Universidad Diego Portales

Escuela de Informática y Telecomunicaciones

Simulación y Biología Programable

Tarea 3

Profesor:
Martín Gutiérrez

Ayudante:
Freddy Aguilar

Autor:
Diego Pastrian Márquez

Fecha de entrega:
Diciembre 2024

Índice

1. Verilog

El siguiente código está en formato verilog e implementa la lógica del edge detector para posteriormente pasarlo a Cello.

```
/*
This file contains example Verilog code.
The circuit represented below is a genetic circuit implementing an edge detector.
*/

// AND gate for PigmentBlack activation
module and_gate (in_A, in_B, out);
    input in_A;
    input in_B;
    output out;

    and(out, in_A, in_B);
endmodule

// NOT gate for Darkness inversion
module not_gate (in, out);
    input in;
    output out;

    not(out, in);
endmodule

// Complete Edge Detector Circuit
module edge_detector (
    input Darkness,    // Darkness signal
    input AHL,         // AHL signal
    output PigmentBlack // Output signal to produce black pigment
);

    wire not_Darkness; // Inverted Darkness signal
    wire Pigment_logic; // Intermediate logic

    // Invert Darkness
    not_gate invert_darkness (
        .in(Darkness),
        .out(not_Darkness)
    );

    // AND gate for PigmentBlack
    and_gate pigment_and (
        .in_A(AHL),
        .in_B(not_Darkness),
        .out(PigmentBlack)
    );

endmodule
```

Figura 1: Código del circuito en formato Verilog.

Cello otorga la siguiente representación de la lógica dada por el código.

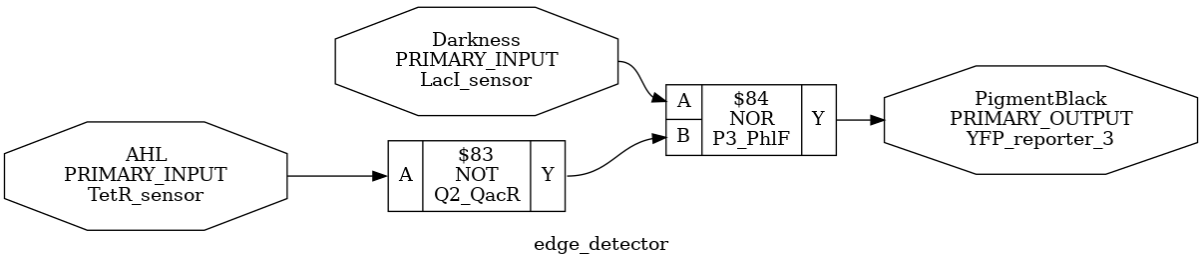

Figura 2: Wiring Diagram Dado por Cello.

Para entender lo anterior, se genera una tabla de verdad donde: p indica la presencia de AHL y q la presencia de oscuridad, el circuito estaría dado por:

p	q	¬pvq	¬(¬pvq)
1	1	1	0
1	0	0	1
0	1	1	0
0	0	1	0

Figura 3: Tabla de verdad asociada al circuito.

Por lo que implementa correctamente la lógica del edge detector ya que el pigmento negro únicamente se prende cuando no hay oscuridad pero sí hay AHL. Los diagramas genéticos entregados por cello se entregan a continuación:

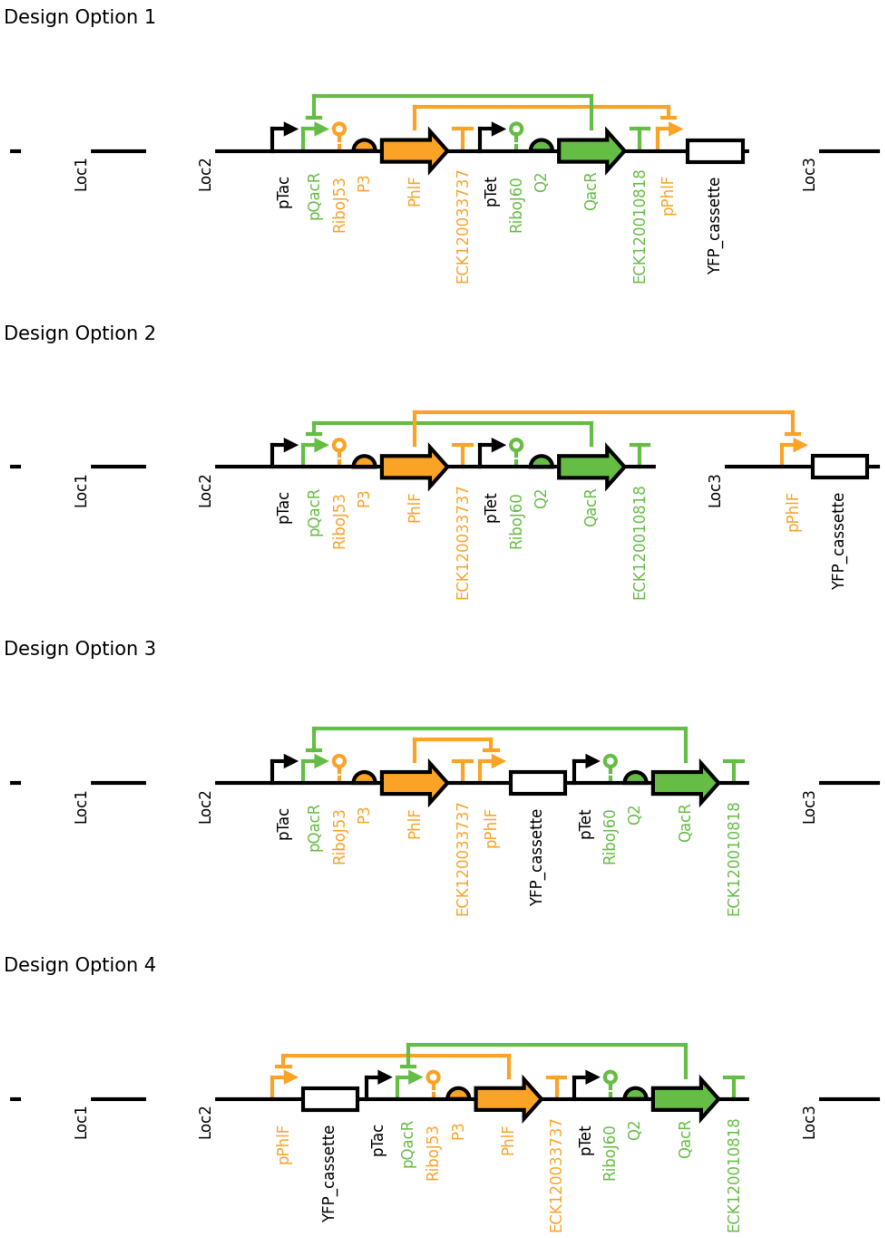

Figura 4: Diseño del circuito por Cello.

2. Pybrick DNA

Se utiliza la segunda opción entregada por cello para pasarla a Pybrick-DNA.

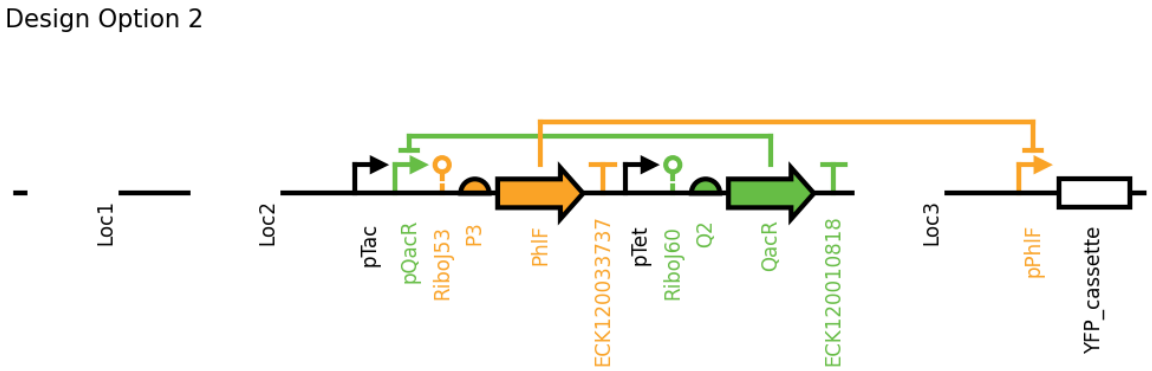

Figura 5: Segunda opción de diseño del circuito.

Este circuito se puede dividir en 3 operones para trabajarlos en Pybrick-DNA.

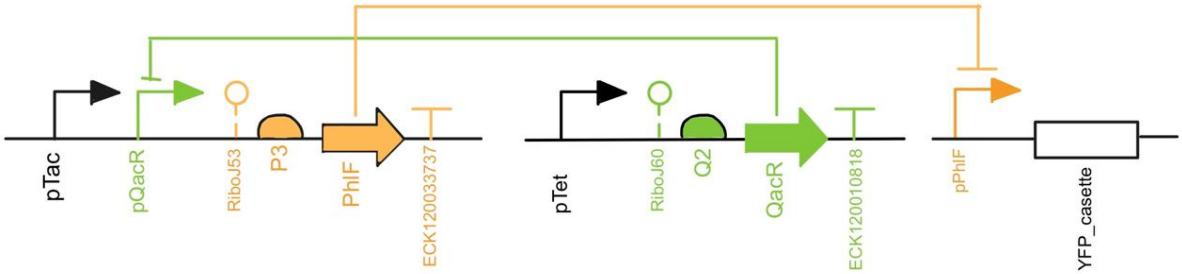

Figura 6: Circuito dividido en operones

Seccionado en 3 operones:

Operón 1:

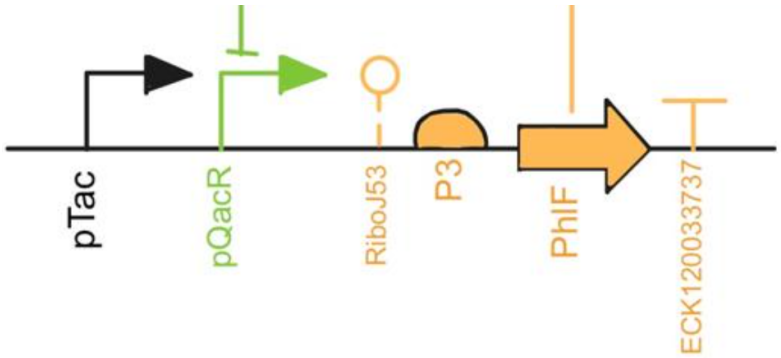

Figura 7: Operón 1.

- Organism: Prokaryote (Utilización de bacterias)
- Promoter: Al haber promotor pTac y pQacR se concatenan sus secuencias para generar un promotor híbrido.
  - pTac: Parte BBa_K864400, su secuencia es: gagctgttgacaattaatcatcggctcgtataatgtgtggaattgtgagcggataacaatt
  - pQacR: Parte BBa_J428050, su secuencia es: ggtatggaagctatacgttaccaattgacagctagctcagtcctactttagtatatagaccgtgcgatcggtctata
- RBS: BBa_B0034, sirve al ser altamente utilizado en la comunidad iGem. Está dado por la secuencia: aaagaggagaaa
- Gen: BBa_K1725040.
- Terminator: BBa_B0015, este es un terminador doble ampliamente utilizado en la comunidad de iGEM. Consiste en dos terminadores fuertes en tándem, lo que proporciona una terminación efectiva de la transcripción.

Figura 8: Esquema del circuito para el operón 1.

Operón 2:

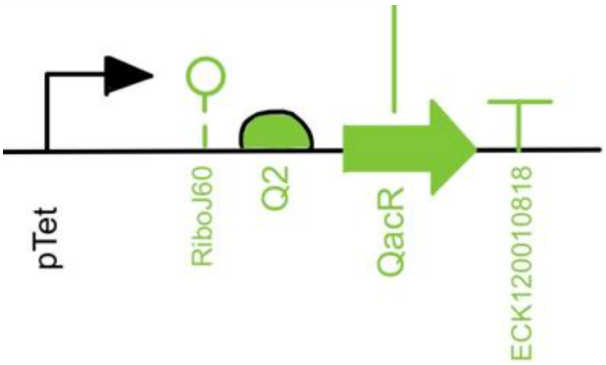

Figura 9: Operon 2

- Organism: Prokayote.
- Promoter: BBa_K2685021
- RBS: BBa_B0034
- Gen: BBa_J428018.
- Terminator: BBa_B0015.

Figura 10: Esquema del circuito para el operón 2.

Operón 3:

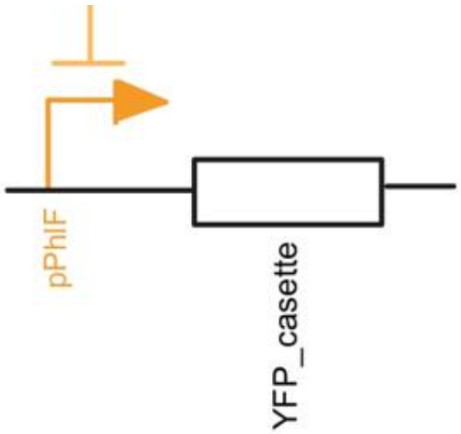

Figura 11: Operón 3.

- Organism: Prokayote.
- Promoter: BBa_K2525016
- RBS: BBa_B0034
- Gen: Bba_K1428018, YFP.
- Terminator: BBa_B0015.

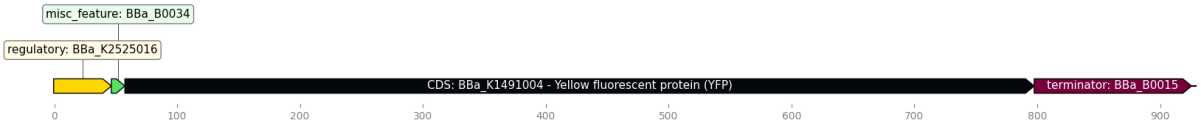

Figura 12: Esquema del circuito para el operón 3.

3. Anexos

1. Github: Repositorio del proyecto