat parameters";
  parameter Real h(unit="J s-1 m-2 °K-1") = heatPar.h
    "Transmission coefficient";
  parameter Real S(unit="m2") = geometry.S "Surface area";

equation
  hCon1.flowQ = h*S*(hCon1.T-hCon2.T);
  hCon1.flowQ = -hCon2.flowQ;

end HeatConvection;
```

PhotoBioLib.Basic.Heat.HeatConvRad

Convection and radiation heat transfers



Parameters

Type	Name	Default	Description
Geometry	geometry		Geometry parameters
HeatPar	heatPar		Heat parameters

Connectors

Type	Name	Description
HeatCon	hCon	Heat connector
input RealInput	Io	Radiation [uE m-2 s-1]
input RealInput	Text	Temperature [°K]

Modelica definition

```
model HeatConvRad "Convection and radiation heat transfers"
  import Modelica.Constants.pi;
  extends PhotoBioLib.Interfaces.Partials.Heat.Heat1Rad1T1;
  extends PhotoBioLib.Icons.Basic.Heat.HeatConvRadIcon;

  parameter PhotoBioLib.Records.Geometry.Geometry geometry
    "Geometry parameters";
  parameter PhotoBioLib.Records.Heat.HeatPar heatPar "Heat parameters";

  PhotoBioLib.Basic.Heat.HeatConvection heatConvection(geometry=geometry,
      heatPar=heatPar);
  PhotoBioLib.Basic.Heat.HeatRadiation heatRad(geometry=geometry, heatPar=
      heatPar);
  PhotoBioLib.Basic.Heat.Environment ambiente;

equation
  connect(heatRad.hCon, hCon);
  connect(heatConvection.hCon2, hCon);
  connect(ambiente.hCon, heatConvection.hCon1);
  connect(ambiente.Ic, heatRad.Ic);
  connect(Text, ambiente.Text);
  connect(Io, ambiente.Io);
end HeatConvRad;
```

PhotoBioLib.Basic.Heat.HeatRadiation

Radiation heat transfer

Parameters

Type	Name	Default	Description
Geometry	geometry		Geometry parameters
HeatPar	heatPar		Heat parameters
Real	alpha	geometry.alpha	Distribution solar factor [1]
Real	aR	heatPar.aR	Solar irradiance absorptivity [1]
Real	S	geometry.S	Surface area [m2]

Connectors

Type	Name	Description
HeatCon	hCon	Heat connector
input RealInput	Ic	Radiation [W m-2]

Modelica definition

```
model HeatRadiation "Radiation heat transfer"
  import Modelica.Constants.pi;
  extends PhotoBioLib.Interfaces.Partials.Heat.Heat1Rad1;
  extends PhotoBioLib.Icons.Basic.Heat.HeatRadiationIcon;

  parameter PhotoBioLib.Records.Geometry.Geometry geometry
    "Geometry parameters";
  parameter PhotoBioLib.Records.Heat.HeatPar heatPar "Heat parameters";
  parameter Real alpha(unit="1") = geometry.alpha "Distribution solar factor";
  parameter Real aR(unit="1") = heatPar.aR "Solar irradiance absorptivity";
  parameter Real S(unit="m2") = geometry.S "Surface area";
equation
  hCon.flowQ = -alpha*aR*S*Ic/4.184;
  // Se divide por 4.184 por diferencia de unidades
end HeatRadiation;
```

PhotoBioLib.Basic.Liquid

Basic liquid elements

Package Content

Name	Description
LiqCapacitor	Liquiq capacitor
LiqResistor	Liquid resistor

PhotoBioLib.Basic.Liquid.LiqCapacitor

Liquiq capacitor

Parameters

Type	Name	Default	Description
InitialLiq	initialLiq	extends PhotoBioLib.Records....	Initial liquid values
HeatPar	heatPar		Heat parameters
Real	Cv	heatPar.Cv	Volumetric heat capacity [J m-3 ºK-1)]
Real	Volini	initialLiq.Volini	Initial liquid volume [m3]
Real	Tini	initialLiq.Tini	Inititat temperature [ºK]

Connectors

Type	Name	Description
LiqCon	lCon	Liquid connector
HeatCon	hCon	Heat connector

Modelica definition

```
model LiqCapacitor "Liquiq capacitor"
/* Pendiente
   No se considera que el volumen puede ser 0 o bajar. Útil para tanques vacios
*/

  import Modelica.Math.log10;
  import PhotoBioLib.Records.Constants;
  replaceable class con =
      PhotoBioLib.Interfaces.Partials.Liquid.Liq1Heat1;
  extends con;

  replaceable class icon =
      PhotoBioLib.Icons.Basic.Liquid.LiqCapacitorIcon "Icon";
  extends icon;

  replaceable class initialValues =
      PhotoBioLib.Records.Initial.InitialLiq;

  parameter initialValues initialLiq "Initial liquid values";

  parameter PhotoBioLib.Records.Heat.HeatPar heatPar "Heat parameters";
  parameter Real Cv(unit="J m-3 ºK-1)")= heatPar.Cv "Volumetric heat capacity";

  parameter Real Volini(unit="m3") = initialLiq.Volini "Initial liquid volume";
  // Calor
  parameter Real Tini(unit="ºK") = initialLiq.Tini "Initital temperature";

  Real Qt(unit="J",start=Cv*Volini*Tini,fixed=false) "Total heat";

  // Event variables

  Boolean eVol "Availibility of liquid";

initial equation
  lCon.Vol = Volini;

  eVol = lCon.Vol>0;

equation

  when {pre(eVol) and lCon.Vol<=0, not pre(eVol) and lCon.Vol>0} then
    eVol = lCon.Vol>0;
  end when;

  // Heat
  der(Qt) = lCon.flowQ + hCon.flowQ;
  hCon.T = lCon.T;

  // Integración de componentes
  der(lCon.Vol) = lCon.flowVol;

  if eVol then
    lCon.T = Qt/(lCon.Vol*Cv);

  else
    lCon.T = 0;

  end if;

end LiqCapacitor;
```

PhotoBioLib.Basic.Liquid.LiqResistor

Liquid resistor

Parameters

Type	Name	Default	Description
HeatPar	heatPar		Heat parameters
Real	Cv	heatPar.Cv	Volumetric heat capacity [J m-3 ºK-1)]

Connectors

Type	Name	Description
LiqCon	con1	Liquid connector on the left
LiqCon	con2	Liquid connector on the rigth
input RealInput	qLiq	Input liquid flow [m3 s-1]

Modelica definition

```
model LiqResistor "Liquid resistor"
  replaceable class con =
      PhotoBioLib.Interfaces.Partials.Liquid.Liq2Flow1;
  extends con;

  replaceable class icon =
      PhotoBioLib.Icons.Basic.Liquid.LiqResistorIcon;
  extends icon;

  parameter PhotoBioLib.Records.Heat.HeatPar heatPar "Heat parameters";
  parameter Real Cv(unit="J m-3 ºK-1)")= heatPar.Cv "Volumetric heat capacity";
```

```
equation
  con1.flowVol = qLiq;
  con2.flowVol = -con1.flowVol;

  con1.flowQ = con1.flowVol*Cv*con1.T;
  con2.flowQ = -con1.flowQ;

end LiqResistor;
```

PhotoBioLib.Examples

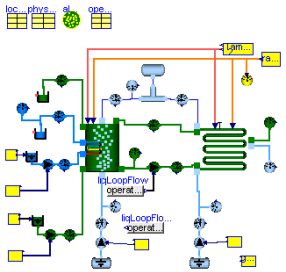
Examples

Package Content

Name	Description
ClosedPhotobioreactor	Closed photobioreactor example
ClosedPhotobioreactor_2	Closed photobioreactor example
NewModel	Closed photobioreactor example
Validation	Real variables

PhotoBioLib.Examples.ClosedPhotobioreactor

Closed photobioreactor example



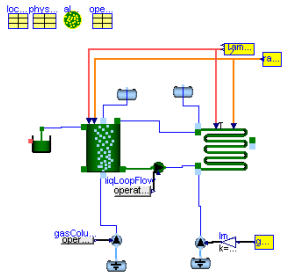
Modelica definition

```
model ClosedPhotobioreactor "Closed photobioreactor example"

//************************************************************************
// Data
//************************************************************************
public
  Records.LocationAndDate locationAndDate;
  PhotoBioLib.Records.PhysicalPar_ClosedPhotobioreactor physical;

  PhotoBioLib.Records.Algae.Algae algae;
   Records.Operation operation(initialLoop(initialLiq(Ctini=4, O2ini=0.2)),
      initialColumn(initialLiq(Ctini=4, O2ini=0.2)));

//************************************************************************

protected
  Modelica.Blocks.Sources.RealExpression liqLoopFlow(y=operation.qMove);

//************************************************************************
// Parts
//************************************************************************
protected
  PhotoBioLib.Parts.Loop loop_(
    algae=algae,
    geometry=physical.geometryLoop,
    initialLoop=operation.initialLoop,
    heatPar=physical.heatParLoop,
    massTransfer=physical.massTransferLoop);
  PhotoBioLib.Parts.Column_HeatExchanger column(
    algae=algae,
    geometry=physical.geometryColumn,
    heatPar=physical.heatParColumn,
    massTransfer=physical.massTransferColumn,
    geometryExchanger=physical.geometryExchanger,
    heatParExchanger=physical.heatParExchanger,
    initialColumn=operation.initialColumn,
    initialExchanger=operation.initialExchanger);

protected
  PhotoBioLib.Parts.Liquid.LiqSource liqSourceExchanger(initialLiq=operation.initialLiqTank);
  PhotoBioLib.Parts.Liquid.LiqPump liqPumpExchanger;

protected
  PhotoBioLib.Parts.Biomass.BioTank liqTankOverflow(
    Tini=13 + 274.15,
    initialLiq=operation.initialMedium,
    Volini=1e-10);
  PhotoBioLib.Parts.Liquid.LiqTank liqTankExchanger(initialLiq=operation.initialLiqTank);
  PhotoBioLib.Parts.Biomass.BioSource liqSourceMedium(initialLiq=operation.initialMedium,
      Tini=13 + 274.15);
  PhotoBioLib.Parts.Gas.GasSource gasTankAir(initialGas=operation.initialAirInyection);
  PhotoBioLib.Parts.Gas.GasSource gasTankCO2(initialGas=operation.initialCO2Inyection);
  PhotoBioLib.Parts.Biomass.BioPump liqPumpMove;
  PhotoBioLib.Parts.Biomass.BioPump liqPumpMedium;
  PhotoBioLib.Parts.Gas.GasPump gasPumpAir;
  PhotoBioLib.Parts.Gas.GasPump gasPumpCO2;
  Parts.Gas.GasTank gasTank_air;
  Parts.Gas.Tpipe Column_air;

//************************************************************************

//************************************************************************
// Sensors
//************************************************************************
public
    PhotoBioLib.Sensors.GasPotentialSensor Pgas_Loop_In;
  PhotoBioLib.Sensors.GasPotentialSensor Pgas_Column;
  PhotoBioLib.Sensors.GasPotentialSensor Pgas_Loop_Out;

  PhotoBioLib.Sensors.GasFlowSensor Fgas_Column_In;
  PhotoBioLib.Sensors.GasFlowSensor Fgas_Loop_In;

  PhotoBioLib.Sensors.RadSensor Frad;

  Sensors.LiqFlowSensor Fliq_Exchanger_In;
  Sensors.LiqPotentialSensor Pliq_Exchanger_Out;
  Sensors.BioPotentialSensor Pbio_Loop_In;
  Sensors.BioPotentialSensor Pbio_Loop_Out;
  Sensors.BioFlowSensor Fbio_Loop_In;
  Sensors.BioFlowSensor Fbio_Medium_In;
  Sensors.BioFlowSensor_ext Fbio_Column_Out_ext;
  Sensors.BioFlowSensor_ext Fbio_Medium_Out_ext;
  Sensors.LiqFlowSensor_ext Fliq_Exchanger_Out_ext;
  Sensors.GasPotentialSensor Pgas_Column_out;

//************************************************************************

//************************************************************************
// Inputs
//************************************************************************

  Utilities.ReadVar Flow_rad(tableName="raddat", file="/data/interpolado.mat");
  Utilities.ReadVar_Plus Tambiente(tableName="Tambiente041");
  Utilities.ReadVar_Plus Tmedio(tableName="TentradaMedio");
  Utilities.ReadVar_Plus T_exchanger_in(tableName="TentradaAgua");
  Utilities.ReadVar_Prod Flow_Medium(tableName="CaudalMedioInput", prod=-1*(
        1e-4)/6);
  Utilities.ReadVar_Prod Flow_Water_Exchanger(tableName="CaudalAguaInput", prod=(
        1e-4)/6);

//************************************************************************

//************************************************************************
// Experiment
//************************************************************************


//************************************************************************

  Sensors.GasFlowSensor_ext Fgas_Column_out;
  Sensors.GasFlowSensor_ext Fgas_Loop_out;
  Utilities.ReadVar_Prod Flow_Air(tableName="CaudalAireInput", prod=(1e-4)/6);
  Utilities.ReadVar_Prod Flow_Air1(prod=(1e-4)/6, tableName="CO2");
protected
  Modelica.Blocks.Sources.RealExpression liqLoopFlow1(y=operation.qBuble);
public
  Utilities.ReadVar_Prod Flow_Air2(prod=(1e-4)/6, tableName="gasdat");
equation
  connect(gasPumpCO2.con1, gasTankCO2.con);
  connect(gasPumpAir.con1, gasTankAir.con);
  connect(loop_.conGprobe,Pgas_Loop_Out. con);
  connect(Fgas_Column_In.con1, gasPumpAir.con2);
  connect(column.conLOut, liqPumpMove.con1);
  connect(liqSourceExchanger.lCon, liqPumpExchanger.con1);
  connect(Fgas_Loop_In.con1, gasPumpCO2.con2);
  connect(Pgas_Loop_In.con,Fgas_Loop_In. con2);
  connect(Fgas_Loop_In.con2, loop_.conGIn);
  connect(Flow_rad.y, Frad.Io);
  connect(Flow_rad.y, loop_.Io);
  connect(Flow_rad.y, column.Io);
  connect(Tambiente.y_sum, loop_.Tamb);
  connect(Tambiente.y_sum, column.T);
  connect(Tmedio.y_sum, liqSourceMedium.T);
  connect(Fliq_Exchanger_In.con2, column.lConIntIn);
  connect(Fbio_Loop_In.con2, loop_.conLIn);
  connect(liqPumpMove.con2, Fbio_Loop_In.con1);
  connect(liqPumpMedium.con2, Fbio_Medium_In.con1);
  connect(liqLoopFlow.y, liqPumpMove.qLiq);
  connect(loop_.conLprobe, Pbio_Loop_Out.lCon);
  connect(T_exchanger_in.y_sum, liqSourceExchanger.T);
  connect(liqPumpExchanger.con2, Fliq_Exchanger_In.con1);
  connect(Fliq_Exchanger_Out.lCon, liqTankExchanger.lCon);
  connect(Fbio_Medium_In.con2, column.conMIn);
  connect(liqSourceMedium.lCon, liqPumpMedium.con1);
  connect(column.conGIn, Pgas_Column.con);
  connect(Fgas_Column_In.con2, column.conGIn);
  connect(Pbio_Loop_In.lCon, column.conLOut);
  connect(Fbio_Column_Out_ext.con1, loop_.conLOut);
```

```
  connect(Fbio_Column_Out_ext.con2, column.conLIn);
  connect(Fbio_Medium_Out_ext.con1, column.conMOut);
  connect(Fbio_Medium_Out_ext.con2, liqTankOverflow.lCon);
  connect(Fliq_Exchanger_Out_ext.con2, liqTankExchanger.lCon);
  connect(Fliq_Exchanger_Out_ext.con1, column.lConIntOut);
  connect(gasTank_air.con, Column_air.con3);
  connect(Fgas_Column_out.con, Column_air.con1);
  connect(Fgas_Column_out.con2, Column_air.con1);
  connect(Column_air.con2, Fgas_Loop_out.con2);
  connect(Fgas_Column_out.con1, column.conGOut);
  connect(Fgas_Loop_out.con1, loop_.conGOut);
  connect(Flow_Medium.y_prod, liqPumpMedium.qLiq);
  connect(Flow_Water_Exchanger.y_prod, liqPumpExchanger.qLiq);
  connect(Flow_Air.y_prod, gasPumpAir.qGas);
  connect(gasPumpCO2.qGas, Flow_Air1.y_prod);
end ClosedPhotobioreactor;
```

PhotoBioLib.Examples.ClosedPhotobioreactor_2

Closed photobioreactor example



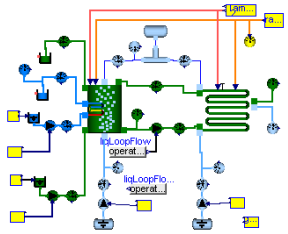
Modelica definition

```
model ClosedPhotobioreactor_2 "Closed photobioreactor example"

//************************************************************************
// Data
//************************************************************************
protected
  PhotoBioLib.Records.PhysicalPar_ClosedPhotobioreactor physical(
    geometryLoop(alpha=0.4),
    heatParExchanger(h=0),
    massTransferColumn(a=0.03));
  Records.LocationAndDate locationAndDate;
  PhotoBioLib.Records.Algae.Algae algae;

//************************************************************************

protected
  Modelica.Blocks.Sources.RealExpression liqLoopFlow(y=operation.qMove);
  Modelica.Blocks.Math.Gain lm_to_m3s(k=0.001/60);
  Modelica.Blocks.Sources.RealExpression gasColumnFlow(y=operation.qBuble);

//************************************************************************
// Parts
//************************************************************************
protected
  PhotoBioLib.Parts.Loop loop_(
    algae=algae,
    geometry=physical.geometryLoop,
    initialLoop=operation.initialLoop,
    heatPar=physical.heatParLoop,
    massTransfer=physical.massTransferLoop);
  PhotoBioLib.Parts.Column column(
    algae=algae,
    geometry=physical.geometryColumn,
    heatPar=physical.heatParColumn,
    massTransfer=physical.massTransferColumn,
    initialColumn=operation.initialColumn);

  PhotoBioLib.Parts.Gas.GasTank gasTankColumn;
  PhotoBioLib.Parts.Gas.GasSource gasTankAir(initialGas=operation.initialAirInyection);
  PhotoBioLib.Parts.Gas.GasSource gasTankCO2(initialGas=operation.initialCO2Inyection);
  PhotoBioLib.Parts.Gas.GasTank gasTankLoop;
  PhotoBioLib.Parts.Biomass.BioPump liqPumpMove;
  PhotoBioLib.Parts.Gas.GasPump gasPumpAir;
  PhotoBioLib.Parts.Gas.GasPump gasPumpCO2;

//************************************************************************

//************************************************************************
// Sensors
//************************************************************************

//************************************************************************

protected
  Records.Operation operation(initialLoop(initialLiq(Tini=13 + 274.15, O2ini=
```

```
          0.28)), initialColumn(initialLiq(Tini=13 + 274.15, O2ini=0.28)));
  Utilities.ReadVar readVar(tableName="raddat", file="/data/interpolado.mat");
  Utilities.ReadVar readVar1(tableName="gasdat", file="/data/interpolado.mat");
  Utilities.ReadVar_Plus Tambiente(tableName="Tambiente041");
public
  PhotoBioLib.Parts.Biomass.BioTank liqTank;
equation

  connect(gasPumpCO2.con1, gasTankCO2.con);
  connect(gasPumpAir.con1, gasTankAir.con);
  connect(liqLoopFlow.y, liqPumpMove.qLiq);
  connect(column.conLOut, liqPumpMove.con1);
  connect(gasColumnFlow.y, gasPumpAir.qGas);
  connect(lm_to_m3s.y, gasPumpCO2.qGas);
  connect(readVar.y, loop_.Io);
  connect(readVar.y, column.Io);
  connect(readVar1.y, lm_to_m3s.u);
  connect(Tambiente.y_sum, loop_.Tamb);
  connect(Tambiente.y_sum, column.T);
  connect(column.conGIn, gasPumpAir.con2);
  connect(loop_.conLIn, liqPumpMove.con2);
  connect(loop_.conLOut, column.conLIn);
  connect(gasTankColumn.con, column.conGOut);
  connect(gasTankLoop.con, loop_.conGOut);
  connect(gasPumpCO2.con2, loop_.conGIn);
  connect(liqTank.lCon, column.conMOut);
end ClosedPhotobioreactor_2;
```

PhotoBioLib.Examples.NewModel

Closed photobioreactor example



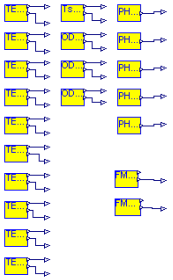
Modelica definition

```
model NewModel "Closed photobioreactor example"

//************************************************************************
// Data
//************************************************************************

//************************************************************************

protected
  Modelica.Blocks.Sources.RealExpression liqLoopFlow(y=operation.qMove);

//************************************************************************
// Parts
//************************************************************************
protected
  PhotoBioLib.Parts.Loop loop_(
    algae=algae,
    geometry=physical.geometryLoop,
    initialLoop=operation.initialLoop,
    heatPar=physical.heatParLoop,
    massTransfer=physical.massTransferLoop);
  PhotoBioLib.Parts.Column_HeatExchanger column(
    algae=algae,
    geometry=physical.geometryColumn,
    heatPar=physical.heatParColumn,
    massTransfer=physical.massTransferColumn,
    geometryExchanger=physical.geometryExchanger,
    heatParExchanger=physical.heatParExchanger,
    initialColumn=operation.initialColumn,
    initialExchanger=operation.initialExchanger);

protected
  PhotoBioLib.Parts.Liquid.LiqSource liqSourceExchanger(initialLiq=operation.initialLiqTank);
  PhotoBioLib.Parts.Liquid.LiqPump liqPumpExchanger;

protected
  PhotoBioLib.Parts.Biomass.BioTank liqTankOverflow(
    Tini=13 + 274.15,
    initialLiq=operation.initialMedium,
    Volini=1e-10);
  PhotoBioLib.Parts.Liquid.LiqTank liqTankExchanger(initialLiq=operation.initialLiqTank);
  PhotoBioLib.Parts.Biomass.BioSource liqSourceMedium(initialLiq=operation.initialMedium,
      Tini=13 + 274.15);
  PhotoBioLib.Parts.Gas.GasSource gasTankAir(initialGas=operation.initialAirInyection);
  PhotoBioLib.Parts.Gas.GasSource gasTankCO2(initialGas=operation.initialCO2Inyection);
  PhotoBioLib.Parts.Biomass.BioPump liqPumpMove;
  PhotoBioLib.Parts.Biomass.BioPump liqPumpMedium;
  PhotoBioLib.Parts.Gas.GasPump gasPumpAir;
```

```
  PhotoBioLib.Parts.Gas.GasPump gasPumpCO2;
  Parts.Gas.GasTank gasTank_air;
  Parts.Gas.Tpipe Column_air;

//************************************************************************

//************************************************************************
// Sensors
//************************************************************************
public
    PhotoBioLib.Sensors.GasPotentialSensor Pgas_Loop_In;
  PhotoBioLib.Sensors.GasPotentialSensor Pgas_Column;
  PhotoBioLib.Sensors.GasPotentialSensor Pgas_Loop_Out;

  PhotoBioLib.Sensors.GasFlowSensor Fgas_Column_In;
  PhotoBioLib.Sensors.GasFlowSensor Fgas_Loop_In;

  PhotoBioLib.Sensors.RadSensor Frad;

  Sensors.LiqFlowSensor Fliq_Exchanger_In;
  Sensors.LiqPotentialSensor Pliq_Exchanger_Out;
  Sensors.BioPotentialSensor Pbio_Loop_In;
  Sensors.BioPotentialSensor Pbio_Loop_Out;
  Sensors.BioFlowSensor Fbio_Loop_In;
  Sensors.BioFlowSensor Fbio_Medium_In;
  Sensors.BioFlowSensor_ext Fbio_Column_Out_ext;
  Sensors.BioFlowSensor_ext Fbio_Medium_Out_ext;
  Sensors.LiqFlowSensor_ext Fliq_Exchanger_Out_ext;
  Sensors.GasPotentialSensor Pgas_Column_out;

//************************************************************************

//************************************************************************
// Inputs
//************************************************************************

  Utilities.ReadVar Flow_rad(tableName="raddat", file="/data/interpolado.mat");
  Utilities.ReadVar_Plus Tambiente(tableName="Tambiente041");
  Utilities.ReadVar_Plus Tmedio(tableName="TentradaMedio");
  Utilities.ReadVar_Plus T_exchanger_in(tableName="TentradaAgua");
  Utilities.ReadVar_Prod Flow_Medium(tableName="CaudalMedioInput", prod=-1*(
        1e-4)/6);
  Utilities.ReadVar_Prod Flow_Water_Exchanger(tableName="CaudalAguaInput", prod=(
        1e-4)/6);

//************************************************************************

//************************************************************************
// Experiment
//************************************************************************


//************************************************************************

  Sensors.GasFlowSensor_ext Fgas_Column_out;
  Sensors.GasFlowSensor_ext Fgas_Loop_out;
  Utilities.ReadVar_Prod Flow_Air(tableName="CaudalAireInput", prod=(1e-4)/6);
  Utilities.ReadVar_Prod Flow_Air1(prod=(1e-4)/6, tableName="CO2");
protected
  Modelica.Blocks.Sources.RealExpression liqLoopFlow1(y=operation.qBuble);
public
  Utilities.ReadVar_Prod Flow_Air2(prod=(1e-4)/6, tableName="gasdat");
equation
  connect(gasPumpCO2.con1, gasTankCO2.con);
  connect(gasPumpAir.con1, gasTankAir.con);
  connect(loop_.conGprobe,Pgas_Loop_Out. con);
  connect(Fgas_Column_In.con1, gasPumpAir.con2);
  connect(column.conLOut, liqPumpMove.con1);
  connect(liqSourceExchanger.lCon, liqPumpExchanger.con1);
  connect(Fgas_Loop_In.con1, gasPumpCO2.con2);
  connect(Pgas_Loop_In.con,Fgas_Loop_In. con2);
  connect(Fgas_Loop_In.con2, loop_.conGIn);
  connect(Flow_rad.y, Frad.Io);
  connect(Flow_rad.y, loop_.Io);
  connect(Flow_rad.y, column.Io);
  connect(Tambiente.y_sum, loop_.Tamb);
  connect(Tambiente.y_sum, column.T);
  connect(Tmedio.y_sum, liqSourceMedium.T);
  connect(Fliq_Exchanger_In.con2, column.lConIntIn);
  connect(Fbio_Loop_In.con2, loop_.conLIn);
  connect(liqPumpMove.con2, Fbio_Loop_In.con1);
  connect(liqPumpMedium.con2, Fbio_Medium_In.con1);
  connect(liqLoopFlow.y, liqPumpMove.qLiq);
  connect(loop_.conLprobe, Pbio_Loop_Out.lCon);
  connect(T_exchanger_in.y_sum, liqSourceExchanger.T);
  connect(liqPumpExchanger.con2, Fliq_Exchanger_In.con1);
  connect(Pliq_Exchanger_Out.lCon, liqTankExchanger.lCon);
  connect(Fbio_Medium_In.con2, column.conMIn);
  connect(liqSourceMedium.lCon, liqPumpMedium.con1);
  connect(column.conGIn, Pgas_Column.con);
  connect(Fgas_Column_In.con2, column.conGIn);
  connect(Pbio_Loop_In.lCon, column.conLOut);
  connect(Fbio_Column_Out_ext.con1, loop_.conLOut);
  connect(Fbio_Column_Out_ext.con2, column.conLIn);
  connect(Fbio_Medium_Out_ext.con1, column.conMOut);
  connect(Fbio_Medium_Out_ext.con2, liqTankOverflow.lCon);
  connect(Fliq_Exchanger_Out_ext.con2, liqTankExchanger.lCon);
  connect(Fliq_Exchanger_Out_ext.con1, column.lConIntOut);
  connect(gasTank_air.con, Column_air.con3);
  connect(Pgas_Column_out.con, Column_air.con1);
  connect(Pgas_Column_out.con2, Column_air.con1);
  connect(Column_air.con2, Fgas_Loop_out.con2);
  connect(Fgas_Column_out.con1, column.conGOut);
  connect(Fgas_Loop_out.con1, loop_.conGOut);
  connect(Flow_Medium.y_prod, liqPumpMedium.qLiq);
  connect(Flow_Water_Exchanger.y_prod, liqPumpExchanger.qLiq);
  connect(Flow_Air.y_prod, gasPumpAir.qGas);
  connect(gasPumpCO2.qGas, Flow_Air1.y_prod);
end NewModel;
```

PhotoBioLib.Examples.Validation

Real variables



Connectors

Type	Name	Description
output RealOutput	TEMP0411	[ºC]
output RealOutput	TEMP0411_K	[ºK]
output RealOutput	TEMP0412	[ºC]
output RealOutput	TEMP0412_K	[ºK]
output RealOutput	TEMP0413	[ºC]
output RealOutput	TEMP0413_K	[ºK]
output RealOutput	TEMP0421	[ºC]
output RealOutput	TEMP0421_K	[ºK]
output RealOutput	TEMP0431	[ºC]
output RealOutput	TEMP0431_K	[ºK]
output RealOutput	TEMP0432	[ºC]
output RealOutput	TEMP0432_K	[ºK]
output RealOutput	TEMP0441	[ºC]
output RealOutput	TEMP0441_K	[ºK]
output RealOutput	TEMP0451	[ºC]
output RealOutput	TEMP0451_K	[ºK]
output RealOutput	TEMP0452	[ºC]
output RealOutput	TEMP0452_K	[ºK]
output RealOutput	TEMP0453	[ºC]
output RealOutput	TEMP0453_K	[ºK]
output RealOutput	TsalidaAgua	[ºC]
output RealOutput	TsalidaAgua_K	[ºK]
output RealOutput	OD0412	
output RealOutput	OD0412_C	[mol m-3]
output RealOutput	OD0432	
output RealOutput	OD0432_C	[mol m-3]
output RealOutput	OD0452	
output RealOutput	OD0452_C	[mol m-3]
output RealOutput	PH0411	[1]
output RealOutput	PH0421	[1]
output RealOutput	PH0431	[1]
output RealOutput	PH0441	[1]
output RealOutput	PH0451	[1]
output RealOutput	yO2	[1]
output RealOutput	yCO2	[1]

Modelica definition

```
model Validation "Real variables"

protected
  Utilities.ReadVar_Plus TEMP1(tableName="TEMP0411");
```

```
  Utilities.ReadVar_Plus TEMP2(tableName="TEMP0412");
  Utilities.ReadVar_Plus TEMP3(tableName="TEMP0413");
  Utilities.ReadVar_Plus TEMP4(tableName="TEMP0421");
  Utilities.ReadVar_Plus TEMP5(tableName="TEMP0431");
  Utilities.ReadVar_Plus TEMP6(tableName="TEMP0432");
  Utilities.ReadVar_Plus TEMP7(tableName="TEMP0441");
  Utilities.ReadVar_Plus TEMP8(tableName="TEMP0451");
  Utilities.ReadVar_Plus TEMP9(tableName="TEMP0452");
  Utilities.ReadVar_Plus TEMP10(tableName="TEMP0453");
  Utilities.ReadVar_Plus TEMP11(tableName="TsalidaAgua");
  Utilities.ReadVar_Prod O2(tableName="OD0412");
  Utilities.ReadVar_Prod O22(tableName="OD0432");
  Utilities.ReadVar_Prod O23(tableName="OD0452");
  Utilities.ReadVar pH(tableName="PH0411");
  Utilities.ReadVar pH1(tableName="PH0421");
  Utilities.ReadVar pH2(tableName="PH0431");
  Utilities.ReadVar pH4(tableName="PH0441");
  Utilities.ReadVar pH5(tableName="PH0451");
  Utilities.ReadVar_Prod FMO2(tableName="FMO2", prod=1/100);
  Utilities.ReadVar_Prod FMCO2(tableName="FMCO2", prod=1/100);
public
  Modelica.Blocks.Interfaces.RealOutput TEMP0411(unit="°C");
  Modelica.Blocks.Interfaces.RealOutput TEMP0411_K(unit="°K");
  Modelica.Blocks.Interfaces.RealOutput TEMP0412(unit="°C");
  Modelica.Blocks.Interfaces.RealOutput TEMP0412_K(unit="°K");
  Modelica.Blocks.Interfaces.RealOutput TEMP0413(unit="°C");
  Modelica.Blocks.Interfaces.RealOutput TEMP0413_K(unit="°K");
  Modelica.Blocks.Interfaces.RealOutput TEMP0421(unit="°C");
  Modelica.Blocks.Interfaces.RealOutput TEMP0421_K(unit="°K");
  Modelica.Blocks.Interfaces.RealOutput TEMP0431(unit="°C");
  Modelica.Blocks.Interfaces.RealOutput TEMP0431_K(unit="°K");
  Modelica.Blocks.Interfaces.RealOutput TEMP0432(unit="°C");
  Modelica.Blocks.Interfaces.RealOutput TEMP0432_K(unit="°K");
  Modelica.Blocks.Interfaces.RealOutput TEMP0441(unit="°C");
  Modelica.Blocks.Interfaces.RealOutput TEMP0441_K(unit="°K");
  Modelica.Blocks.Interfaces.RealOutput TEMP0451(unit="°C");
  Modelica.Blocks.Interfaces.RealOutput TEMP0451_K(unit="°K");
  Modelica.Blocks.Interfaces.RealOutput TEMP0452(unit="°C");
  Modelica.Blocks.Interfaces.RealOutput TEMP0452_K(unit="°K");
  Modelica.Blocks.Interfaces.RealOutput TEMP0453(unit="°C");
  Modelica.Blocks.Interfaces.RealOutput TEMP0453_K(unit="°K");
  Modelica.Blocks.Interfaces.RealOutput TsalidaAgua(unit="°C");
  Modelica.Blocks.Interfaces.RealOutput TsalidaAgua_K(unit="°K");
  Modelica.Blocks.Interfaces.RealOutput OD0412;
  Modelica.Blocks.Interfaces.RealOutput OD0412_C(unit="mol m-3");
  Modelica.Blocks.Interfaces.RealOutput OD0432;
  Modelica.Blocks.Interfaces.RealOutput OD0432_C(unit="mol m-3");
  Modelica.Blocks.Interfaces.RealOutput OD0452;
  Modelica.Blocks.Interfaces.RealOutput OD0452_C(unit="mol m-3");
  Modelica.Blocks.Interfaces.RealOutput PH0411(unit="1");
  Modelica.Blocks.Interfaces.RealOutput PH0421(unit="1");
  Modelica.Blocks.Interfaces.RealOutput PH0431(unit="1");
  Modelica.Blocks.Interfaces.RealOutput PH0441(unit="1");
  Modelica.Blocks.Interfaces.RealOutput PH0451(unit="1");
  Modelica.Blocks.Interfaces.RealOutput yO2(   unit="1");
  Modelica.Blocks.Interfaces.RealOutput yCO2(unit="1");

equation
  connect(TEMP1.y, TEMP0411);
  connect(TEMP0411_K, TEMP0411_K);
  connect(TEMP1.y_sum, TEMP0411_K);
  connect(TEMP2.y, TEMP0412);
  connect(TEMP2.y_sum, TEMP0412_K);
  connect(TEMP3.y, TEMP0413);
  connect(TEMP3.y_sum, TEMP0413_K);
  connect(TEMP4.y, TEMP0421);
  connect(TEMP4.y_sum, TEMP0421_K);
  connect(TEMP5.y, TEMP0431);
  connect(TEMP5.y_sum, TEMP0431_K);
  connect(TEMP6.y, TEMP0432);
  connect(TEMP6.y_sum, TEMP0432_K);
  connect(TEMP7.y, TEMP0441);
  connect(TEMP7.y_sum, TEMP0441_K);
  connect(TEMP8.y, TEMP0451);
  connect(TEMP8.y_sum, TEMP0451_K);
  connect(TEMP9.y, TEMP0452);
  connect(TEMP9.y_sum, TEMP0452_K);
  connect(TEMP10.y, TEMP0453);
  connect(TEMP10.y_sum, TEMP0453_K);
  connect(TEMP11.y, TsalidaAgua);
  connect(TEMP11.y_sum, TsalidaAgua_K);
  connect(O2.y, OD0412);
  connect(O2.y_prod, OD0412_C);
  connect(O22.y, OD0432);
  connect(O22.y_prod, OD0432_C);
  connect(O23.y, OD0452);
  connect(O23.y_prod, OD0452_C);
  connect(pH.y, PH0411);
  connect(pH1.y, PH0421);
  connect(pH2.y, PH0431);
  connect(pH4.y, PH0441);
  connect(pH5.y, PH0451);
  connect(FMO2.y_prod, yO2);
  connect(FMCO2.y_prod, yCO2);
end Validation;
```

PhotoBioLib.Icons

Icons package

Package Content

Name	Description
Basic	Basic elements icons
Parts	Parts icons
Records	Records icons
Sensors	Sensors icons

PhotoBioLib.Icons.Basic

Basic elements icons

Package Content

Name	Description
Biomass	Basic biomass elements
Gas	Basic gas elements
Heat	Basic heat elements
Liquid	Basic liquid elements
Liquid2	Basic biomass elements

PhotoBioLib.Icons.Basic.Biomass

Basic biomass elements

Package Content

Name	Description
AlgaeSourceIcon	Algae activity icon
BioCapacitorIcon	Biomass capacitor icon
BioGasResistorIcon	BioGasResistor icon
BioResistorIcon	Biomass resistor icon
ColumnSectionIcon	Column section icon
FlowRegulatorIcon	Flow regulator icon
LiqGasResistorIcon	LiqGasResistor icon
LoopSectionIcon	Loop section icon

PhotoBioLib.Icons.Basic.Biomass.AlgaeSourceIcon

Algae activity icon

Modelica definition

```
partial model AlgaeSourceIcon "Algae activity icon"

end AlgaeSourceIcon;
```

PhotoBioLib.Icons.Basic.Biomass.BioCapacitorIcon

Biomass capacitor icon

Modelica definition

```
partial model BioCapacitorIcon "Biomass capacitor icon"

equation

end BioCapacitorIcon;
```

PhotoBioLib.Icons.Basic.Biomass.BioGasResistorIcon

BioGasResistor icon

Modelica definition

```
partial model BioGasResistorIcon "BioGasResistor icon"

equation

end BioGasResistorIcon;
```

PhotoBioLib.Icons.Basic.Biomass.BioResistorIcon

Biomass resistor icon

Modelica definition

```
partial model BioResistorIcon "Biomass resistor icon"
equation
end BioResistorIcon;
```

PhotoBioLib.Icons.Basic.Biomass.ColumnSectionIcon

Column section icon

Modelica definition

```
partial model ColumnSectionIcon "Column section icon"
equation
end ColumnSectionIcon;
```

PhotoBioLib.Icons.Basic.Biomass.FlowRegulatorIcon

Flow regulator icon

Modelica definition

```
partial model FlowRegulatorIcon "Flow regulator icon"
equation
end FlowRegulatorIcon;
```

PhotoBioLib.Icons.Basic.Biomass.LiqGasResistorIcon

LiqGasResistor icon

Modelica definition

```
partial model LiqGasResistorIcon "LiqGasResistor icon"
equation
end LiqGasResistorIcon;
```

PhotoBioLib.Icons.Basic.Biomass.LoopSectionIcon

Loop section icon

Modelica definition

```
partial model LoopSectionIcon "Loop section icon"
equation
end LoopSectionIcon;
```

PhotoBioLib.Icons.Basic.Gas

Basic gas elements

Package Content

Name	Description
GasCapacitorIcon	Gas capacitor icon
GasResistorIcon	Gas resistor icon

PhotoBioLib.Icons.Basic.Gas.GasCapacitorIcon

Gas capacitor icon

Modelica definition

```
partial model GasCapacitorIcon "Gas capacitor icon"
equation
end GasCapacitorIcon;
```

PhotoBioLib.Icons.Basic.Gas.GasResistorIcon

Gas resistor icon

Modelica definition

```
partial model GasResistorIcon "Gas resistor icon"
equation
end GasResistorIcon;
```

PhotoBioLib.Icons.Basic.Heat

Basic heat elements

Package Content

Name	Description
EnvironmentIcon	Environment icon
HeatConvectionIcon	HeatConvection icon
HeatConvRadIcon	HeatConcRad icon
HeatRadiationIcon	HeatRadiation icon

PhotoBioLib.Icons.Basic.Heat.EnvironmentIcon

Environment icon

Modelica definition

```
partial model EnvironmentIcon "Environment icon"
end EnvironmentIcon;
```

PhotoBioLib.Icons.Basic.Heat.HeatConvectionIcon

HeatConvection icon

Modelica definition

```
partial model HeatConvectionIcon "HeatConvection icon"
equation
end HeatConvectionIcon;
```

PhotoBioLib.Icons.Basic.Heat.HeatConvRadIcon

HeatConcRad icon

Modelica definition

```
partial model HeatConvRadIcon "HeatConcRad icon"
equation
end HeatConvRadIcon;
```

PhotoBioLib.Icons.Basic.Heat.HeatRadiationIcon

HeatRadiation icon

Modelica definition

```
partial model HeatRadiationIcon "HeatRadiation icon"
equation
end HeatRadiationIcon;
```

PhotoBioLib.Icons.Basic.Liquid

Basic liquid elements

Package Content

Name	Description
LiqCapacitorIcon	Liquid capacitor icon
LiqResistorIcon	Liquid resistor icon

PhotoBioLib.Icons.Basic.Liquid.LiqCapacitorIcon

Liquid capacitor icon

Modelica definition

```
partial model LiqCapacitorIcon "Liquid capacitor icon"
equation
end LiqCapacitorIcon;
```

PhotoBioLib.Icons.Basic.Liquid.LiqResistorIcon

Liquid resistor icon

Modelica definition

```
partial model LiqResistorIcon "Liquid resistor icon"
equation
end LiqResistorIcon;
```

PhotoBioLib.Icons.Basic.Liquid2

Basic biomass elements

Package Content

Name	Description
BioCapacitorIcon	Biomass capacitor icon
BioGasResistorIcon	BioGasResistor icon
BioResistorIcon	Biomass resistor icon
ColumnSectionIcon	Column section icon
FlowRegulatorIcon	Flow regulator icon
LoopSectionIcon	Loop section icon

PhotoBioLib.Icons.Basic.Liquid2.BioCapacitorIcon

Biomass capacitor icon

Modelica definition

```
partial model BioCapacitorIcon "Biomass capacitor icon"
equation
end BioCapacitorIcon;
```

PhotoBioLib.Icons.Basic.Liquid2.BioGasResistorIcon

BioGasResistor icon

Modelica definition

```
partial model BioGasResistorIcon "BioGasResistor icon"
equation
end BioGasResistorIcon;
```

PhotoBioLib.Icons.Basic.Liquid2.BioResistorIcon

Biomass resistor icon

Modelica definition

```
partial model BioResistorIcon "Biomass resistor icon"
equation
end BioResistorIcon;
```

PhotoBioLib.Icons.Basic.Liquid2.ColumnSectionIcon

Column section icon

Modelica definition

```
partial model ColumnSectionIcon "Column section icon"
equation
end ColumnSectionIcon;
```

PhotoBioLib.Icons.Basic.Liquid2.FlowRegulatorIcon

Flow regulator icon

Modelica definition

```
partial model FlowRegulatorIcon "Flow regulator icon"
equation
end FlowRegulatorIcon;
```

PhotoBioLib.Icons.Basic.Liquid2.LoopSectionIcon

Loop section icon

Modelica definition

```
partial model LoopSectionIcon "Loop section icon"
equation
end LoopSectionIcon;
```

PhotoBioLib.Icons.Parts

Parts icons

Package Content

Name	Description
BioPumpIcon	Biomass pump icon
BioSourceIcon	Biomass source icon
BioTankIcon	Biomass tank icon
Column_HeatExchangerIcon	Column_HeatExchanger icon
ColumnIcon	Column icon
GasPumpIcon	Gas pump icon
GasSourceIcon	Gas source icon
GasTankIcon	Gas tank icon
HeatExchangerIcon	Heat exchanger icon
LiqPumpIcon	Liquid pump icon
LiqSourceIcon	Liquid source icon
LiqTankIcon	Liquid tank icon
LoopIcon	Loop icon
RadiationIcon	Radiation icon
TpipeIcon	Pipe with T form

PhotoBioLib.Icons.Parts.BioPumpIcon

Biomass pump icon

Modelica definition

```
partial model BioPumpIcon "Biomass pump icon"
equation
end BioPumpIcon;
```

PhotoBioLib.Icons.Parts.BioSourceIcon

Biomass source icon

Modelica definition

```
partial model BioSourceIcon "Biomass source icon"
equation
end BioSourceIcon;
```

PhotoBioLib.Icons.Parts.BioTankIcon

Biomass tank icon

Modelica definition

```
partial model BioTankIcon "Biomass tank icon"
equation
end BioTankIcon;
```

PhotoBioLib.Icons.Parts.Column_HeatExchangerIcon

Column_HeatExchanger icon

Modelica definition

```
partial model Column_HeatExchangerIcon "Column_HeatExchanger icon"
  extends PhotoBioLib.Icons.Parts.ColumnIcon;
end Column_HeatExchangerIcon;
```

PhotoBioLib.Icons.Parts.ColumnIcon

Column icon

Modelica definition

```
partial model ColumnIcon "Column icon"
equation
end ColumnIcon;
```

PhotoBioLib.Icons.Parts.GasPumpIcon

Gas pump icon

Modelica definition

```
partial model GasPumpIcon "Gas pump icon"
equation
end GasPumpIcon;
```

PhotoBioLib.Icons.Parts.GasSourceIcon

Gas source icon

Modelica definition

```
partial model GasSourceIcon "Gas source icon"
equation
end GasSourceIcon;
```

PhotoBioLib.Icons.Parts.GasTankIcon

Gas tank icon

Modelica definition

```
partial model GasTankIcon "Gas tank icon"
equation
end GasTankIcon;
```

PhotoBioLib.Icons.Parts.HeatExchangerIcon

Heat exchanger icon

Modelica definition

```
partial model HeatExchangerIcon "Heat exchanger icon"
equation
end HeatExchangerIcon;
```

PhotoBioLib.Icons.Parts.LiqPumpIcon

Liquid pump icon

Modelica definition

```
partial model LiqPumpIcon "Liquid pump icon"
equation
end LiqPumpIcon;
```

PhotoBioLib.Icons.Parts.LiqSourceIcon

Liquid source icon

Modelica definition

```
partial model LiqSourceIcon "Liquid source icon"
equation
end LiqSourceIcon;
```

PhotoBioLib.Icons.Parts.LiqTankIcon

Liquid tank icon

Modelica definition

```
partial model LiqTankIcon "Liquid tank icon"
equation
end LiqTankIcon;
```

PhotoBioLib.Icons.Parts.LoopIcon

Loop icon

Modelica definition

```
partial model LoopIcon "Loop icon"
equation
end LoopIcon;
```

PhotoBioLib.Icons.Parts.RadiationIcon

Radiation icon

Modelica definition

```
partial model RadiationIcon "Radiation icon"
equation
end RadiationIcon;
```

PhotoBioLib.Icons.Parts.TpipeIcon

Pipe with T form

Modelica definition

```
partial model TpipeIcon "Pipe with T form"
end TpipeIcon;
```

PhotoBioLib.Icons.Records

Records icons

Package Content

Name	Description
AlgaeIcon	Algae icon

PhotoBioLib.Icons.Records.AlgaeIcon

Algae icon

Modelica definition

```
partial model AlgaeIcon "Algae icon"

end AlgaeIcon;
```

PhotoBioLib.Icons.Sensors

Sensors icons

Package Content

Name	Description
BioFlowSensorIcon	Biomass flow sensor icon
BioPotentialSensorIcon	Biomass potential sensor icon
GasFlowSensorIcon	Gas flow sensor icon
GasPotentialSensorIcon	Gas potential sensor icon
LiqFlowSensorIcon	Liquid flow sensor icon
LiqPotentialSensorIcon	Liquid potential sensor icon
RadSensorIcon	Radiation sensor icon

PhotoBioLib.Icons.Sensors.BioFlowSensorIcon

Biomass flow sensor icon

Modelica definition

```
partial model BioFlowSensorIcon "Biomass flow sensor icon"
equation
end BioFlowSensorIcon;
```

PhotoBioLib.Icons.Sensors.BioPotentialSensorIcon

Biomass potential sensor icon

Modelica definition

```
partial model BioPotentialSensorIcon "Biomass potential sensor icon"
equation
end BioPotentialSensorIcon;
```

PhotoBioLib.Icons.Sensors.GasFlowSensorIcon

Gas flow sensor icon

Modelica definition

```
partial model GasFlowSensorIcon "Gas flow sensor icon"
equation
end GasFlowSensorIcon;
```

PhotoBioLib.Icons.Sensors.GasPotentialSensorIcon

Gas potential sensor icon

Modelica definition

```
partial model GasPotentialSensorIcon "Gas potential sensor icon"
equation
end GasPotentialSensorIcon;
```

PhotoBioLib.Icons.Sensors.LiqFlowSensorIcon

Liquid flow sensor icon

Modelica definition

```
partial model LiqFlowSensorIcon "Liquid flow sensor icon"
equation
end LiqFlowSensorIcon;
```

PhotoBioLib.Icons.Sensors.LiqPotentialSensorIcon

Liquid potential sensor icon

Modelica definition

```
partial model LiqPotentialSensorIcon "Liquid potential sensor icon"
equation
end LiqPotentialSensorIcon;
```

PhotoBioLib.Icons.Sensors.RadSensorIcon

Radiation sensor icon

Modelica definition

```
partial model RadSensorIcon "Radiation sensor icon"
equation
end RadSensorIcon;
```

PhotoBioLib.Interfaces

Interfaces package

Package Content

Name	Description
BioCon	Liquid with biomass connector
GasCon	Gas connector
HeatCon	Heat connector
LiqCon	Liquid connector with heat transference
Partials	Interfaces package

PhotoBioLib.Interfaces.BioCon

Liquid with biomass connector

Contents

Type	Name	Description

Real	T	Temperatura [K]
flow Real	flowQ	Heat flow [J s-1]
Real	Vol	Liquid volume [m-3]
flow Real	flowVol	Liquid flow [m3 s-1]
Real	Cb	Biomass concentration [Kg m-3]
flow Real	flowCb	Biomass flow [Kg s-1]
Real	O2	Oxygen concentration [mol m-3]
flow Real	flowO2	Oxygen flow [mol s-1]
Real	CO2	Carbon dioxide concentration [mol m-3]
flow Real	flowCO2	Carbon dioxiden flow [mol s-1]
Real	Ct	Total carbon concentration [mol m-3]
flow Real	flowCt	Total carbon flow [mol s-1]
Real	H	Hidrogen concentration [mol m-3]
flow Real	flowH	Hidrogen flow [mol s-1]

Modelica definition

```
connector BioCon "Liquid with biomass connector"
  extends PhotoBioLib.Interfaces.LiqCon;

  Real Cb(unit="Kg m-3") "Biomass concentration";
  flow Real flowCb(unit="Kg s-1") "Biomass flow";
  Real O2(unit="mol m-3") "Oxygen concentration";
  flow Real flowO2(unit="mol s-1") "Oxygen flow";
  Real CO2(unit="mol m-3") "Carbon dioxide concentration";
  flow Real flowCO2(unit="mol s-1") "Carbon dioxiden flow";
  Real Ct(unit="mol m-3") "Total carbon concentration";
  flow Real flowCt(unit="mol s-1") "Total carbon flow";
  Real H(unit="mol m-3") "Hidrogen concentration";
  flow Real flowH(unit="mol s-1") "Hidrogen flow";

end BioCon;
```

PhotoBioLib.Interfaces.GasCon

Gas connector

Contents

Type	Name	Description
Real	O2	Oxygen concentration [mol m-3]
flow Real	flowO2	Oxygen flow [mol s-1]
Real	CO2	Carbon dioxide concentration [mol m-3]
flow Real	flowCO2	Carbon dioxide concentration [mol s-1]
Real	N2	Nitrogen concentration [mol m-3]
flow Real	flowN2	Nitrogen flow [mol s-1]
Real	Vol	Gas volume [m-3]
flow Real	flowVol	Gas flow [m3 s-1]

Modelica definition

```
connector GasCon "Gas connector"

  Real O2(unit="mol m-3") "Oxygen concentration";
  flow Real flowO2(unit="mol s-1") "Oxygen flow";
  Real CO2(unit="mol m-3") "Carbon dioxide concentration";
  flow Real flowCO2(unit="mol s-1") "Carbon dioxide concentration";
  Real N2(unit="mol m-3") "Nitrogen concentration";
  flow Real flowN2(unit="mol s-1") "Nitrogen flow";
  Real Vol(unit="m-3") "Gas volume";
  flow Real flowVol(unit="m3 s-1") "Gas flow";
end GasCon;
```

PhotoBioLib.Interfaces.HeatCon

Heat connector

Contents

Type	Name	Description
Real	T	Temperatura [K]
flow Real	flowQ	Heat flow [J s-1]

Modelica definition

```
connector HeatCon "Heat connector"
  Real T(unit="K") "Temperatura";
  flow Real flowQ(unit="J s-1") "Heat flow";
end HeatCon;
```

PhotoBioLib.Interfaces.LiqCon

Liquid connector with heat transference

Contents

Type	Name	Description
Real	T	Temperatura [K]
flow Real	flowQ	Heat flow [J s-1]
Real	Vol	Liquid volume [m-3]
flow Real	flowVol	Liquid flow [m3 s-1]

Modelica definition

```
connector LiqCon "Liquid connector with heat transference"
  extends PhotoBioLib.Interfaces.HeatCon;
  Real Vol(unit="m-3") "Liquid volume";
  flow Real flowVol(unit="m3 s-1") "Liquid flow";

end LiqCon;
```

PhotoBioLib.Interfaces.Partials

Interfaces package

Package Content

Name	Description
Biomass	Partials with biomass connectors
Gas	Partials with gas connectors
Heat	Partials with heat connectors
Liquid	Partials with liquid connectors
Others	Others partials

PhotoBioLib.Interfaces.Partials.Biomass

Partials with biomass connectors

Package Content

Name	Description
Bio1	Interface with a biomass connector
Bio1Gas1	Interface with a biomass and a gas connectors
Bio1GasIn_Vel_InOut	Flow regulator interface
Bio1Heat1	Interface with a biomass and a heat connectors
Bio1Heat1_Inv	Interface with a biomass and a heat connectors
Bio1OutN	Interface with a biomass connector with N outputs
Bio1Rad1	Interface with a biomass connector and a input radiation
Bio1T1	Interface with a biomass connector and a input temperature
Bio2	Interface with two biomass connectors
Bio2Flow1	Interface with two biomass connectors and a input liquid flow
Bio2Flow1Heat1	Exchanger interface
Bio2Gas2	Interface with two biomass and two gas connectors
Bio2Gas2Rad1T1	Interface with two gas connectors, two biomass connectors, a input radiation and a input temperature
Bio2Gas2Rad1T1Heat1_V	Interface with two gas connectors, two biomass connectors, a heat connector, a input radiation and a input temperature
Bio2OutN	Interface with two biomass connectors with N outputs

---

PhotoBioLib.Interfaces.Partials.Biomass.Bio1

Interface with a biomass connector

Connectors

Type	Name	Description

BioCon	lCon	Liquid connector

Modelica definition

```
partial model Bio1 "Interface with a biomass connector"

  PhotoBioLib.Interfaces.BioCon lCon "Liquid connector";
end Bio1;
```

PhotoBioLib.Interfaces.Partials.Biomass.Bio1Gas1

Interface with a biomass and a gas connectors

Connectors

Type	Name	Description
BioCon	conL	Liquid connector
GasCon	conG	Gas connector

Modelica definition

```
partial model Bio1Gas1
  "Interface with a biomass and a gas connectors"

  PhotoBioLib.Interfaces.BioCon conL "Liquid connector";
  GasCon conG "Gas connector";
end Bio1Gas1;
```

PhotoBioLib.Interfaces.Partials.Biomass.Bio1GasIn_Vel_InOut

Flow regulator interface

Connectors

Type	Name	Description
input RealInput	Qin	Input liquid flow [m3 s-1]
output RealOutput	Qlout	Output liquid flow [m3 s-1]
output RealOutput	Qgout	Output gas flow [m3 s-1]
GasCon	gasCon	Gas connector

Modelica definition

```
partial model Bio1GasIn_Vel_InOut "Flow regulator interface"

  Modelica.Blocks.Interfaces.RealInput Qlin(unit="m3 s-1") "Input liquid flow";
  Modelica.Blocks.Interfaces.RealOutput Qlout(unit="m3 s-1")
    "Output liquid flow";
  Modelica.Blocks.Interfaces.RealOutput Qgout(unit="m3 s-1") "Output gas flow";
  GasCon gasCon "Gas connector";
end Bio1GasIn_Vel_InOut;
```

PhotoBioLib.Interfaces.Partials.Biomass.Bio1Heat1

Interface with a biomass and a heat connectors

Connectors

Type	Name	Description
BioCon	lCon	Liquid connector
HeatCon	hCon	Heat connector

Modelica definition

```
partial model Bio1Heat1
  "Interface with a biomass and a heat connectors"

  PhotoBioLib.Interfaces.BioCon lCon "Liquid connector";
  HeatCon hCon "Heat connector";

end Bio1Heat1;
```

PhotoBioLib.Interfaces.Partials.Biomass.Bio1Heat1_Inv

Interface with a biomass and a heat connectors

Connectors

Type	Name	Description
BioCon	lCon	Liquid connector
HeatCon	hCon	Heat connector

Modelica definition

```
partial model Bio1Heat1_Inv
  "Interface with a biomass and a heat connectors"

  PhotoBioLib.Interfaces.BioCon lCon "Liquid connector";
  HeatCon hCon "Heat connector";

end Bio1Heat1_Inv;
```

PhotoBioLib.Interfaces.Partials.Biomass.Bio1OutN

Interface with a biomass connector with N outputs

Parameters

Type	Name	Default	Description
Integer	nout	1	Number of outputs

Connectors

Type	Name	Description
output RealOutput	y[nout]	Outputs
BioCon	lCon	Liquid connector

Modelica definition

```
partial model Bio1OutN
  "Interface with a biomass connector with N outputs"
 parameter Integer nout(min=1) = 1 "Number of outputs";
  Modelica.Blocks.Interfaces.RealOutput y[nout](redeclare type SignalType =
        Real) "Outputs";
  PhotoBioLib.Interfaces.BioCon lCon "Liquid connector";
end Bio1OutN;
```

PhotoBioLib.Interfaces.Partials.Biomass.Bio1Rad1

Interface with a biomass connector and a input radiation

Connectors

Type	Name	Description
BioCon	lCon	Liquid connector
input RealInput	Io	Input radiation [uE m-2 s-1]

Modelica definition

```
partial model Bio1Rad1
```

```
  "Interface with a biomass connector and a input radiation"
  extends PhotoBioLib.Interfaces.Partials.Biomass.Bio1;
  Modelica.Blocks.Interfaces.RealInput Io(unit="uE m-2 s-1") "Input radiation";
end Bio1Rad1;
```

PhotoBioLib.Interfaces.Partials.Biomass.Bio1T1

Interface with a biomass connector and a input temperature

Connectors

Type	Name	Description
BioCon	lCon	Liquid connector
input RealInput	T	Input temperature [ºK]

Modelica definition

```
partial model Bio1T1
  "Interface with a biomass connector and a input temperature"

  PhotoBioLib.Interfaces.BioCon lCon "Liquid connector";

  Modelica.Blocks.Interfaces.RealInput T(unit="ºK") "Input temperature";
end Bio1T1;
```

PhotoBioLib.Interfaces.Partials.Biomass.Bio2

Interface with two biomass connectors

Connectors

Type	Name	Description
BioCon	con1	Liquid connector on the left
BioCon	con2	Liquid connector on the rigth

Modelica definition

```
partial model Bio2 "Interface with two biomass connectors"

  PhotoBioLib.Interfaces.BioCon con1 "Liquid connector on the left";
  PhotoBioLib.Interfaces.BioCon con2 "Liquid connector on the rigth";
end Bio2;
```

PhotoBioLib.Interfaces.Partials.Biomass.Bio2Flow1

Interface with two biomass connectors and a input liquid flow

Connectors

Type	Name	Description
BioCon	con1	Liquid connector on the left
BioCon	con2	Liquid connector on the rigth
input RealInput	qLiq	Input liquid flow [m3 s-1]

Modelica definition

```
partial model Bio2Flow1
  "Interface with two biomass connectors and a input liquid flow"
extends PhotoBioLib.Interfaces.Partials.Biomass.Bio2;

  Modelica.Blocks.Interfaces.RealInput qLiq(unit="m3 s-1") "Input liquid flow";
end Bio2Flow1;
```

PhotoBioLib.Interfaces.Partials.Biomass.Bio2Flow1Heat1

Exchanger interface

Connectors

Type	Name	Description
BioCon	lConIn	Input liquid connector
BioCon	lConOut	Output liquid connector
HeatCon	hCon	Heat connector

Modelica definition

```
partial model Bio2Flow1Heat1 "Exchanger interface"

  PhotoBioLib.Interfaces.BioCon lConIn "Input liquid connector";
  PhotoBioLib.Interfaces.BioCon lConOut "Output liquid connector";
  HeatCon hCon "Heat connector";
end Bio2Flow1Heat1;
```

PhotoBioLib.Interfaces.Partials.Biomass.Bio2Gas2

Interface with two biomass and two gas connectors

Connectors

Type	Name	Description
BioCon	conL1	Liquid connector on the left
BioCon	conL2	Liquid connector on the rigth
GasCon	conG1	Gas connector on the left
GasCon	conG2	Gas connector on the rigth

Modelica definition

```
partial model Bio2Gas2
  "Interface with two biomass and two gas connectors"

  PhotoBioLib.Interfaces.BioCon conL1 "Liquid connector on the left";
  PhotoBioLib.Interfaces.BioCon conL2 "Liquid connector on the rigth";
  GasCon conG1 "Gas connector on the left";
  GasCon conG2 "Gas connector on the rigth";

end Bio2Gas2;
```

PhotoBioLib.Interfaces.Partials.Biomass.Bio2Gas2Rad1T1

Interface with two gas connectors, two biomass connectors, a input radiation and a input temperature

Connectors

Type	Name	Description
BioCon	conL1	Liquid connector on the left
BioCon	conL2	Liquid connector on the rigth
GasCon	conG1	Gas connector on the left
GasCon	conG2	Gas connector on the rigth
input RealInput	Io	Input radiation [uE m-2 s-1]
input RealInput	T	Input temperatura [ºK]

Modelica definition

```
partial model Bio2Gas2Rad1T1
  "Interface with two gas connectors, two biomass connectors, a input radiation and a input temperature"
extends PhotoBioLib.Interfaces.Partials.Biomass.Bio2Gas2;

  Modelica.Blocks.Interfaces.RealInput Io(unit="uE m-2 s-1") "Input radiation";
  Modelica.Blocks.Interfaces.RealInput T(unit="ºK") "Input temperatura";
end Bio2Gas2Rad1T1;
```

PhotoBioLib.Interfaces.Partials.Biomass.Bio2Gas2Rad1T1Heat1_V

Interface with two gas connectors, two biomass connectors, a heat connector, a input radiation and a input temperature

Connectors

Type	Name	Description
BioCon	conLIn	Liquid connector on the left
BioCon	conLOut	Liquid connector on the rigth
GasCon	conGOut	Gas connector on the left
GasCon	conGIn	Gas connector on the rigth
input RealInput	Io	Input radiation [uE m-2 s-1]
input RealInput	T	Input temperature [°K]
HeatCon	hCon	Heat connector

Modelica definition

```
partial model Bio2Gas2Rad1T1Heat1_V
  "Interface with two gas connectors, two biomass connectors, a heat connector, a input radiation and a input temperature"

  PhotoBioLib.Interfaces.BioCon conLIn "Liquid connector on the left";
  PhotoBioLib.Interfaces.BioCon conLOut "Liquid connector on the rigth";
  GasCon conGOut "Gas connector on the left";
  GasCon conGIn "Gas connector on the rigth";
  Modelica.Blocks.Interfaces.RealInput Io(unit="uE m-2 s-1") "Input radiation";
  Modelica.Blocks.Interfaces.RealInput T(unit="°K") "Input temperature";

  HeatCon hCon "Heat connector";
end Bio2Gas2Rad1T1Heat1_V;
```

PhotoBioLib.Interfaces.Partials.Biomass.Bio2OutN

Interface with two biomass connectors with N outputs

Parameters

Type	Name	Default	Description
Integer	nout	1	Number of outputs

Connectors

Type	Name	Description
output RealOutput	y[nout]	Outputs
BioCon	con1	Liquid connector on the left
BioCon	con2	Liquid connector on the rigth

Modelica definition

```
partial model Bio2OutN
  "Interface with two biomass connectors with N outputs"
  parameter Integer nout(min=1) = 1 "Number of outputs";
  Modelica.Blocks.Interfaces.RealOutput y[nout](redeclare type SignalType =
        Real) "Outputs";
  PhotoBioLib.Interfaces.BioCon con1 "Liquid connector on the left";
  PhotoBioLib.Interfaces.BioCon con2 "Liquid connector on the rigth";
end Bio2OutN;
```

PhotoBioLib.Interfaces.Partials.Gas

Partials with gas connectors

Package Content

Name	Description
Gas1	Interface with a gas connector
Gas1OutN	Interface with a gas connector with N outputs
Gas2	Interface with two gas connectors
Gas2Flow1	Interface with two gas connector and a input gas flow
Gas2OutN	Interface with two gas connectors with N outputs
Gas3	Interface with three gas connectors

PhotoBioLib.Interfaces.Partials.Gas.Gas1

Interface with a gas connector

Connectors

Type	Name	Description
GasCon	con	Gas connector

Modelica definition

```
partial model Gas1 "Interface with a gas connector"

  PhotoBioLib.Interfaces.GasCon con "Gas connector";
end Gas1;
```

PhotoBioLib.Interfaces.Partials.Gas.Gas1OutN

Interface with a gas connector with N outputs

Parameters

Type	Name	Default	Description
Integer	nout	1	Number of outputs

Connectors

Type	Name	Description
output RealOutput	y[nout]	Outputs
GasCon	con	Gas connector

Modelica definition

```
partial model Gas1OutN
  "Interface with a gas connector with N outputs"

  parameter Integer nout(min=1) = 1 "Number of outputs";
  Modelica.Blocks.Interfaces.RealOutput y[nout](redeclare type SignalType =
        Real) "Outputs";

  PhotoBioLib.Interfaces.GasCon con "Gas connector";
end Gas1OutN;
```

PhotoBioLib.Interfaces.Partials.Gas.Gas2

Interface with two gas connectors

Connectors

Type	Name	Description
GasCon	con1	Gas connector on the left
GasCon	con2	Gas connector on the rigth

Modelica definition

```
partial model Gas2 "Interface with two gas connectors"

  PhotoBioLib.Interfaces.GasCon con1 "Gas connector on the left";
  PhotoBioLib.Interfaces.GasCon con2 "Gas connector on the rigth";
end Gas2;
```

PhotoBioLib.Interfaces.Partials.Gas.Gas2Flow1

Interface with two gas connector and a input gas flow

Connectors

Type	Name	Description
GasCon	con1	Gas connector on the left
GasCon	con2	Gas connector on the rigth
input RealInput	qGas	Input gas flow [m3 s-1]

Modelica definition

```
partial model Gas2Flow1
  "Interface with two gas connector and a input gas flow"
extends PhotoBioLib.Interfaces.Partials.Gas.Gas2;

  Modelica.Blocks.Interfaces.RealInput qGas(unit="m3 s-1") "Input gas flow";
end Gas2Flow1;
```

PhotoBioLib.Interfaces.Partials.Gas.Gas2OutN

Interface with two gas connectors with N outputs

Parameters

Type	Name	Default	Description
Integer	nout	1	Number of outputs

Connectors

Type	Name	Description
output RealOutput	y[nout]	
GasCon	con1	Gas connector on the left
GasCon	con2	Gas connector on the rigth

Modelica definition

```
partial model Gas2OutN
  "Interface with two gas connectors with N outputs"
  parameter Integer nout(min=1) = 1 "Number of outputs";
  Modelica.Blocks.Interfaces.RealOutput y[nout];

  PhotoBioLib.Interfaces.GasCon con1 "Gas connector on the left";
  PhotoBioLib.Interfaces.GasCon con2 "Gas connector on the rigth";
end Gas2OutN;
```

PhotoBioLib.Interfaces.Partials.Gas.Gas3

Interface with three gas connectors

Connectors

Type	Name	Description
GasCon	con1	Gas connector on the left
GasCon	con2	Gas connector on the rigth
GasCon	con3	Gas connector on top

Modelica definition

```
partial model Gas3 "Interface with three gas connectors"

  PhotoBioLib.Interfaces.GasCon con1 "Gas connector on the left";
  PhotoBioLib.Interfaces.GasCon con2 "Gas connector on the rigth";
  GasCon con3 "Gas connector on top";
end Gas3;
```

PhotoBioLib.Interfaces.Partials.Heat

Partials with heat connectors

Package Content

Name	Description
Heat1Rad1	Interface with a heat connector and a input radiation
Heat1Rad1T1	Interface with a heat connector, a input radiation and a input temperature
Heat1Rad2T1	Interface with a heat connector, a input radiation, a input temperature and a output radiation
Heat2	Interface with two heat connectors

PhotoBioLib.Interfaces.Partials.Heat.Heat1Rad1

Interface with a heat connector and a input radiation

Connectors

Type	Name	Description
HeatCon	hCon	Heat connector
input RealInput	Ic	Radiation [W m-2]

Modelica definition

```
partial model Heat1Rad1
  "Interface with a heat connector and a input radiation"

  HeatCon hCon "Heat connector";
  Modelica.Blocks.Interfaces.RealInput Ic(unit="W m-2") "Radiation";
end Heat1Rad1;
```

PhotoBioLib.Interfaces.Partials.Heat.Heat1Rad1T1

Interface with a heat connector, a input radiation and a input temperature

Connectors

Type	Name	Description
HeatCon	hCon	Heat connector
input RealInput	Io	Radiation [uE m-2 s-1]
input RealInput	Text	Temperature [°K]

Modelica definition

```
partial model Heat1Rad1T1
  "Interface with a heat connector, a input radiation and a input temperature"

  HeatCon hCon "Heat connector";
  Modelica.Blocks.Interfaces.RealInput Io(unit="uE m-2 s-1") "Radiation";
  Modelica.Blocks.Interfaces.RealInput Text(unit="°K") "Temperature";
end Heat1Rad1T1;
```

PhotoBioLib.Interfaces.Partials.Heat.Heat1Rad2T1

Interface with a heat connector, a input radiation, a input temperature and a output radiation

Connectors

Type	Name	Description
HeatCon	hCon	Heat connector

input RealInput	Io	Input radiation [uE m-2 s-1]
input RealInput	Text	Temperature [°K]
output RealOutput	Ic	Output radiation [W m-2]

Modelica definition

```
partial model Heat1Rad2T1
  "Interface with a heat connector, a input radiation, a input temperature and a output radiation"

  HeatCon hCon "Heat connector";
  Modelica.Blocks.Interfaces.RealInput Io(unit="uE m-2 s-1") "Input radiation";
  Modelica.Blocks.Interfaces.RealInput Text(unit="9K") "Temperature";
  Modelica.Blocks.Interfaces.RealOutput Ic(unit="W m-2") "Output radiation";
end Heat1Rad2T1;
```

PhotoBioLib.Interfaces.Partials.Heat.Heat2

Interface with two heat connectors

Connectors

Type	Name	Description
HeatCon	hCon1	Heat connector on the left
HeatCon	hCon2	Heat connector on the right

Modelica definition

```
partial model Heat2 "Interface with two heat connectors"

  HeatCon hCon1 "Heat connector on the left";
  HeatCon hCon2 "Heat connector on the right";
end Heat2;
```

PhotoBioLib.Interfaces.Partials.Liquid

Partials with liquid connectors

Package Content

Name	Description
Liq1	Interface with a liquid connector
Liq1Heat1	Interface with a liquid and a heat connectors
Liq1Heat1_Inv	Interface with a liquid and a heat connectors
Liq1OutN	Interface with a liquid connector and N outputs
Liq1T1	Interface with a liquid connector and a input temperature
Liq2	Interface with two liquid connectors
Liq2Flow1	Interface with two liquid connector and a input liquid flow
Liq2Flow1Heat1	Exchanger interface
Liq2OutN	Interface with two liquid connectors with N outputs

PhotoBioLib.Interfaces.Partials.Liquid.Liq1

Interface with a liquid connector

Connectors

Type	Name	Description
LiqCon	lCon	Liquid connector

Modelica definition

```
partial model Liq1 "Interface with a liquid connector"

  PhotoBioLib.Interfaces.LiqCon lCon "Liquid connector";
end Liq1;
```

PhotoBioLib.Interfaces.Partials.Liquid.Liq1Heat1

Interface with a liquid and a heat connectors

Connectors

Type	Name	Description
LiqCon	lCon	Liquid connector
HeatCon	hCon	Heat connector

Modelica definition

```
partial model Liq1Heat1
  "Interface with a liquid and a heat connectors"

  PhotoBioLib.Interfaces.LiqCon lCon "Liquid connector";
  HeatCon hCon "Heat connector";

end Liq1Heat1;
```

PhotoBioLib.Interfaces.Partials.Liquid.Liq1Heat1_Inv

Interface with a liquid and a heat connectors

Connectors

Type	Name	Description
LiqCon	lCon	Liquid connector
HeatCon	hCon	Heat connector

Modelica definition

```
partial model Liq1Heat1_Inv
  "Interface with a liquid and a heat connectors"

  PhotoBioLib.Interfaces.LiqCon lCon "Liquid connector";
  HeatCon hCon "Heat connector";

end Liq1Heat1_Inv;
```

PhotoBioLib.Interfaces.Partials.Liquid.Liq1OutN

Interface with a liquid connector and N outputs

Parameters

Type	Name	Default	Description
Integer	nout	1	Number of outputs

Connectors

Type	Name	Description
output RealOutput	y[nout]	Outputs
LiqCon	lCon	Liquid connector

Modelica definition

```
partial model Liq1OutN
  "Interface with a liquid connector and N outputs"

  parameter Integer nout(min=1) = 1 "Number of outputs";
  Modelica.Blocks.Interfaces.RealOutput y[nout](redeclare type SignalType =
        Real) "Outputs";

  PhotoBioLib.Interfaces.LiqCon lCon "Liquid connector";

end Liq1OutN;
```

PhotoBioLib.Interfaces.Partials.Liquid.Liq1T1

Interface with a liquid connector and a input temperature

Connectors

Type	Name	Description
LiqCon	lCon	Liquid connector
input RealInput	T	Input temperature [°K]

Modelica definition

```
partial model Liq1T1
  "Interface with a liquid connector and a input temperature"

  PhotoBioLib.Interfaces.LiqCon lCon "Liquid connector";

  Modelica.Blocks.Interfaces.RealInput T(unit="°K") "Input temperature";
end Liq1T1;
```

PhotoBioLib.Interfaces.Partials.Liquid.Liq2

Interface with two liquid connectors

Connectors

Type	Name	Description
LiqCon	con1	Liquid connector on the left
LiqCon	con2	Liquid connector on the rigth

Modelica definition

```
partial model Liq2 "Interface with two liquid connectors"

  PhotoBioLib.Interfaces.LiqCon con1 "Liquid connector on the left";
  PhotoBioLib.Interfaces.LiqCon con2 "Liquid connector on the rigth";
end Liq2;
```

PhotoBioLib.Interfaces.Partials.Liquid.Liq2Flow1

Interface with two liquid connector and a input liquid flow

Connectors

Type	Name	Description
LiqCon	con1	Liquid connector on the left
LiqCon	con2	Liquid connector on the rigth
input RealInput	qLiq	Input liquid flow [m3 s-1]

Modelica definition

```
partial model Liq2Flow1
  "Interface with two liquid connector and a input liquid flow"
extends PhotoBioLib.Interfaces.Partials.Liquid.Liq2;

  Modelica.Blocks.Interfaces.RealInput qLiq(unit="m3 s-1") "Input liquid flow";
end Liq2Flow1;
```

PhotoBioLib.Interfaces.Partials.Liquid.Liq2Flow1Heat1

Exchanger interface

Connectors

Type	Name	Description
LiqCon	lConIn	Input liquid connector
LiqCon	lConOut	Output liquid connector
HeatCon	hCon	Heat connector

Modelica definition

```
partial model Liq2Flow1Heat1 "Exchanger interface"

  PhotoBioLib.Interfaces.LiqCon lConIn "Input liquid connector";
  PhotoBioLib.Interfaces.LiqCon lConOut "Output liquid connector";
  PhotoBioLib.Interfaces.HeatCon hCon "Heat connector";
end Liq2Flow1Heat1;
```

PhotoBioLib.Interfaces.Partials.Liquid.Liq2OutN

Interface with two liquid connectors with N outputs

Parameters

Type	Name	Default	Description
Integer	nout	1	Number of outputs

Connectors

Type	Name	Description
output RealOutput	y[nout]	
LiqCon	con1	Liquid connector on the left
LiqCon	con2	Liquid connector on the rigth

Modelica definition

```
partial model Liq2OutN
  "Interface with two liquid connectors with N outputs"
  parameter Integer nout(min=1) = 1 "Number of outputs";
  Modelica.Blocks.Interfaces.RealOutput y[nout];

  PhotoBioLib.Interfaces.LiqCon con1 "Liquid connector on the left";
  PhotoBioLib.Interfaces.LiqCon con2 "Liquid connector on the rigth";

end Liq2OutN;
```

PhotoBioLib.Interfaces.Partials.Others

Others partials

Package Content

Name	Description
Column	Column interface
Column_HeatExchanger	Column with exchanger interface
Flow2	Interface with a input biomass and a input gas flows
Loop	Loop interface
Rad1	Interface with a output radiation

PhotoBioLib.Interfaces.Partials.Others.Column

Column interface

Connectors

Type	Name	Description
BioCon	conLIn	Input liquid from the loop
BioCon	conMOut	Liquid overflow
BioCon	conMIn	Input medium
BioCon	conLOut	Output liquid to the loop

GasCon	conGIn	Input gas
GasCon	conGOut	Output gas
input RealInput	Io	Solar radiation [uE m-2 s-1]
input RealInput	T	Environment temperature [°K]

Modelica definition

```
partial model Column "Column interface"

  PhotoBioLib.Interfaces.BioCon conLIn "Input liquid from the loop";
  PhotoBioLib.Interfaces.BioCon conMOut "Liquid overflow";
  PhotoBioLib.Interfaces.BioCon conMIn "Input medium";
  PhotoBioLib.Interfaces.BioCon conLOut "Output liquid to the loop";
  GasCon conGIn "Input gas";
  GasCon conGOut "Output gas";
  Modelica.Blocks.Interfaces.RealInput Io(unit="uE m-2 s-1") "Solar radiation";
  Modelica.Blocks.Interfaces.RealInput T(unit="°K") "Environment temperature";
end Column;
```

PhotoBioLib.Interfaces.Partials.Others.Column_HeatExchanger

Column with exchanger interface

Connectors

Type	Name	Description
BioCon	conLIn	Input liquid from the loop
BioCon	conMOut	Liquid overflow
BioCon	conMIn	Input medium
BioCon	conLOut	Output liquid to the loop
GasCon	conGIn	Input gas
GasCon	conGOut	Output gas
input RealInput	Io	Solar radiation [uE m-2 s-1]
input RealInput	T	Environment temperature [°K]
LiqCon	lConIntIn	Input liquid to the exchanger
LiqCon	lConIntOut	Output liquid from the exchanger

Modelica definition

```
partial model Column_HeatExchanger "Column with exchanger interface"
  extends PhotoBioLib.Interfaces.Partials.Others.Column;
  PhotoBioLib.Interfaces.LiqCon lConIntIn "Input liquid to the exchanger";
  PhotoBioLib.Interfaces.LiqCon lConIntOut "Output liquid from the exchanger";

end Column_HeatExchanger;
```

PhotoBioLib.Interfaces.Partials.Others.Flow2

Interface with a input biomass and a input gas flows

Connectors

Type	Name	Description
input RealInput	Qliq	Input liquid flow
input RealInput	Qgas	Input gas flow

Modelica definition

```
partial model Flow2
  "Interface with a input biomass and a input gas flows"

  Modelica.Blocks.Interfaces.RealInput Qliq "Input liquid flow";
  Modelica.Blocks.Interfaces.RealInput Qgas "Input gas flow";
end Flow2;
```

PhotoBioLib.Interfaces.Partials.Others.Loop

Loop interface

Connectors

Type	Name	Description
BioCon	conLOut	Liquid output
GasCon	conGOut	Gas output
GasCon	conGIn	Gas input
BioCon	conLIn	Liquid input
input RealInput	Io	Input radiation [uE m-2 s-1]
GasCon	conGprobe	Gas probe
BioCon	conLprobe	Liquid probe
input RealInput	Tamb	Input temperature [°K]

Modelica definition

```
partial model Loop "Loop interface"

  PhotoBioLib.Interfaces.BioCon conLOut "Liquid output";
  GasCon conGOut "Gas output";
  GasCon conGIn "Gas input";
  PhotoBioLib.Interfaces.BioCon conLIn "Liquid input";
  Modelica.Blocks.Interfaces.RealInput Io(unit="uE m-2 s-1") "Input radiation";
  GasCon conGprobe "Gas probe";
  PhotoBioLib.Interfaces.BioCon conLprobe "Liquid probe";
  Modelica.Blocks.Interfaces.RealInput Tamb(unit="°K") "Input temperature";
end Loop;
```

PhotoBioLib.Interfaces.Partials.Others.Rad1

Interface with a output radiation

Connectors

Type	Name	Description
output RealOutput	Io	Radiation [uE m-2 s-1]

Modelica definition

```
partial model Rad1 "Interface with a output radiation"
  Modelica.Blocks.Interfaces.RealOutput Io(unit="uE m-2 s-1") "Radiation";
end Rad1;
```

PhotoBioLib.Parts

Parts package

Package Content

Name	Description
Biomass	Biomass parts package
Column	Column
Column_HeatExchanger	Column with heat exchanger
Gas	Gas parts package
HeatExchanger	Heat exchanger
Liquid	Liquid parts package
Loop	Loop with nE sections
Radiation	Radiation package

PhotoBioLib.Parts.Biomass

Biomass parts package

Package Content

Name	Description

BioPump	Biomass pump
BioSource	Biomass source
BioTank	Biomass tank

PhotoBioLib.Parts.Biomass.BioPump

Biomass pump

Parameters

Type	Name	Default	Description
HeatPar	heatPar		Heat parameters
Real	Cv	heatPar.Cv	Volumetric heat capacity [J m-3 ºK-1)]

Connectors

Type	Name	Description
BioCon	con1	Liquid connector on the left
BioCon	con2	Liquid connector on the rigth
input RealInput	qLiq	Input liquid flow [m3 s-1]

Modelica definition

```
model BioPump "Biomass pump"
    extends PhotoBioLib.Basic.Biomass.BioResistor(redeclare class icon =
          PhotoBioLib.Icons.Parts.BioPumpIcon);
end BioPump;
```

PhotoBioLib.Parts.Biomass.BioSource

Biomass source

Parameters

Type	Name	Default	Description
InitialBio	initialLiq		Initial liquid values
Real	Cbini	initialLiq.Cbini	Initial biomass concentration [Kg m-3]
Real	O2ini	initialLiq.O2ini	Initial oxygen concentration [mol m-3]
Real	CO2ini	initialLiq.CO2ini	Initial carbon dioxide concentration [mol m-3]
Real	Ctini	initialLiq.Ctini	Initial total carbon concentration [mol m-3]
Real	Hini	initialLiq.Hini	Initial hidrogen concentration [mol m-3]
Real	Volini	initialLiq.Volini	Initial liquid volume [m3]
Real	Tini	initialLiq.Tini	Initial liquid temperature [ºK]

Connectors

Type	Name	Description
BioCon	lCon	Liquid connector
input RealInput	T	Input temperature [ºK]

Modelica definition

```
model BioSource "Biomass source"
  extends PhotoBioLib.Interfaces.Partials.Biomass.BiolTl;
  extends PhotoBioLib.Icons.Parts.BioSourceIcon;

  parameter PhotoBioLib.Records.Initial.InitialBio initialLiq
    "Initial liquid values";
  import Modelica.Math.log10;

  parameter Real Cbini(unit="Kg m-3") = initialLiq.Cbini
    "Initial biomass concentration";
  parameter Real O2ini(unit="mol m-3") = initialLiq.O2ini
    "Initial oxygen concentration";
  parameter Real CO2ini(unit="mol m-3") = initialLiq.CO2ini
    "Initial carbon dioxide concentration";
  parameter Real Ctini(unit="mol m-3") = initialLiq.Ctini
    "Initial total carbon concentration";
  parameter Real Hini(unit="mol m-3") = initialLiq.Hini
    "Initial hidrogen concentration";
  parameter Real Volini(unit="m3") = initialLiq.Volini "Initial liquid volume";

  Real pH(unit="1") "pH";

  parameter Real Tini(unit="ºK") = initialLiq.Tini "Initial liquid temperature";

initial equation
  lCon.Vol = Volini;

equation
  // Calor
  lCon.T = T;

  // Cálculo de concentraciones
  lCon.Cb = Cbini;
  lCon.O2 = O2ini;
  lCon.CO2 = CO2ini;
  lCon.Ct = Ctini;
  lCon.H = Hini;

  // Cálculo de pH
  pH = -log10(lCon.H);

  der(lCon.Vol) = lCon.flowVol;

end BioSource;
```

PhotoBioLib.Parts.Biomass.BioTank

Biomass tank

Parameters

Type	Name	Default	Description
InitialBio	initialLiq	extends PhotoBioLib.Records....	Initial liquid values
HeatPar	heatPar		Heat parameters
Real	Cv	heatPar.Cv	Volumetric heat capacity [J m-3 ºK-1)]
Real	Volini	1E-40	Initial liquid volume [m3]
Real	Tini	initialLiq.Tini	Initital temperature [ºK]
Real	Cbini	initialLiq.Cbini	Initial biomass concentration [Kg m-3]
Real	O2ini	initialLiq.O2ini	Initial oxygen concentration [mol m-3]
Real	CO2ini	initialLiq.CO2ini	Initial carbon dioxide concetration [mol m-3]
Real	Ctini	initialLiq.Ctini	Initial total inorganic carbon concentration [mol m-3]
Real	Hini	initialLiq.Hini	Initial hidrogen concentration [mol m3-]

Connectors

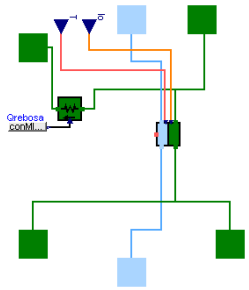
Type	Name	Description
BioCon	lCon	Liquid connector
HeatCon	hCon	Heat connector

Modelica definition

```
model BioTank "Biomass tank"
  extends PhotoBioLib.Basic.Biomass.BioCapacitor(
    Volini=1E-40,
    redeclare class icon = PhotoBioLib.Icons.Parts.BioTankIcon,
    redeclare class con = PhotoBioLib.Interfaces.Partials.Biomass.BiolHeatl_Inv);
end BioTank;
```

PhotoBioLib.Parts.Column

Column



Parameters

Type	Name	Default	Description
GeometryColumn	geometry		Geometrical parameters
HeatParColumn	heatPar		Heat parameters
MassTransferColumn	massTransfer		Mass transfer parameters
Algae	algae		Alage parameters
InitialColumn	initialColumn		Initial values of the column

Connectors

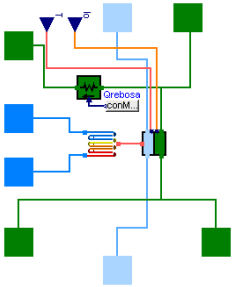
Type	Name	Description
BioCon	conLIn	Input liquid from the loop
BioCon	conMOut	Liquid overflow
BioCon	conMIn	Input medium
BioCon	conLOut	Output liquid to the loop
GasCon	conGIn	Input gas
GasCon	conGOut	Output gas
input RealInput	Io	Solar radiation [uE m-2 s-1]
input RealInput	T	Environment temperature [ºK]

Modelica definition

```
model Column "Column"
  extends PhotoBioLib.Interfaces.Partials.Others.Column;
  extends PhotoBioLib.Icons.Parts.ColumnIcon;

  parameter PhotoBioLib.Records.Geometry.GeometryColumn geometry
    "Geometrical parameters";
  parameter PhotoBioLib.Records.Heat.HeatParColumn heatPar "Heat parameters";
  parameter PhotoBioLib.Records.MassTransfer.MassTransferColumn massTransfer
    "Mass transfer parameters";

  parameter PhotoBioLib.Records.Algae.Algae algae "Alage parameters";
  parameter PhotoBioLib.Records.Initial.InitialColumn initialColumn
    "Initial values of the column";
  constant PhotoBioLib.Records.Initial.InitialGas initialGas=initialColumn.initialGas
    "Initial gas values";
  constant PhotoBioLib.Records.Initial.InitialBio initialLiq=initialColumn.initialLiq
    "Initial liquid values";


  Modelica.Blocks.Sources.RealExpression Qrebosa(y=conMIn.flowVol);
  PhotoBioLib.Basic.Biomass.BioResistor liqResistor(heatPar=heatPar);
  PhotoBioLib.Basic.Biomass.ColumnSection ColumnSection(
    initialGas=initialGas,
    initialLiq=initialLiq,
    algae=algae,
    geometry=geometry,
    heatPar=heatPar,
    massTransfer=massTransfer);
equation

  connect(liqResistor.con2, conMOut);
  connect(Qrebosa.y, liqResistor.qLiq);
  connect(ColumnSection.conLOut, conLOut);
  connect(conLIn,ColumnSection. conLIn);
  connect(ColumnSection.conGIn, conGIn);
  connect(conMIn,ColumnSection. conLOut);
  connect(liqResistor.con1,ColumnSection. conLIn);
  connect(conGOut,ColumnSection. conGOut);
  connect(Io,ColumnSection. Io);
  connect(ColumnSection.T, T);
end Column;
```

PhotoBioLib.Parts.Column_HeatExchanger

Column with heat exchanger



Parameters

Type	Name	Default	Description
GeometryColumn	geometry		Geometrical parameters
HeatParColumn	heatPar		Heat parameters
MassTransferColumn	massTransfer		Mass transfer parameters
GeometryExchanger	geometryExchanger		Exchanger geometrical parameters
HeatParExchanger	heatParExchanger		Exchanger heat parameters
InitialColumn	initialColumn		Initial values
InitialExchanger	initialExchanger		Initial exchanger parameters
Algae	algae		Algae parameters

Connectors

Type	Name	Description
BioCon	conLIn	Input liquid from the loop
BioCon	conMOut	Liquid overflow
BioCon	conMIn	Input medium
BioCon	conLOut	Output liquid to the loop
GasCon	conGIn	Input gas
GasCon	conGOut	Output gas
input RealInput	Io	Solar radiation [uE m-2 s-1]
input RealInput	T	Environment temperature [ºK]
LiqCon	lConIntIn	Input liquid to the exchanger
LiqCon	lConIntOut	Output liquid from the exchanger

Modelica definition

```
model Column_HeatExchanger "Column with heat exchanger"
  extends PhotoBioLib.Interfaces.Partials.Others.Column_HeatExchanger;
  extends PhotoBioLib.Icons.Parts.Column_HeatExchangerIcon;

  parameter PhotoBioLib.Records.Geometry.GeometryColumn geometry
    "Geometrical parameters";
  parameter PhotoBioLib.Records.Heat.HeatParColumn heatPar "Heat parameters";
  parameter PhotoBioLib.Records.MassTransfer.MassTransferColumn massTransfer
    "Mass transfer parameters";

  parameter PhotoBioLib.Records.Geometry.GeometryExchanger geometryExchanger
    "Exchanger geometrical parameters";
  parameter PhotoBioLib.Records.Heat.HeatParExchanger heatParExchanger
    "Exchanger heat parameters";
  parameter PhotoBioLib.Records.Initial.InitialColumn initialColumn
    "Initial values";
  constant PhotoBioLib.Records.Initial.InitialGas initialGas=initialColumn.initialGas
    "Initial gas values";
  constant PhotoBioLib.Records.Initial.InitialBio initialLiq=initialColumn.initialLiq
    "Initial liquid values";
  parameter PhotoBioLib.Records.Initial.InitialExchanger initialExchanger
    "Initial exchanger parameters";
  parameter PhotoBioLib.Records.Algae.Algae algae "Algae parameters";

  Modelica.Blocks.Sources.RealExpression Qrebosa(y=conMIn.flowVol);
  PhotoBioLib.Basic.Biomass.BioResistor liqResistor(heatPar=heatPar);
  PhotoBioLib.Basic.Biomass.ColumnSection ColumnSection(
    initialGas=initialGas,
    initialLiq=initialLiq,
    algae=algae,
    geometry=geometry,
    heatPar=heatPar,
    massTransfer=massTransfer);
  PhotoBioLib.Parts.HeatExchanger Heat_Exchanger(
    initialExchanger=initialExchanger,
    geometry=geometryExchanger,
```

```
    heatPar=heatParExchanger) "Heat Exchanger";

equation
  connect(liqResistor.con2, conMOut);
  connect(Qrebosa.y, liqResistor.qLiq);
  connect(ColumnSection.conLOut, conLOut);
  connect(conLIn,ColumnSection. conLIn);
  connect(ColumnSection.conGIn, conGIn);
  connect(conMIn,ColumnSection. conLOut);
  connect(liqResistor.con1,ColumnSection. conLIn);
  connect(conGOut,ColumnSection. conGOut);
  connect(Io,ColumnSection. Io);
  connect(ColumnSection.T, T);
  connect(Heat_Exchanger.hCon,ColumnSection. hCon);
  connect(lConIntOut,Heat_Exchanger. lConOut);
  connect(lConIntIn,Heat_Exchanger. lConIn);
end Column_HeatExchanger;
```

PhotoBioLib.Parts.Gas

Gas parts package

Package Content

Name	Description
GasPump	Gas pump
GasSource	Gas source
GasTank	Gas tank
Tpipe	Pipe with T form

PhotoBioLib.Parts.Gas.GasPump

Gas pump

Parameters

Type	Name	Default	Description
replaceable class icon		GasResistorIcon	

Connectors

Type	Name	Description
GasCon	con1	Gas connector on the left
GasCon	con2	Gas connector on the rigth
input RealInput	qGas	Input gas flow [m3 s-1]
replaceable class icon		

Modelica definition

```
model GasPump "Gas pump"
  extends PhotoBioLib.Basic.Gas.GasResistor(redeclare class icon =
        PhotoBioLib.Icons.Parts.GasPumpIcon);
end GasPump;
```

PhotoBioLib.Parts.Gas.GasSource

Gas source

Parameters

Type	Name	Default	Description
InitialGas	initialGas		Initial gas values
Real	O2ini	initialGas.O2ini	Initial oxygen concentration [Kg m-3]
Real	CO2ini	initialGas.CO2ini	Initial carbon dioxide concentration [mol m-3]
Real	N2ini	initialGas.N2ini	Initial nitrogen concentration [mol m-3]
Real	Volini	initialGas.Volini	Initial gas volume [m3]

Connectors

Type	Name	Description
GasCon	con	Gas connector

Modelica definition

```
model GasSource "Gas source"
  extends PhotoBioLib.Interfaces.Partials.Gas.Gasl;
  extends PhotoBioLib.Icons.Parts.GasSourceIcon;
  parameter PhotoBioLib.Records.Initial.InitialGas initialGas
    "Initial gas values";

  // Parámetros para marcar potenciales iniciales
  parameter Real O2ini(unit="Kg m-3") = initialGas.O2ini
    "Initial oxygen concentration";
  parameter Real CO2ini(unit="mol m-3") = initialGas.CO2ini
    "Initial carbon dioxide concentration";
  parameter Real N2ini(unit="mol m-3") = initialGas.N2ini
    "Initial nitrogen concentration";
  parameter Real Volini(unit="m3") = initialGas.Volini "Initial gas volume";

initial equation
  con.Vol = Volini;

equation
  con.O2 = O2ini;
  con.CO2 = CO2ini;
  con.N2 = N2ini;

  der(con.Vol) = con.flowVol;
end GasSource;
```

PhotoBioLib.Parts.Gas.GasTank

Gas tank

Parameters

Type	Name	Default	Description
replaceable class icon		GasCapacitorIcon	Icon
InitialGas	initialGas		Initial gas values
Real	O2ini	initialGas.O2ini	Initial oxygen concentration [mol m-3]
Real	CO2ini	initialGas.CO2ini	Initial carbon dioxide concentration [mol m-3]
Real	N2ini	initialGas.N2ini	Inititial nitrogen concentration [mol m-3]
Real	Volini	initialGas.Volini	Initial gas volume [m3]
Real	eps	0	
Real	eps2	10*eps	

Connectors

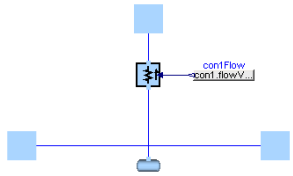
Type	Name	Description
GasCon	con	Gas connector
replaceable class icon		Icon

Modelica definition

```
model GasTank "Gas tank"
  extends PhotoBioLib.Basic.Gas.GasCapacitor(redeclare class icon =
        PhotoBioLib.Icons.Parts.GasTankIcon);
end GasTank;
```

PhotoBioLib.Parts.Gas.Tpipe

Pipe with T form



Connectors

Type	Name	Description

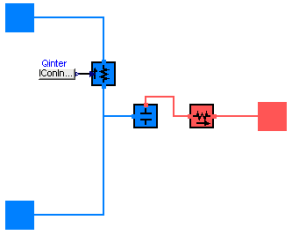
GasCon	con1	Gas connector on the left
GasCon	con2	Gas connector on the rigth
GasCon	con3	Gas connector on top

Modelica definition

```
model Tpipe "Pipe with T form"
  extends Interfaces.Partials.Gas.Gas3;
  extends Icons.Parts.TpipeIcon;
  /*
equation
  -con3.flowVol =con1.flowVol + con2.flowVol;
  -con3.flowO2 =con1.flowO2 + con2.flowO2;
  -con3.flowCO2 =con1.flowCO2 + con2.flowCO2;
  -con3.flowN2 =con1.flowN2 + con2.flowN2;
*/
  Basic.Gas.GasResistor gasResistor1;
protected
  Modelica.Blocks.Sources.RealExpression con1Flow(y=con1.flowVol + con2.flowVol);
public
  GasTank gasTank(
    Volini=0.1,
    O2ini=10.5,
    CO2ini=0.015,
    N2ini=39.485);
equation
  connect(con1, gasTank.con);
  connect(con2, gasTank.con);
  connect(gasResistor1.con1, gasTank.con);
  connect(con3, gasResistor1.con2);
  connect(gasResistor1.qGas, con1Flow.y);
end Tpipe;
```

PhotoBioLib.Parts.HeatExchanger

Heat exchanger



Parameters

Type	Name	Default	Description
GeometryExchanger	geometry		Geometrical parameters
HeatParExchanger	heatPar		Heat parameters
InitialExchanger	initialExchanger		Initial values

Connectors

Type	Name	Description
LiqCon	lConIn	Input liquid connector
LiqCon	lConOut	Output liquid connector
HeatCon	hCon	Heat connector

Modelica definition

```
model HeatExchanger "Heat exchanger"
  extends PhotoBioLib.Interfaces.Partials.Liquid.Liq2Flow1Heat1;
  extends PhotoBioLib.Icons.Parts.HeatExchangerIcon;

  parameter PhotoBioLib.Records.Geometry.GeometryExchanger geometry
    "Geometrical parameters";
  parameter PhotoBioLib.Records.Heat.HeatParExchanger heatPar "Heat parameters";
  parameter PhotoBioLib.Records.Initial.InitialExchanger initialExchanger
    "Initial values";

  PhotoBioLib.Basic.Liquid.LiqCapacitor liqCapacitor(initialLiq=
        initialExchanger);
  PhotoBioLib.Basic.Liquid.LiqResistor liqResistor(heatPar=heatPar);
  PhotoBioLib.Basic.Heat.HeatConvection heatConv(heatPar=heatPar, geometry=
        geometry);
  Modelica.Blocks.Sources.RealExpression Qinter(
    y=lConIn.flowVol);

equation
  connect(Qinter.y, liqResistor.qLiq);
  connect(lConIn, liqCapacitor.lCon);
  connect(liqResistor.con1, liqCapacitor.lCon);
  connect(liqResistor.con2, lConOut);
  connect(liqCapacitor.hCon, heatConv.hCon1);
  connect(heatConv.hCon2, hCon);
end HeatExchanger;
```

PhotoBioLib.Parts.Liquid

Liquid parts package

Package Content

Name	Description
LiqPump	Liquid pump
LiqSource	Liquid source
LiqTank	Liquid tank

PhotoBioLib.Parts.Liquid.LiqPump

Liquid pump

Parameters

Type	Name	Default	Description
replaceable class con		Liq2Flow1	
replaceable class icon		LiqResistorIcon	
HeatPar	heatPar		Heat parameters
Real	Cv	heatPar.Cv	Volumetric heat capacity [J m-3 °K-1)]

Connectors

Type	Name	Description
LiqCon	con1	Liquid connector on the left
LiqCon	con2	Liquid connector on the rigth
input RealInput	qLiq	Input liquid flow [m3 s-1]
replaceable class icon		

Modelica definition

```
model LiqPump "Liquid pump"
   extends PhotoBioLib.Basic.Liquid.LiqResistor(redeclare class icon =
         PhotoBioLib.Icons.Parts.LiqPumpIcon);
end LiqPump;
```

PhotoBioLib.Parts.Liquid.LiqSource

Liquid source

Parameters

Type	Name	Default	Description
InitialLiq	initialLiq		Initial liquid values
Real	Volini	initialLiq.Volini	Initial liquid volume [m3]
Real	Tini	initialLiq.Tini	Initial liquid temperature [°K]

Connectors

Type	Name	Description
LiqCon	lCon	Liquid connector
input RealInput	T	Input temperature [°K]

Modelica definition

```
model LiqSource "Liquid source"
  extends PhotoBioLib.Interfaces.Partials.Liquid.LiqlTl;
  extends PhotoBioLib.Icons.Parts.LiqSourceIcon;

  parameter PhotoBioLib.Records.Initial.InitialLiq initialLiq
    "Initial liquid values";

  parameter Real Volini(unit="m3") = initialLiq.Volini "Initial liquid volume";

  parameter Real Tini(unit="ºK") = initialLiq.Tini "Initial liquid temperature";

initial equation
  lCon.Vol = Volini;

equation
  // Calor
  lCon.T = T;

  der(lCon.Vol) = lCon.flowVol;

end LiqSource;
```

PhotoBioLib.Parts.Liquid.LiqTank

Liquid tank

Parameters

Type	Name	Default	Description
replaceable class con		Liq1Heat1	
replaceable class icon		LiqCapacitorIcon	Icon
replaceable class initialValues		InitialLiq	
InitialLiq	initialLiq	extends PhotoBioLib.Records....	Initial liquid values
HeatPar	heatPar		Heat parameters
Real	Cv	heatPar.Cv	Volumetric heat capacity [J m-3 ºK-1)]
Real	Volini	initialLiq.Volini	Initial liquid volume [m3]
Real	Tini	initialLiq.Tini	Initital temperature [ºK]

Connectors

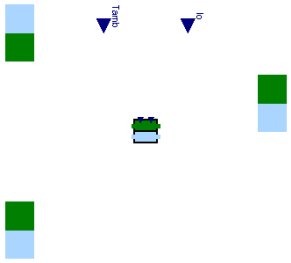
Type	Name	Description
replaceable class con		
LiqCon	lCon	Liquid connector
HeatCon	hCon	Heat connector
replaceable class icon		Icon

Modelica definition

```
model LiqTank "Liquid tank"
  extends PhotoBioLib.Basic.Liquid.LiqCapacitor(redeclare class icon =
        PhotoBioLib.Icons.Parts.LiqTankICon, redeclare class con =
        PhotoBioLib.Interfaces.Partials.Liquid.LiqlHeat1_Inv);
end LiqTank;
```

PhotoBioLib.Parts.Loop

Loop with nE sections



Parameters

Type	Name	Default	Description
InitialLoop	initialLoop		Initial values
Algae	algae		Algae parameters
GeometryLoop	geometry		Geometrical parameters
HeatParLoop	heatPar		Heat parameters
MassTransferLoop	massTransfer		Mass transfer parameters

Connectors

Type	Name	Description
BioCon	conLOut	Liquid output
GasCon	conGOut	Gas output
GasCon	conGIn	Gas input
BioCon	conLIn	Liquid input
input RealInput	Io	Input radiation [uE m-2 s-1]
GasCon	conGprobe	Gas probe
BioCon	conLprobe	Liquid probe
input RealInput	Tamb	Input temperature [ºK]

Modelica definition

```
model Loop "Loop with nE sections"
  extends PhotoBioLib.Icons.Parts.LoopIcon;
  extends PhotoBioLib.Interfaces.Partials.Others.Loop;

  parameter PhotoBioLib.Records.Initial.InitialLoop initialLoop
    "Initial values";
  constant PhotoBioLib.Records.Initial.InitialGas initialGas=initialLoop.initialGas
    "Initial gas values";
  constant PhotoBioLib.Records.Initial.InitialBio initialLiq=initialLoop.initialLiq
    "Initial liquid values";
  parameter PhotoBioLib.Records.Algae.Algae algae "Algae parameters";

  parameter PhotoBioLib.Records.Geometry.GeometryLoop geometry
    "Geometrical parameters";
  parameter PhotoBioLib.Records.Heat.HeatParLoop heatPar "Heat parameters";
  parameter PhotoBioLib.Records.MassTransfer.MassTransferLoop massTransfer
    "Mass transfer parameters";

  constant Integer nE = geometry.nE "Number of sections";
  constant Integer pointProbe= nE "Probe point";

  PhotoBioLib.Basic.Biomass.LoopSection s[nE](
    geometry=fill(geometry, nE),
    heatPar=fill(heatPar, nE),
    massTransfer=fill(massTransfer, nE),
    initialGas=fill(initialGas, nE),
    initialLiq=fill(initialLiq, nE),
    algae=fill(algae, nE));

equation
  // Conexión de secciones
  connect(s[1].conL1, conLIn);
  connect(s[1].conG1, conGIn);
  for i in 1:nE - 1 loop
    connect(s[i].conL2, s[i + 1].conL1);
    connect(s[i].conG2, s[i + 1].conG1);
  end for;
  connect(s[nE].conL2, conLOut);
  connect(s[nE].conG2, conGOut);

  // Conexión de radiación y velocidad
  for i in 1:nE loop
    connect(s[i].Io, Io);
    connect(s[i].T, Tamb);
  end for;

  // Conexión de la sonda
  connect(s[pointProbe].conL1, conLprobe);
  connect(s[pointProbe].conG1, conGprobe);
end Loop;
```

PhotoBioLib.Parts.Radiation

Radiation package

Package Content

Name	Description
Radiation	Radiation
toDegrees	toDegrees fucntion
toRadians	toRadians function

PhotoBioLib.Parts.Radiation.Radiation

Radiation

Parameters

Type	Name	Default	Description
Integer	startDay	1	
Integer	startSecond	1	
Real	latitude	36.767	
Real	longitudeLocal	-2.4	
Real	longitudeUso	0	

Connectors

Type	Name	Description
output RealOutput	Io	Radiation [uE m-2 s-1]

Modelica definition

```
model Radiation "Radiation"
  import Modelica.Constants.pi;
  import Modelica.Math.acos;

  extends PhotoBioLib.Interfaces.Partials.Others.Radl;
  extends PhotoBioLib.Icons.Parts.RadiationIcon;

  parameter Integer startDay = 1;
  parameter Integer startSecond = 1;
  parameter Real latitude = 36.767;
  parameter Real longitudeLocal = -2.4;
  parameter Real longitudeUso = 0;
  Real G0;

  Real latitudeRad;
  constant Real Bo = 1367;
  // Cambiar
  constant Real Ktd = 0.53;
  // Cambiar
  Real Fd;
  Real second;
  Real minute;
  Real day;
  Real hour;
  Real omega;
  Real gamma;
  Real delta;
  Real omega_s;
  Real cosin_fi_ZS;
  Real fi_ZS;
  Real Eo;
  Real Bo0;
  Real Bod0;
  Real Gd0;
  Real Dd0;
  Real rD;
  Real D0;
  Real aux;
  Real a;
  Real b;
  Real rG;
  Real B0;
  //Real minute2;

algorithm
  Io:=G0*2;

equation

  latitudeRad = PhotoBioLib.Parts.Radiation.toRadians(latitude);
  second = time + startSecond;
  minute = second / 60;
  day = startDay + div(minute, 24 * 60);
  hour = (minute - day * 24 * 60) / 60;
  omega = 15 * (hour - 12) - longitudeLocal - longitudeUso;
  gamma = (2 * pi * (day - 1)) / 365;
  delta = 0.006918 - 0.399912 * cos(gamma) + 0.070257 * sin(gamma) - 0.006758 * cos(2 * gamma) + 0.000907 * sin(2 * gamma) - 0.002697
  omega_s = -acos(-tan(delta) * tan(latitudeRad));
  cosin_fi_ZS = cos(delta)*cos(latitudeRad)*cos(
    PhotoBioLib.Parts.Radiation.toRadians(omega)) + sin(delta)*sin(latitudeRad);
  fi_ZS = acos(cosin_fi_ZS);
  Eo = 1.00011 + 0.034221 * cos(gamma) + 0.00128 * sin(gamma) + 0.000719 * cos(2 * gamma) + 0.000077 * sin(2 * gamma);
  Bo0 = Bo * Eo * cosin_fi_ZS;
  Bod0 = 24.0/pi*Bo*Eo*(-(1*pi)/180.0*PhotoBioLib.Parts.Radiation.toDegrees(
    omega_s)*sin(latitudeRad)*sin(delta) - cos(latitudeRad)*cos(delta)*sin(
    omega_s)));
  Gd0 = Ktd * Bod0;
  Dd0 = Fd * Gd0;
  rD = max(0, Bo0 / Bod0);
  D0 = max(0, rD * Dd0);
  aux = sin(omega_s + PhotoBioLib.Parts.Radiation.toRadians(60));
  a = 0.409 - 0.5016 * aux;
  b = 0.6609 - 0.4767 * aux;
  rG = rD*(a + b*cos(PhotoBioLib.Parts.Radiation.toRadians(omega)));
  G0 = max(0, rG * Gd0);
  B0 = G0 - D0;
  Fd = if Ktd <= 0.14 then 0.952 else if Ktd > 0.8 then 0.141 else 0.868 + 1.335 * Ktd - 5.782 * Ktd * Ktd + 3.721 * Ktd * Ktd * Ktd;
end Radiation;
```

PhotoBioLib.Parts.Radiation.toDegrees

toDegrees fucntion

Inputs

Type	Name	Default	Description
Real	rad		

Outputs

Type	Name	Description
Real	deg	

Modelica definition

```
function toDegrees "toDegrees fucntion"
  import Modelica.Constants.pi;
  input Real rad;
  output Real deg;
algorithm
  deg:=(rad * 360) / (2 * pi);
end toDegrees;
```

PhotoBioLib.Parts.Radiation.toRadians

toRadians function

Inputs

Type	Name	Default	Description
Real	deg		

Outputs

Type	Name	Description
Real	rad	

Modelica definition

```
function toRadians "toRadians function"
  import Modelica.Constants.pi;
  output Real rad;
  input Real deg;
algorithm
  rad:=(2 * pi * deg) / 360;
end toRadians;
```

PhotoBioLib.Records

Records package

Package Content

Name	Description
Algae	Algae parameters package
Constants	Constants
Geometry	Geometrical paramaeters package
Heat	Heat parameters package
Initial	Initial values package
LocationAndDate	Location and date values
MassTransfer	Mass transfer parameters package
Operation	Operation parameters
PhysicalPar_ClosedPhotobioreactor	Physical parameters of a closed photobioreactor

---

PhotoBioLib.Records.Algae

Algae parameters package

Package Content

Name	Description
Algae	Algae parameters

PhotoBioLib.Records.Algae.Algae

Algae parameters

Parameters

Type	Name	Default	Description
Real	A1	4.99E7	Preexponential factor [1]
Real	A2	1.66E13	Preexponential factor [1]
Real	B1	2.4098	Preexponential factor [1]
Real	B2	533.009	Preexponential factor [1]
Real	C1	6.2684	Activation energie [1]
Real	C2	68.8062	Activation energie [1]
Real	Ea1	4.27E4	Activation energie [mol J-1]
Real	Ea2	7.71E4	Activation energie [mol J-1]
Real	Ka	133.0324	Extinction coefficient [m2 Kg-1]
Real	Ki	173.9504	Form parameter [uE m-2 s-1]
Real	KO2	0.7202	Oxygen inhibition constant [mol m-3]
Real	m	0.0015	Form parameter [1]
Real	n	0.9779	Form exponent [1]
Real	nO2	100	Form parameter [1]
Real	PO2max	4.37E-5	Maximum photosynthesis rate [KgO2 KgB-1 s-1)=1/s]
Real	r	0.01	Respiration factor [1]
Real	R	8.314472	Gas constant [J K-1 mol-1]
Real	Ybo	0.9713	Biomass yield coefficient [KgB KgO2-1]
Real	z	5.4333	Form parameter [1]

Modelica definition

```
record Algae "Algae parameters"
  extends PhotoBioLib.Icons.Records.AlgaeIcon;

  parameter Real A1(unit="1") = 4.99E7 "Preexponential factor";
  parameter Real A2(unit="1") = 1.66E13 "Preexponential factor";
  parameter Real B1(unit="1") = 2.4098 "Preexponential factor";
  parameter Real B2(unit="1") = 533.009 "Preexponential factor";
  parameter Real C1(unit="1") = 6.2684 "Activation energie";
  parameter Real C2(unit="1") = 68.8062 "Activation energie";
  parameter Real Ea1(unit="mol J-1") = 4.27E4 "Activation energie";
  parameter Real Ea2(unit="mol J-1") = 7.71E4 "Activation energie";
  parameter Real Ka(unit="m2 Kg-1") = 133.0324 "Extinction coefficient";
  parameter Real Ki(unit="uE m-2 s-1") = 173.9504 "Form parameter";
  parameter Real KO2(unit="mol m-3") = 0.7202 "Oxygen inhibition constant";
  parameter Real m(unit="1") = 0.0015 "Form parameter";
  parameter Real n(unit="1") = 0.9779 "Form exponent";
  parameter Real nO2(unit="1") = 100 "Form parameter";
  parameter Real PO2max(unit="KgO2 KgB-1 s-1)=1/s") = 4.37E-5
    "Maximum photosynthesis rate";
  parameter Real r(unit="1") = 0.01 "Respiration factor";
  parameter Real R(unit="J K-1 mol-1") = 8.314472 "Gas constant";
  parameter Real Ybo(unit="KgB KgO2-1") = 0.9713 "Biomass yield coefficient";
  parameter Real z(unit="1") = 5.4333 "Form parameter";

end Algae;
```

PhotoBioLib.Records.Constants

Constants

Modelica definition

```
record Constants "Constants"
  // Constantes de Henry
  constant Real HO2(unit="mol atm-1 m-3") = 1.07 "Henry constant for oxygen";
  constant Real HCO2(unit="mol atm-1 m-3") = 38.36
    "Henry constant for carbon dioxide";
  constant Real Pt(unit="atm") = 1 "Atmospheric pressure";

  // Masa molar y volumen
  constant Real Vmol(unit="m3 mol-1") = 20E-3 "Molar volumen";
  constant Real MO2(unit="Kg mol-1") = 32E-3 "Molecular weight of oxygen";
  constant Real MCO2(unit="Kg mol-1") = 44E-3
    "Molecular weight of carbon dyoxide";

  // Equilibrium constants H
  constant Real Kw(unit="1") = 10^(-14) "Equilibrium constant";
  constant Real K1(unit="1") = 10^(-6.381) "Equilibrium constant";
  constant Real K2(unit="1") = 10^(-10.377) "Equilibrium constant";

  // En realidad parámetros de operacion??

  // Molar fraction in the air
  constant Real yO2(unit="1") = 0.2097 "Molar fraction of oxygen in the air";
  constant Real yCO2(unit="1") = 0.0003
    "Molar fraction of carbon dioxide in the air";
  constant Real yN2(unit="%") = 0.79 "Molar fraction of nitrogen in the air";

  // Molar fraction in the CO2
  constant Real yO2CO2(unit="1") = 0
    "Molar fraction of oxygen in the CO2 injection";
  constant Real yCO2CO2(unit="1") = 0.95
    "Molar fraction of carbon dioxide in the CO2 injection";
  constant Real yN2CO2(unit="1") = 0.05
    "Molar fraction of nitrogen in the CO2 injection";

end Constants;
```

PhotoBioLib.Records.Geometry

Geometrical paramaeters package

Package Content

Name	Description
Geometry	Geometrical parameters
GeometryColumn	Geometrical parameters of the column
GeometryExchanger	Geometrical parameters of the exchanger
GeometryLoop	Geometrical parameters of the loop

PhotoBioLib.Records.Geometry.Geometry

Geometrical parameters

Parameters

Type	Name	Default	Description
Real	alpha	0.9725	Distribution solar factor [1]
Real	dt	0.084	Total diameter [m]
Real	L	400	Length [m]
Integer	nE	20	Number of elements
Real	At	pi*(dt/2)^2	Total cross-sectional area [m2]
Real	dX	L/nE	Lenght of section [m]
Real	S	dt*pi*dX	Surface of section [m2]
Real	Vol	At*dX	Volume of section [m3]

Modelica definition

```
record Geometry "Geometrical parameters"
  extends Modelica.Icons.Record;
  import Modelica.Constants.pi;

  parameter Real alpha(unit="1") = 0.9725 "Distribution solar factor";
  parameter Real dt(unit="m") = 0.084 "Total diameter";
  parameter Real L(unit="m") = 400 "Length";
  parameter Integer nE = 20 "Number of elements";
  parameter Real At(unit="m2") = pi*(dt/2)^2 "Total cross-sectional area";
  parameter Real dX(unit="m") = L/nE "Lenght of section";
  parameter Real S(unit="m2") = dt*pi*dX "Surface of section";
  parameter Real Vol(unit="m3") = At*dX "Volume of section";
end Geometry;
```

PhotoBioLib.Records.Geometry.GeometryColumn

Geometrical parameters of the column

Parameters

Type	Name	Default	Description
Real	alpha	0.1052	Distribution solar factor [1]
Real	dt	0.4	Total diameter [m]
Real	L	3.2	Length [m]
Integer	nE	1	Number of elements
Real	At	pi*(dt/2)^2	Total cross-sectional area [m2]
Real	dX	L/nE	Lenght of section [m]

Real	S	dt*pi*dX	Surface of section [m2]
Real	Vol	At*dX	Volume of section [m3]

Modelica definition

```
record GeometryColumn "Geometrical parameters of the column"
  extends PhotoBioLib.Records.Geometry.Geometry(
    alpha=0.1052,
    dt=0.4,
    L=3.2,
    nE=1);
end GeometryColumn;
```

PhotoBioLib.Records.Geometry.GeometryExchanger

Geometrical parameters of the exchanger

Parameters

Type	Name	Default	Description
Real	alpha	0.9725	Distribution solar factor [1]
Real	dt	0.084	Total diameter [m]
Real	L	400	Length [m]
Integer	nE	20	Number of elements
Real	At	pi*(dt/2)^2	Total cross-sectional area [m2]
Real	dX	L/nE	Lenght of section [m]
Real	S	3.1919	Surface of section [m2]
Real	Vol	20.3E-3	Volume of section [m3]

Modelica definition

```
record GeometryExchanger "Geometrical parameters of the exchanger"
  extends Geometry(
    S=3.1919,
    Vol=20.3E-3);
    //nE=1,
end GeometryExchanger;
```

PhotoBioLib.Records.Geometry.GeometryLoop

Geometrical parameters of the loop

Parameters

Type	Name	Default	Description
Real	alpha	0.9725	Distribution solar factor [1]
Real	dt	0.084	Total diameter [m]
Real	L	400	Length [m]
Integer	nE	20	Number of elements
Real	At	pi*(dt/2)^2	Total cross-sectional area [m2]
Real	dX	L/nE	Lenght of section [m]
Real	S	dt*pi*dX	Surface of section [m2]
Real	Vol	At*dX	Volume of section [m3]

Modelica definition

```
record GeometryLoop "Geometrical parameters of the loop"
  extends PhotoBioLib.Records.Geometry.Geometry(
    alpha=0.9725,
    dt=0.084,
    L=400,
    nE=20);
end GeometryLoop;
```

PhotoBioLib.Records.Heat

Heat parameters package

Package Content

Name	Description
HeatPar	Heat parameters
HeatParColumn	Heat parameters of the column
HeatParExchanger	Heat parameters of the exchanger
HeatParLoop	Heat parameters of the loop

PhotoBioLib.Records.Heat.HeatPar

Heat parameters

Parameters

Type	Name	Default	Description
Real	aR	0.5411	Solar irradiance absortivity [1]
Real	Cv	4184e3	Volumetric heat capacity [J m-3 ºK-1]
Real	h	6.6189	Transmission coefficient [J s-1 m-2 ºK-1]

Modelica definition

```
record HeatPar "Heat parameters"
  extends Modelica.Icons.Record;
  parameter Real aR(unit="1") = 0.5411 "Solar irradiance absortivity";
  parameter Real Cv(unit="J m-3 ºK-1")=4184e3 "Volumetric heat capacity";
  parameter Real h(unit="J s-1 m-2 ºK-1") = 6.6189 "Transmission coefficient"; //poner a 0 o quitar??
end HeatPar;
```

PhotoBioLib.Records.Heat.HeatParColumn

Heat parameters of the column

Parameters

Type	Name	Default	Description
Real	aR	0.5411	Solar irradiance absortivity [1]
Real	Cv	4184e3	Volumetric heat capacity [J m-3 ºK-1]
Real	h	11.1886	Transmission coefficient [J s-1 m-2 ºK-1]

Modelica definition

```
record HeatParColumn "Heat parameters of the column"
  extends HeatPar(
    h=11.1886);
end HeatParColumn;
```

PhotoBioLib.Records.Heat.HeatParExchanger

Heat parameters of the exchanger

Parameters

Type	Name	Default	Description
Real	aR	0.5411	Solar irradiance absortivity [1]
Real	Cv	4184e3	Volumetric heat capacity [J m-3 ºK-1]
Real	h	449.9170	Transmission coefficient [J s-1 m-2 ºK-1]

Modelica definition

```
record HeatParExchanger "Heat parameters of the exchanger"
  extends HeatPar(
    h=449.9170);
end HeatParExchanger;
```

PhotoBioLib.Records.Heat.HeatParLoop

Heat parameters of the loop

Parameters

Type	Name	Default	Description
Real	aR	0.5411	Solar irradiance absortivity [1]
Real	Cv	4184e3	Volumetric heat capacity [J m-3 ºK-1]
Real	h	6.6189	Transmission coefficient [J s-1 m-2 ºK-1]

Modelica definition

```
record HeatParLoop "Heat parameters of the loop"
  extends HeatPar(
    h = 6.6189);
    //aR = 0.5411,
```

```
    //Cv = 4184,
end HeatParLoop;
```

PhotoBioLib.Records.Initial

Initial values package

Package Content

Name	Description
InitialAirInyection	Initial gas values of the air inyector
InitialBio	Initial liquid and biomass values
InitialBioTank	Initial values of a liquid tank with biomass
InitialCO2Inyection	Initial gas values of the CO2 inyector
InitialColumn	Initial values of the column
InitialExchanger	Initial values of the exchanger
InitialGas	Initial gas values
InitialGasTank	Initial values of a gas tank
InitialLiq	Initial liquid values
InitialLiqTank	Initial values of a liquid tank
InitialLoop	Initial values of the loop
InitialMedium	Initial values of the medium
Initials	Initial gas and liquid values

PhotoBioLib.Records.Initial.InitialAirInyection

Initial gas values of the air inyector

Parameters

Type	Name	Default	Description
Real	O2ini	10.5	Initial oxygen concentration in the gas phase [mol m-3]
Real	CO2ini	0.015	Initial carbon dioxide concentration in the gas phase [mol m-3]
Real	N2ini	39.485	Initial nitrogen concentration in the gas phase [mol m-3]
Real	Volini	0	Initial gas volume [m3]

Modelica definition

```
record InitialAirInyection "Initial gas values of the air inyector"
  extends PhotoBioLib.Records.Initial.InitialGas(
    O2ini=10.5,
    CO2ini=0.015,
    N2ini=39.485,
    Volini=0);
end InitialAirInyection;
```

PhotoBioLib.Records.Initial.InitialBio

Initial liquid and biomass values

Parameters

Type	Name	Default	Description
Real	O2ini	70*Constants.HO2*Constants.y...	Initial oxygen concentration in the liquid phase [mol m-3]
Real	CO2ini	Ctini/(1 + Constants.K1/Hini...	Initial carbon dioxide concentration in the liquid phase [mol m-3]
Real	Ctini	6	Initial totoal inorganic carbon concentration in the liquid phase [mol m-3]
Real	Hini	4.2530E-8	Initial hidrogen concentration in the liquid phase [mol m-3]
Real	Cbini	1	Initial biomass concentration [Kg m-3]
Real	Volini	0	Initial liquid volume [m3]
Real	Tini	14 + 274.15	Initial temperature of the liquid [ºK]

Modelica definition

```
record InitialBio "Initial liquid and biomass values"

  extends Modelica.Icons.Record;
  import PhotoBioLib.Records.Constants;
  parameter Real O2ini(unit="mol m-3") = 70*Constants.HO2*Constants.yO2/100
    "Initial oxygen concentration in the liquid phase";
  parameter Real CO2ini(unit="mol m-3")=Ctini/(1+Constants.K1/Hini +Constants.K1*Constants.K2/Hini^2)
    "Initial carbon dioxide concentration in the liquid phase";
  parameter Real Ctini(unit="mol m-3")=6
    "Initial totoal inorganic carbon concentration in the liquid phase";
  parameter Real Hini(unit="mol m-3")=4.2530E-8
    "Initial hidrogen concentration in the liquid phase";
  parameter Real Cbini(unit="Kg m-3")=1 "Initial biomass concentration";
  parameter Real Volini(unit="m3")=0 "Initial liquid volume";
  parameter Real Tini(unit="ºK")=14+274.15 "Initial temperature of the liquid";
end InitialBio;
```

PhotoBioLib.Records.Initial.InitialBioTank

Initial values of a liquid tank with biomass

Parameters

Type	Name	Default	Description
Real	O2ini	70*Constants.HO2*Constants.y...	Initial oxygen concentration in the liquid phase [mol m-3]
Real	CO2ini	Ctini/(1 + Constants.K1/Hini...	Initial carbon dioxide concentration in the liquid phase [mol m-3]
Real	Ctini	6	Initial totoal inorganic carbon concentration in the liquid phase [mol m-3]
Real	Hini	4.2530E-8	Initial hidrogen concentration in the liquid phase [mol m-3]
Real	Cbini	1	Initial biomass concentration [Kg m-3]
Real	Volini	0	Initial liquid volume [m3]
Real	Tini	14 + 274.15	Initial temperature of the liquid [ºK]

Modelica definition

```
record InitialBioTank "Initial values of a liquid tank with biomass"
  extends PhotoBioLib.Records.Initial.InitialBio;
end InitialBioTank;
```

PhotoBioLib.Records.Initial.InitialCO2Inyection

Initial gas values of the CO2 inyector

Parameters

Type	Name	Default	Description
Real	O2ini	0	Initial oxygen concentration in the gas phase [mol m-3]
Real	CO2ini	47.5	Initial carbon dioxide concentration in the gas phase [mol m-3]
Real	N2ini	2.5	Initial nitrogen concentration in the gas phase [mol m-3]
Real	Volini	0	Initial gas volume [m3]

Modelica definition

```
record InitialCO2Inyection "Initial gas values of the CO2 inyector"
  extends PhotoBioLib.Records.Initial.InitialGas(
    O2ini=0,
    CO2ini=47.5,
    N2ini=2.5,
    Volini=0);
end InitialCO2Inyection;
```

PhotoBioLib.Records.Initial.InitialColumn

Initial values of the column

Modelica definition

```
record InitialColumn "Initial values of the column"
  extends Initials(
    initialGas(
      O2ini = 0,
      CO2ini = 0,
      N2ini = 0,
      Volini = 0),
    initialLiq(
      O2ini=68*Constants.HO2*Constants.yO2/100,
      Cbini=1,
      Ctini=6,
      Hini=4.2530E-8,
      Tini=14+274.15));

end InitialColumn;
```

PhotoBioLib.Records.Initial.InitialExchanger

Initial values of the exchanger

Parameters

Type	Name	Default	Description

Real	Volini	20.3E-3	Initial liquid volume [m3]
Real	Tini	14 + 274.15	Initial temperature of the liquid [ºK]

Modelica definition

```
record InitialExchanger "Initial values of the exchanger"
  extends PhotoBioLib.Records.Initial.InitialLiq(Tini=14 + 274.15, Volini=
        20.3E-3);
end InitialExchanger;
```

PhotoBioLib.Records.Initial.InitialGas

Initial gas values

Parameters

Type	Name	Default	Description
Real	O2ini	0	Initial oxygen concentration in the gas phase [mol m-3]
Real	CO2ini	0	Initial carbon dioxide concentration in the gas phase [mol m-3]
Real	N2ini	0	Initial nitrogen concentration in the gas phase [mol m-3]
Real	Volini	0	Initial gas volume [m3]

Modelica definition

```
record InitialGas "Initial gas values"
  extends Modelica.Icons.Record;
  parameter Real O2ini(unit="mol m-3") = 0
    "Initial oxygen concentration in the gas phase";
  parameter Real CO2ini(unit="mol m-3") = 0
    "Initial carbon dioxide concentration in the gas phase";
  parameter Real N2ini(unit="mol m-3") = 0
    "Initial nitrogen concentration in the gas phase";
  parameter Real Volini(unit="m3")=0 "Initial gas volume";
end InitialGas;
```

PhotoBioLib.Records.Initial.InitialGasTank

Initial values of a gas tank

Parameters

Type	Name	Default	Description
Real	O2ini	0	Initial oxygen concentration in the gas phase [mol m-3]
Real	CO2ini	0	Initial carbon dioxide concentration in the gas phase [mol m-3]
Real	N2ini	0	Initial nitrogen concentration in the gas phase [mol m-3]
Real	Volini	0	Initial gas volume [m3]

Modelica definition

```
record InitialGasTank "Initial values of a gas tank"
  extends InitialGas(
    O2ini=0,
    CO2ini=0,
    N2ini=0,
    Volini=0);
end InitialGasTank;
```

PhotoBioLib.Records.Initial.InitialLiq

Initial liquid values

Parameters

Type	Name	Default	Description
Real	Volini	0	Initial liquid volume [m3]
Real	Tini	14 + 274.15	Initial temperature of the liquid [ºK]

Modelica definition

```
record InitialLiq "Initial liquid values"

  extends Modelica.Icons.Record;
  import PhotoBioLib.Records.Constants;

  parameter Real Volini(unit="m3")=0 "Initial liquid volume";
  parameter Real Tini(unit="9K")=14+274.15 "Initial temperature of the liquid";
end InitialLiq;
```

PhotoBioLib.Records.Initial.InitialLiqTank

Initial values of a liquid tank

Parameters

Type	Name	Default	Description
Real	Volini	0	Initial liquid volume [m3]
Real	Tini	14 + 274.15	Initial temperature of the liquid [ºK]

Modelica definition

```
record InitialLiqTank "Initial values of a liquid tank"
  extends PhotoBioLib.Records.Initial.InitialLiq;
end InitialLiqTank;
```

PhotoBioLib.Records.Initial.InitialLoop

Initial values of the loop

Modelica definition

```
record InitialLoop "Initial values of the loop"
  extends Initials(
    initialGas(
      O2ini = 0,
      CO2ini = 0,
      N2ini = 0,
      Volini = 0),
    initialLiq(
      O2ini=70*Constants.HO2*Constants.yO2/100,
      Cbini=1,
      Ctini=6,
      Hini=4.2530E-8,
      Tini=14+274.15));

end InitialLoop;
```

PhotoBioLib.Records.Initial.InitialMedium

Initial values of the medium

Parameters

Type	Name	Default	Description
Real	O2ini	0.2812	Initial oxygen concentration in the liquid phase [mol m-3]
Real	CO2ini	Ctini/(1 + Constants.K1/Hini...	Initial carbon dioxide concentration in the liquid phase [mol m-3]
Real	Ctini	8	Initial totoal inorganic carbon concentration in the liquid phase [mol m-3]
Real	Hini	4.2530E-8	Initial hidrogen concentration in the liquid phase [mol m-3]
Real	Cbini	0	Initial biomass concentration [Kg m-3]
Real	Volini	0	Initial liquid volume [m3]
Real	Tini	14 + 274.15	Initial temperature of the liquid [ºK]

Modelica definition

```
record InitialMedium "Initial values of the medium"
  extends PhotoBioLib.Records.Initial.InitialBio(
    O2ini=0.2812,
    Cbini=0,
    Ctini=8,
    Tini=14 + 274.15);
end InitialMedium;
```

PhotoBioLib.Records.Initial.Initials

Initial gas and liquid values

Parameters

Type	Name	Default	Description
InitialGas	initialGas		
InitialBio	initialLiq		

Modelica definition

```
record Initials "Initial gas and liquid values"
  extends Modelica.Icons.Record;
  parameter PhotoBioLib.Records.Initial.InitialGas initialGas;
  parameter PhotoBioLib.Records.Initial.InitialBio initialLiq;
```

```
end Initials;
```

PhotoBioLib.Records.LocationAndDate

Location and date values

Modelica definition

```
record LocationAndDate "Location and date values"
  extends Modelica.Icons.Record;
end LocationAndDate;
```

PhotoBioLib.Records.MassTransfer

Mass transfer parameters package

Package Content

Name	Description
MassTransfer	Mass transfer parameters
MassTransferColumn	Mass transfer parameters of the column
MassTransferLoop	Mass transfer parameters of the loop

PhotoBioLib.Records.MassTransfer.MassTransfer

Mass transfer parameters

Parameters

Type	Name	Default	Description
Real	a	0.012	Form parameter [1]
Real	b	0.8450	Form parameter [1]
Real	KCO2	0.91	TRansfer coefficient for CO2 [1]
Real	Vinf	0.651	Bubble accession rate [m s-1]
Real	C0	0.996	Drift flux model parameter [1]

Modelica definition

```
record MassTransfer "Mass transfer parameters"
  extends Modelica.Icons.Record;
  parameter Real a(unit="1")= 0.012 "Form parameter";
  parameter Real b(unit="1")= 0.8450 "Form parameter";
  parameter Real KCO2(unit="1") = 0.91 "TRansfer coefficient for CO2";
  parameter Real Vinf(unit="m s-1") = 0.651 "Bubble accession rate";
  parameter Real C0(unit="1") = 0.996 "Drift flux model parameter";
end MassTransfer;
```

PhotoBioLib.Records.MassTransfer.MassTransferColumn

Mass transfer parameters of the column

Parameters

Type	Name	Default	Description
Real	a	0.0606	Form parameter [1]
Real	b	0.7533	Form parameter [1]
Real	KCO2	0.9017	TRansfer coefficient for CO2 [1]
Real	Vinf	0.651	Bubble accession rate [m s-1]
Real	C0	0.996	Drift flux model parameter [1]

Modelica definition

```
record MassTransferColumn "Mass transfer parameters of the column"
  extends PhotoBioLib.Records.MassTransfer.MassTransfer(
    a=0.0606,
    b=0.7533,
    KCO2=0.9017,
    Vinf=0.651,
    C0=0.996);
end MassTransferColumn;
```

PhotoBioLib.Records.MassTransfer.MassTransferLoop

Mass transfer parameters of the loop

Parameters

Type	Name	Default	Description
Real	a	0.012	Form parameter [1]
Real	b	0.8450	Form parameter [1]
Real	KCO2	0.91	TRansfer coefficient for CO2 [1]
Real	Vinf	0.651	Bubble accession rate [m s-1]
Real	C0	0.996	Drift flux model parameter [1]

Modelica definition

```
record MassTransferLoop "Mass transfer parameters of the loop"
  extends PhotoBioLib.Records.MassTransfer.MassTransfer(
    a=0.012,
    b=0.8450,
    KCO2=0.91);
end MassTransferLoop;
```

PhotoBioLib.Records.Operation

Operation parameters

Parameters

Type	Name	Default	Description
InitialLoop	initialLoop		Initial values of the loop
InitialColumn	initialColumn		Initial values of the column
InitialLiqTank	initialLiqTank		Initial values of the liquid tank
InitialBioTank	initialBioTank		Initial values of the biomass tank
InitialMedium	initialMedium		Initial values of the medium
InitialExchanger	initialExchanger		Initial values of the exchanger
InitialGasTank	initialGasTank		Initial values of the gas tank
InitialAirInyection	initialAirInyection		Initial values od the gas inyector
InitialCO2Inyection	initialCO2Inyection		Initial values of the CO2 inyector
Real	qMedio	5*(1e-4)/6	Medium flow [m3 s-1]
Real	qInter	30*(1e-4)/6	Exchanger liquid flow [m3 s-1]
Real	qMove	332*(1e-4)/6	Liquid circulation flow [m3 s-1]
Real	qBuble	140*(1e-4)/6	Air inyector flow [m3 s-1]
Real	qCO2	0	CO2 inyector flow [m3 s-1]

Modelica definition

```
record Operation "Operation parameters"
  extends Modelica.Icons.Record;
  // Initial values
  parameter PhotoBioLib.Records.Initial.InitialLoop initialLoop
    "Initial values of the loop";
  parameter PhotoBioLib.Records.Initial.InitialColumn initialColumn
    "Initial values of the column";

  // Liquid
  parameter PhotoBioLib.Records.Initial.InitialLiqTank initialLiqTank
    "Initial values of the liquid tank";
  parameter PhotoBioLib.Records.Initial.InitialBioTank initialBioTank
    "Initial values of the biomass tank";

  parameter PhotoBioLib.Records.Initial.InitialMedium initialMedium
    "Initial values of the medium";
  parameter PhotoBioLib.Records.Initial.InitialExchanger initialExchanger
    "Initial values of the exchanger";

  // Gas
  parameter PhotoBioLib.Records.Initial.InitialGasTank initialGasTank
    "Initial values of the gas tank";
  parameter PhotoBioLib.Records.Initial.InitialAirInyection initialAirInyection
    "Initial values od the gas inyector";
  parameter PhotoBioLib.Records.Initial.InitialCO2Inyection initialCO2Inyection
    "Initial values of the CO2 inyector";

  // Caudales y otras entradas
  parameter Real qMedio(unit="m3 s-1") = 5*(1e-4)/6 "Medium flow";
  parameter Real qInter(unit="m3 s-1") = 30*(1e-4)/6 "Exchanger liquid flow";
  parameter Real qMove(unit="m3 s-1") = 332*(1e-4)/6 "Liquid circulation flow";
  parameter Real qBuble(unit="m3 s-1") = 140*(1e-4)/6 "Air inyector flow";
  //parameter Real Tamb = 10+274.15;

  parameter Real qCO2(unit="m3 s-1") = 0 "CO2 inyector flow";

end Operation;
```

PhotoBioLib.Records.PhysicalPar_ClosedPhotobioreactor

Physical parameters of a closed photobioreactor

Parameters

Type	Name	Default	Description
GeometryLoop	geometryLoop		Geometrical parameters of the loop
GeometryColumn	geometryColumn		Geometrical parameters of the column
GeometryExchanger	geometryExchanger		Geometrical parameters of the exchanger
HeatParLoop	heatParLoop		Heat parameters of the loop
HeatParColumn	heatParColumn		Heat parameters of the column
HeatParExchanger	heatParExchanger		Heat parameters of the exchanger
MassTransferLoop	massTransferLoop		Mass transfer parameters of the loop
MassTransferColumn	massTransferColumn		Mass transfer parameters of the column

Modelica definition

```
record PhysicalPar_ClosedPhotobioreactor
  "Physical parameters of a closed photobioreactor"
  extends Modelica.Icons.Record;

  // Geometry
  parameter PhotoBioLib.Records.Geometry.GeometryLoop geometryLoop
    "Geometrical parameters of the loop";
  parameter PhotoBioLib.Records.Geometry.GeometryColumn geometryColumn
    "Geometrical parameters of the column";
  parameter PhotoBioLib.Records.Geometry.GeometryExchanger geometryExchanger
    "Geometrical parameters of the exchanger";

  // Heat
  parameter PhotoBioLib.Records.Heat.HeatParLoop heatParLoop
    "Heat parameters of the loop";
  parameter PhotoBioLib.Records.Heat.HeatParColumn heatParColumn
    "Heat parameters of the column";
  parameter PhotoBioLib.Records.Heat.HeatParExchanger heatParExchanger
    "Heat parameters of the exchanger";

  // MassTransfer
  parameter PhotoBioLib.Records.MassTransfer.MassTransferLoop massTransferLoop
    "Mass transfer parameters of the loop";
  parameter PhotoBioLib.Records.MassTransfer.MassTransferColumn
    massTransferColumn "Mass transfer parameters of the column";

end PhysicalPar_ClosedPhotobioreactor;
```

PhotoBioLib.Sensors

Sensors package

Package Content

Name	Description
BioFlowSensor	Biomass flow sensor
BioFlowSensor_ext	Biomass flow sensor
BioPotentialSensor	Biomass potential sensor
GasFlowSensor	Gas flow sensor
GasFlowSensor_ext	Gas flow sensor
GasPotentialSensor	Gas potencial sensor
LiqFlowSensor	Liquid flow sensor
LiqFlowSensor_ext	Liquid flow sensor
LiqPotentialSensor	Liquid potential sensor
RadSensor	Radiation sensor

PhotoBioLib.Sensors.BioFlowSensor

Biomass flow sensor

Parameters

Type	Name	Default	Description
replaceable class con		PhotoBioLib.Interfaces.Parti...	
Integer	nout	7	Number of outputs
replaceable class icon		LiqFlowSensorIcon	

Connectors

Type	Name	Description
replaceable class con		
output RealOutput	y[nout]	Outputs
BioCon	con1	Liquid connector on the left
BioCon	con2	Liquid connector on the rigth
replaceable class icon		

Modelica definition

```
model BioFlowSensor "Biomass flow sensor"
  extends PhotoBioLib.Sensors.LiqFlowSensor(redeclare class icon =
        PhotoBioLib.Icons.Sensors.BioFlowSensorIcon, redeclare class con =
        PhotoBioLib.Interfaces.Partials.Biomass.Bio2OutN (nout=7));

  Real flowO2(unit="mol s-1") "Oxygen molar flow";
  Real flowCt(unit="mol s-1") "Total carbon molar flow";
  Real flowCb(unit="Kg s-1") "Biomass flow";

equation
  y[3]=con1.flowCb;
  y[4]=con1.flowO2;
  y[5]=con1.flowCO2;
  y[6]=con1.flowCt;
  y[7]=con1.flowH;

  flowO2 = con1.flowO2;
  flowCt = con1.flowCt;
  flowCb = con1.flowCb;

end BioFlowSensor;
```

PhotoBioLib.Sensors.BioFlowSensor_ext

Biomass flow sensor

Parameters

Type	Name	Default	Description
replaceable class con		PhotoBioLib.Interfaces.Parti...	
Integer	nout	7	Number of outputs
replaceable class icon		LiqFlowSensorIcon	
Real	Cv	4184e3	

Connectors

Type	Name	Description
replaceable class con		
output RealOutput	y[nout]	Outputs
BioCon	con1	Liquid connector on the left
BioCon	con2	Liquid connector on the rigth
replaceable class icon		

Modelica definition

```
model BioFlowSensor_ext "Biomass flow sensor"
  extends PhotoBioLib.Sensors.LiqFlowSensor(redeclare class icon =
        PhotoBioLib.Icons.Sensors.BioFlowSensorIcon, redeclare class con =
        PhotoBioLib.Interfaces.Partials.Biomass.Bio2OutN (nout=7));

  Real flowO2(unit="mol s-1") "Oxygen molar flow";
  Real flowCt(unit="mol s-1") "Total carbon molar flow";
  Real flowCb(unit="Kg s-1") "Biomass flow";

  Real T;
  Real Tc;
  parameter Real Cv = 4184e3;
  //lCon.T = Qt/(lCon.Vol*Cv);

equation
  y[3]=con1.flowCb;
  y[4]=con1.flowO2;
  y[5]=con1.flowCO2;
  y[6]=con1.flowCt;
  y[7]=con1.flowH;

  flowO2 = con1.flowO2;
  flowCt = con1.flowCt;
```

```
  flowCb = con1.flowCb;

  T = con1.flowQ/(con1.flowVol*Cv);
  Tc = T-274.15;

end BioFlowSensor_ext;
```

PhotoBioLib.Sensors.BioPotentialSensor

Biomass potential sensor

Parameters

Type	Name	Default	Description
replaceable class con		PhotoBioLib.Interfaces.Parti...	
Integer	nout	8	Number of outputs
replaceable class icon		LiqPotentialSensorIcon	

Connectors

Type	Name	Description
replaceable class con		
output RealOutput	y[nout]	Outputs
BioCon	lCon	Liquid connector
replaceable class icon		

Modelica definition

```
model BioPotentialSensor "Biomass potential sensor"
  extends PhotoBioLib.Sensors.LiqPotentialSensor(redeclare class icon =
        PhotoBioLib.Icons.Sensors.BioPotentialSensorIcon, redeclare class con
      = PhotoBioLib.Interfaces.Partials.Biomass.BiolOutN (nout=8));

  import Modelica.Math.log10;
  Real O2(unit="mol m-3") "Oxygen concentration";
  Real CO2(unit="mol m-3") "Oxygen concentration";
  Real Ct(unit="mol m-3") "Total carbon concentration";
  Real H(unit="mol m-3") "Hidrogen concentration";
  Real Cb(unit="Kg m-3") "Biomass concentration";
  Real pH(unit="1") "pH";

equation
  y[4]=lCon.Cb;
  y[5]=lCon.O2;
  y[6]=lCon.CO2;
  y[7]=lCon.Ct;
  y[8]=lCon.H;

  lCon.flowO2=0;
  lCon.flowCO2=0;
  lCon.flowCt=0;
  lCon.flowH=0;
  lCon.flowCb=0;

  O2 = lCon.O2;
  CO2 = lCon.CO2;
  Cb = lCon.Cb;
  Ct = lCon.Ct;
  H = lCon.H;
  pH = -log10(lCon.H);

end BioPotentialSensor;
```

PhotoBioLib.Sensors.GasFlowSensor

Gas flow sensor

Parameters

Type	Name	Default	Description
Integer	nout	4	Number of outputs

Connectors

Type	Name	Description
output RealOutput	y[nout]	
GasCon	con1	Gas connector on the left
GasCon	con2	Gas connector on the rigth

Modelica definition

```
model GasFlowSensor "Gas flow sensor"

  Real flowVol(unit="m3 s-1") "Gas volume flow";
  Real flowO2(unit="mol s-1") "Oxygen molar flow";
  Real flowCO2(unit="mol s-1") "Carbon dioxide molar flow";
  Real flowN2(unit="mol s-1") "Nitrogen molar flow";

  extends PhotoBioLib.Interfaces.Partials.Gas.Gas2OutN(nout=4);
  extends PhotoBioLib.Icons.Sensors.GasFlowSensorIcon;

equation
  flowVol = con1.flowVol;
  flowO2 = con1.flowO2;
  flowCO2 = con1.flowCO2;
  flowN2 = con1.flowN2;

  y[1] = con1.flowVol;
  y[2] = con1.flowO2;
  y[3] = con1.flowCO2;
  y[4] = con1.flowN2;

  connect(con1, con2);
end GasFlowSensor;
```

PhotoBioLib.Sensors.GasFlowSensor_ext

Gas flow sensor

Parameters

Type	Name	Default	Description
Integer	nout	4	Number of outputs

Connectors

Type	Name	Description
output RealOutput	y[nout]	
GasCon	con1	Gas connector on the left
GasCon	con2	Gas connector on the rigth

Modelica definition

```
model GasFlowSensor_ext "Gas flow sensor"

  Real flowVol(unit="m3 s-1") "Gas volume flow";
  Real flowO2(unit="mol s-1") "Oxygen molar flow";
  Real flowCO2(unit="mol s-1") "Carbon dioxide molar flow";
  Real flowN2(unit="mol s-1") "Nitrogen molar flow";

  Real yO2;
  Real yCO2;
  Real yN2;

  extends PhotoBioLib.Interfaces.Partials.Gas.Gas2OutN(nout=4);
  extends PhotoBioLib.Icons.Sensors.GasFlowSensorIcon;

equation
  if flowVol>0 then
    yO2 = con1.flowO2/(con1.flowO2+con1.flowCO2+con1.flowN2);
    yCO2 = con1.flowCO2/(con1.flowO2+con1.flowCO2+con1.flowN2);
    yN2 = con1.flowN2/(con1.flowO2+con1.flowCO2+con1.flowN2);
  else
    yO2=0;
    yCO2=0;
    yN2=0;
  end if;
  flowVol = con1.flowVol;
  flowO2 = con1.flowO2;
  flowCO2 = con1.flowCO2;
  flowN2 = con1.flowN2;

  y[1] = con1.flowVol;
  y[2] = con1.flowO2;
  y[3] = con1.flowCO2;
  y[4] = con1.flowN2;

  connect(con1, con2);
end GasFlowSensor_ext;
```

PhotoBioLib.Sensors.GasPotentialSensor

Gas potencial sensor

Parameters

Type	Name	Default	Description
Real	epsV	1E-10	Filter value [1]
Integer	nout	14	Number of outputs

Connectors

Type	Name	Description
output RealOutput	y[nout]	Outputs
GasCon	con	Gas connector

Modelica definition

```
model GasPotentialSensor "Gas potencial sensor"
  Real Vol(unit="m3") "Gas volume";
  Real O2(unit="mol m-3") "Oxygen concentration";
  Real CO2(unit="mol m-3") "Carbon dioxide concentration";
  Real N2(unit="mol m-3") "Nitrogen concentration";
  Real yO2(unit="1") "Oxygen molar fraction";
  Real yCO2(unit="1") "Carbon dioxide molar fraction";
  Real yN2(unit="1") "Nitrogen molar fraction";
  Real yT(unit="1") "Total molar fraction";

  Real Volf(unit="m3") "Gas volume (filtered)";
  Real O2f(unit="mol m-3") "Oxygen concentration (filtered)";
  Real CO2f(unit="mol m-3") "Carbon dioxide concentration (filtered)";
  Real N2f(unit="mol m-3") "Nitrogen concentration (filtered)";
  Real yO2f(unit="1") "Oxygen molar fraction (filtered)";
  Real yCO2f(unit="1") "Carbon dioxide molar fraction (filtered)";
  Real yN2f(unit="1") "Nitrogen molar fraction (filtered)";
  Real yTf(unit="1") "Total molar fraction (filtered)";

  parameter Real epsV(unit="1") = 1E-10 "Filter value";
  Boolean gas "Gas availibility";

  extends PhotoBioLib.Interfaces.Partials.Gas.GasIOutN(nout=14);
  extends PhotoBioLib.Icons.Sensors.GasPotentialSensorIcon;

initial equation
  gas = con.Vol>epsV;

equation
  when {pre(gas) and con.Vol<=epsV, not pre(gas) and con.Vol>epsV} then
    gas = con.Vol >epsV;
  end when;

  Vol = con.Vol;
  O2 = con.O2;
  CO2 = con.CO2;
  N2 =con.N2;

  y[1] = con.Vol;
  y[2] = con.O2;
  y[3] = con.CO2;
  y[4] =con.N2;

  yO2 = if (con.O2 + con.CO2 + con.N2)>0 then con.O2/(con.O2 + con.CO2 + con.N2) else 0;
  yCO2 =if (con.O2 + con.CO2 + con.N2)>0 then con.CO2/(con.O2 + con.CO2 + con.N2) else 0;
  yN2 = if (con.O2 + con.CO2 + con.N2)>0 then con.N2/(con.O2 + con.CO2 + con.N2) else 0;

  yT =yO2 + yCO2 + yN2;

  y[5] = yO2;
  y[6] = yCO2;
  y[7] = yN2;

  con.flowVol=0;
  con.flowO2=0;
  con.flowCO2=0;
  con.flowN2=0;

  if gas then
    Volf=Vol;
    O2f=O2;
    CO2f=CO2;
    N2f=N2;
    yO2f=yO2;
    yCO2f=yCO2;
    yN2f=yN2;
    yTf=yT;
  else
    Volf=0;
    O2f=0;
    CO2f=0;
    N2f=0;
    yO2f=0;
    yCO2f=0;
    yN2f=0;
    yTf=0;
  end if;

  y[8]=Volf;
  y[9]=O2f;
  y[10]=CO2f;
  y[11]=N2f;
  y[12]=yO2f;
  y[13]=yCO2f;
  y[14]=yN2f;

end GasPotentialSensor;
```

PhotoBioLib.Sensors.LiqFlowSensor

Liquid flow sensor

Parameters

Type	Name	Default	Description
Integer	nout	2	Number of outputs

Connectors

Type	Name	Description
output RealOutput	y[nout]	
LiqCon	con1	Liquid connector on the left
LiqCon	con2	Liquid connector on the rigth

Modelica definition

```
model LiqFlowSensor "Liquid flow sensor"
  replaceable class con =
      PhotoBioLib.Interfaces.Partials.Liquid.Liq2OutN (nout=2);
  extends con;
  replaceable class icon =
      PhotoBioLib.Icons.Sensors.LiqFlowSensorIcon;
  extends icon;

  Real flowVol(unit="m3 s-1") "Liquid volume";
  Real flowQ(unit="J s-1") "Heat flow";

equation
  flowVol = con1.flowVol;
  flowQ =  con1.flowQ;
  y[1] = con1.flowVol;
  y[2] =  con1.flowQ;

  connect(con1, con2);
end LiqFlowSensor;
```

PhotoBioLib.Sensors.LiqFlowSensor_ext

Liquid flow sensor

Parameters

Type	Name	Default	Description
Integer	nout	2	Number of outputs
Real	Cv	4184e3	

Connectors

Type	Name	Description
output RealOutput	y[nout]	
LiqCon	con1	Liquid connector on the left
LiqCon	con2	Liquid connector on the rigth

Modelica definition

```
model LiqFlowSensor_ext "Liquid flow sensor"
```

```
  replaceable class con =
      PhotoBioLib.Interfaces.Partials.Liquid.Liq2OutN (nout=2);
  extends con;
  replaceable class icon =
      PhotoBioLib.Icons.Sensors.LiqFlowSensorIcon;
  extends icon;

  Real flowVol(unit="m3 s-1") "Liquid volume";
  Real flowQ(unit="J s-1") "Heat flow";

  Real T;
  Real Tc;
  parameter Real Cv = 4184e3;

equation
  flowVol = con1.flowVol;
  flowQ =  con1.flowQ;
  y[1] = con1.flowVol;
  y[2] =  con1.flowQ;

  T = con1.flowQ/(con1.flowVol*Cv);
  Tc = T-274.15;

  connect(con1, con2);
end LiqFlowSensor_ext;
```

PhotoBioLib.Sensors.LiqPotentialSensor

Liquid potential sensor

Parameters

Type	Name	Default	Description
Integer	nout	3	Number of outputs

Connectors

Type	Name	Description
output RealOutput	y[nout]	Outputs
LiqCon	lCon	Liquid connector

Modelica definition

```
model LiqPotentialSensor "Liquid potential sensor"
  import Modelica.Math.log10;
  replaceable class con =
      PhotoBioLib.Interfaces.Partials.Liquid.Liq1OutN (nout=3);
  extends con;
  replaceable class icon =
      PhotoBioLib.Icons.Sensors.LiqPotentialSensorIcon;
  extends icon;

  Real Vol(unit="m-3") "Liquid volume";
  Real T(unit="0K") "Liquid temperature";
  Real Tc(unit="0C") "Liquid temperature (Celsius)";

equation
  lCon.flowQ = 0;
  lCon.flowVol=0;

  Vol = lCon.Vol;
  T = lCon.T;
  Tc = T-274.15;

  y[1] = lCon.Vol;
  y[2] = lCon.T;
  y[3] = T-274.15;
end LiqPotentialSensor;
```

PhotoBioLib.Sensors.RadSensor

Radiation sensor

Connectors

Type	Name	Description
input RealInput	Io	Radiation

Modelica definition

```
model RadSensor "Radiation sensor"
  extends PhotoBioLib.Icons.Sensors.RadSensorIcon;

public
  Modelica.Blocks.Interfaces.RealInput Io(redeclare type SignalType = Real (
          unit="uE m-2 s-1")) "Radiation";


end RadSensor;
```

PhotoBioLib.UsersGuide

Users Guide of package Photobioreactor

Package Content

Name	Description
Contact	Contact
Literature	Literature
OverView	Overview of library
ReleaseNotes	Release Notes

PhotoBioLib.UsersGuide.Contact

Contact

Main Authors:

Agustín Pérez Castro <agustinperezcastro@bec.uned.es>
Department of Computer Science and Automatic Control (DIA)
School of Computer Science (ETSII)
Spanish University of Distance Learning (UNED)
28040 Madrid
SPAIN

Acknowledgements:

The authors are very grateful to the Spanish Ministry of Economy and Competitiveness for financing this work through the Programme "Formacion dePersonal Investigador (FPI)". This work has been partially funded by the following projects: DPI2011-27818-C02-01 and DPI2011-27818-C02-02.

PhotoBioLib.UsersGuide.Literature

Literature

The Photobioreactor library is based on the following references:

PhotoBioLib.UsersGuide.OverView

Overview of Library Photobioreactor

In this section, an overview of the most important features of this library is given.

(will be added as soon as possible).

PhotoBioLib.UsersGuide.ReleaseNotes

Release notes

Version 1.0.0, 2014-04-15

- A first version has been implemented.

PhotoBioLib.Utilities

Utilities package

Package Content

Name	Description

ReadVar	Read variable from Mat -v4 file
ReadVar_Plus	Read variable from Mat -v4 file
ReadVar_Prod	Read variable from Mat -v4 file

PhotoBioLib.Utilities.ReadVar

Read variable from Mat -v4 file

Parameters

Type	Name	Default	Description
String	LibrarySystemPath	Modelica.Utilities.System.ge...	
String	file	"/data/interpolado.mat"	
String	tableName	"raddat"	

Connectors

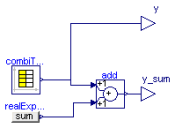
Type	Name	Description
output RealOutput	y	

Modelica definition

```
model ReadVar "Read variable from Mat -v4 file"
  parameter String LibrarySystemPath = Modelica.Utilities.System.getWorkDirectory();
  parameter String file =  "/data/interpolado.mat";
  parameter String tableName = "raddat";


  Modelica.Blocks.Sources.CombiTimeTable combiTable1D(
    tableOnFile=true,
    tableName=tableName,
    fileName=LibrarySystemPath + file);
  Modelica.Blocks.Interfaces.RealOutput y;
equation

  connect(combiTable1D.y[1], y);
end ReadVar;
```

PhotoBioLib.Utilities.ReadVar_Plus

Read variable from Mat -v4 file



Parameters

Type	Name	Default	Description
String	LibrarySystemPath	Modelica.Utilities.System.ge...	
String	file	"/data/interpolado.mat"	
String	tableName	"raddat"	
Real	sum	274.15	

Connectors

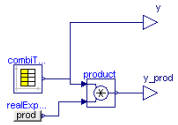
Type	Name	Description
output RealOutput	y_sum	
output RealOutput	y	

Modelica definition

```
model ReadVar_Plus "Read variable from Mat -v4 file"
  parameter String LibrarySystemPath = Modelica.Utilities.System.getWorkDirectory();
  parameter String file =  "/data/interpolado.mat";
  parameter String tableName = "raddat";
  parameter Real sum = 274.15;


  Modelica.Blocks.Sources.CombiTimeTable combiTable1D(
    tableOnFile=true,
    tableName=tableName,
    fileName=LibrarySystemPath + file);
  Modelica.Blocks.Interfaces.RealOutput y_sum;
  Modelica.Blocks.Math.Add add;
  Modelica.Blocks.Interfaces.RealOutput y;
equation

  connect(add.y, y_sum);
  connect(add.u1, combiTable1D.y[1]);
  connect(realExpression.y, add.u2);
  connect(combiTable1D.y[1], y);
end ReadVar_Plus;
```

PhotoBioLib.Utilities.ReadVar_Prod

Read variable from Mat -v4 file



Parameters

Type	Name	Default	Description
String	LibrarySystemPath	Modelica.Utilities.System.ge...	
String	file	"/data/interpolado.mat"	
String	tableName	"raddat"	
Real	prod	Constants.HO2*Constants.yO2*...	

Connectors

Type	Name	Description
output RealOutput	y_prod	
output RealOutput	y	

Modelica definition

```
model ReadVar_Prod "Read variable from Mat -v4 file"
  import PhotoBioLib.Records.Constants "Constants";
  parameter String LibrarySystemPath = Modelica.Utilities.System.getWorkDirectory();
  parameter String file =  "/data/interpolado.mat";
  parameter String tableName = "raddat";

  parameter Real prod = Constants.HO2*Constants.yO2*1/100;


  Modelica.Blocks.Sources.CombiTimeTable combiTable1D(
    tableOnFile=true,
    tableName=tableName,
    fileName=LibrarySystemPath + file);
  Modelica.Blocks.Interfaces.RealOutput y_prod;
  Modelica.Blocks.Sources.RealExpression realExpression(y=prod);
  Modelica.Blocks.Math.Product product;
  Modelica.Blocks.Interfaces.RealOutput y;
equation

  connect(product.y, y_prod);
  connect(combiTable1D.y[1], product.u1);
  connect(realExpression.y, product.u2);
  connect(y, combiTable1D.y[1]);
end ReadVar_Prod;
```