Universidad
Politécnica
de Cartagena

industriales
etsii UPCT

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

TÍTULO:

Modelado y simulación de dos placas fotovoltaicas controladas por Matlab

Adrián Pedrero Campoy

Director:
Javier Molina Vilaplana
Titulación:
Ingeniería Industrial

Índice general

Agradecimientos

En primer lugar quiero agradecérselo a Irene, mi pareja, porque me ha apoyado en momentos de flaqueza y me ha potenciado en los momentos fuertes en la realización de este proyecto, e incluso me ha ayudado de forma técnica alguna vez y no quiero ni pensar qué se encontrarían los lectores si no fuese por ella. A mi madrina Fina por darme su apoyo incondicional todos estos años y por la paciencia que ha tenido conmigo, y a mi abuela Loli porque ha financiado con discreción y cariño la otra faceta de la vida universitaria, fuera de los libros.

No quiero olvidar a los grandes profesionales de la educación que me han llevado hasta aquí, a Javier Molina, mi director de proyecto, y a Jorge Feliú, que me enseñó a usar las herramientas utilizadas en este proyecto, ambos del Departamento de Control de la UPCT, así como José Antonio Parra, antiguo compañero de clase, que me echó un buen cable con Simulink. Quiero recordar también a mi profesor de filosofía del instituto Floridablanca, Onofre porque sin pretenderlo, me dio un gran ejemplo y me contagió sus ansias de sabiduría. Me gustaría mencionar a alguno de mis profesores del colegio Santa Joaquina de Vedruna, pero sería injusto mencionar solo a unos porque son todos grandísimos y son muy culpables de la persona que soy ahora, y ninguno menos importante que otro. Mención especial al que quiera que sea el que organizó en 1º de Bachiller (IES Floridablanca) una visita a la UPCT, porque ese día me enamoré de la ingeniería, y así sigo hoy.

Solo me quedaría agradecer a mis amigos y amigas que han estado conmigo, pero no voy a decir nombres para evitar una larga lista donde seguramente olvide a alguien. Amigo mío, si estás leyendo esto, probablemente sea porque me acordé de ti también, de lo contrario no lo estarías leyendo. Os agradezco a todos que me soportéis hablar de ingeniería, sistemas y del proyecto sin rechistar. Acepto vuestra gratitud por haceros un poco más sabios también. Mención especial a aquellos que se presenten a la exposición del proyecto, por supuesto.

Capítulo 1

Introducción

1.1. Tecnología de captación solar fotovoltaica

El Sol es una fuente de energía inagotable a escala humana: cada año emite hacia la tierra una energía que ni toda la población mundial podría consumir en toda su vida, por lo que resulta evidente que el uso de la energía solar es una vía inteligente.

Hay muchas formas de utilizar la energía procedente del Sol, ya que el viento y el ciclo hidrológico dependen de esta energía, pero también se puede aprovechar de forma más directa mediante colectores solares y *paneles fotovoltaicos*.

Los paneles fotovoltaicos están formados por dispositivos semiconductores tipo diodo que, al recibir radiación solar, se excitan y provocan saltos electrónicos, generando una pequeña diferencia de potencial entre sus extremos. El acoplamiento en serie de estos elementos permite la obtención de diferencias de potencial más elevadas. Esto genera una corriente continua, que se puede transformar a corriente alterna e inyectar en la red para su distribución.

Hoy en día, se está extendiendo cada vez más el uso de dispositivos de energía alternativa en todo el mundo, tanto a nivel industrial como personal. Cada vez se pueden ver más colectores solares y placas fotovoltaicas en casas y edificios, así como en huertos solares. Esto no es sino una ventaja frente a la dependencia de las energías fósiles, ya que la energía proveniente del Sol es, como se comentó antes, inagotable a escala humana, además de que su explotación se compromete mejor con el medio ambiente. Además, son muy útiles en lugares en los que resulta difícil el acceso a la red de distribución eléctrica. Desafortunadamente, el rendimiento de los dispositivos fotovoltaicos pocas

veces supera el 20 %.

La optimización de estos dispositivos ha hecho que se extienda su uso, como la extensión de su uso da como necesidad que se optimice su funcionamiento. Para abordar su optimización, se puede trabajar sobre la energía de entrada y sobre la energía de salida.

Para actuar sobre la **energía de salida**, se trabaja mejorando los materiales semiconductores, para que conviertan mejor la energía de entrada, es decir, mejorando la eficiencia de la transformación.

Para actuar sobre la **energía de entrada**, se trabaja procurando que le llegue el máximo de energía a los dispositivos. Inicialmente esto se hacía diseñando placas curvas parabólicas para optimizar la colección de la radiación y disponiéndolas inclinadas un cierto ángulo llamado *ángulo óptimo*[1].

Figura 1.1: Placas planas y parábolicas

Actualmente se están utilizando seguidores solares (*Solar trackers)*, que se encargan de que la placa mantenga siempre su orientación hacia el Sol a lo largo del día, mediante sensores de luz *(LCR)*. Esto ha permitido el aumento de la recolección de energía solar a lo largo del día, dando unos resultados a ciertas horas del día que sin seguidor no han sido satisfactorios.

Estos seguidores hacen girar la placa mediante uno o dos ejes motorizados y sensorizados. Uno de ellos la hacer girar en la dirección Norte-Sur y el otro en la dirección Este-Oeste(ángulo de orientación). El uso de seguidores de un eje en instalaciones fotovoltaicas es capaz de aumentar el rendimiento en hasta un 30 % con respecto a las placas fijas inclinadas con ángulo óptimo, mientras que el uso de dos ejes permite un mejor seguimiento del sol y se llega a la mejora del 40 %.

1.1.1. Back-Tracking

Sin embargo, la necesidad de recolectar cada vez más energía ha llevado a un problema: la sombra. Colocándose grupos de placas en serie, en ciertos momentos del día la inclinación del Sol hace que unas placas hagan sombra a

[1]El ángulo óptimo de inclinación es un ángulo fijo que maximiza la recolecta de luz teniendo en cuenta la latitud del lugar. No confundir con el ángulo de orientación.

otras, teniendo como problemas la reducción de recolección de potencia de la placa sombreada, así como la aparición de zonas calientes (*hot spots*), cuya aparición puede ser peligrosa para la placa[2].

La primera solución fue imponer una mayor distancia entre las placas, pero si se quiere optimizar el espacio, y con él la energía recolectada, siempre se tiende a querer reducir esta distancia. Así es como apareció el *Back-Tracking*, un algoritmo que se introduce en el seguimiento solar para evitar que las placas cercanas se hagan sombra unas a otras, lo que da lugar a los problemas antes mencionados.

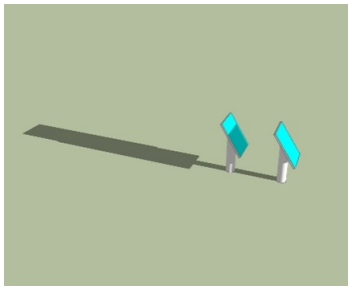

Figura 1.2: Sin Back-Tracking

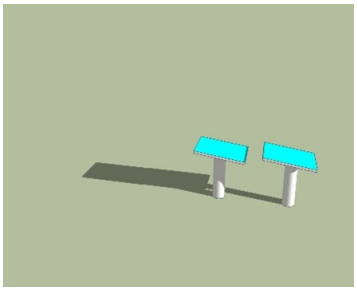

Figura 1.3: Con Back-Tracking

En la imagen 1.2 se puede ver la situación de necesidad de Back-Tracking[1], es decir, momentos del día en los que el Sol está a elevaciones bajas. Cuando se aplica *Back-Tracking*, las placas se disponen de forma que no se sombreen, sacrificando que los rayos del Sol lleguen perpendiculares a la placa para conseguirlo. Esto se puede comprobar en la imagen 1.3. En grandes instalaciones se puede llegar a conseguir un aumento de la eficiencia de 4 % o 5 %.

La reducción de potencia cuando una placa es sombreada puede comportarse de diferentes maneras. Una manera realista de verlo es, suponiendo que cuando parte de una célula es sombreada, esta célula no da potencia alguna, por lo que la potencia de salida sería una función escalonada en función de la sombra[2]. Una forma más optimista es plantear que el porcentaje de potencia perdida es igual al porcentaje de placa sombreada.

En la siguiente figura proveniente de un ensayo en **Matlab** se puede observar como disminuye la potencia de salida (en tanto por uno) cuando aumenta el sombreado. La señal amarilla representa el punto de vista optimista, mientras que la rosa representa el caso en el que se cancela toda la potencia de una fila de células cuando parte de ellas son sombreadas.

[2]Una placa está compuesta por varias células fotovoltaicas, y cada una genera una parte de la energía total

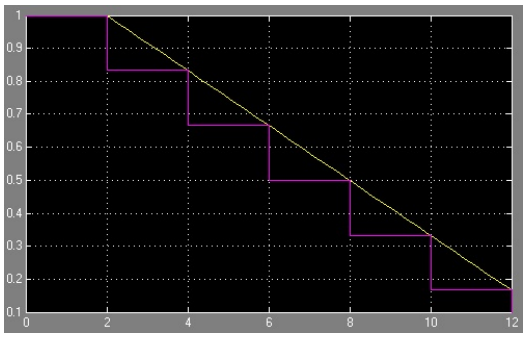

Figura 1.4: Disminución de potencia por sombra

La placa fotovoltaica simulada en el ensayo de la figura 1.4 tiene 60 células, dispuestas en 6 filas de 10 células, por lo que, en el caso realista, cuando se sombree la placa, esta perderá una sexta parte de su potencia, como se puede ver en la figura. Cuantas más filas de células hay, menos pérdida habrá en cada escalón, hasta llegar al caso optimista, en el que la relación entre el porcentaje de sombra y el de potencia es igual a 1. En este caso optimista se hace menos atractiva la idea de aplicar Back-Tracking, debido a que las pérdidas por sombra solo pueden ser menores que en el caso realista, es decir, un 1 % de sombra conlleva un 1 % de pérdida de energía, mientras que en el caso realista se perdería más de un 17 %.

1.2. Resumen del proyecto

En este proyecto se han modelado dos placas fotovoltaicas situadas en el dominio de la Muralla del Mar, de la Universidad Politécnica de Cartagena. Estas han dado el problema de sombrearse entre ellas en ciertos momentos del día, por lo que se ha adoptado por introducir un seguidor con las funcionalidades de *Tracking y Back-Tracking*.

La herramienta utilizada ha sido Matlab y Simulink. **Matlab** es una herramienta de software matemático que ofrece un entorno de desarrollo integrado con un lenguaje de programación propio (lenguaje **M**) muy similar al lenguaje C. Dentro de **Matlab**, existe la herramienta **Simulink**, que es un entorno de programación visual de más alto nivel, ya que se programa con bloques interconectados. Esta herramienta se usa para construir modelos y sistema de control de forma más sencilla y visual.

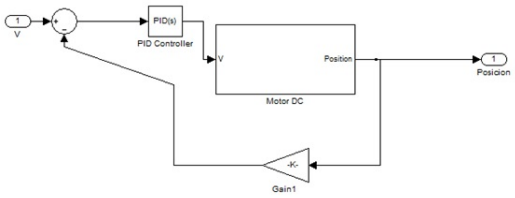

Figura 1.5: Ejemplo de programación en Simulink: Motor DC

Simulink permite la simulación de un modelo construido durante un tiempo ficticio de 10 segundos (pudiendo ser cambiado esto) con la posibilidad de graficar cualquier resultado, por lo que resulta muy útil para hacer modelos como para sintonizar *PID*, comprobaciones...

Entre sus bloques principales o más conocidos están los generadores de señal: Constante, Sinusoidal, Escalón, Pulso... y para leer resultados está el Scope, donde se visualizan las gráficas, o el Display, donde se pueden ver resultados en forma numérica de cualquier señal.

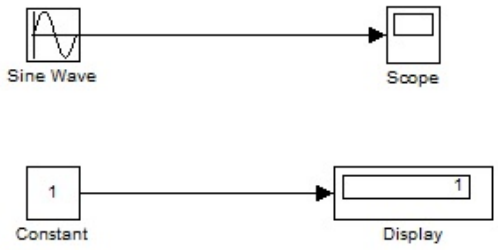

Figura 1.6: Fuentes de señal conectadas a las interfaces gráficas

Simulink también cuenta con bloques que reciben y manipulan la señal. Hay bloques que realizan operaciones matemáticas, así como integradores, derivadores y amplificadores, elementos que componen gran parte de un modelo. Incluso se pueden introducir bloques definidos por el usuario mediante código M, el código característico de Matlab.

Adicionalmente, se ha usado una librería de Simulink llamada **Stateflow**. Esto es una herramienta de control lógico para modelar sistemas reactivos mediante una variación de la conocida máquina de estados. Ha sido usada para el seguimiento solar.

1.2.1. Partes del modelo

El modelo realizado consta de varias partes. Estas partes son fácilmente diferenciables debido a que, al ser programación con bloques, estas partes pueden ser los mismos bloques. Cada parte mencionada será, usualmente, un bloque que contiene más bloques en su interior: un *subsistema*. A continuación serán descritas las partes principales del sistema o modelo:

- Un conjunto de (dos, concretamente) placas fotovoltaicas iguales.
- Dos motores, que se encargarán de mover las placas para seguir al Sol. Incluye cada uno un *PID* para su correcto funcionamiento.
- Calculador de condiciones externas. Calcula la Hora Solar en cada momento, temperatura de las células solares, ángulo de elevación y azimutal.
- Calculador de insolación solar λ , en función de las condiciones externas mencionadas y de la inclinación de las placas en cada momento.
- Calculador de sombra. Obtiene en cada momento la sombra (si la hay) de cada una de las placas, reduciendo la insolación que estas reciben.
- Bloque de control. Bloque realizado en Stateflow, Simulink, que se encargará de hacer el seguimiento y el *Back-Tracking* en función de la potencia que tengan cada una de las placas.

El algoritmo de control realizado para este proyecto tiene la opción de optimizar la energía recibida, lo que es útil para casos en los que los escalones de pérdidas son muy grandes y suponen una inclinación ineficiente de las placas con pérdidas aún mayores. Esto es, el algoritmo vela por que la potencia salvada por evitar el sombreado sea mayor que la sacrificada por inclinación ineficiente, en el caso en el que se diese tal evento.

Por lo tanto, el algoritmo tiene como entrada las potencias de ambas placas, en lugar de señales de sensores de luz, y su salida son las posiciones que deben de adoptar los motores. En la figura 1.3 se pueden ver dos placas ejemplo bajo un algoritmo de Back-Tracking, el cual somete a las dos placas al mismo ángulo corregido de inclinación, mientras que en el algoritmo usado en este proyecto **se controlan por separado**.

Capítulo 2

Descripción del modelo

En este capítulo se van a describir detalladamente todas las partes del modelo. Como se ha dicho anteriormente, el proyecto ha sido realizado con Simulink, herramienta perteneciente a Matlab, por lo que en primer lugar es necesaria una pequeña introducción a su entorno. Posteriormente, será expuesta la realización y explicación de cada una de las partes del modelo: Motor CC, calculadores de posición solar, sombra, insolación, y temperatura de la célula, Finalmente se describirá el modelo de la placa solar.

2.1. Bloques de Simulink

En esta sección se van a exponer los bloques utilizados para el modelo, así como una breve explicación de los mismos:

- Fuentes. Se han utilizado hasta 3 fuentes distintas:

  a) Constante

  b) Tren de pulso

  c) In(*Input*)

- Sumideros de señal. Son bloques cuyo uso es la visualización de las señales a las que se conectan. Han sido utilizados tres bloques sumidero:

  a) *Display*

  b) *Scope*

  c) Out(*Output*)

- *Continuous*. En esta biblioteca hay bloques que modifican la señal integrando y derivando con respecto del tiempo, así como otras operaciones como retrasos o adelantos de señal. Se han utilizado dos de sus bloques:

  a) Integrador
  b) Controlador *PID*

Figura 2.1: Bloques de Continuous

- *Math Operations*:Es una biblioteca de bloques que realizan operaciones matemáticas. Se han usado de aquí los siguientes bloques:

  1. *Abs*
  2. *Add* y *Sum*
  3. *Product* y *Divide*
  4. *Gain*

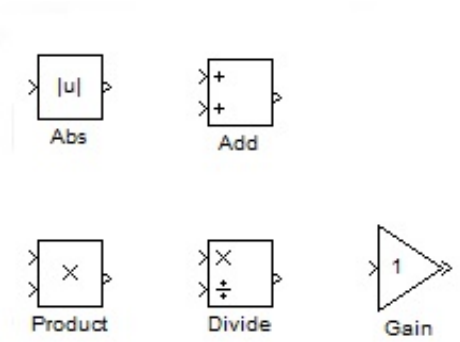

Figura 2.2: Bloques de Math Operations

- *Signal Routing*.Formada por los bloques que manipulan el camino que toma una o varias señales. De esta biblioteca se han cogido dos bloques:

a) *Mux* (multiplexor)

b) *Demux*

- *User-defined functions*. Una biblioteca de bloques que permiten al usuario usar código para programar su funcionamiento. De aquí solo será usado el bloque *Embedded Matlab Function*, función incrustada de Matlab, programable con código M.

2.2. Motor de corriente continua

Se ha escogido un Motor de corriente continua como actuador del sistema de control. La variable de control va a ser el **voltaje de entrada V**, mientras que la salida y parámetro de interés será el **ángulo de salida** θ. El modelado de un motor de corriente continua es representado en la siguiente imagen, donde se pueden identificar los parámetros de entrada y salida mencionados:

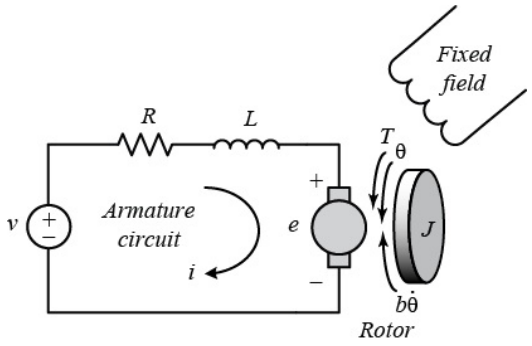

Figura 2.3: Modelo de un motor CC

A continuación, se explican el resto de parámetros:

- **R** es la resistencia eléctrica del motor. Se ha adoptado un valor de 0,5Ω
- **L** es la inductancia, con un valor de $1{,}5mH$.
- **J** es el momento de inercia del rotor del motor y la carga, y se le atribuye un valor de $0{,}00025\frac{Nm}{rad/s^2}$
- El parámetro **b** es la constante de fricción viscosa del motor, cuyo valor es $0{,}001\frac{Nm}{rad/s}$.

El torque generado por un motor de corriente continua es proporcional a la corriente de armadura y a la fuerza del campo magnético. Se ha asumido que el campo magnético es constante, y que, por lo tanto, el torque solo es proporcional a la corriente de armadura de la siguiente forma:

$$T = K_t i$$

La fuerza electromotriz **e** es proporcional a la velocidad angular del eje, por lo que se puede escribir:

$$e = K_e \frac{d\theta}{dt}$$

Siendo $K_t = 0{,}5$ la constante del torque del motor y $K_e = 0{,}05$ la constante de fuerza electromotriz, iguales en este modelo.

Para construir este modelo en Matlab, primero modelamos las integrales de la aceleración angular y de la ratio de cambio de la corriente de armadura:

$$\int \frac{d^2\theta}{dt^2} dt = \frac{d\theta}{dt} \tag{2.1}$$

$$\int \frac{di}{dt} dt = i \tag{2.2}$$

Después, se aplica la *Ley de Newton* y la *Ley de Kirchhoff* al motor para general las siguientes ecuaciones:

$$J\frac{d^2\theta}{dt^2} = T - b\frac{d\theta}{dt} \longrightarrow \frac{d^2\theta}{dt^2} = \frac{1}{J}(K_t i - b\frac{d\theta}{dt}) \tag{2.3}$$

$$L\frac{di}{dt} = -Ri + V - e \longrightarrow \frac{di}{dt} = \frac{1}{L}(-Ri + V - K_e\frac{d\theta}{dt}) \tag{2.4}$$

Para llevar a cabo este modelo en Simulink, se necesita crear un modelo (**.mdl**) de Simulink. El primer paso es acceder al explorador del modelo (*Model explorer*) para establecer el valor de los parámetros, para que así sea más fácil cambiarlos, sin necesidad de tocar los bloques. Para acceder al explorador, se sigue la ruta $\hookrightarrow View/ModelExplorer$ o bien usando la combinación de teclas *Ctrl+H*. Una vez allí se selecciona *Model Workspace* en el menú de la izquierda. Aparecerá una ventana como la siguiente:

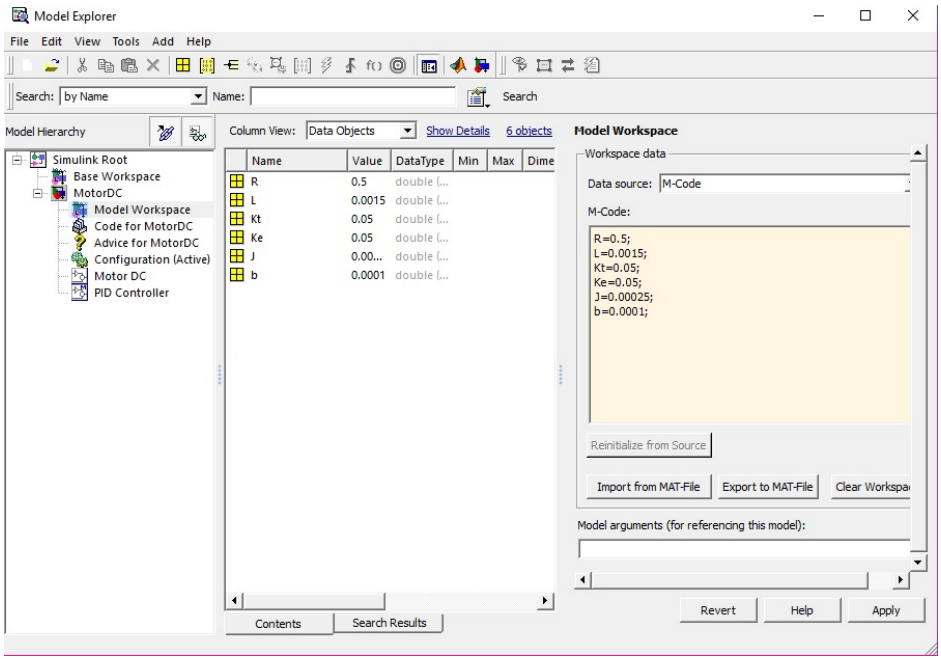

Figura 2.4: Ventana del Model Explorer

Para introducir los datos, hay que seleccionar el desplegable de *Data source*, a la derecha de la ventana, seleccionar **M-code**, y a continuación definir las variables pertinentes tal y como se hace en Matlab. En la imagen de ejemplo se ven ya introducidas, y también puede verse como se reflejan en el cuadro principal, en el centro de la ventana. Se aplican los cambios.

Ahora se empieza a construir el modelo con bloques de Simulink. Para el modelo del motor se usan *Integradores* para integrar las ecuaciones diferenciales, el bloque *Add* para sumar y restar señales, y el bloque *Gain* para multiplicarlas por los parámetros. Como el motor es solo un subsistema del modelo general, es necesario introducirle y extraerle señales, por lo que también se colocan los bloques *Input y Output*, dándoles el nombre de **V** y **Posición**, que son la entrada y salida respectivamente. El subsistema motor DC quedaría de la siguiente forma:

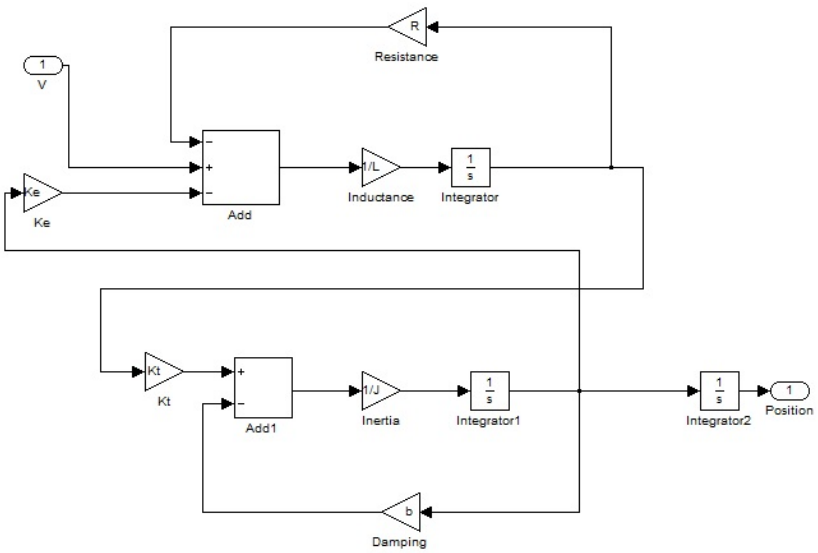

Figura 2.5: Modelo en Simulink de un motor CC

Si se seleccionan todos los elementos y se pulsa el botón derecho del ratón, aparece la opción de *crear un subsistema*, con lo que todos se agruparan en un solo bloque cuya entrada es V y salida la posición.

Para que se estabilice en una posición ante un valor de tensión, es necesario aplicar una realimentación. Como la salida es un valor de posición angular (radianes) y la entrada es un nivel de tensión (voltios) en la realimentación es necesario poner una ganancia, que está definida por la constante de conversión ángulo-voltaje, determinada por los niveles de voltaje a los que puede ser excitado el motor. Se ha decidido que el motor puede ser excitado en el rango $[-18, 18]V$, y cada voltio significará un giro de 5º del rotor (y la placa), por lo que teniendo en cuenta la conversión a radianes, esta ganancia será $K = 36/\pi$. Una vez hecha la realimentación, se coloca un escalón unitario, se simula, y se obtiene la siguiente respuesta:

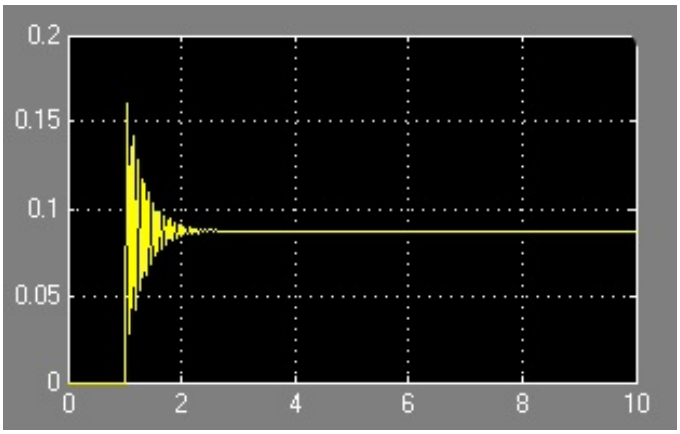

Figura 2.6: Respuesta del motor ante escalón unitario: Sobre-oscilación

Se puede observar que el estacionario tiene el valor correcto, equivalente en radianes a 5º, pero el transitorio da problemas. Hay una clara sobre-oscilación que hay que evitar, y también un sobre-pico que llega a un valor de 0,16 radiantes, casi el doble del valor estacionario, cosa que puede ser perjudicial.

Para evitar estos efectos, se coloca un bloque *PID*, descrito anteriormente. Este bloque es colocado en el modelo justo antes del motor. En las opciones del bloque hay tres campos en los que se puede definir el término **proporcional**, **integrador** y **derivativo** del *PID*. Se puede introducir cualquier valor, pero se puede ver un botón llamado *tune*, que, pulsando en él, Simulink calcula varias posibilidades de sintonización del *PID*. Una vez que está calculado, el usuario puede variar el tiempo de establecimiento de la señal si fuera necesario.

Con este bloque introducido, se vuelve a hacer la simulación y se observa el resultado:

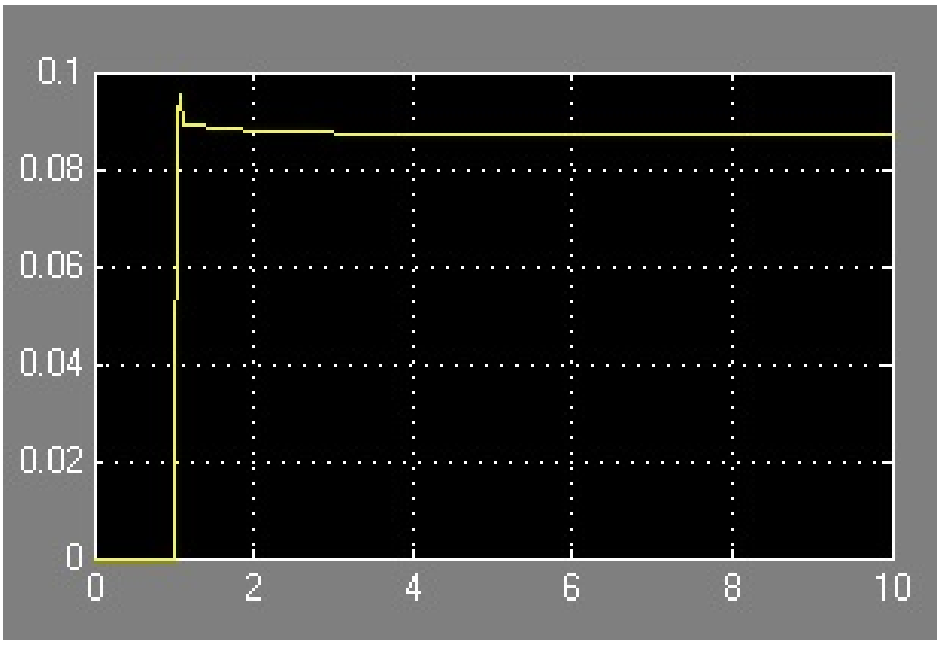

Figura 2.7: Señal corregida del motor

La ventana de sintonización del PID que recoge los parámetros adoptados es la siguiente:

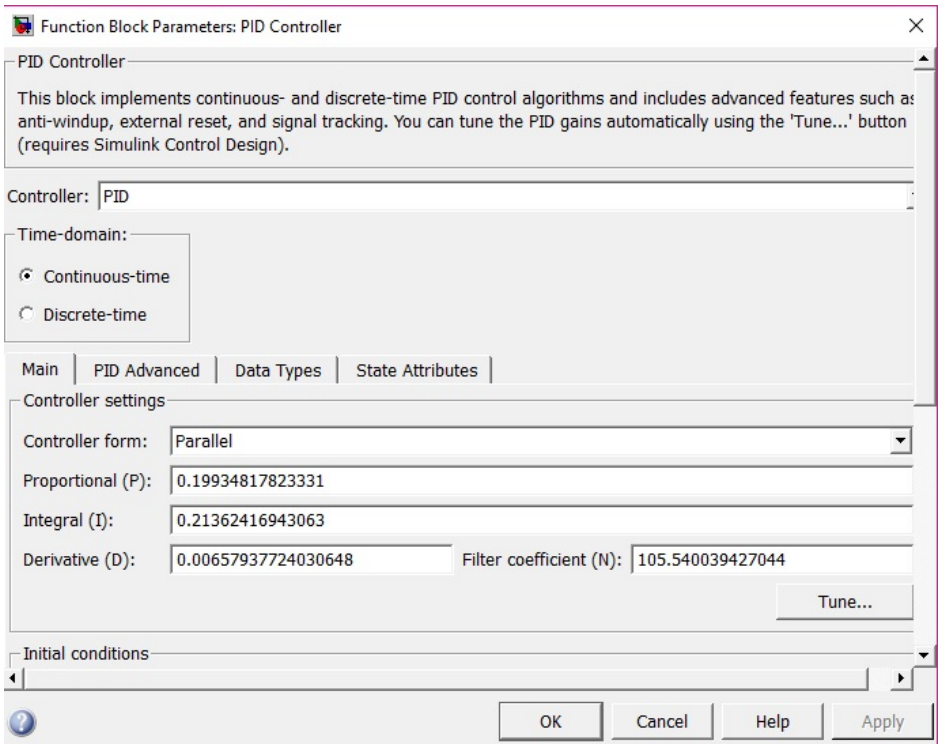

Figura 2.8: Ventana de sintonización del PID

Así, se daría por terminado el modelo del motor, estructura final se puede visualizar en la imagen 1.5 del capítulo anterior.

2.3. Calculador de la posición solar

Con el objeto de seguir al Sol y hacer cálculos pertinentes de irradiación y sombra, es necesario saber en cada momento los ángulos relacionados con el Sol.

Estos ángulos son:

- Elevación
- Azimuth
- Hora solar

La hora solar está relacionada con la hora oficial, y es la necesaria para el seguimiento solar, mientras que la elevación y el azimut servirán para calcular la irradiación de la que disponen las placas.

El cálculo de estos parámetros ha sido realizado con una función incrustada de Matlab, es decir, programado con código M, y devuelve como dato los 3 ángulos mencionados, teniendo en cuenta la fecha y la hora del ordenador en el que tenga lugar la simulación, así como el lugar y su latitud correspondiente.

No obstante, este bloque por sí solo proporciona los valores de esos ángulos únicamente una vez: cuando es llamado debido al inicio de la simulación. Debido a esto, al sistema de control le llegaría siempre los mismos valores. Para evitar esto, se ha introducido una S-Function, un bloque de Simulink personalizable. Se encargará de tener actualizados todos los datos de fecha y hora en todo momento.

2.4. Calculador de la sombra

Para este cálculo se ha optado por cambiar los ejes de referencia del Sol. Cuando este está perpendicular al plano normal a la Tierra se va a decir que la hora solar es 0^o, es decir, es mediodía. Entonces el amanecer se da para -90^o y el Sol se oculta por completo a partir de 90^o.

Para dos placas fotovoltaicas siguiendo la trayectoria del Sol, con una distancia entre ellas **L**, con una altura **h** con respecto al suelo y con dimensiones [**a** $\times$ **b**] (siendo la dimensión 'a' la perpendicular al papel), existe una hora solar crítica de sombreado.

Al amanecer, la placa del Oeste aparece sombreada, ya que el Sol sale por el Este, y la primera placa iluminada hace sombra a la posterior. Conforme se va haciendo de día, y suponiendo que ambas placas siguen la trayectoria del Sol, el sombreado de la placa del Oeste se va reduciendo, hasta alcanzar un valor nulo para un valor de $-\alpha_{crit}$ de la hora solar. Al atardecer ocurre algo similar. Durante el seguimiento solar a pleno día no hay sombreado, pero por la tarde, al alcanzar la hora solar un valor de $+\alpha_{crit}$, la placa del Este comienza a ser sombreada, desde un valor nulo hasta el máximo cuando cae la noche.

Esto ocurre durante el seguimiento normal de ambas placas al movimiento del Sol, pero puede resultar en un comportamiento inadecuado de las placas, debido a la pérdida de potencia y a la peligrosidad de los llamados "puntos" calientes. Resulta necesario entonces aplicar *Back-Tracking*, el algoritmo que evita o reduce el sombreado mutuo. Para calcular la sombra de cada placa en cada momento, se ha diseñado un calculador mediante código M, por lo que el bloque calculador es una función de Matlab incrustada. Dentro puede encontrarse el código M, que resulta en un vector "sombra", que contiene en tanto por uno el **porcentaje libre de sombra** de cada placa, para

multiplicar por la **insolación** antes de introducir esta en el modelo de la placa solar.

Las entradas de este bloque, y por tanto los parámetros de los que depende la sombra son: la hora solar y las orientaciones de las placas, en radianes, y los parámetros geométricos antes mencionados L, b y h, que podrán ser cambiados, ya que son introducidos mediante bloques "Constant" por Simulink.

El problema del cálculo de sombra es un problema geométrico, que va a ser descrito a continuación: Se sitúan las dos placas sobre el plano, haciendo coincidir el origen de los ejes con el principio del soporte de la placa situada al Oeste. A su derecha, a una distancia L, se sitúa la placa del Este. Serán indexadas como placa 1 (Oeste) y placa 2 (Este). Las placas están situadas sobre los soportes de dimensión h y la dimensión de las mismas contenidas en el plano es $2b$.

Para calcular sombras el primer paso es calcular el ángulo crítico a partir del cual se presenta el sombreado. Para ello, es supuesto que ambas placas siguen perfectamente al Sol, es decir, $E_1 = E_2 = \alpha$. Se toma, por simplicidad de cálculo, el caso del atardecer, ya que se ha hecho coincidir la placa del Oeste con el eje de coordenadas.

El caso de ángulo crítico será aquel en el cual la sombra de una de las placas proyectada por la parte superior de la misma coincide exactamente con la parte inferior de la otra placa, en el caso supuesto de que ambas placas tienen la misma orientación perfecta hacia el Sol. Esto es un problema geométrico de *resolución de un triángulo*, cuyos catetos son conocidos. Siendo la distancia entre las placas igual al valor L, y con la dimensión b antes mencionada (h no va a tener ninguna importancia) se identifica el triángulo a resolver, como se puede ver en la figura siguiente.

El cateto horizontal tiene un valor de $L - 2b\cos(\alpha)$ mientras que el otro cateto tiene un valor de $2b\sin(\alpha)$. La trigonometría afirma que la tangente de un ángulo es igual a su cateto opuesto dividido entre su cateto contiguo, por lo que:

$$\tan(\alpha) = \frac{L - 2b\cos(\alpha)}{2b\sin(\alpha)} \tag{2.5}$$

Se descompone la tangente de forma que $\tan(\alpha) = \sin(\alpha)/\cos(\alpha)$, y se recombina la igualdad de forma que queda:

$$2b(\sin^2(\alpha) + \cos^2(\alpha)) = L\cos(\alpha)$$

Sabiendo por trigonometría que $\sin^2(\alpha) + \cos^2(\alpha) = 1$ se puede determinar

finalmente que:

$$\alpha_{critico} = \arccos\left(\frac{2b}{L}\right) \tag{2.6}$$

Ahora hay que calcular los dos casos en los que se puede dar sombra: *Amanecer* y *Atardecer*. Siguiendo con el atardecer usado como ejemplo, se procede a la definición de los siguientes puntos y rectas necesarios para el cálculo instantáneo de la sombra, que en caso del atardecer se dará únicamente en la placa del Este, es decir, la placa 2.

$$y_1 = h + x\tan(E_1) \tag{2.7}$$

Donde y_1 es la recta que define la placa 1 y E_1 es el ángulo de orientación asociado a la placa 1. Si se define el punto $A(b\cos(\alpha), h + b\sin(\alpha))$ como el punto más alto de la placa 1, se puede trazar una recta con dirección solar que pase por el punto A, dando lugar a la recta proyección de la sombra:

$$y_{s1} = h + b\sin(E_1) + \frac{b\cos(E_1)}{\tan(\alpha)} - \frac{x}{\tan(\alpha)} \tag{2.8}$$

El siguiente paso es calcular la ecuación de la recta asociada a la placa 2, para hallar el punto de intersección entre esta y la última recta calculada. La recta de la placa 2 es la siguiente:

$$y_2 = h + (x - L)\tan(E_2) \tag{2.9}$$

Igualando las ecuaciones de la recta 2.8 y 2.9 se obtiene el punto de intersección de la sombra:

$$x_{int} = L\tan(E_2)\tan(\alpha) + b(\sin(E_1)\tan(\alpha) + \cos(E_1) \tag{2.10}$$

$$y_{int} = h + (x_{int} - L)\tan(E_2) \tag{2.11}$$

Este punto, si $\alpha > \alpha_{critico}$, estará contenido en la placa 2, es decir, el punto de intersección estará por encima del punto más bajo de la placa 2. Esto quiere decir que, si existe sombra, y_{int} será mayor que y_ω, donde este último parámetro es la altura a la que se encuentra el punto más bajo de la segunda placa. En este caso, el punto omega(ω) es:

$$x_\omega = L - b\cos(E_2) \tag{2.12}$$

$$y_\omega = h - b\sin(E_2) \tag{2.13}$$

Una vez calculado todo esto, el porcentaje de sombra es la distancia entre el punto de intersección y el punto omega, dividido entre la longitud $2b$ de la placa. Por cuestiones de utilidad, se calcula el **porcentaje de placa no sombreada** como:

$$S = \frac{\|x_{int} - x_\omega, y_{int} - y_\omega\|}{2b} \tag{2.14}$$

Para el caso del amanecer se toman las placas orientadas hacia la derecha como en la figura siguiente:

Teniendo la recta y_2 y a su vez la recta proyección de la sombra y_{s2} construida de la misma manera que en el caso del atardecer, se puede hallar la intersección de esta con y_1. En este caso el punto de intersección resultará:

$$x_{int} = \frac{L - b(\cos(E_2) - \sin(E_1)\tan(\alpha))}{1 + \tan(E_1)\tan(\alpha)} \tag{2.15}$$

$$y_{int} = h + x_{int}\tan(E_1) \tag{2.16}$$

La sombra de la placa del Oeste se calcula de la misma forma, teniendo en cuenta que el punto omega ahora es el punto más bajo de la misma placa.

Nota. *En el escenario de sombra realista se hará un pequeño ajuste en el código en el cual cuando la sombra represente entre un 0 % y un 17 %, esta pase a adoptar directamente este último valor. Cuando exceda este, pasará a valer 33 % y así sucesivamente. Estos valores equivalen a una sexta y dos sextas partes de sombra en la placa.*

2.5. Cálculo de la insolación recibida por las placas

Este subsistema utiliza las variables solares que se calcula el bloque mencionado anteriormente, la inclinación de cada placa (parámetro que enviarán aquí los motores), así como la sombra. La insolación que recibiera una placa

solar sería máxima si esta estuviera perfectamente perpendicular al vector Sol, es decir, si los rayos solares incidieran perpendicularmente. Esto normalmente no es así, razón por la que se usa *Tracking* o seguimiento.

Para seguir la elevación se hace seguimiento con un eje, pero si se desea una perpendicularidad perfecta, también hay que subsanar el ángulo azimutal, cuya solución sería hacer seguimiento con dos ejes, horizontal y vertical. En este modelo solo se ha adoptado el seguimiento con un eje (horizontal).

La insolación dependerá fundamentalmente de:

- La insolación máxima que recibiría en el caso de perpendicularidad ideal. Se ha adoptado un valor de $\lambda_{max} = 1000W/m^2$
- La diferencia entre la elevación y el ángulo de orientación de las placas. Casi siempre esta diferencia será distinta de 0 debido a que las posibles posiciones de las placas están definidas de 5 en 5 grados.
- La inclinación fija β(no confundir con orientación).
- La elevación solar.
- La sombra que perciban.

La diferencia de ángulos mencionada es la responsable de que la insolación recibida no sea la máxima, por lo que cualquier valor de que sea distinto de 0 causará que la insolación sea más baja. Esto ocurre tanto cuando esta diferencia es positiva como negativa, es decir, una diferencia entre la elevación y el ángulo de orientación de las placas de 7º conllevará la misma reducción de la insolación si esta diferencia fuera de -7º. Además, la insolación varía sinusoidalmente con respecto a esta diferencia angular debido al carácter cíclico del Sol, permaneciendo inmutable si esta última vale 0. Por esto, se determina que hay que multiplicar la insolación máxima por el coseno del valor absoluto de tal valor. Esto se acabaría aquí si las placas estuviesen situadas en el ecuador de la Tierra, pero al no ser así hay algo más que tener en cuenta. La ubicación es Cartagena, con una latitud de 36,7º. Esto afectará a la Elevación, que no coincidirá con el ángulo de la hora solar. Según *pveducation.org*[3], la insolación que recibe el módulo solar es:

$$S_{module} = S_{incident} \sin(\alpha + \beta)$$

donde α es la elevación, dependiente de la latitud y de la declinación solar, y β es la inclinación fija que se le haya podido dar a la placa[1].

[1]En principio no tiene por qué dársele un valor distinto de 0 a este ángulo, pero la simulaciones y la información encontrada en *pveducation.org* en que un valor de 23º mejoraba la colecta de energía solar a lo largo de todo el año

La ecuación que combina todos estos efectos queda como:

$$\lambda = \lambda_{max} \left(\cos(|Elevacion - E_{placas}|)\right) \sin(\alpha + \beta) \tag{2.17}$$

Esto es fácilmente traducido a bloques en Simulink, como se puede ver en la siguiente imagen:

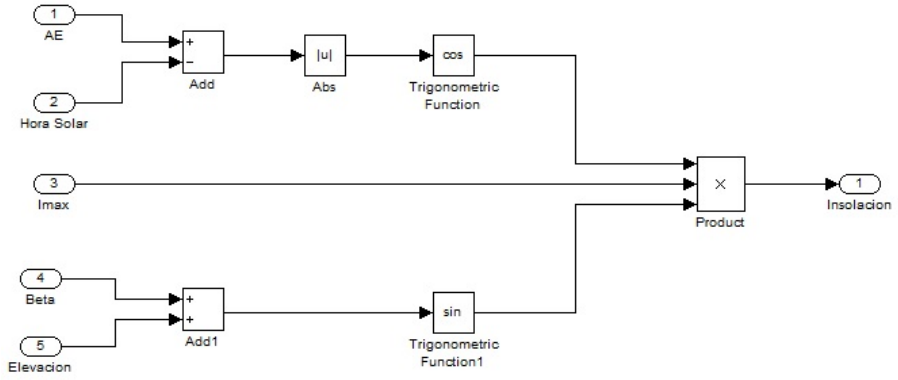

Figura 2.9: Modelo de calculador de insolacion

Este subsistema dará como resultado una insolación próxima a $1000W/m^2$ en horas cercanas al mediodía solar, disminuyendo conforme la hora se aleje de este.

2.6. Calculador de la temperatura de la célula

El subsistema que va a ser descrito aquí calcula la temperatura en el interior de la placa fotovoltaica, ya que esta disminuye su rendimiento si se eleva demasiado. Es un subsistema sencillo, cuya salida es la temperatura de célula T_c y cuyas entradas son la temperatura ambiente T_a y la insolación λ que reciba la placa, proveniente del subsistema descrito antes que este.

El cálculo de esta temperatura se realiza mediante la siguiente fórmula:

$$T_c = T_a + \frac{\lambda}{\lambda_{max}}(NOCT - T_{ref}) \tag{2.18}$$

Donde *NOCT* es la temperatura medida en condiciones de operación normales (*Normal Operation Cell Temperature*), dato que da el fabricante, y

T_{ref} es la temperatura medida en las condiciones de referencia. De la hoja de características de la placa elegida se identifican estos parámetros:

$$NOCT = 46^oC; T_{ref} = 20^oC$$

El subsistema compuesto por los bloques encargados de hacer esta operación es el siguiente:

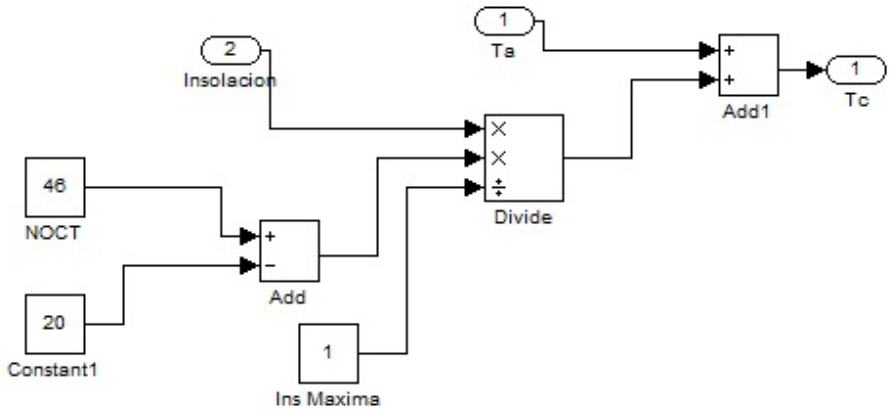

Figura 2.10: Modelo de calculador de temperatura de célula

2.7. Placa fotovoltaica

Un panel fotovoltaico está formado por un conjunto de células fotovoltaicas que producen electricidad a partir de la luz que incide sobre ellas mediante el efecto fotoeléctrico, que consiste en la emisión de electrones de un material al incidir sobre él una onda electromagnética de cierta energía.

Los metales conducen muy bien la electricidad debido a que sus electrones deslocalizados de la capa más alta pueden moverse libremente, ya que ocupan niveles altos de energía y eso les permite escapar fácilmente del átomo al que están ligados. El conjunto de estos niveles se llama **banda de conducción**, en la que también existen niveles vacíos próximos a los que pueden ir a parar los electrones en movimiento. Los electrones localizados en niveles más bajos de energía y por tanto más ligados al átomo, se dice que están en la **banda de valencia**. En los metales estas bandas están solapadas.

Los aislantes tienen la banda de conducción vacía y su banda de valencia llena, todos sus átomos están ligados al átomo, impidiendo su movimiento,

la conducción eléctrica. Sus bandas están alejadas, por eso los electrones no pueden saltar entre ellas como ocurre en los metales.

Los semiconductores presentan una situación intermedia. En la banda de conducción hay pocos electrones, y la separación entre bandas existe, pero es menor que en el caso de los aislantes. En tal caso, cuando se les aplica energía es posible la corriente eléctrica en materiales semiconductores, como va a ser en el caso de las placas fotovoltaicas, cuyo material es semiconductor.

Las placas fotovoltaicas están formadas por semiconductores. Cada célula está compuesta de al menos 2 delgadas láminas de *silicio*. Una de ellas dopada con menos electrones que el Silicio, denominada *capa P*, y otra dopada con más, denominada *capa N*. La luz incide sobre la capa P, haciendo que fluyan electrones hacia la capa N, pero no pudiendo volver. De esta forma se crea una diferencia de potencial entre ambas capas y si se conectara una carga al dispositivo se crearía una corriente eléctrica.

Un modelo eléctrico simplificado sería el de una fuente de corriente con un diodo de unión p-n en paralelo, una resistencia en paralelo y una resistencia en serie. La resistencia en serie R_s está asociada a diferentes efectos como la resistencia de los contactos, de los propios semiconductores y de los diodos metálicos que constituyen la *malla de metalización frontal*. La resistencia en paralelo R_p está relacionada con las imperfecciones de la unión p-n y representa las corrientes de fuga. En este modelo se van a obviar estas resistencias, se va a tomar el caso ideal, es decir $R_S = 0$ y $R_p = \infty$.

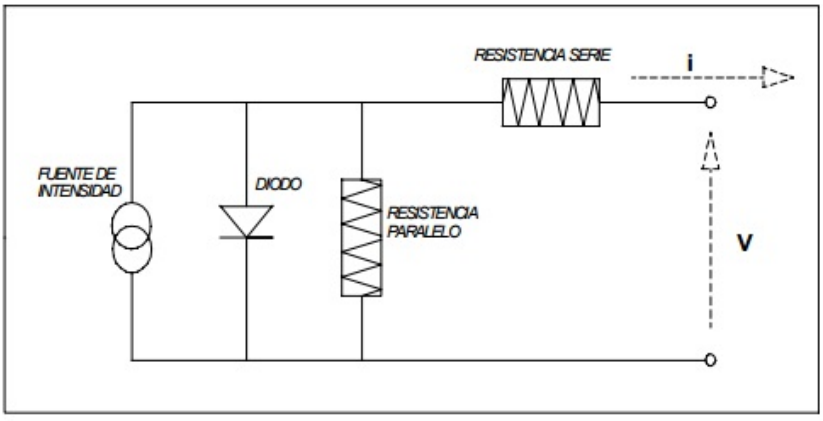

Figura 2.11: Modelo eléctrico de una célula

Sin embargo, existe otra fuente de pérdidas de corriente como se verá en

las fórmulas del modelado de la célula: la corriente de saturación.

La expresión para calcular la corriente que se obtiene de la placa es la siguiente:

$$I = I_{PH} - I_S \left(\exp\left(\frac{q(V + IR_S)}{kT_cA} \right) - 1 \right) - \frac{(V + IR_S)}{R_P} \tag{2.19}$$

Donde los parámetros que aparecen son:

I_{PH}=Corriente generada por el efecto fotovoltaico.

I_s=Corriente de saturación.

$q = 1{,}6E - 19C$ (carga de un electrón).

$k = 1{,}38E - 23J/K$ (constante de Boltzman).

T_C=Temperatura de la célula.

$A = 1{,}2$ (factor ideal del diodo para Silicio monocristalino).

En el segundo sumando, se da que: $\exp\left(\frac{q(V + IR_S)}{kT_cA} \right) \gg 1$. Además, se ha considerado para este modelado $R_S = 0$ y $R_P = \infty$, por lo que la expresión quedaría más simplificada:

$$I = I_{PH} - I_S \left(\exp\left(\frac{qV}{kT_cA} \right) \right) \tag{2.20}$$

Por el momento se sabe que k, q y A son constantes conocidas, y como entradas se tienen la tensión V, la temperatura de célula T_C y la insolación λ. Se desconocen ambas corrientes presentes en la anterior fórmula, y tendrán que ser calculadas.

Corriente fotogenerada

Esta es la corriente que depende de la insolación recibida por la placa. Antes de su cálculo hay que tener en consideración que el fabricante proporciona un dato, la intensidad de cortocircuito I_{SC}, que, despreciando las pérdidas y el diodo del modelo de la célula, coincidiría con la corriente fotogenerada en condiciones normales de operación. Dicho esto la expresión para el cálculo de la primera corriente mencionada es:

$$I_{PH} = \frac{\lambda}{\lambda_{max}} I_{SC}(1 + \alpha(T_c - T_{ref})) \tag{2.21}$$

Donde I_{SC} es la corriente de cortocircuito, $I_{SC} = 7{,}34A$ en este modelo debido

a las características del panel comercial seleccionado. A su vez, $\alpha = 0{,}04$ también es proporcionado por el fabricante y es el coeficiente de temperatura de la corriente de cortocircuito. Se puede interpretar como el cambio que sufra este corriente cuando la temperatura no es la de ensayo o referencia, ya sea por encima o por debajo de esta.

Corriente de saturación

La corriente de saturación va a ser responsable de algunas ineficiencias de la placa. Va a aumentar conforme aumente la temperatura de célula, por lo que, a mayor temperatura, menos eficiencia. La expresión sería la siguiente:

$$I_S = I_{RS} \left(\frac{T_c}{T_{ref}} \right)^3 \exp \left(\frac{qE_G}{kA} \left(\frac{1}{T_{ref}} - \frac{1}{T_c} \right) \right) \tag{2.22}$$

Donde I_{RS} es la llamada *Reversal Saturation Current*, y E_G es el *gap de energía*[2] del semiconductor que hay que aplicar para que se produzca la corriente de un electrón. Esta corriente de saturación inversa corresponde a la fuga de portadores de carga a través de la unión p-n, como resultado de una recombinación de los mismos en una zona neutra del semiconductor. Se calcula mediante la expresión:

$$I_{RS} = \frac{I_{SC}}{\exp \left(\left(\frac{qV_{OC}}{N_s AkT_c} \right) - 1 \right)} \tag{2.23}$$

Donde I_{SC} y V_{OC} son la *intensidad de cortocircuito* y la *tensión de circuito abierto*, respectivamente, datos que proporciona el fabricante en su hoja de características. El parámetro N_S representa el número de células en paralelo que hay en una placa fotovoltaica, y en el caso de la placa seleccionada, este parámetro es igual a 1. Para construir todo esto en Simulink, el primero paso es establecer los parámetros fijos. Como se hizo construyendo el motor, se accede al explorador del modelo, y como fue explicado anteriormente, se introducen mediante código M.

Las expresiones del cálculo de las corrientes de saturación han sido introducidas mediante el bloque función de Matlab incrustada, y el resto de operaciones se han realizado con bloques de Simulink comunes, de los expuestos al principio del capítulo. No se va a exponer visualmente la construcción de este modelo debido a su extensión, pero está disponible en los ficheros del proyecto.

[2]Se ha adoptado un valor de $E_G = 1{,}4eV$

Capítulo 3

Control discreto con Stateflow

En este capítulo será descrito cómo se ha diseñado y llevado a cabo el control para este proyecto. Inicialmente se dará un esbozo con la herramienta Grafcet. En la siguiente sección se explicará ligeramente fundamentos básicos de Stateflow, y finalmente en la última sección se expondrá el Chart de control con Stateflow paso por paso.

3.1. Diseño del control con Grafcet

El control de las placas está diseñado por Estados discretos y transiciones entre ellos. Para el diseño teórico del mismo se ha recurrido a Grafcet, que es una herramienta de diseño gráfico secuencial. De esta herramienta surgen modelos consistentes en cajitas o Estados que cumplen cierta secuencia, que pueden estar activados o desactivados y que están conectados por transiciones, que pueden tener o no condiciones para poder darse. Además, cada una de estas cajitas o Estados puede tener asociada una o varias acciones.

Grafcet resulta una herramienta perfecta para el diseño en este proyecto, ya que Stateflow, la herramienta perteneciente a Simulink, tiene un comportamiento muy similar, por lo que todo lo que se haga en Grafcet será fácilmente exportable al programa.

Hay que tener en cuenta que el "lenguaje" de Stateflow no tiene las mismas reglas que Grafcet. Algunas consideraciones de Grafcet que no son necesarias en Stateflow son las siguientes:

- Entre un Estado y otro debe de haber una transición y no más de una.
- El final del proceso debe acabar en el estado cero.
- El estado 0 no debe tener acciones.

A partir de ahora van a ser expuestos y explicados los diseños en Grafcet iniciales que se usaron para posteriormente los modelos finales en Stateflow. Por motivos de claridad en el gráfico, no se incluyen las acciones asociadas a cada Estado, pero serán mencionadas en esta sección, y aclaradas finalmente en posteriores secciones.

Grafcet de Tracking

Uno de los Grafcet de funcionamiento principales. Este se encarga de enviar la señal de mover los motores (E) para que las placas sigan al Sol. En la siguiente imagen se puede ver el diseño:

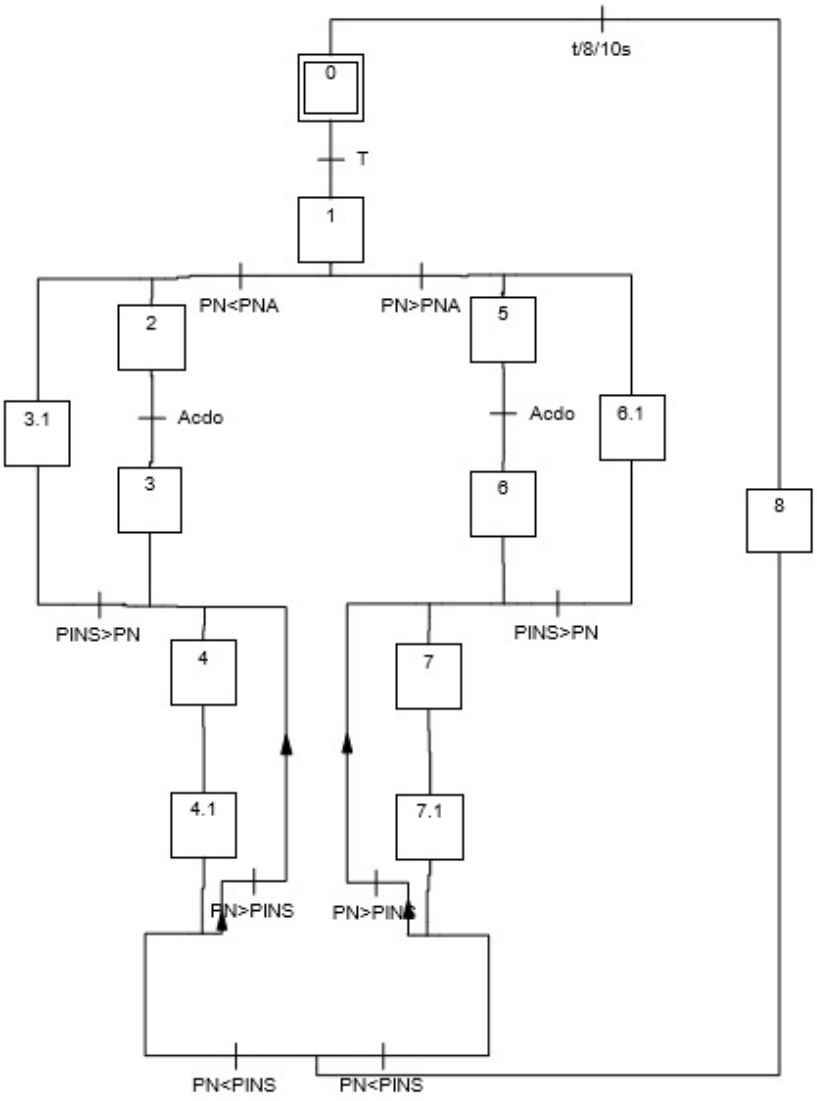

Figura 3.1: Grafcet del algoritmo de Tracking

Los condiciones en las transiciones significan lo siguiente:

- T. Da la señal para empezar a ejecutar el Tracking. Viene de otro Grafcet funcionando en paralelo.
- Acdo. Significa que los valores de las potencias están recién actualizados y preparados para ser comparados.
- PINS, PN, PNA. Potencias recogidas de las placas en distintos momentos para comprobar que aumenta.

- t/8/10s. Temporizador ligado al Estado 8 de duración 10 segundos, necesarios y tomados con seguridad para que las potencias se actualicen, ya que dependen del movimiento de los motores.

Además, varios estados tienen acciones asociadas, que son las siguientes:

- Estado 2. Reduce la variable E (posición de los motores) en 1 unidad debido a que la potencia ha disminuido $PN < PNA$ (las placas giran hacia el Este). Se pone a 1 la variable interna *Act* que dará la señal a otro Grafcet para actualizar las potencias, es decir, leer PINS.

- Estado 5. Aumenta la variable E en 1 unidad debido a que la potencia ha aumentado $PN > PNA$, por lo que las placas se mueven siguiendo al Sol. También pone *Act* a 1.

- Estados 3 y 6. Ya se han actualizado las potencias, así que la variable *Act* vuelve a valer 0.

- Estados 3.1 y 6.1. En estos estados se crea la variable interna "Iterado[1]", que pasa a valer 1 si alguno de estos estados se activa, es decir, si se itera, lo que querría decir que el punto óptimo está en la dirección de la iteración, por lo que no se dará iteración en la dirección contraria (ver más adelante en esta lista). A su vez, se actualiza PN para volver a ser comparada posteriormente.

- Estados 4 y 7. Si ha existido iteración anteriormente (Iterado=1) su función es únicamente deshacer el último movimiento de E, porque es el que ha resultado en una disminución de Potencia (si esta disminución no se da, no se sale del bucle), y así quedarse en el punto óptimo. Si Iterado=0, significa que el punto óptimo está en la dirección contraria al primer movimiento, por lo que se inicia una iteración en dirección contraria. Por tanto, estos Estados mueven E en la dirección contraria a la que se llevó a cabo en 3 y 6 volviendo a necesitar actualizar PINS (Act=1).

- Estados 4.1 y 7.1. Vuelven a poner Act=0 y esperan a que se comparen potencias con el nuevo PINS.

- Estado 8. Pone la variable *Tend* (Tracking End) a 1, para indicar que se ha terminado, y se vuelve al Estado 0.

[1]Se inicializa a 0 en el Estado 0, lo cual es ilegal en Grafcet, pero legal en Stateflow donde se ha llevado a cabo. En Grafcet habría que crear un Estado más para justo después del 0 para inicializarla

El funcionamiento entonces es el siguiente: cuando es la hora de hacer Tracking (T) se determina si la potencia actual es menor o mayor que la establecida al final del último Tracking o la inicialización. Si es mayor se intenta seguir al Sol para ver si se puede hacer crecer más. Si sigue creciendo se forma un bucle en el que las placas siguen moviéndose siguiendo más la trayectoria solar, hasta que se lee una bajada de potencia, se subsana el último movimiento, se indica el final del Tracking (Tend) y se acaba hasta la próxima vez. Si el primer movimiento de placas da lugar a una bajada, se inicia un bucle en la dirección contraria. Al salir de este bucle se envía Tend y se acaba.

Si por el contrario se detecta una bajada de potencia, las placas giran en sentido contrario a la trayectoria del Sol para ver si reencuentra el punto óptimo. De ver su potencia disminuida reculan y se acaba el Tracking. De verla aumentada se entra en bucle hasta que no lo sea y se acaba.

Grafcet de BackTracking

El segundo de los dos Grafcet de funcionamiento principales. Este es el que se encarga de cambiar la posición de ambas placas separadamente para evitar que en ciertas horas del día exista sombra, minimizándola.

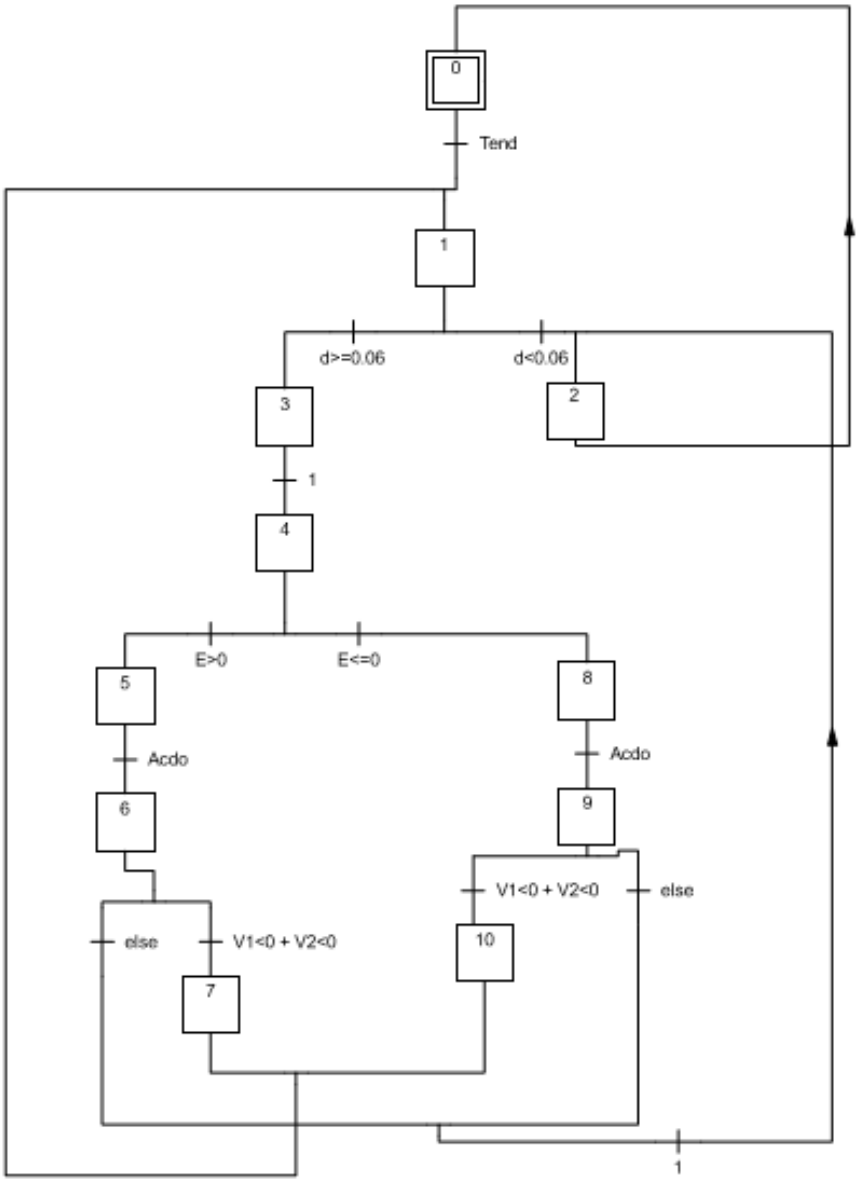

Figura 3.2: Grafcet del algoritmo de Back-Tracking

A continuación se explican las transiciones del modelo:

- Tend. Fin del tracking, da inicio al Back-Tracking de ser necesario.
- d. Variable encargada de comprobar si es necesario hacer Back-Tracking. Evalúa si la diferencia entre ambas potencias es significativa como para creer que existe sombra.

- E. Si E es negativa significa que es por la mañana. De ser positiva es por la tarde o noche. Se discrimina así para decidir cual placa debe moverse para evitar la sombra de la otra, ya que por la mañana la culpa la tiene la del Este y por la noche la del Oeste.
- Acdo. Variable que indica que las potencias se han actualizado y están listas para ser evaluadas.
- V1 y V2. Variables que evalúan el correcto comportamiento del Back-Tracking. Si ambas son positivas significa que la potencia de la placa sombreada crece y la otra decrece, resultado esperado.
- else. Algunas de las dos V1 o V2 son negativas.

Y las acciones asociadas a Estados:

- Estado 1. Se declara la variable d para ser evaluada justo después. También se hace PN=PINS para tener libre esta última para poder actualizarla y ser comparada con la nueva PN.
- Estado 2. Se pone a 1 la variable NOBT, que significa que el BackTracking ha terminado.
- Estado 5. Es por la noche así que la placa del Oeste, o placa 1, reduce su posición (E(1)) en una unidad. También se pone Act a 1.
- Estado 8. Es por la mañana así que la placa de Este, o placa 2, aumenta su posición (E(2)) en una unidad. También se pone Act a 1.
- Estados 6 y 9. La variable Acdo y Tend pasan a 0, mientras que Tend pasa a 1. Se declaran las variables V1 y V2 según sea por la mañana o por la noche (ver en Descripción del Control por diagramas de Estado).
- Estado 7 y 10. Reculan el último movimiento que se haya dado en E.

La explicación de su funcionamiento es: Cuando el Grafcet Tracking termina da la señal a este para empezar. Se declara d para comprobar si las potencias son significativamente distintas. Si no lo son se sale del Back-Tracking hasta nuevo aviso, pero si hay diferencia se entra a discriminar si es mañana o tarde. En función de esto, se moverá en bucle la placa que esté sombreando a la otra hasta que el movimiento no resulte en un aumento de potencia de la placa sombreada y/o disminución de la sombreadora, en cuyo caso se sale del Back-Tracking. El modelo hecho en Stateflow además evalúa que la potencia ganada en una placa sea superior a la perdida en la otra.

Grafcet de Sincronización y Control

Este es un Grafcet sencillo de tan solo 2 Estados, que se encarga de determinar cada uno de los ciclos de control, explicados más adelante. Quedaría como en la siguiente imagen:

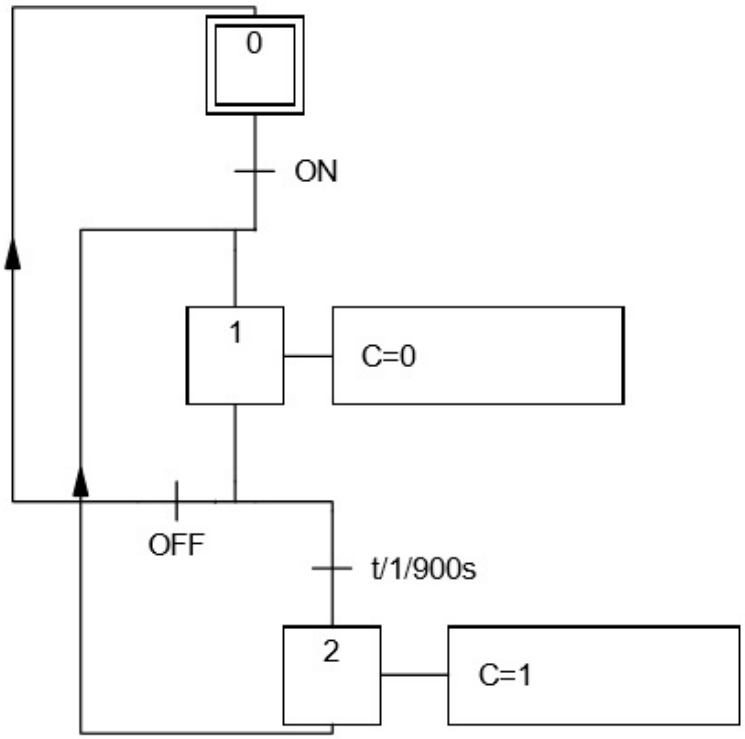

Figura 3.3: Grafcet de Sincronización y control

Cuando el botón ON de la botonera de mando del usuario es pulsado se activa y T se inicializa a 0. A su vez, un temporizador asociado al Estado 1 se activa y cuenta 900 segundos, 15 minutos, y al terminar estos se pasará al Estado 2, donde se establece a 1 la variable C: se llama al Grafcet L1, que se explicará más adelante y que se encargará de llamar a Tracking. Inmediatamente después se vuelve al Estado 1. En este, se espera de nuevo al temporizador o a que el usuario pulse el botón OFF, que hará pasar el curso de acción al Estado 0 y apagará el sistema[2].

[2]Para respetar las reglas de Grafcet se debería poner una condición en la realimentación al 0, pero el Software utilizado para construir el Grafcet no precisa de ello

Grafcet de inicialización

Es un esquema sencillo consistente en dos Estados que se encargarán de pasarle al Grafcet L1 el evento (o variable) de inicializar. Se puede visualizar en la siguiente imagen:

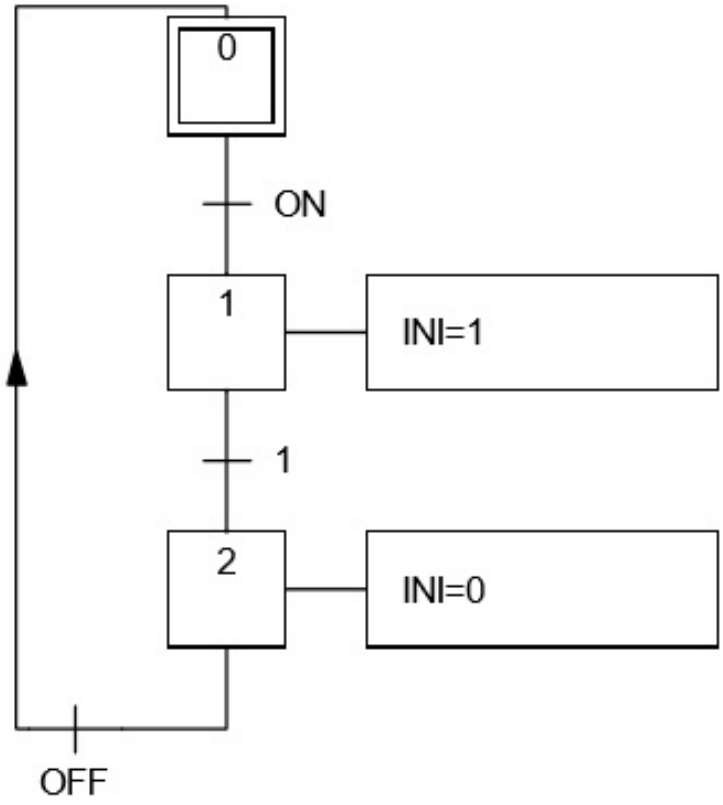

Figura 3.4: Grafcet de Inicializacion

Cuando el usuario pulsa ON se activa durante un momento la variable INI, suficiente para enviarle el evento al Grafcet L1. Cuando el usuario pulsa OFF se vuelve al Estado 0.

Grafcet L1

Este es el que se encarga de la lectura y almacenamiento de las variables PN y PNA, que son respectivamente: Potencia evaluada en el instante N

y potencia evaluada en el instante Anterior a N. También se encarga de comunicar al Grafcet de Tracking que debe comenzar.

Se puede visualizar en la siguiente imagen:

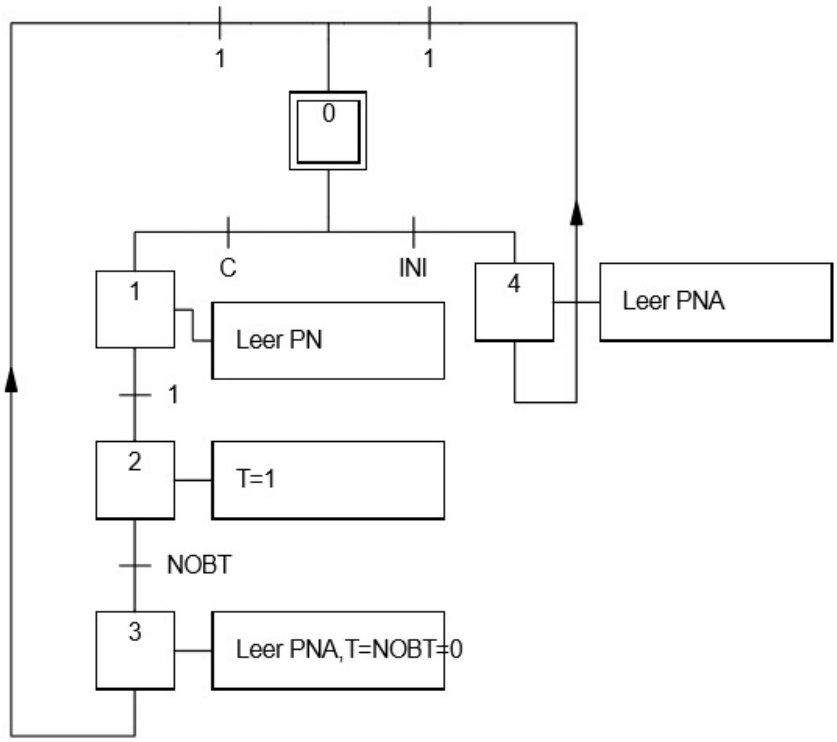

Figura 3.5: Grafcet de Lectura 1 y control

El funcionamiento es el siguiente: cuando el usuario pulsa ON la variable INI pasa a valer 1 durante un instante, lo que hace que este Grafcet lea la potencia PNA que será comparada después, y vuelve al estado inicial. Pasados 15 minutos la variable C valdrá 1 y se ejecutará el resto de la secuencia, se lee PN para compararla en el Tracking con PNA, se pone T a 1 que inicia el Tracking y cuando este y el Back-Tracking finalicen se cumplirá la condición NOBT y se resetearán NOBT y T a 0, mientras que se actualizara PNA.

Grafcet L2

Este sencillo Grafcet es el encargado de actualizar la variable PINS cada vez que se de un movimiento de las placas. Esta variable significa *potencia*

instantánea. En el Grafcet de Back-Tracking su valor se guarda en el de PN antes de actualizarla.

Se puede visualizar en la siguiente imagen:

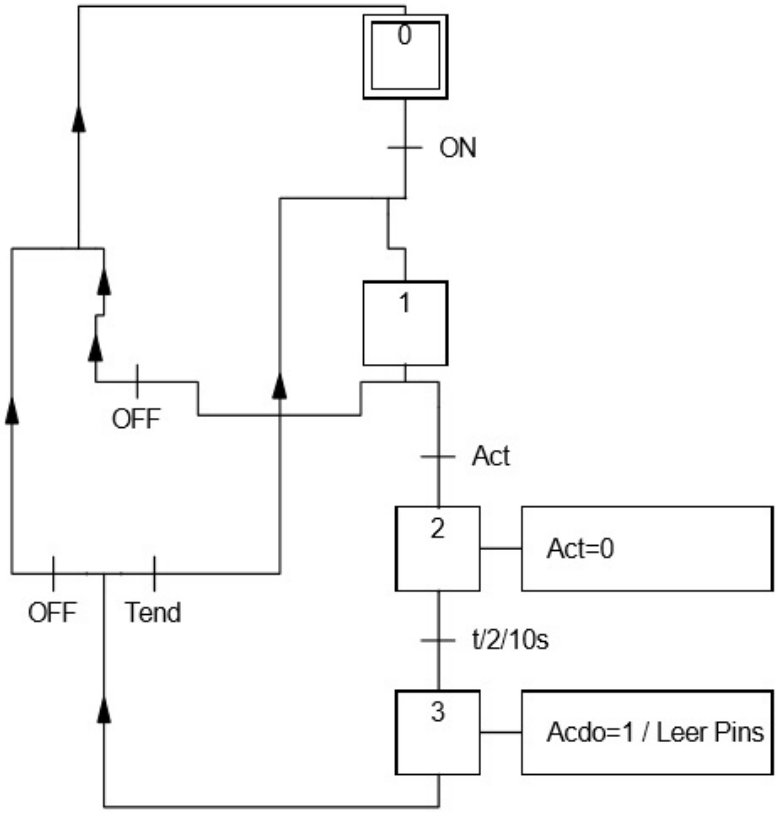

Figura 3.6: Grafcet de Lectura 2

Cuando se pulsa ON pasa al Estado 1, que espera al cumplimiento de la condición *Act* (Actualizar) para pasar al Estado 2, donde espera 10 segundos (debido a que en este momento las placas se están moviendo) y en el Estado 3 lee PINS y pone Acdo (Actualizado) a 1 para indicar que la tarea ha sido realizada. Espera al cumplimiento de la variable Tend para pasar de nuevo al Estado 1. En ambos Estados 1 y 3 se contempla pasar al Estado 0 cuando el usuario pulse OFF.
Todos los Grafcet hasta ahora descritos funcionan en paralelo, es decir, todos sus Estados 0 o iniciales se activan al mismo tiempo.

3.2. Introducción a Stateflow

Stateflow es un entorno dentro de Simulink para modelar y simular lógica de decisión combinatoria y secuencial basado en máquinas de estado y diagramas de flujo. Trabaja también con bloques enlazados entre ellos, pero de forma más parecida a Grafcet[3].

Al abrir la librería Stateflow aparecen únicamente dos bloques: *Chart* y *Truth Table*, pero en este proyecto solo ha sido utilizado el primer bloque. Este, se puede introducir en un .mdl de Simulink, y es personalizable entrando en él, haciéndole doble clic. El conjunto de bloques inter-relacionados va a ser llamado Diagrama de Estados[4] Aparecerá el entorno mencionado donde, en la barra lateral, se pueden seleccionar varios elementos. Algunos de ellos van a ser descritos aquí:

- *State*: Es el bloque básico de Stateflow y puede estar activo o inactivo. La programación se basa en interconectar estos bloques o estados. Dentro de ellos se pueden definir acciones e incluso más estados, como se verá más adelante. Pinchando en uno de ellos y arrastrando hacia otro se crea una transición.

- *History Junction*: Es un "sello" que se le puede poner a un estado que contenga sub-estados. Guarda la información del último sub-estado que ha estado activo.

- *Default Transition*: Determina el estado inicial por defecto cuando empiece la simulación. Es representado como un pequeño punto negro unido a una flecha que apunta al Estado seleccionado como inicial.

- *Connective Junction*: Mediante ellos se pueden diversificar o unificar las transiciones en un punto. Sirve para crear puntos de decisión entre caminos.

El resto de elementos que se proporcionan no van a ser utilizados para este proyecto.

3.2.1. Bloques de estado (States)

El primer paso hacia la creación de un modelo es insertar los bloques de estado o *States*. Cuando un bloque es colocado es posible escribir dentro,

[3]Grafcet es un modelo de representación gráfica basado en estados y transiciones.

[4]Dentro del bloque Chart va a existir al menos un Diagrama de Estados, pero pueden crearse Estados padre, y estos contienen dentro otro diagrama de Estados.

pero teniendo en cuenta que la primera línea es para el nombre del bloque. Es importante saber y tener en cuenta que **cada Diagrama de Estados debe tener un Estado inicial por defecto**, y esto se hace colocando un *Default Transition* en el bloque que se quiera sea el inicial.

Nombre

El nombre del bloque debe ser escrito en la primera línea del bloque, sin espacios. No es posible que el nombre de un bloque sea un número escrito con lenguaje numérico, pero sí alfabético. Además, dos Estados dentro de un mismo Diagrama de Estados no pueden ser definidos por el mismo nombre.

Acciones

Por debajo de la línea del nombre del Estado es posible escribir distintos tipos de acciones asociadas a él. Hay varios tipos de acciones y se pueden utilizar cualesquiera de ellos, que son los siguientes:

- Acciones entry: Son acciones que se ejecutan cuando el Estado que las contiene pasa de estar inactivo a estar activo. Para definirlas se escribe "en:" o "entry:" y a continuación la acción o acciones a realizar, separándose con punto y coma si son varias.

- Acciones during: Estas acciones se ejecutan siempre mientras el estado que las contenga este activo. Se definen de la misma forma que la anteriores, pero escribiendo "du:" o "during:" al principio.

- Acciones exit: Se ejecutan cuando el Estado que las contiene pasa de activo a inactivo, y en este caso el comando es "exit:" o "ex:"

- Acciones on: Estas acciones van ligadas a una condición. Ejecutan una acción si, durante el tiempo en el que Estado que la contiene está activo, se da una determinada condición. El formato es "on condición: acción". La condición puede ser, por ejemplo, un evento[5].

- Acciones bind: Sirve para atar un evento al Estado que lo contiene y sus sub-Estados, y que solo pueda ser transmitido por los mismos.

Las acciones que pueden usarse están limitadas, ya que Stateflow no carga por sí mismo todas las librerías que usa Matlab en su código M, pero para este proyecto las acciones utilizadas han sido asignaciones de valores y están

[5]Los eventos serán descritos posteriormente en este capítulo.

introducidas satisfactoriamente. No obstante, es posible introducir cualquier acción utilizando cualquier función de Matlab utilizando "Matlab Function", que es una de las herramientas ofrecidas por Stateflow para su programación, contenida en la barra lateral del mismo (donde se encuentran los Estados y demás), pero no ha sido descrita debido a que no ha sido utilizada en este proyecto.

Jerarquía

Es posible establecer una jerarquía entre estados: Establecer un Estado padre que contenga varios sub-Estados, y que de esta forma cuando el Estado padre se active, lo haga así uno (o varios, si están en paralelo) de los sub-Estados, formando un diagrama de Estados dentro de un solo estado. Para hacer esto, se coloca un Estado y se extiende, arrastrando de la esquina inferior derecha hasta hacerlo más grande. Después, se seleccionan los bloques que se quieran someter a este Estado padre y se arrastran dentro, teniendo en cuenta que los límites del Estado padre no deben ser tocados por límites de otros estados ni por transiciones: no debe quedar ninguna parte de la selección fuera.

A partir de ese momento ya se tiene un Estado padre y uno o varios sub-Estados . Hay que recordar que **uno de estos sub-Estados debe de ser definido como el inicial por defecto**.

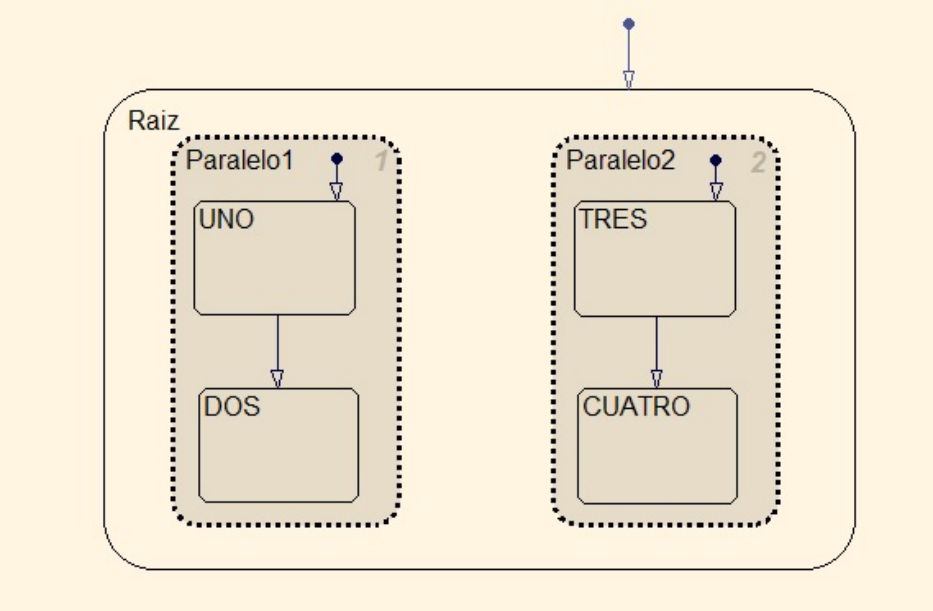

Figura 3.7: Ejemplo Stateflow: Jerarquía

En la figura 3.7 se pueden observar algunas cosas acerca de lo comentado hasta ahora: Existe un gran Estado padre llamado Raiz que contiene dos sub-Estados Paralelo1 y Paralelo2. Como su propio nombre indica, van en paralelo, se pueden reproducir al mismo tiempo (y por eso no necesitan estar interconectados ni tener definido cual es el inicial, ya que ambos lo son) y esto se puede identificar por sus delimitaciones discontinuas. Dentro de cada uno de estos Estados paralelos, hay otros dos Estados. Son un conjunto de Estados que forman un diagrama de Estados y no están en paralelo, por lo que es necesario definir cual de ellos es el inicial por defecto, tal y como está hecho, tanto en esos sub-Estados, como en el Estado padre Raiz. Existe la opción de ocultar los bloques de sub-Estados, dando click derecho sobre el padre y siguiendo la ruta $\triangleright MakeContent \triangleright Subcharted$. Esta opción hará que los sub-Estados se oculten, dando lugar a un diagrama de Estados más limpio y sencillo. Haciendo doble click sobre un Estado padre se puede ver los Estados que este tiene dentro.

Nota. *Haciendo doble click sobre un Estado padre, este se oscurece y se agrupa con los sub-Estados. Mientras esté así se consideran agrupados y se mueven juntos si son arrastrados con el ratón.*

Al simular el diagrama de Estados de la figura 3.7 se activa **Raiz**. Al hacerlo, se activan todos sus Estados dispuestos en paralelo, es decir **Paralelo1** y **Paralelo2** y, a su vez, se activan los Estados iniciales por defecto de cada uno de los paralelos: UNO y TRES. De estos dos últimos, existe una transición hasta DOS y CUATRO respectivamente, y como no hay condición en la transición, esta ocurre inmediatamente, por lo que UNO Y TRES se desactivan, y DOS y CUATRO quedarían activos para el resto de la simulación.

Descomposición exclusiva (OR) y paralela (AND)

Como se ha visto en el ejemplo anterior, es posible que exista un Estado padre cuyos sub-Estados contenidos estén en paralelo y se activen a la vez. Esto es una característica que se puede elegir de los Estados padre haciendo click derecho sobre él, seleccionando *Decomposition* y eligiendo entre descomposición exclusiva OR o paralela AND. Por defecto, Stateflow selecciona OR.

Si la elección es la descomposición exclusiva OR, deberá haber un sub-Estado inicial definido, y solo puede existir uno activo al mismo tiempo. Cuando la elección es la descomposición paralela AND, todos los sub-Estados se activarán al mismo tiempo.

3.2.2. Transiciones

Una transición en Stateflow es una línea que conecta un objeto gráfico con otro, y significa un cambio de estado del sistema. Pueden emanar y conectar con Estados y con puntos de unión (*Connective Junction*).

Una transición vacía entre dos Estados conllevará una consecución inmediata del Estado del que emana hacia el Estado destino, pero es posible introducir condiciones en estas lineas de transición, para que esta solo ocurra cuando se cumpla. Para esto, se selecciona la línea de transición y el programa permite al usuario escribir la condición. Algunas condiciones utilizadas para este proyecto son:

- *Condición lógica*: Se puede escribir un condición lógica entre corchetes, como igualdad o desigualdad. La variable a analizar por la condición lógica debe estar definida como *data* previamente.

- Evento: Se puede escribir en la transición el nombre de un evento ya definido, y se realizará esta transición cuando el evento ocurra.

- Temporizador: mediante el comando **after(T,sec)** se puede hacer que la transición ocurra cuando pasan T segundos desde que se activó el Estado del que emana la transición.

También es posible introducir una acción en la transición, disponiéndola entre llaves { }. Cuando es hecho de esta manera, esta acción se llevará a cabo cuando se cumpla la condición asociada a la transición (ocurriendo inmediatamente si no hay condición alguna) sin tener en cuenta que se active el siguiente Estado. Si la acción asociada entre llaves es precedida por una barra "\", entonces la acción solo se llevará a cabo cuando se haya desactivado el Estado del que emana la acción y activado un Estado receptor.

Puntos de unión

Hay varias formas de usar los puntos de unión en Stateflow. Es posible que la salida de un Estado desemboque en uno de estos puntos, y del mismo partan varias salidas con condiciones asociadas y que lleguen a diferentes Estados. También es posible que haya varios Estados que desemboquen en un solo punto de unión, y del cual salgan otras transiciones hacia otros Estados. Las entradas al punto de unión no están numeradas, pero las salidas sí que se numeran automáticamente, debido a que solo una de estas transiciones puede ocurrir a la vez, o habría un conflicto en la simulación.

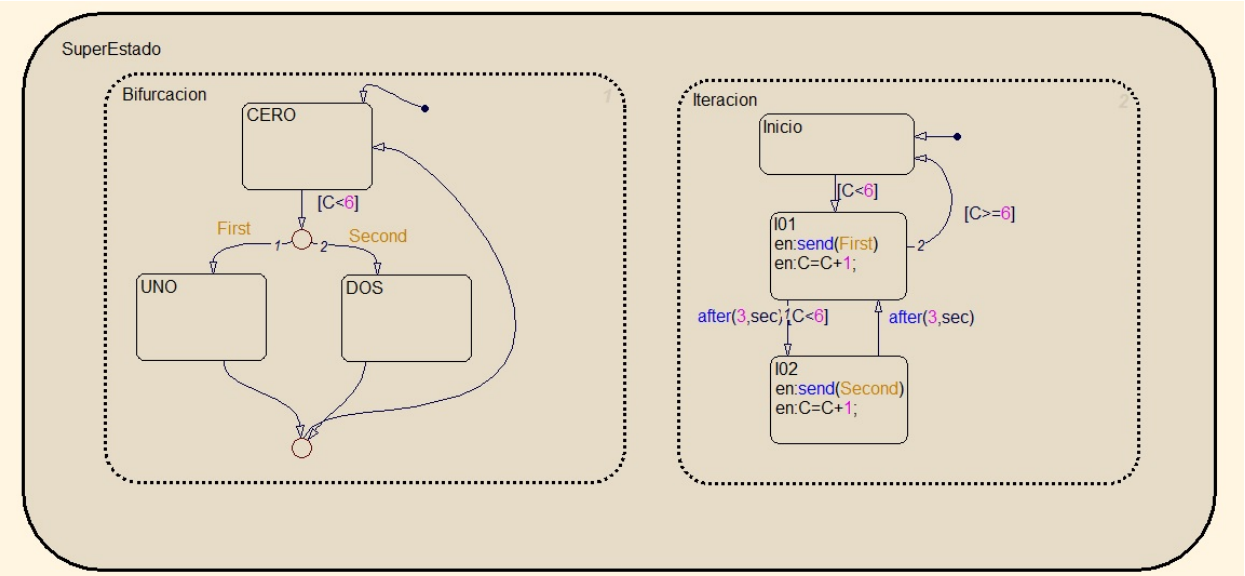

Figura 3.8: Ejemplo Stateflow: Eventos, acciones, transiciones

En la figura 3.8 se puede observar un ejemplo más elaborado de Stateflow. Hay un super-Estado que contiene otros dos funcionando a la vez, en paralelo (obsérvese que estos dos aparecen delimitados con línea discontinua). Estos a su vez contienen más sub-Estados dentro. Lo primero hay que tener en cuenta que se han definido los eventos *First* y *Second*, y la variable *C*, cuyo valor inicial es 0. El super-Estado "Bifurcacion", mientras la variable C tenga un valor menor a 6, espera a la ocurrencia de First o Second y según cual ocurra se activa el Estado "UNO" u "DOS". El encargado de transmitir los eventos necesarios para que esto ocurra es el super-Estado "Iteración".

Este super-Estado solo abandona el Estado "Inicio" si $C < 6$, y mientras tanto ocurre una iteración. Cuando se activa "I01", este estado envía el evento First necesario en el anterior super-Estado, incrementa en 1 la variable C, y además se inicia el contador de 3 segundos necesario para pasar al siguiente Estado, el "I02". En este, ocurre algo similar: se envia Second y espera durante 3 segundo para volver al Estado anterior. Cuando C haya alcanzado el valor 6, la iteración acaba.

Definir datos y eventos

Hasta ahora se ha hablado mucho de eventos, y del papel que estos tienen en un diagrama de Estados de Stateflow, pero no se ha determinado como estos se definen y como se pueden caracterizar. Para añadir un evento que se

pueda leer en un diagrama es necesario buscar en la barra superior el botón *Add*, del que saldrá un menú desplegable. En él, se puede encontrar Event y Data, con lo cuales se definen eventos y otras variables, respectivamente. Al pulsar Event, saldrá otro menú desplegable con las siguientes opciones:

- *Local...* : Sirve para definir un evento que solo será utilizado en el ámbito del diagrama de Estados en el que se esté trabajando. No se reflejará de ninguna manera en el exterior de este.

- *Input from Simulink...* : Define un evento que puede leer el diagrama de Estados en el que se está trabajando, pero procede de fuera del diagrama, de Simulink. En el bloque *Chart* en el entorno Simulink aparecerá una entrada en la parte superior del bloque. La lectura de tal evento puede ocurrir por un escalón ascendente en la señal de entrada, por una descendente o por cualquiera de ellas (a elección del usuario): *Rising Edge, Falling Edge, Either Edge*. También existe la opción *Function Call()*, pero no va a ser comentada aquí.

- *Output to Simulink...* : Define un evento que será enviado desde el diagrama de Estados hacia fuera del *Chart*. Aparecerá una salida en el bloque en el entorno de Simulink, que podrá ser conectada a un subsistema que tenga la capacidad de leer eventos de Stateflow[6] y activarse con el mismo.

A partir de que un evento sea definido, este puede ser utilizado ya en el diagrama de Estados, y su nombre aparecerá de un tono amarillo cuando sea mencionado, tal y como se puede observar en la figura 3.8.
Para transmitir eventos se usa el comando send(Evento,Estado), donde "Evento" es el nombre del evento que se quiere transmitir y Estado es el estado que se quiere que sea el receptor de tal evento. Esto se llamaría **transmisión directa**, ya que se indica el nombre del receptor, pero en el caso de la figura 3.8 se ha omitido, por lo tanto es una **transmisión indirecta**. Es mucho más recomendable usar la directa, pero en el ejemplo utilizado, al ser pequeño, la indirecta no resulta inadecuada. Se puede introducir el comando de envío de eventos en un Estado, pero también puede ser incluido dentro de una transición, siempre que se escriba entre llaves { }.

Volviendo al menú *Add*, se puede observar la aparición de Data en el menú desplegable. Al seleccionar esa opción, se despliega otro menú donde

[6]Se puede añadir un bloque *Enable* a un subsistema ya realizado, o elegir directamente *Enabled subsystem* para que un sub-sistema de Simulink tenga entrada para eventos

se pueden ver varias opciones, pero solo serán comentadas las 3 primeras, que además son similares a las opciones que ofrecía el añadir eventos: Local, Input from Simulink y Output to Simulink. Una vez seleccionado el ámbito deseado para la variable introducida, se pueden seleccionar otras características, como los valores máximos y mínimos que puede alcanzar o el valor inicial. En el ejemplo de la figura 3.8 la variable C ha sido inicializada a 0 en su definición.

3.3. Descripción del Control por diagramas de Estado

En esta sección va a ser descrito el corazón del control. Como ha sido mencionado, esta diseñado y realizado mediante Estados, en Stateflow, y utiliza la mayoría de los elementos explicados en la sección anterior. Desde el punto de vista de Simulink, este sistema de control es un *Chart* que recibe dos variables del modelo descrito en el Capítulo 2, y manda al mismo una variable, concretamente a los motores. Posee también entrada de eventos en la parte superior. En el entorno de Stateflow, dentro del Chart, hay un super-Estado llamado **Raiz** que contiene 6 **sub-Estados que actúan en paralelo**, teniendo cada uno de ellos una función específica en el sistema de control. En las siguientes sub-secciones van a ser descritos cada uno de los elementos que componen el super-Estado *Raíz*.

3.3.1. Inicialización

Este sub-Estado contiene otros dos estados, llamados ON y OFF, siendo este último el elegido por defecto mediante el indicador default transition. Para el funcionamiento correcto de este bloque se introducen 3 eventos. Dos de ellos son entradas provenientes de Simulink: los eventos *On* y *Off*. Estos eventos son los que hacen la transición entre los únicos dos Estados existentes. Cuando ocurre el evento *On*, se activa el Estado ON si no se esta ya en él, y si se esta y ocurre *Off*, hay una transición hasta el Estado OFF.

En el Estado OFF se quiere que se inicialice la variable E, que es la que va a controlar el ángulo que tengan los motores[7]. El valor inicial de este vendrá de fuera del Chart: hay que introducir una variable proveniente de Simulink y otra que vaya hacia el mismo. Se declaran entonces E_{ini} y E, respectivamente.

[7]Se inicializa en el Estado OFF y no en otro distinto porque cuando el Chart despierte, que lo hace manualmente, inicialice y este preparado antes de pulsar *On*

Ya declaradas estas, se introduce la inicialización en OFF con la siguiente línea de comando:

```
en: E=Eini
```

y así la inicialización está completa. Cuando se entra en ON es necesario comunicar al resto del Chart que ya se ha inicializado y que debe empezar a ejecutarse. Se declara un evento local llamado *Inicia* y se escribe la línea

```
en: send(Inicia,L1)
```

que envía la señal al Estado L1 para empezar a funcionar.

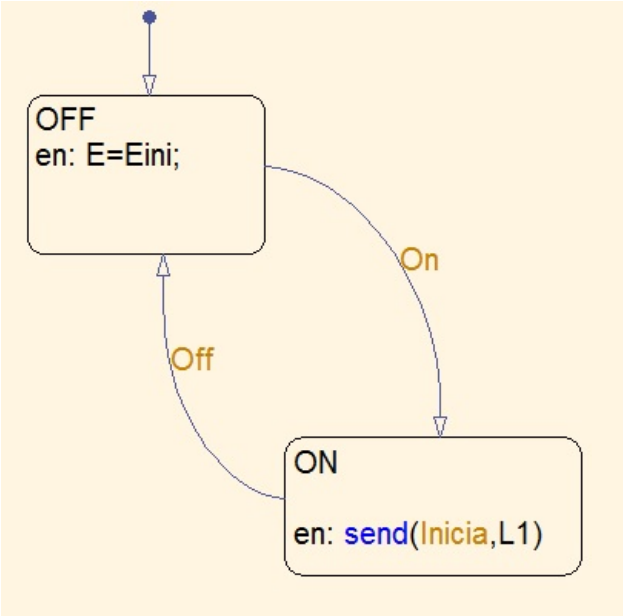

Figura 3.9: Interior del Estado Inicializacion

3.3.2. Sincronización y control SYC

Este Estado es el encargado de mandar la señal periódicamente de que se aplique el algoritmo de Tracking. Va a tener 3 Estados: ON OFF y Checkeo. El comportamiento entre los Estados ON y OFF es similar al del Estado anterior, con una única diferencia: si no ocurre el evento *Off* antes de pasado cierto tiempo, ocurre una transición momentánea al Estado Checkeo, que envía la señal mencionada, y se vuelve otra vez a ON. Para que la transición a Checkeo ocurra cada cierto tiempo, hay que escribir una condición en la transición:

```
after(900 ,sec)
```

En este Estado la transición de vuelta a ON no tiene condición, ocurre instantáneamente, por lo que la transición de arriba ocurrirá una vez cada 900 segundos, o mejor dicho, 15 minutos. Al ocurrir y entrar en Checkeo, es necesario enviar la señal a otro Estado para que continue el seguimiento. Esto se hace definiendo el evento local *Check*, que ocurrirá cada 15 minutos, y será enviado mediante el comando escrito en el Estado Checkeo:

```
en: send(Check,L1)
```

y será recibido por L1, que se describirá a continuación.

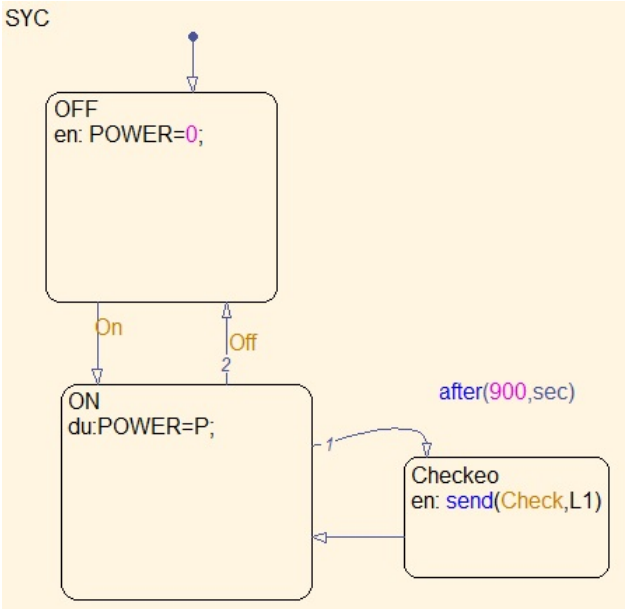

Figura 3.10: Sincronización y control SYC

Además, mientras el Estado OFF esté activo, la potencia de salida del Chart será 0. En cualquier otro caso será igual a la potencia original que resulta del modelo.

3.3.3. Estado de lectura de datos L1

Este Estado es el que se encarga de leer los datos de potencia en la entrada del Chart en diferentes momentos para, en otro Estado, comparar su evolución, por lo que recibirá eventos de otros Estados. Se definirán 3 variables locales relacionadas con la potencia:

- P_N (PN): Potencia evaluada en el comienzo del tracking, que será comparada con una potencia anterior.

- P_{N-1} (PNA): Potencia resultante de la inicialización o fin del tracking, con la que será comparada PN para seguir la evolución.

- P_{INS} (PINS): Potencia evaluada en mitad del tracking o back-tracking para evaluar la mejora o no de la potencia con las operaciones que estos dos procesos lleven a cabo. Esta variable será usada más adelante, no en este Estado.

Estas variables serán igualadas a la entrada P cada vez que sean definidas, que será como acciones de entrada en varios Estados que podrán verse a partir de ahora.

Serán utilizados 3 eventos: *Inicia, Check y NOBT*. Este último aún no ha sido definido, pero como resumen introductorio se puede decir que es recibido cuando el Tracking y el Back-Tracking ya han terminado para esa iteración[8].

Se comienza en el estado por defecto, que al llegarle el evento Inicia pasará momentáneamente al Estado EstablecerPNA y volverá inmediatamente al Estado por defecto. En él, como el propio nombre indica, se establece la variable PNA, para compararla posteriormente con PN. Cuando se recibe *Check*, se pasa al Estado LeerPN, donde se colocan dos acciones de entrada: una para establecer PN y otra para enviar el evento local Track al Estado Tracking, que se definirá posteriormente. Para esto hay, por supuesto, que definir el evento local mencionado, ya que de lo contrario no funcionaría[9].

Después de la espera de 1 segundo, se pasa al Estado Track (no confundir con el evento del mismo nombre), donde se espera a que ocurra el evento *NOBT* para pasar al Estado CicloOFF, donde se vuelve a establecer PNA y se pasa instantáneamente al Estado por defecto.

[8]Una iteración es la que ocurre cada 15 minutos.

[9]Cuando un evento está correctamente definido, siempre que es escrito dentro del entorno Stateflow es visualizado de un color amarillo anaranjado.

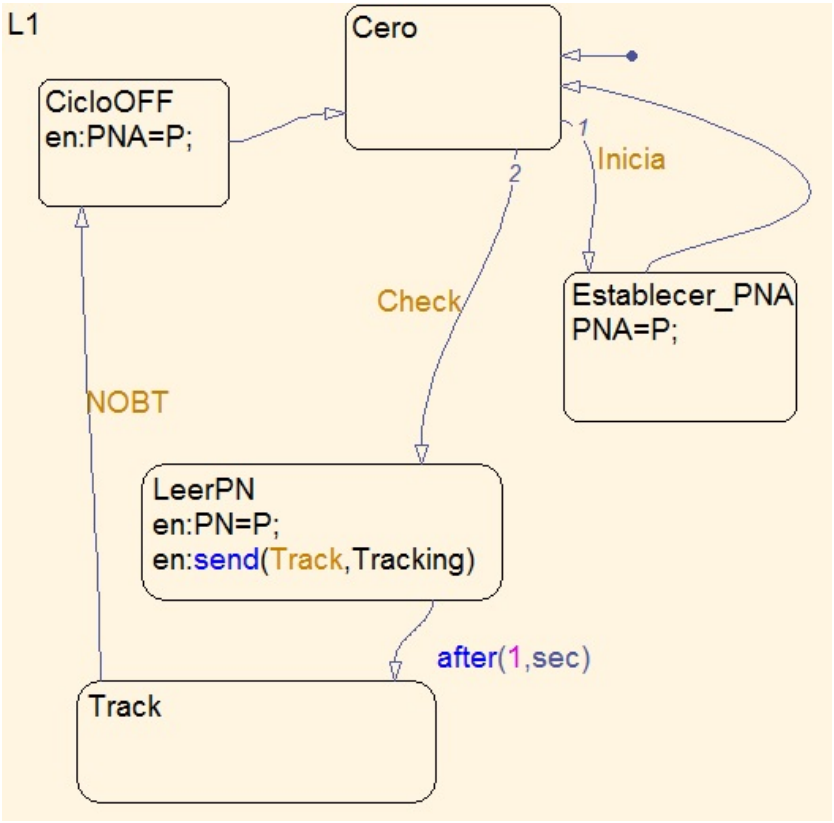

Figura 3.11: Estado de lectura de datos L1

3.3.4. Estado de lectura de datos L2

Mientras se aplican los algoritmos de Tracking y Back-Tracking la potencia cambia, ya que se modifican los ángulos de orientación de las placas fotovoltaicas. Surge la necesidad de ir comparando la potencia resultante de un movimiento de la placa con la potencia que esta tenía antes de ese movimiento. Para ello será utilizada una variable definida anteriormente: **PINS**. Cada vez que se realice un movimiento en los Estados Tracking o BT (Back-Tracking) es necesario evaluar la nueva potencia. Se define el evento *Act* (Actualizar) como la señal de que se ha de actualizar después de un movimiento. A su vez, se define el evento *Acdo* (Actualizado) como la señal de que ya se ha definido esta potencia instantánea PINS para que los algoritmos puedan seguir ejecutándose.

Cuando ocurre el evento *Act* proveniente de otros Estados, se pasa del Estado por defecto L20 al Estado vacío L21. Está vacío porque representa

un estado de espera, debido a que los motores tienen que mover a las placas y estas estabilizar su potencia, algo que se requiere antes de ser leída. Para esto, hacia el siguiente Estado L22 se coloca una transición con *after* y un valor de 10 segundos. En L22 se establece PINS y se envía el evento *Acdo* tanto al Estado Tracking como al BT. Tanto L21 como L22 se van a ver relacionados con un bucle realizado por el Estado Tracking, por lo que se redirige desde L22 hasta L21 de nuevo si ocurre el evento *Act* mientras se está en el primero.

Para volver al Estado por defecto desde el L22, se define otro evento llamado *Tend* (Tracking End) emitido desde otros Estados hasta este, de forma que se vuelva al L20, a la espera del siguiente movimiento de los motores.

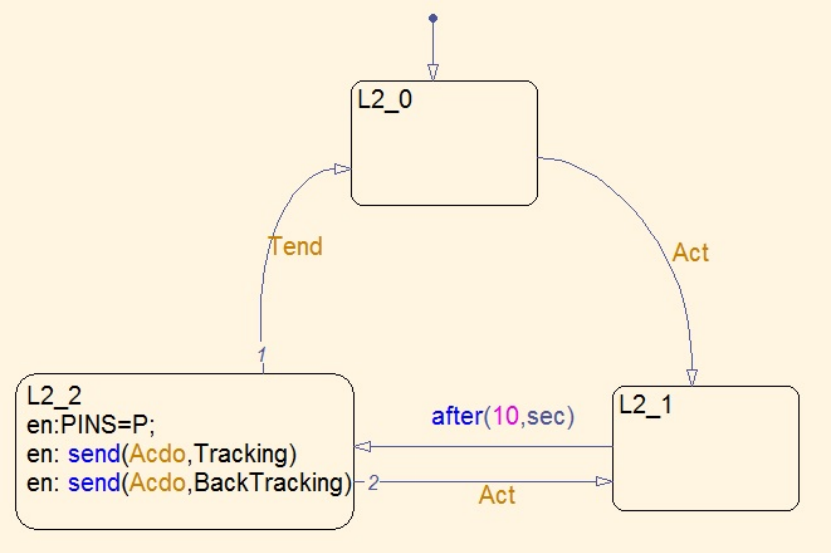

Figura 3.12: Estado de lectura de datos L2

3.3.5. Tracking

Este es el Estado en el que queda recogido el algoritmo de seguimiento solar utilizado. Este algoritmo se encarga de modificar la variable de salida E, que es la importante, ya que es la que controla los motores. La variable E solo puede adoptar valores enteros de valor absoluto menor o igual que 18, es decir:

$$E \in [-18, 18]$$

Una adopción de un valor de 18 significar una orientación perfecta al Oeste y el -18 una orientación al Este, mientras que un valor de 0 pondría a la

placa orientada de forma perpendicular al suelo. En Tracking se van a tomar acciones de seguimiento y/o retracción en las cuales se incrementará o decrementará E en valores enteros mediante los comandos respectivos:

$$E = E + 1;$$

$$E = E - 1;$$

El Estado por defecto aquí se llama Reposo. Cuando ocurre el evento *Track* enviado por L1 se pasa a un Connective Junction, una unión de conexiones, de donde salen dos ramas de decisión lógica:

- Rama 1 `[PN[1]>PNA[1]]`. Ocurre si la potencia actual PN es mayor que la potencia de la iteración anterior PNA[10] [11]. Esto significa que el seguimiento está dando un aumento de la potencia, por lo que en el siguiente Estado se forzará un poco el seguimiento para comprobar si eso aumentaría la potencia recolectada. De ser así, se entra en bucle hasta que se observe que la potencia no ha aumentado, sino que ha disminuido. Esto puede pasar incluso tras el primer paso de la iteración. Como debido a esta iteración, el último paso por definición dará lugar a una disminución de potencia, el siguiente sub-Estado realizará una retracción en el seguimiento y se volverá a Reposo.
  Si no ha habido iteración significa que el punto óptimo no está en la dirección en la que giraron las placas en primera instancia. La variable *Iterado*[12] vale 1 si se ha iterado, siendo 0 en caso contrario. Cuando es 0 esta variable, se puede iniciar una iteración en el sentido contrario al que se iteró en primera instancia de forma similar, y con las mismas condiciones de transición, como se puede ver en la imagen 3.13.

- Rama 2 `[PN[1]<=PNA[1]]`. Significa que el seguimiento ha dado lugar por algún motivo a una disminución de potencia con respecto a la anterior iteración, por lo que es necesaria una retracción del seguimiento. Si esta retracción da lugar a un aumento en la potencia, se entrará en bucle hasta que deje de dar lugar a tal cosa, llevando a cabo justo después una corrección en el seguimiento en el sentido contrario, de forma

[10] Ya que en este punto ambas potencias contenidas en ambos vectores de potencias PN y PNA son iguales utilizar la indexada en 0 o 1 da el mismo resultado.

[11] En lenguaje C++ el primer elemento de un vector está indexado con el número 0, pero en lenguaje M el primero está indexado en 1. Sin embargo Stateflow adopta la indexación de C++.

[12] Esta variable interna controlará si se ha iterado o no en primera instancia, lo que es de mucho valor para localizar en que dirección se encuentra el punto óptimo de orientación.

similar pero inversa a lo llevado a cabo en la rama 1.
Si no se ha dado la iteración, se iniciará una segunda en sentido contrario, tal como pasó en la primera rama.

En cada entrada a cada uno de los bucles mencionados se envía a L2 el evento *Act* con la conocida línea de comando

```
send(Act,L2)
```

y no se continua con el bucle hasta que L2 envía de vuelta el evento *Acdo*.

Cada rama desemboca en el Estado Reposo, donde se envía el evento *Tend* a L2 para que este también pase a su Estado por defecto hasta nuevo aviso. En estas transiciones hasta reposo se colocan dos elementos:

- La condición de transición `after(10,sec)`, ya que justo antes acaba de darse un movimiento del motor y antes de que ocurra otra lectura es recomendable que la potencia se estabilice.

- El envío de evento `send(Act,BackTracking)` dando por finalizado el Tracking en esa iteración y dando paso al Back-Tracking.

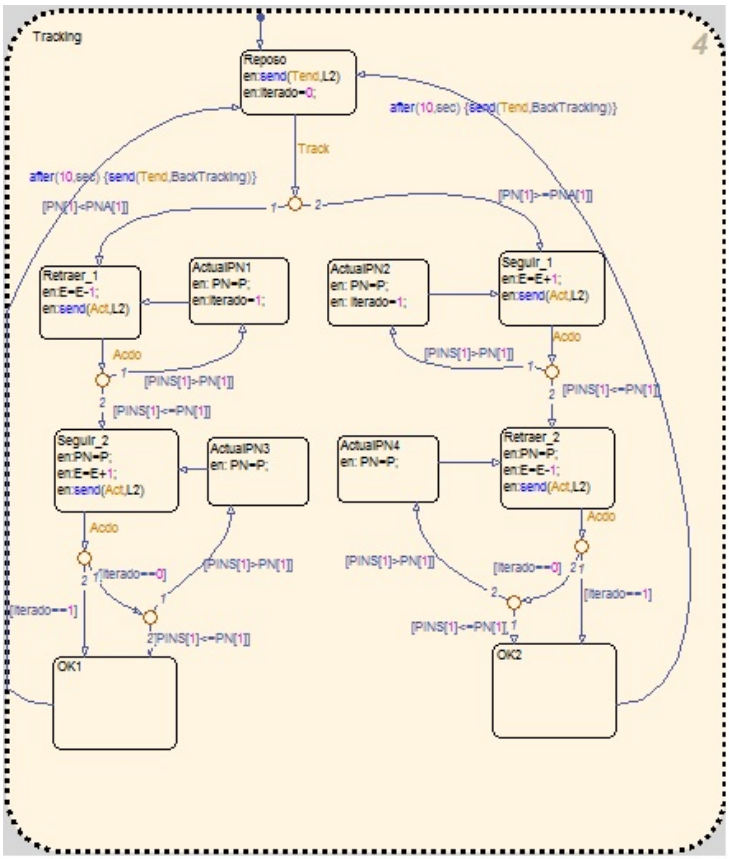

Figura 3.13: Algoritmo de Tracking en Stateflow

3.3.6. BackTracking

Hasta el momento, la variable que contiene la posición del motor y orientación de las placas E, que es un vector de dos elementos, era modificada sumando o restando un número entero a ambos elementos. En este Estado se discrimina qué elemento es modificado, con objeto de evitar la sombra y maximizar la potencia en función del momento del día. En los momentos en los que el Sol está bajo[13] una de las placas es sombreada. Si es por la mañana, la placa afectada será la del Oeste, o placa 1 según la nomenclatura de este proyecto. Por la tarde, se sombre la placa 2, o placa del Este.
Antes de explicar el funcionamiento de este Estado es necesario introducir de tres variables introducidas específicamente para este bloque. Las tres son variables locales y son las siguientes:

[13]Se considera que el Sol está bajo cuando el seguimiento solar con unos parámetros geómetricos determinados conlleva que una de las placas sea sombreada por la otra

- *d*: Es un parámetro introducido para comprobar si existe sombreado entre las placas teniendo en cuenta la potencia que llega de cada una de ellas. Considerando que las fluctuaciones pequeñas de potencia debidas a diversos factores podrían disparar el Back-Tracking sin necesidad, se ha diseñado el parámetro *d* de forma que solo se dispare cuando la diferencia entre ambas potencias supere cierto nivel. Se define entonces esta variable como:

  ```
  d=(PINS[0]-PINS[1])/PINS[0]
  ```

  y discriminará entre realizar o no el algoritmo según sea igual o mayor (sí se lleva a cabo) o menor (no lo hace)

- *V1*: Cuando la placa que sombrea a la otra varía su posición óptima para no hacerlo, esta baja su potencia, ya que los rayos del Sol no le llegarían perfectamente perpendiculares. Después de que se haga este cambio de orientación se establece esta variable para comprobar si la disminución de potencia ha ocurrido realmente como era esperada. Se define de la siguiente forma:

  ```
  V1=PN[i]-PINS[i]
  ```

  teniendo i un valor de 1 si es por la mañana y un valor de 0 por la noche, identificándose con la placa que sombrea. Se comprueba que el Back-Tracking funciona si este número es positivo.

- *V2*: Cuando la placa que sombrea a la otra varía su posición óptima para no hacerlo, la placa sombreada reduce su sombra y a su vez aumenta su potencia. Después de que se haga este cambio de orientación se establece esta variable para comprobar si el aumento de potencia ha ocurrido realmente como era esperado. Se define de la siguiente forma:

  ```
  V2=PINS[i]-PN[i]
  ```

  tomando i un valor de 0 si es por la mañana y un valor de 1 por la noche, identificándose con la placa sombreada. Para que se ejecute el algoritmo de Back-Tracking ambas variables *V1* y *V2* deberán ser mayores que 0, y el parámetro *d* mayor que 0.06.

El funcionamiento es el siguiente: Cuando ocurre el evento Tend, se pasa del Estado por defecto BT0 a BT1 donde se establece el parámetro d. Si las potencias no son distintas no es necesario hacer Back-Tracking, por lo que si el parámetro d es menor a 0.06 se pasaría al Estado NOBT, que envía el

evento *NOBT* al Estado L1 para acabar el proceso y esperar a la siguiente iteración. Sin embargo, si d resulta suficientemente grande se ejecutará el algoritmo.

Si es por la mañana, la placa 2 incrementa su seguimiento E_2 en 1 punto, o 5 grados, y se envía el evento *Act* a L2 para actualizar PINS. Una vez actualizado se establecen las variables V1 y V2. Por la tarde la diferencia es que la placa 1 retrae su seguimiento y que V1 y V2 se calculan de forma distinta, tal y como se explicó anteriormente. Ambas ramas de mañana y tarde desembocan en un punto de decisión donde se evalúan V1 y V2 para comprobar que el algoritmo funciona correctamente. Posteriormente se evalúan otra vez estas variables en valor absoluto para comprobar que la potencia que sube una de las placas es mayor que la potencia que pierde la otra placa para ello. Tanto si esto falla como si alguna de las dos es negativa se rompe el proceso, se deshacen los cambios hechos y se activa el Estado NOBT, que envía el evento de su mismo nombre y se acaba el proceso a la espera de la siguiente iteración.

3.3.7. Comunicación con Simulink

El bloque Chart de Stateflow utilizado aquí se vale de variables y eventos para comunicarse con el exterior.

Variables de entrada

Una de las variables es E_{ini} que consiste en el ángulo inicial que tomarán ambas placas en el momento en el que se despierte el Chart, inicializando la variables de salida E al valor que tenga. Se calcula en una función de Matlab externa.
La otra variables es un vector que contiene dos variables: las potencias de ambas placas. El Chart usa este vector varias veces para actualizar sus variables internas PN, PNA y PINS.

Variables de salida

La variable de salida relacionada con la orientación es E. Esta variable recoge los dos valores de orientación de ambas placas, siendo calculada dentro del Chart de control. Es conducida a la entrada de los motores como valor deseado, y puede variar de -18 a +18 pudiendo adoptar únicamente números enteros.

Cuadro 3.1: Tabla de eventos

Objeto	**Ámbito**	**Explicación**
Awake	Entrada	Activa el Chart
On	Entrada	Activa el control
Off	Entrada	Desactiva el control, aunque el Chart sigue activo
Inicia	Local	Da la orden de inicializar la posición de las placas
Check	Local	Hace que se lean datos de potencia y controla el evento Track
Track	Local	Da la señal para llevar a cabo el algoritmo de seguimiento o Tracking
Act	Local	Indica que se actualicen las variables de potencia para su comparació
Acdo	Local	Indica que las variables que se querían actualizar ya lo han hecho
Tend	Local	Se encarga de anunciar que el algoritmo de Tracking ha terminado
NOBT	Local	Anuncia que el algoritmo de Back-Tracking ya no es necesario

La otra variable de salida es la potencia. La potencia resultante de ambas placas se podría tomar directamente de la salida del modelo, pero pasando por el Chart se asegura el hecho de que es igual a 0 mientras el usuario no permita la recolección de potencia mediante una interfaz gráfica.

Eventos de entrada

Con el fin de que un usuario pueda controlar la activación o desactivación del Chart de control se ha introducido una interfaz gráfica. Esta se llama GUIDE y está incluida en las herramientas que ofrece Matlab. Está herramienta muestra al usuario una ventana con distintas funcionalidades, como pulsadores u otros tipos de botones, que pueden ser personalizables[14].

El Chart de control tiene 3 eventos de entrada: AWAKE, ON y OFF y los tres van a ser introducidos vía GUIDE. El primero despierta el Chart, mientras que los otros dos tienen un efecto de encendido y apagado sobre el mismo. Mientras la máquina esté en ON, la potencia de salida del Chart seguirá a su entrada, siendo igual a 0 de no ser así.

[14]La utilización en este proyecto de la herramienta GUIDE será explicada en el ANEXO

Cuadro 3.2: Tabla de datos

Objeto	**Ámbito**	**Explicación**
Eini	Entrada	Posición por defecto de los motores al inicio
E	Salida	Posición de los motores y placas
P	Entrada	Potencia que proviene de las placas
PN	Local	Potencia en el intervalo actual
PNA	Local	Potencia en el intervalo anterior
PINS	Local	Potencia instantánea en el intervalo actual
POWER	Salida	Salida de potencia: sigue a la entrada P
Iterado	Local	Indicador interno para hallar el punto óptimo
d	Local	Valora la necesidad o no de Back-Tracking
V1	Local	Valora el comportamiento del Back-Tracking
V2	Local	Valora el comportamiento del Back-Tracking

Capítulo 4

Resultados

En este capítulo se mostrarán los resultados de simulaciones hechas a diferentes horas del día, con objeto de evaluar su comportamiento. Serán expuestas gráficas de la evolución de la potencia junto a la evolución de la posición de los motores (u orientación de las placas) con respecto del tiempo en horas en las que el Sol difícilmente proyectaría sombra de una placa hacia otra, mientras que otras gráficas hechas más tarde mostrarán la bifurcación en las potencias que esto supone.

Por motivos de limitaciones del hardware, las simulaciones no superaran una duración de 2 horas.

4.1. Análisis del punto crítico de Back-Tracking

Como se ha estado comentando durante toda la memoria, existe un momento en el día a partir del cual una placa sombrea a la otra. Por la mañana empiezan sombreándose hasta ese momento, en el que dejan de hacerlo y continúan así hasta la tarde, en la que existen el momento en el que empiezan a sombrearse de nuevo. Esto aparece calculado en el bloque Sombra del modelo de Simulink y la fórmula es sencilla:

$$\alpha_{critico} = \arccos\left(\frac{2b}{L}\right)$$

donde b es la dimensión longitudinal de la placa, y L es la distancia entre ellas. En este modelo se ha seleccionado una placa con una dimensión $b = 1m$ y una distancia $L = 4m$. Para estos datos el ángulo crítico se da para $E = \pm 12$ lo que equivale a 60º medidos desde el eje vertical, como se ha estado haciendo en este proyecto. No obstante, introduciendo la formula de $\alpha_{critico}$ en Matlab,

se puede hacer un pequeño y sencillo análisis de sensibilidad que puede ser de utilidad a la hora de exportar este proyecto a otras condiciones geométricas.

Tomando $b = 1$ como parámetro, se varía L desde 1 hasta 10 metros para ver cual sería en ángulo crítico y su evolución.

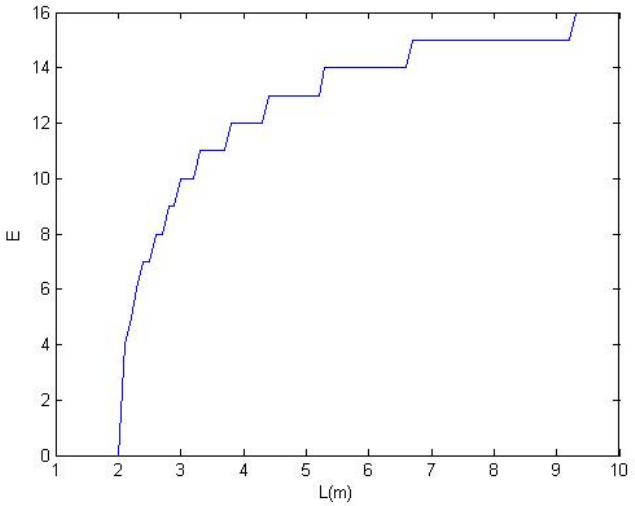

Figura 4.1: Análisis de sensibilidad de $\alpha_{critico}$ con L

Para L=4, E da un valor de 12, como se ha mencionado. Viendo el análisis, también vendría bien tomar un valor de L de 5 metros, pero se ha seguido optando por el actual debido a dos razones:

- Optimización de espacio.
- A pesar del seguimiento, la potencia cae cuando el Sol se aleja del Cénit. Cuanto mayor sea la latitud, pierde sentido separar las placas más de la cuenta, porque las horas que se salvan del sombreado dan considerablemente menos potencia.

A pesar de estas razones, la latitud de Cartagena y la simpleza del proyecto (solamente hay dos placas) permitirían perfectamente una mayor separación.

El otro análisis de sensibilidad sería manteniendo la separación $L = 4m$ constante y variar b. Se ha variado desde 0.5 hasta 2 metros. El resultado es el siguiente:

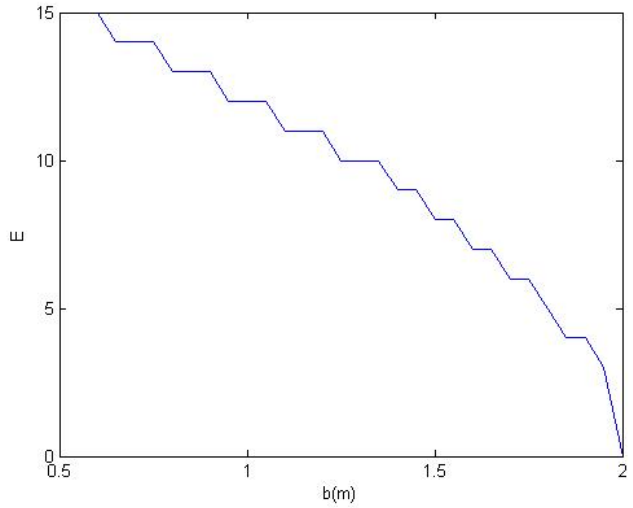

Figura 4.2: Análisis de sensibilidad de $\alpha_{critico}$ con L

Como se puede ver, para b=1, E tiene un valor de 12, como fue ya calculado.

4.2. Simulaciones de Tracking

Estas simulaciones están hechas a horas en las que la sombra proyectada por una de las placas no alcanza a sombrear en absoluto a la otra placa, es decir, para valores de E comprendidos entre -12 y 12. Para esta modalidad es interesante observar las siguientes variables:

- Potencia: Solo una de las dos, puesto que son iguales.
- Orientación de las placas o posición del motor E. También son iguales en esta modalidad.
- Temperatura de las placas o temperatura de célula.
- Insolación recibida: que variará con E, la temperatura y los ángulos solares.

La potencia y la orientación E han sido representadas en la misma gráfica en todas la simulaciones, pues da una mejor visualización de la evolución de ambas. La potencia va a ser siempre expresada en vatios (W). La variable E va a ser expresada en radianes pero con una pequeña consideración: cuando

se grafica junto a la potencia, como es el caso ahora pero no lo será más adelante, se multiplica por 100 para una mejor visualización conjunta.

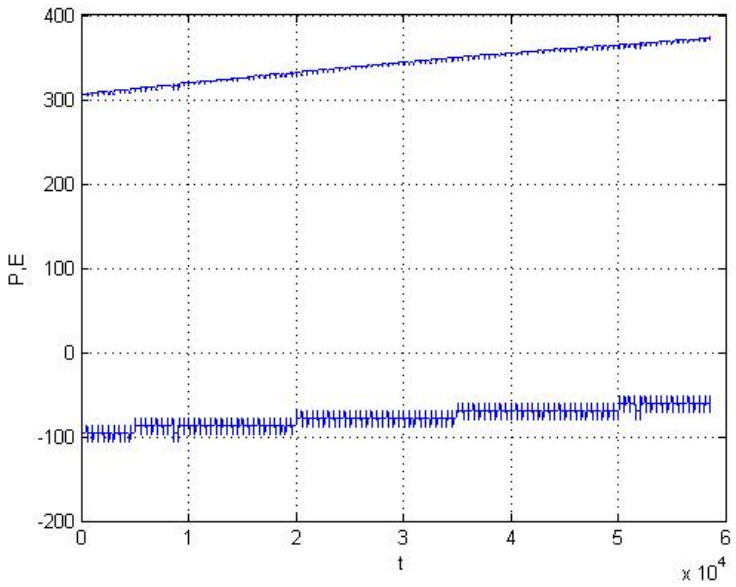

Figura 4.3: Tracking: Potencia y Posición E (10:00-11:30)

En la gráfica 4.3 se puede visualizar la evolución de la potencia y la posición E desde las 10:00 de la mañana hasta las 12:00, hora y media después. Observando la potencia se puede ver su paulatino aumento de forma clara. Mirando a la señal E, se observa cierto patrón. Esto es debido a que cada cierto tiempo el algoritmo de tracking varía E para comprobar si la potencia aumenta, reculando el movimiento en caso contrario, de ahí la pequeña perturbación.

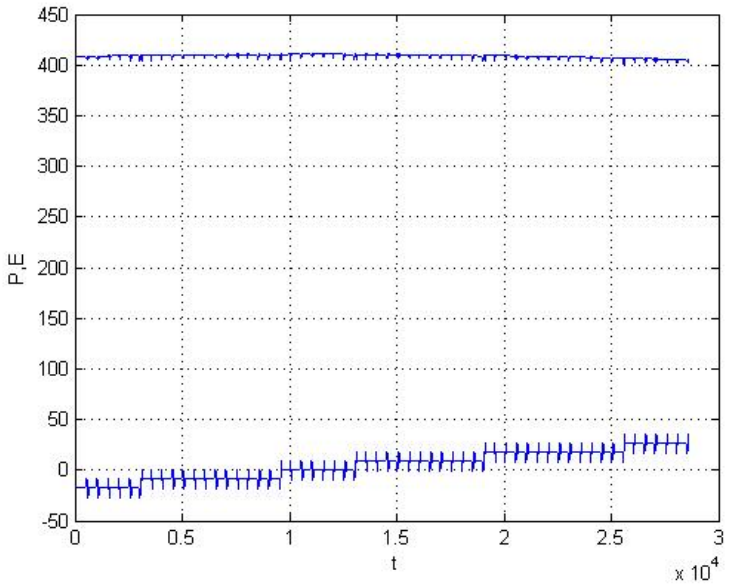

Figura 4.4: Tracking: Potencia y Posición E (13:00-15:00)

En la gráfica 4.4 se observa cómo la potencia llega a su máximo, que alcanza aproximadamente un valor de 460 W. También se puede observar como el ángulo o posición E se mueve en torno a 0, que equivale en este modelo a la posición horizontal de la placa. Al final de la gráfica se puede ver que tiende a bajar la potencia, y el ángulo aumenta, como era de esperar.

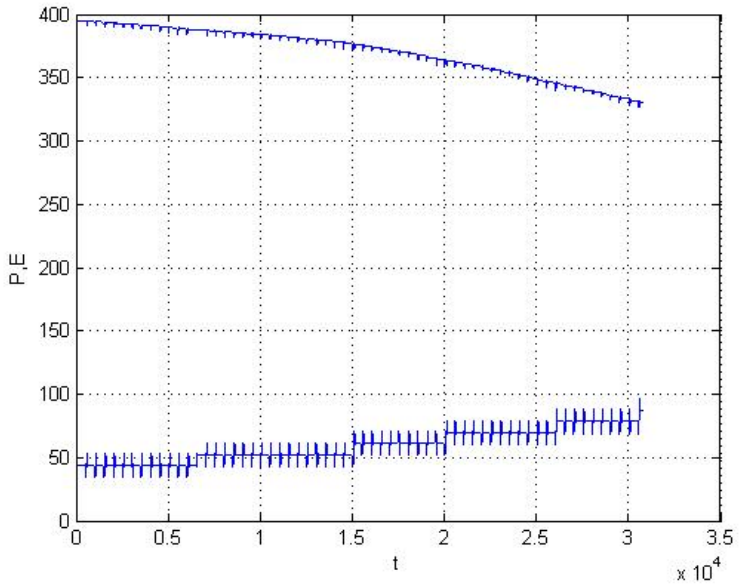

Figura 4.5: Tracking: Potencia y Posición E (15:30-17:00)

En esta última gráfica de potencia en modo Tracking, la 4.5, se puede ver como va bajando considerablemente la potencia. Si hubiera llegado a las 18:00 se habría visto una bifurcación en ambas señales, pero eso se expondrá en el apartado de Back-Tracking.

También existen resultados acerca de la insolación recibida y la temperatura de la célula. Una consideración importante sería señalar que se ha establecido T_a como la temperatura ambiente, fija a 25^oC, por lo que, y sobre todo en verano, la temperatura real de la célula sería mayor a la de la simulación. Se va a poder observar en las siguientes gráficas que ambas señales son parecidas, y así es, debido a que el modelo del cálculo de la temperatura de célula relaciona esta directamente con la insolación recibida de la forma $T_c = T_a + K\lambda$, siendo esta λ la insolación.

A continuación se exponen las 3 gráficas de la temperatura de la célula ($T_a = 25^oC$):

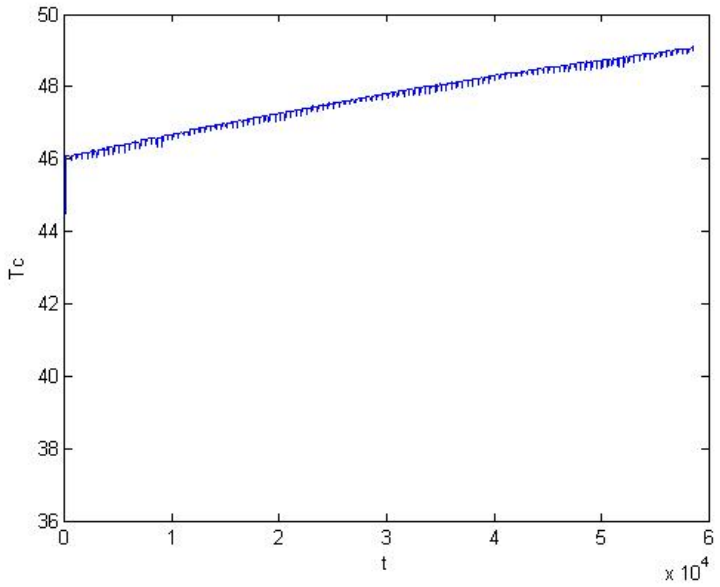

Figura 4.6: Tracking: Temperatura de célula(10:30-11:30)

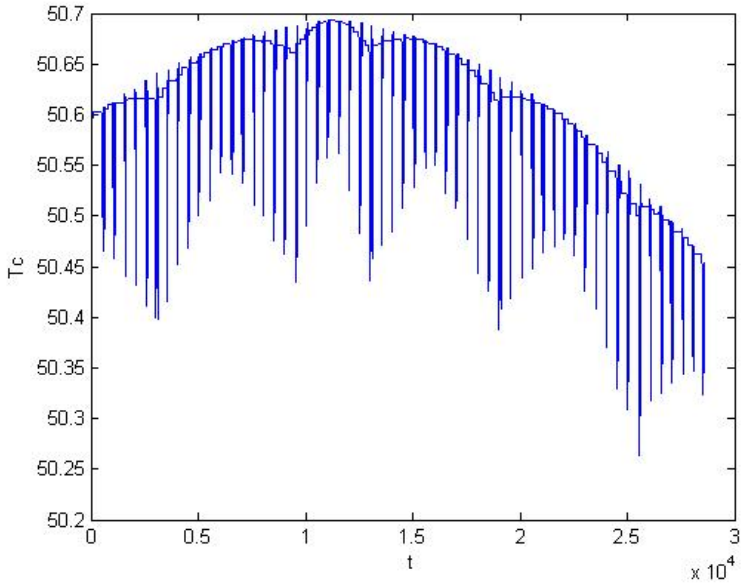

Figura 4.7: Tracking: Temperatura de célula (13:00-15:00)

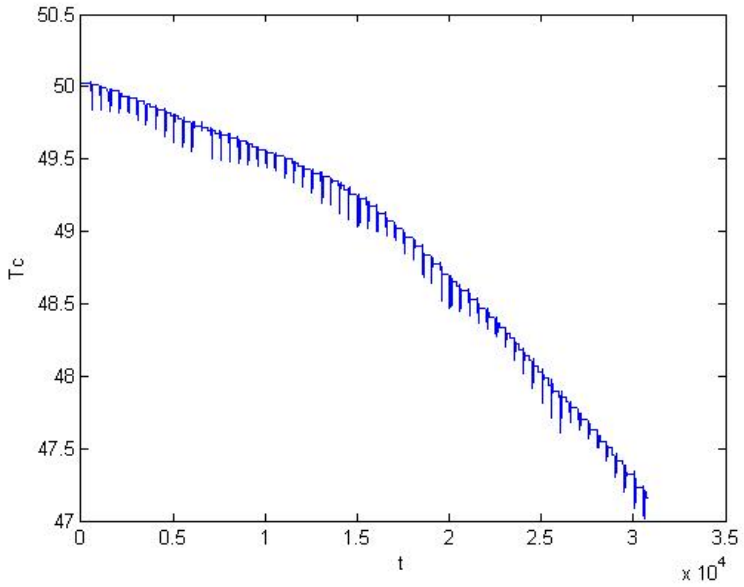

Figura 4.8: Tracking: Temperatura de célula (15:30-17:00)

El hecho de que la temperatura sea capaz de alcanzar altos valores comparada con la ambiente puede acarrear un descenso de la potencia, pero en cualquier caso la potencia visualizada anteriormente ya llevaba implementado este fenómeno, pues en el modelo de las placas va programado.

4.3. Simulaciones de Back-Tracking

En esta sección van a ser representados los resultados de simulaciones realizadas en horas críticas del día, en las que una de las placas se va a ver sombreada por la otra. Como fue expuesto anteriormente, el sombreado se da para $E = \pm 12$. Por la tarde en horario de verano, esto se da a las 18:00 aproximadamente, mientras que por la mañana se da antes de las 10:00.

En este tipo de simulaciones, es interesante analizar y graficar las siguientes variables:

- Potencias P_1 y P_2, que esta vez serán distintas, aunque se intenta minimizar su diferencia.
- Orientaciones E_1 y E_2, también distintas ahora.
- Insolaciones o sombras: son complementarias, por lo que cualquiera puede dar una información similar.

Lo primero que hay que mostrar es la potencia colectada. Se van a exponer las gráficas de potencias P_1 y P_2 por la tarde:

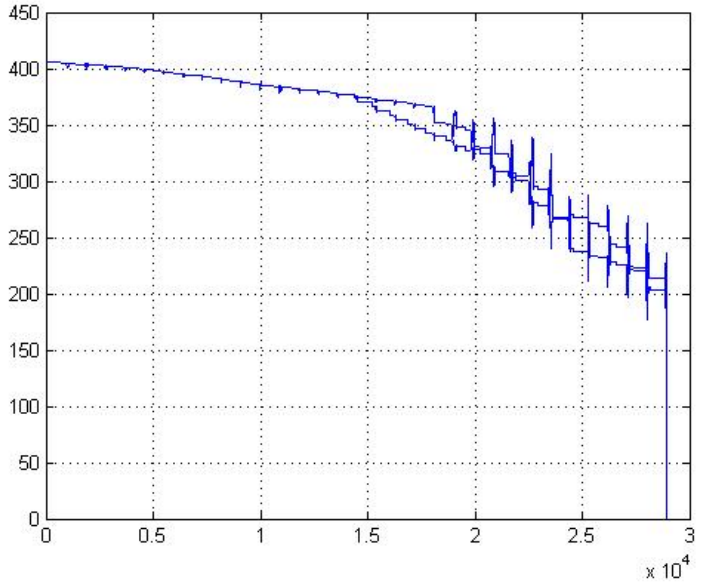

Figura 4.9: BackTracking: Potencia (17:40+)

A su vez, los ángulos u orientaciones de las placas (E) se representan de forma similar en las siguientes gráficas:

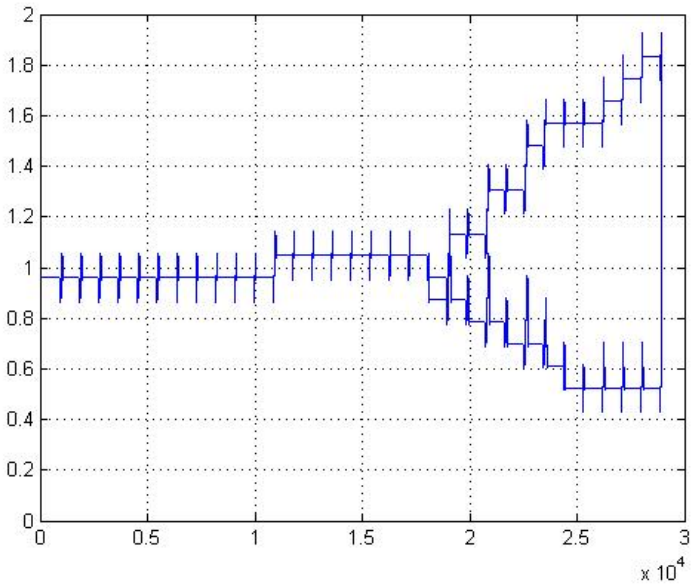

Figura 4.10: BackTracking: E (17:40+)

En las representaciones de la potencia se puede observar que a partir de cierto instante se separan, pero tienden a no estar muy alejadas. Esto es debido a que el algoritmo de Back-Tracking está diseñado de forma que se minimice esta diferencia y se maximice la suma de ambas. Sin embargo la variable E presenta un comportamiento distinto: divergen. Esto es porque una de las placas seguirá tendiendo a seguir al Sol mientras que la otra sacrifica su seguimiento para no sombrear a su compañera.
No obstante, haciendo la comparación de potencia obtenida en cada instante, se puede observar que **la suma de las potencias de ambas placas es mayor cuando no se está haciendo algoritmo de Back-Tracking**, sino Tracking simplemente, dejando que se sombreen las placas entre ellas y pesar de ello. Esto ocurre debido a que en los ensayos hasta el momento, el calculador de sombra estaba diseñado de forma que conllevara una reducción de potencia proporcional a la parte sombreada de la placa, pero esto no es así. Como se comentó anteriormente en este trabajo, el sombreado puede tener comportamientos menos optimistas, hasta el caso extremo de que cualquier sombra cancele la potencia total.
Para la disposición geométrica adoptada (separación entre placas igual a 4 metros, dimensión transversal igual a 1 metro) se descarta el algoritmo que evita cualquier sombra, ya que acarrea unas pérdidas de potencia muy grandes, sino que se conservará el método de optimización de la potencia.
A su vez, haciendo los ensayos pertinentes ha quedado claro que para la

disposición adoptada por la tarde la placa empieza a ser sombreada aproximadamente a las 18:00 en junio. Esto equivale a una orientación de las placas de 60º con respecto a la vertical, y en nomenclatura del proyecto en E=12. Un ensayo a las 18:20 aproximadamente, cuando la orientación del vector Sol es $E_{SOL} = 13$ muestra una sombra proyectada sobre la placa 2. Para evitar esta sombra, se optó por mover únicamente la placa 1, y eso conllevó una mejora del 0,6 % en la potencia de la segunda placa, siendo las pérdidas mayores en la primera que las ganancias en la segunda. Sin embargo, si las dos placas se reorientan con el mismo ángulo se logra un mayor aumento de potencia (1,2 %) por lo que se va a optar por **controlarlas a las dos a la vez**.

Dicho esto, la representación de la potencia y la orientación de las placas para la disposición geométrica indicada pero controlando las placas a la vez, y también utilizando un calculador de sombra más realista[1] sería la siguiente:

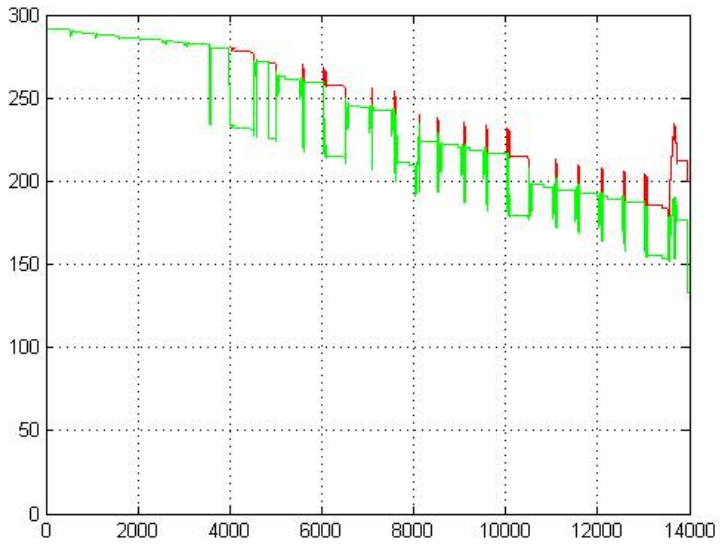

Figura 4.11: BackTracking: Potencia (sombra realista)

[1]En este caso cuando se sombrea parte de una célula del módulo fotovoltaico se cancela la potencia de esa célula. Como los módulos adoptados tienen 6 filas de células, un poco de sombra conllevará una disminución del 17 % de la potencia.

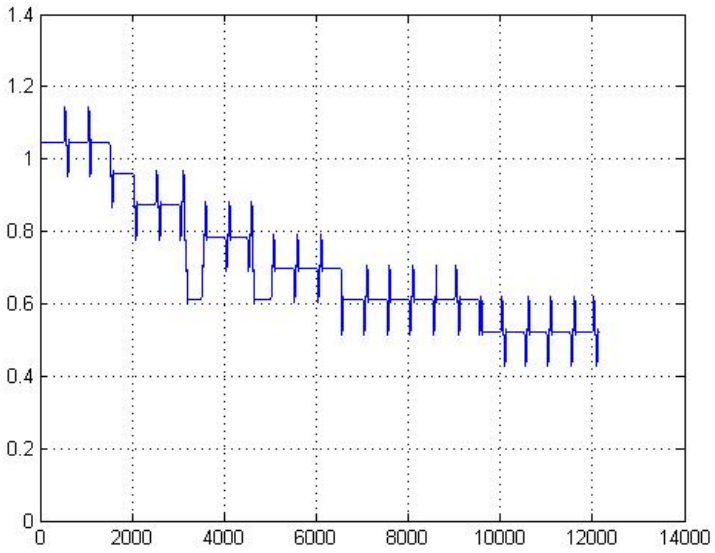

Figura 4.12: BackTracking: Posición E, orientación (sombra realista)

Para comprobar que el Back-Tracking está justificado en este caso, se va a realizar una simulación sin que este se ejecute, es decir, con ambas placas haciendo un perfecto seguimiento del Sol. Las gráficas resultantes son las siguientes:

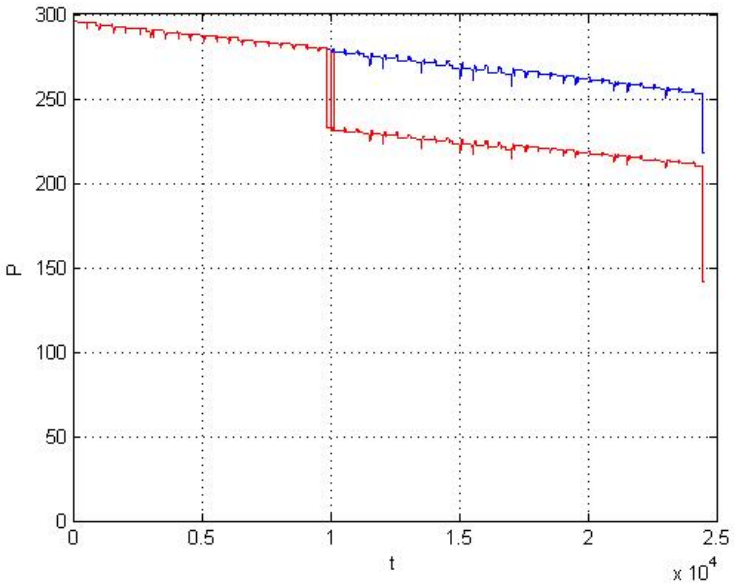

Figura 4.13: Sin BackTracking: Potencia (sombra realista)

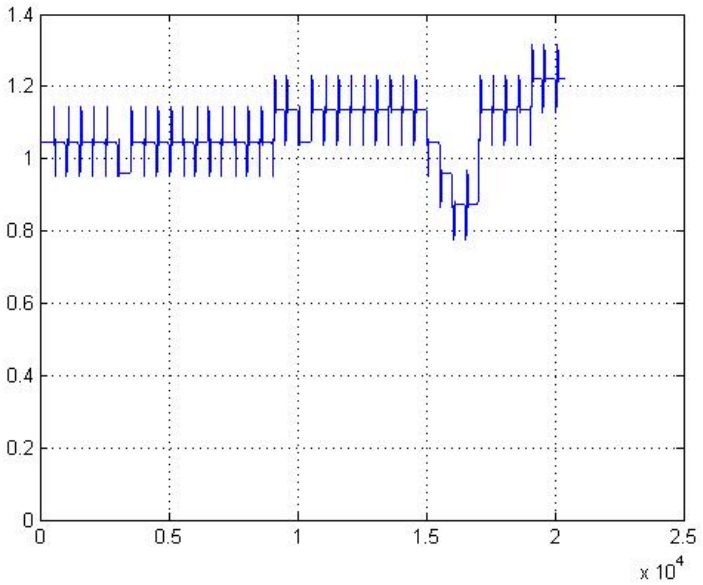

Figura 4.14: Sin BackTracking: Posición E, orientación (sombra realista)

En estas gráficas se puede comprobar que en cierto momentos se pierde una gran cantidad de potencia de golpe, y estos momentos son cuando la

sombra empieza a existir en la primera fija de células y cuando va a hacer en la segunda.

Nota. *Después de las simulaciones y resultados obtenidos, ha habido pequeños cambios en algunas partes del proyecto que han dado lugar a estos nuevos resultados. Fundamentalmente, estos cambios pertenecen al Estado* Back-Tracking. *Después de las modificaciones este ha quedado como en la siguiente figura:*

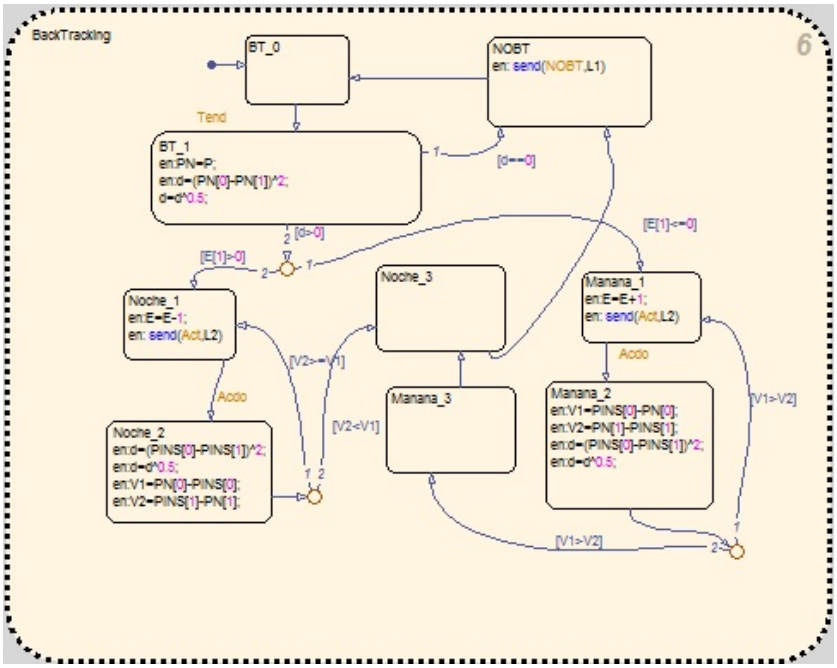

Figura 4.15: Back-Tracking Corregido

Capítulo 5

Conclusiones

En este capítulo final se van a comentar algunas de los pensamientos que se pueden plantear tras leer este proyecto. Primero se va a comentar los resultados obtenidos en las simulaciones y posteriormente se van a comentar las líneas futuras, las vías por las que este proyecto podría ser aún más desarrollado.

5.1. Comentarios sobre el proyecto

Sobre el comportamiento en modo Tracking se comprueban unos resultados satisfactorios. Se controlan ambas placas al unísono con una sola variable y con únicamente la entrada de potencia para hacer el seguimiento, y cada cierto tiempo maximiza la potencia recibida en ese instante.

En Back-Tracking ha demostrado un mal comportamiento. Si se deja funcionar en Tracking recolecta más potencia entre las dos placas que si se implementa un algoritmo de Back-Tracking, ya sea con el criterio de evitar toda sombra, o de maximizar la suma de potencias. Esto ocurre por dos motivos fundamentales:

- Control distinto de las placas: Se planteó en un principio controlarlas de forma distinta durante el Back-Tracking, es decir, que pudieran tener distintas orientaciones. Sin embargo, esto resultó perjudicial para obtener la máxima potencia posible. Cuando solo se mueve una de las placas en esta configuración geométrica, la sombra disipada no es suficiente, ya que siempre la potencia sacrificada de la que cambia su orientación para evitar sombreado es mayor. Al controlar ambas placas con la misma orientación se evitar sombra más eficientemente sin llegar a orientaciones fuera de lugar. No obstante, aún así la potencia sacrificada siempre es mayor que la ganada por evitar la sombra.

- Calculador de sombreado optimista: Como ya se ha comentado varias veces en este documento, se plantea en un principio un calculador que disminuye la potencia de una placa en el mismo porcentaje que sombra recibe. Esto en la realidad no es así, y planteando un modelo de sombreado más realista se ha visto que el algoritmo de Back-Tracking mejora la suma de potencias recibidas.

Por lo tanto, cambiando la configuración inicial, y bajo las condiciones establecidas geométricas y geográficas, se puede afirmar que:

- El algoritmo de Tracking implementado mejora la potencia y energía recibida a lo largo del día con respecto a una placa fija.
- El algoritmo de Back-Tracking implementado mejor la potencia y energía que podría recibirse en horas críticas del día si en lugar de este se continuara con el algoritmo implementado de Tracking.

5.2. Líneas futuras

Existen muchísimas opciones para llevar un paso más allá este proyecto, y aquí van a ser planteadas algunas de ellas. Dentro de lo posible, se van a ordenar estas opciones de forma que la primera esté relacionada más íntimamente con el software y la última con lo más externo del hardware: las placas.

PLC con código M

En este proyecto, el Chart de Stateflow actúa como un PLC (*Programmable Logic Controller*), pero no es la única opción que tiene Matlab para hacerlo. Otra forma de diseñar un PLC es con código M: Un bloque incrustado como los utilizados podría valer, pero también lo haría un bloque función de Matlab, que llama a un archivo '.m´. Más que una línea futura, se presenta como una vía alternativa a Stateflow.
Un artículo publicado por la Universidade Nova de Lisboa y la Universidad Politécnica de Cataluña[4] expone esto, y presenta unos bloques de código M que funcionan de forma similar a los Estados de un Grafcet. El Estado 0 se representa por $m0$, el siguiente por $m1$, etc...y están diseñados para que cuando se active el siguiente, el anterior se desactive. Un ejemplo que es presentado en ese artículo se presenta aquí con algunos cambios:

```
m0=(m2 & di5)|(m0 &  m1);
```

```
m1=(m0 & di1 & di2 & di3)|(m1 &  m2);
m2=(m1 & di4)|(m2 &  m0);
```

Esta pieza de código es sencilla de interpretar: El Estado $m0 = 1(TRUE)$ en la inicialización, y se establece por su segunda condición, porque él es 1 pero $m1$ es 0. Cuando $di1$ (Digital input 1) se pulsa al mismo tiempo que $di2$ y $di3$ se cumple una de las condiciones que pedía $m1$, por lo que esta pasa a ser 1 durante un instante. En este instante, la condición que mantenía activo a $m0$ se anula y pasa a 0. Al mismo tiempo, como $m1$ es 1, su segunda condición se cumple, ya que $m2$ es 0. Después de ese instante, uno o varios de los inputs se puede desactivar, pero el Estado está ya consolidado, y a la espera de que se active el di4 para pasar al $m2$.

De esta forma, se exporta un Grafcet sencillo de 3 Estados a código M. Para asociar acciones a los Estados bastaría con un comando `if`, o si lo que se quiere es poner una variable a 1 (SET), simplemente valdría una linea del estilo:

```
LUZ=m1;
```

Es una vía de acción interesante, y sería necesario valorar la efectividad, eficiencia, complejidad y coste computacional que resultaría de esto en un proyecto más complejo.

Simescape

Es una librería dentro de Simulink que contiene bloques muy útiles dentro del campo de la ingeniería. Se divide de la siguiente forma:

- Foundation Library. Contiene elementos básicos de varias ramas, como condensadores en la rama eléctrica, por ejemplo.
- SimDriveline. Contiene bloques relacionados con la mecánica rotacional y transacional, como componentes de vehículos, engranajes o actuadores.
- SimElectronics. Se compone de elementos como circuitos integrados, drivers, sensores y, por ejemplo, células solares.
- SimHydraulics. Componentes hidráulicos como válvulas o acumuladores.
- SimMechanics. Contiene elementos de mecánica de máquinas.

Con bloques de Simescape se podrían construir modelos de lo que se ha hecho con las librerías básicas de Simulink, sobre todo los motores y las placas.

Conexión con placas reales o maquetas

El siguiente paso interesante sería la adaptación a la realidad fuera de la simulación y el modelo. Para ello, se cogería únicamente el Chart de control con Stateflow, cuya entrada es la potencia de las placas. El valor inicial de E se podría introducir dentro del Chart para adaptarlo y que este solo tengo una entrada. La salida sería E y se conectaría a un motor de corriente continua con un PID.

Las herramientas que Matlab (o Simulink) ofrecen para esto son:

- Instrument control toolbox: Permite la conexión de Matlab con instrumentos como osciloscopios, generadores de funciones o un suministro de potencia. Puede comunicarse con drivers, o comandos de SCPI escritos. Matlab puede enviar datos generados por él mismo, o leer datos provenientes de un instrumento para su visualización.

- Data Acquisition toolbox: Permite a Matlab conectarse con instrumentos de hardware de adquisición de datos. Se puede configurar la lectura para su análisis en Matlab, así como enviar datos a través de las salida digitales o analógicas del hardware de adquisición de datos.

- Real Time Workshop (RTW): Genera código en C y C++ que puede ser utilizado en aplicaciones en tiempo real o simulado. Puede interactuar con el exterior de Matlab y Simulink.

Exportación a un conjunto más grande de placas fotovoltaicas

Una vez conectado con la realidad, se podría plantear la opción de ampliar el alcance a un proyecto de más de dos placas. Hay dos vías para ampliarlo:

- Instalar dos filas de placas de N placas cada fila, una al lado de la otra, de forma que cada placa de la fila 1 solo pueda sombrear a la de atrás, de la fila 2. En este caso no haría falta mucha adaptación, ya que todas las placas de una misma fila pueden tener la misma orientación, y de hecho el caso simplificado de este escenario es el realizado en este proyecto. No obstante, sería recomendable hacer una estudio de sombreado diagonal: en ciertas horas del día la placa (1,i) podría sombrear a la placa (2,i+1). Este efecto se puede dar conforme la latitud del lugar es mayor.

- Instalar dos columnas de placas de M placas cada columna. En este caso, en horas críticas del día se crearía una sombra en cadena. Esto hace pensar que es un escenario poco eficiente ya que ocuparía el mismo espacio aproximadamente la configuración anterior mencionada. No

obstante, sí que se puede plantear la existencia de una matriz de placas de MxN placas.
En este caso, optimizando con cualquier criterio el Back-Tracking, el resultado sería ver una placa en el extremo de cada columna con el ángulo óptimo de orientación hacia el Sol, mientras que el resto de placas de la columna tienen una orientación distinta para no sombrear a la placa de detrás. Entonces para este caso, existen las siguientes vías de acción:

- Hacer un estudio de la potencia ganada debido a que una de las placas conserve la orientación óptima. Para cierto número de placas puede dejar de ser interesante un control en el que se algunas placas adopten distintos ángulos, debido a la simpleza de que todas adopten el mismo.
- Hacer que cada una de las placas adopte un ángulo distinto para optimizar la energía recolectada a la hora de hacer Back-Tracking. No obstante esto no es viable, debido a que es difícil hacer que el ajuste fino para orientar la placa sea menor a 5º.

Appendices

Apéndice A

GUIDE

La herramienta utilizada para hacer la botonera a modo de interfaz ha sido GUIDE. Pertence a Matlab y permite crear aplicaciones interactivas con varios tipos de modos de interacción: botones, pulsadores, radio botones, barras deslizadoras... La interfaz realizada para este proyecto es muy sencilla: solo dispone de pulsadores (*Push Button*) y van a controlar los eventos de entrada del Chart.
En este Anexo se van a describir los pasos realizados para llevar a cabo la interfaz de este proyecto.

1. Abrir GUIDE. Para esto, basta con escribir la palabra "guide" en la línea de comandos en Matlab, o bien pulsa su icono en la barra de herramientas, junto al de Simulink.

2. Aparecerá una ventana con varias opciones. Hay que pulsar sobre "Blank GUI (default)" y después en OK.

3. Se verá una ventana como la siguiente:

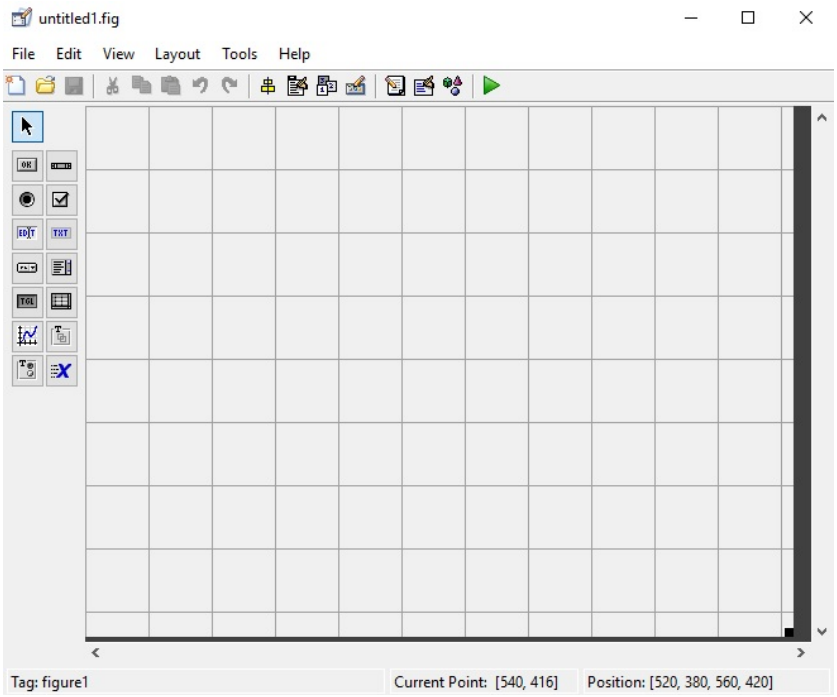

Figura A.1: Ventana de edición de Guide

En la barra de herramientas lateral se pueden ver todos los elementos disponibles para personalizar la GUI, pero solo será utilizado el primero, el Push button.

4. Habiendo elegido el Push button, colocando el puntero sobre la zona de trabajo se puede dimensionar tal botón, pulsando y arrastrando. Para este proyecto se necesitan 5 botones: **Awake** para despertar el Chart, **Load** para cargar el modelo, **Run** para empezar la simulación, **On** para que el Chart se encargue de llevar a cabo los algoritmos de Tracking y Back-Tracking ,**Off** para que deje de hacerlo y *Stop* para parar la simulación. Un primer diseño quedaría:

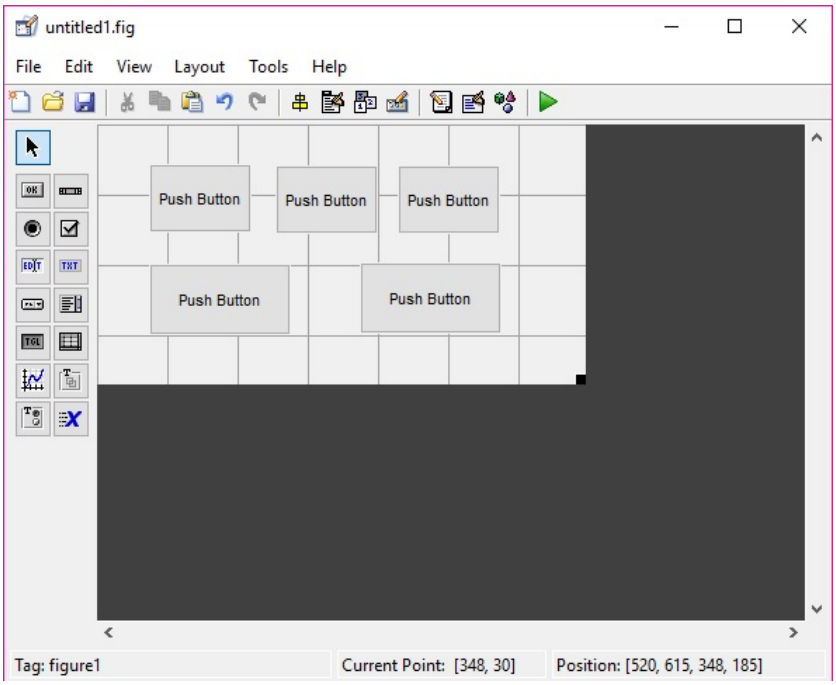

Figura A.2: Ventana de edición de Guide con Push Buttons

5. Personalizar el aspecto de los botones. Al hacer click derecho sobre cada botón, se abre un menú. En él, aparece abajo *Property Inspector*. Aparece una lista de opciones personalizable, pero solo van a ser utilizadas dos: *ForegroundColor* y *String*. La primera cambia el color del texto, y la segunda cambia el texto en sí. Después de añadir el último Push Button y personalizarlos, quedarían así:

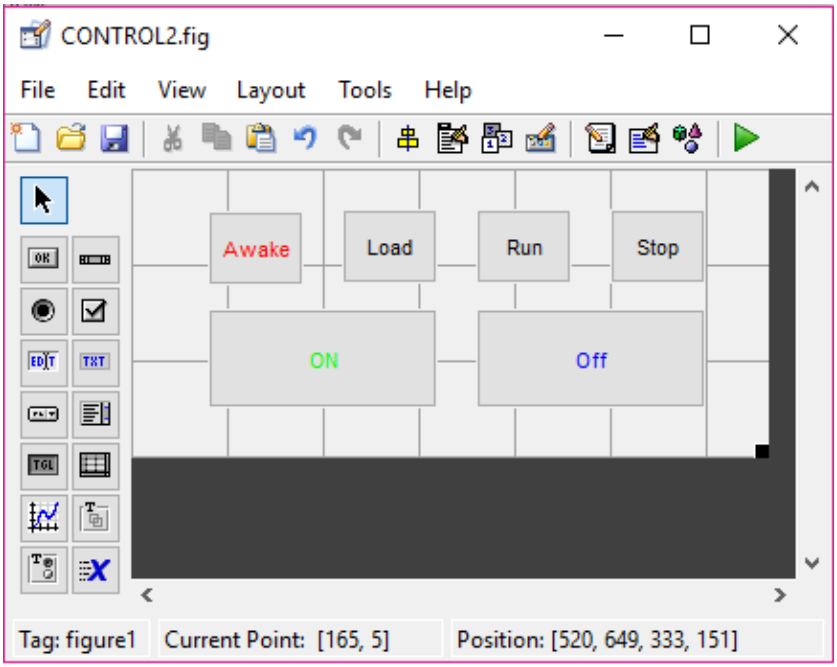

Figura A.3: Ventana de edición de Guide con Push Buttons personalizados

6. Diseño de la funcionalidad de cada Push Button. Para esto es necesario modificar el archivo en código M generado por GUI. Se puede acceder a cada parte de ese código de forma intuitiva haciendo click derecho sobre el elemento que se quiera configurar y siguiendo la ruta *View CallBacks* → *CallBack*. Esto abre el código por la parte en la que actúa el botón, la que se ejecutará cuando sea pulsado. El código asociado a cada botón es el siguiente:

   - Load
     ```
     load_system('PFC');
     ```
   - Run
     ```
     set_param('PFC','SimulationCommand','start');
     ```
   - Awake
     ```
     v=1;value=num2str(v);
     set_param('PFC/Botones/AWAKE','Gain', value);
     ```
   - On
     ```
     v=1;value=num2str(v);
     set_param('PFC/Botones/ON','Gain', value);
     v=0;value=num2str(v);
     ```

```
set_param('PFC/Botones/OFF','Gain', value);
```

- Off

```
v=0;value=num2str(v);
set_param('PFC/Botones/ON','Gain', value);
v=1;value=num2str(v);
set_param('PFC/Botones/OFF','Gain', value);
```

- Stop

```
v=0;value=num2str(v);
set_param('PFC/Botones/ON','Gain', value);
set_param('PFC/Botones/OFF','Gain', value);
set_param('PFC/Botones/AWAKE','Gain', value);
save_system('PFC');
set_param('PFC','SimulationCommand','stop');
```

El funcionamiento de esta configuración es sencilla: Controlan que una ganancia valga 0 o 1 para cortar o permitir el paso de un tren de pulsos que activará el evento correspondiente.

7. Se guarda y se ejecuta. Debería salir la ventana con los botones programados. Para este caso, el primero en pulsar debería ser Load, seguido de Run. Awake y On para que el Chart ejecute los algoritmos, Off para apagar el Chart y finalmente Stop para terminar con la simulación.

Figura A.4: Interfaz final

Bibliografía

[1] Lauritzen Inc, *www.lauritzen.biz/static/solutions/backtracking.pdf*, junio 2016.

[2] L. Narvarte y E. Lorenzo, Tracking and Ground Cover Ratio, *PROGRESS IN PHOTOVOLTAICS: RESEARCH AND APPLICATIONS*, junio 2008.

[3] PVeducation, *http://www.pveducation.org/*, junio 2016.

[4] Herminio Martínez, Antoni Grau y Joao Martins, Celson Lima, *A Matlab/Simulink framework for PLC controlled processes*,*http://www.intechopen.com*.