Diseño con microcontroladores

Controladores PID

Marco Xavier Rivera González
marco.rivera@upm.es

Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

POLITÉCNICA

UNIVERSIDAD
POLITÉCNICA
DE MADRID

Controladores: Actuadores, sistema, o planta de acción

Los actuadores, sistema o planta de acción: son los dispositivos que generan una salida con respecto a la entrada.

- Entrada: voltaje
- Salida: velocidad del motor

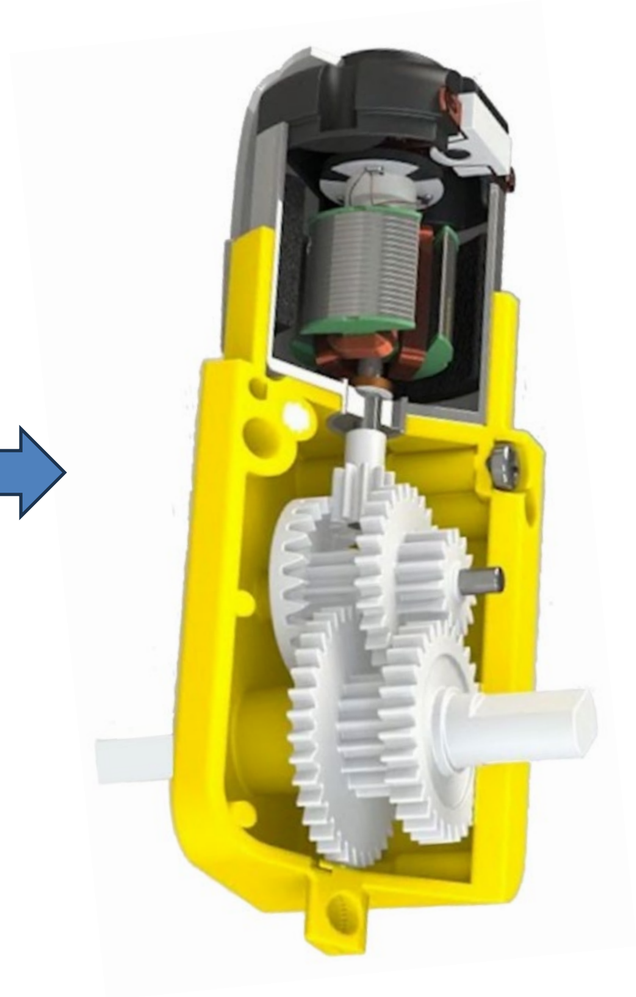
Controladores: Controladores a lazo Abierto

Planta

Velocidad

¿Qué valor de voltaje, y corriente necesita el motor para girar a 100 RPM?

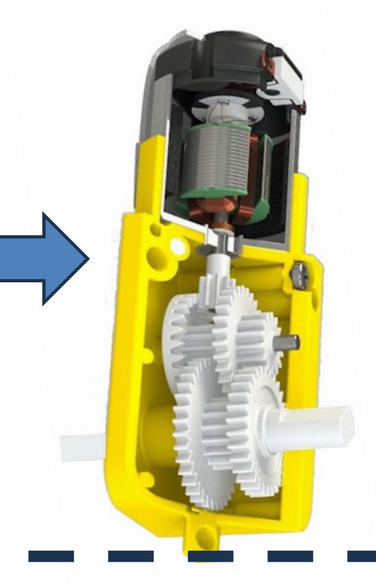
Controladores: Controladores a lazo Abierto



Velocidad

Es necesario conocer el funcionamiento exacto de la Planta (fricciones, desgaste, perturbaciones)

Controladores: Controladores a lazo Abierto



Velocidad

Referencia

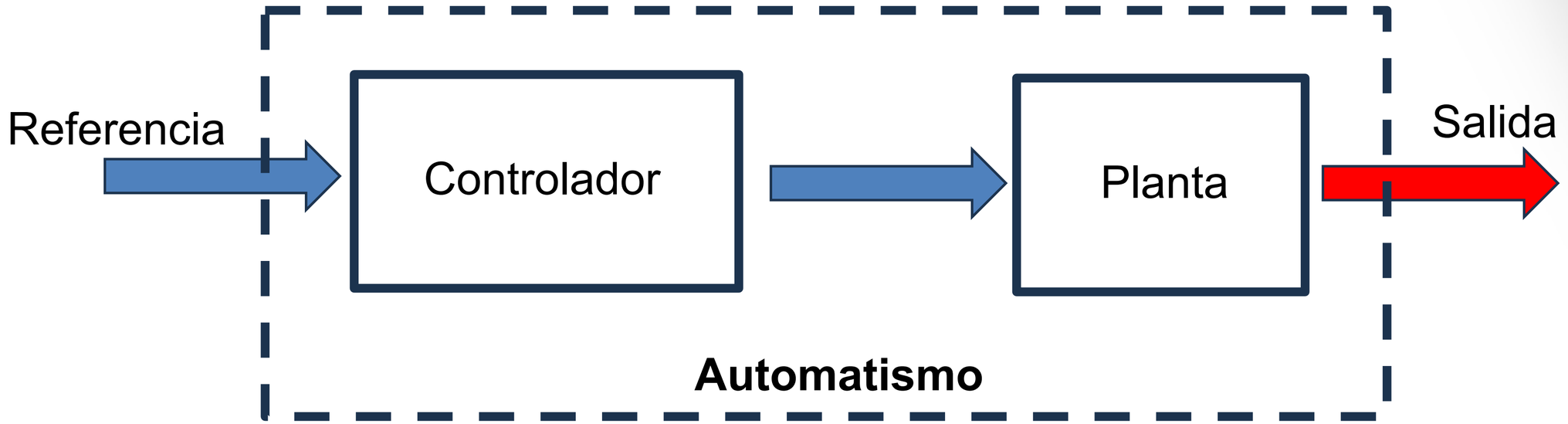
Controlador

Planta

Salida

Automatismo

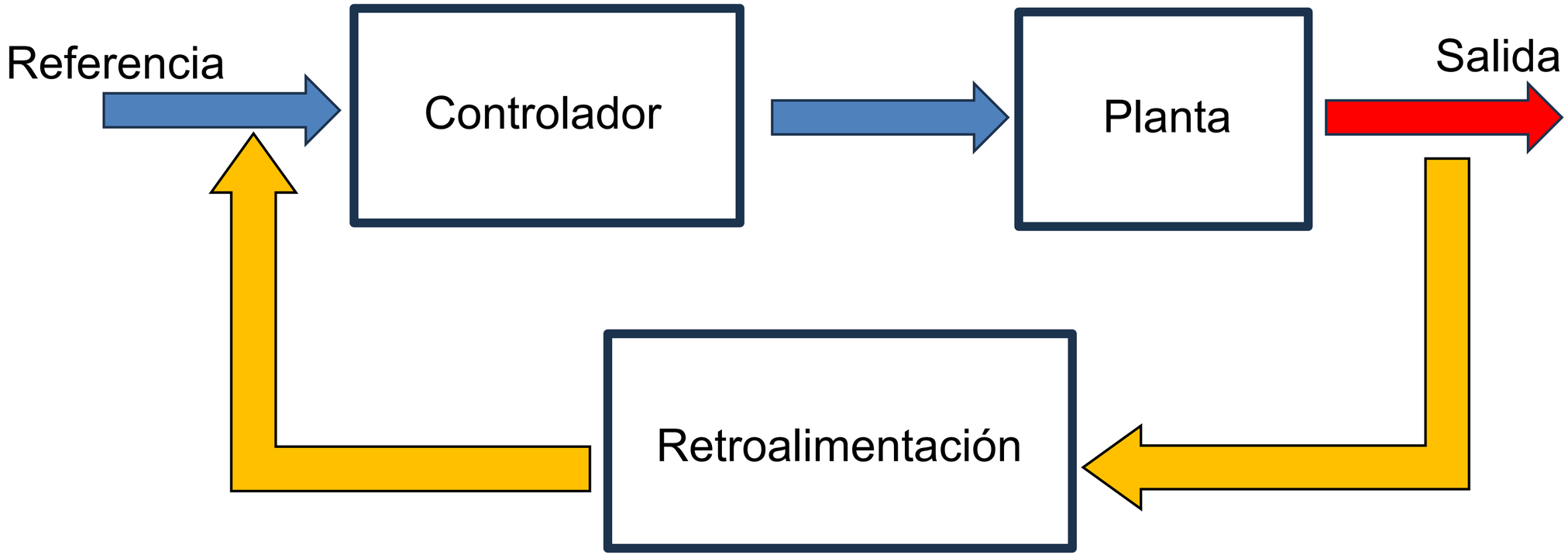
Controladores: Controladores a lazo Abierto



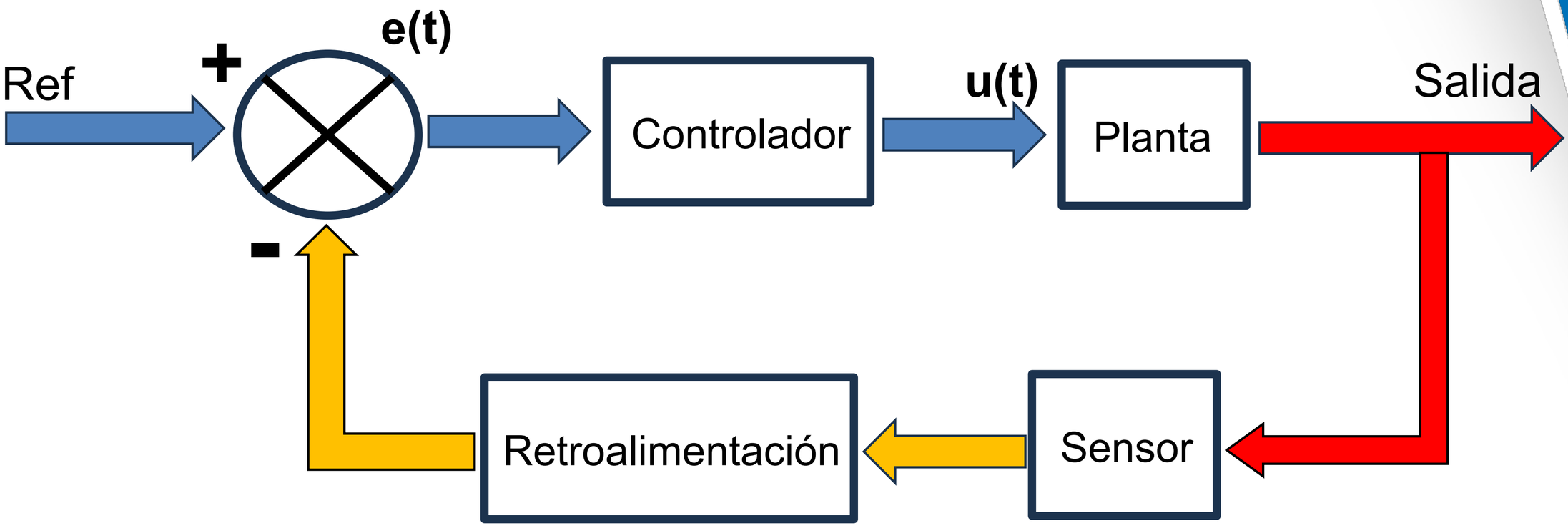
- El mayor problema de los controladores es que son **“ciegos”.**
- **La Planta define la física:** El PID solo ve números, pero la Planta es la que tiene inercia, pérdidas de calor, fricción o retrasos.
- ¿Cómo se puede compensar perturbaciones externas, perdidas, etc?

Controladores: Controladores a lazo Abierto

¿Cómo hacemos que un sistema se comporte como queremos a pesar de las perturbaciones?



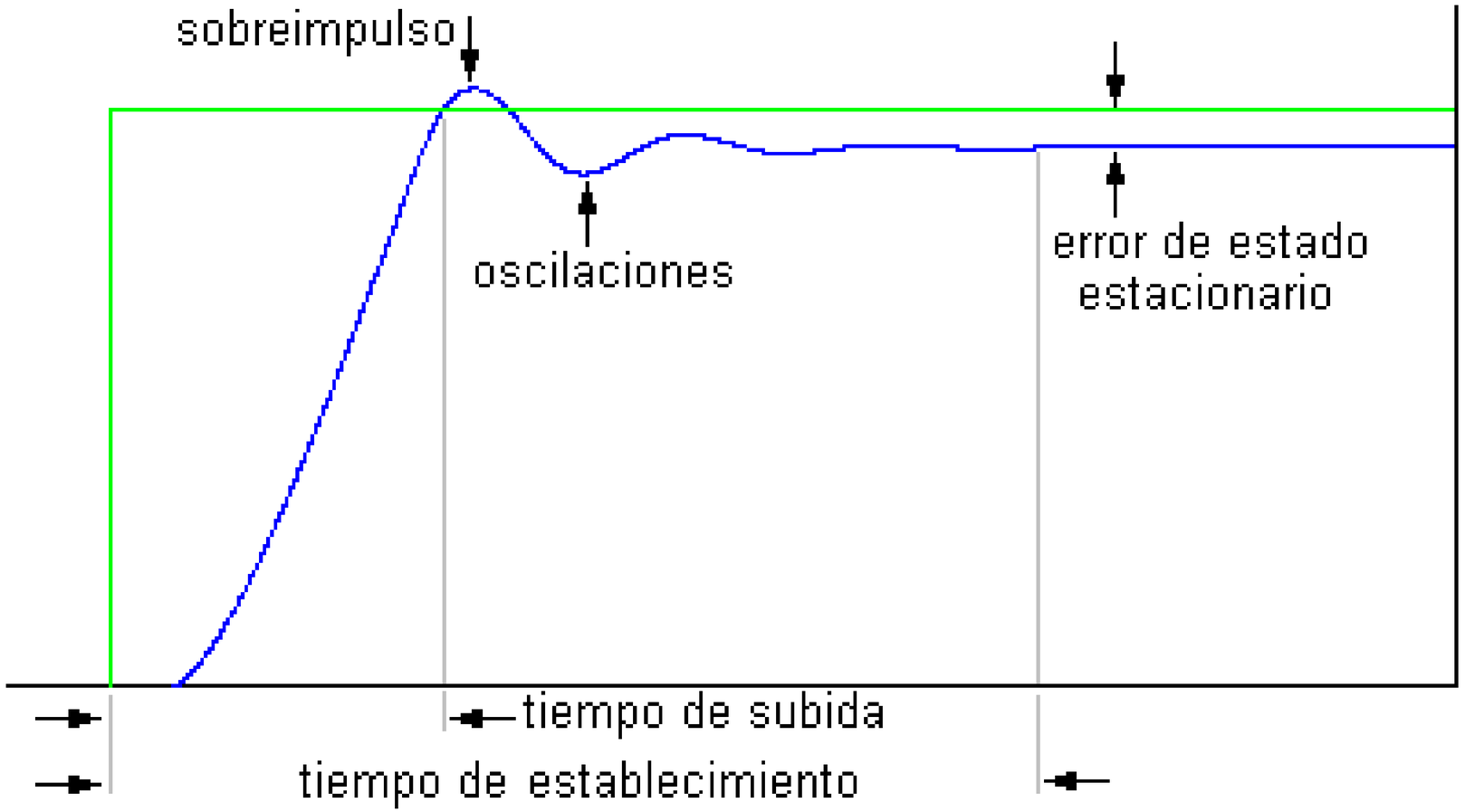
Controladores: Controladores a lazo Abierto



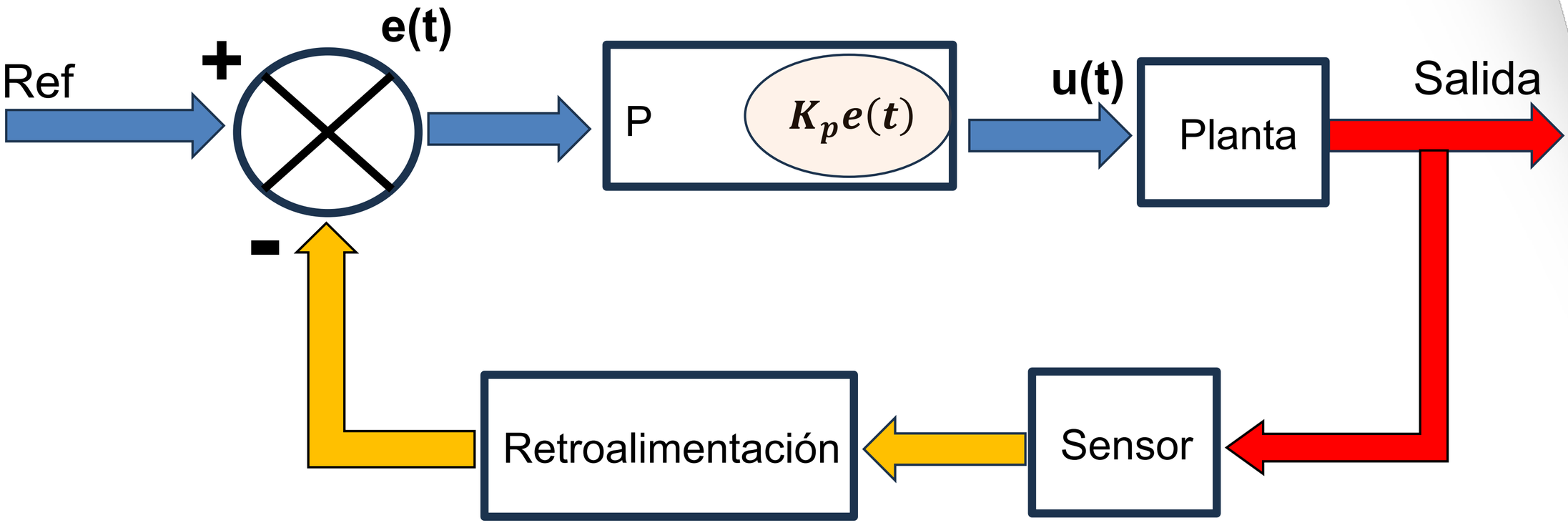
$$e(t) = \textbf{Señal de referencia} - \textbf{Señal de salida}$$

Controladores: Señal de respuesta de los controladores

$$e(t) = Señal\ de\ referencia - Señal\ de\ salida$$

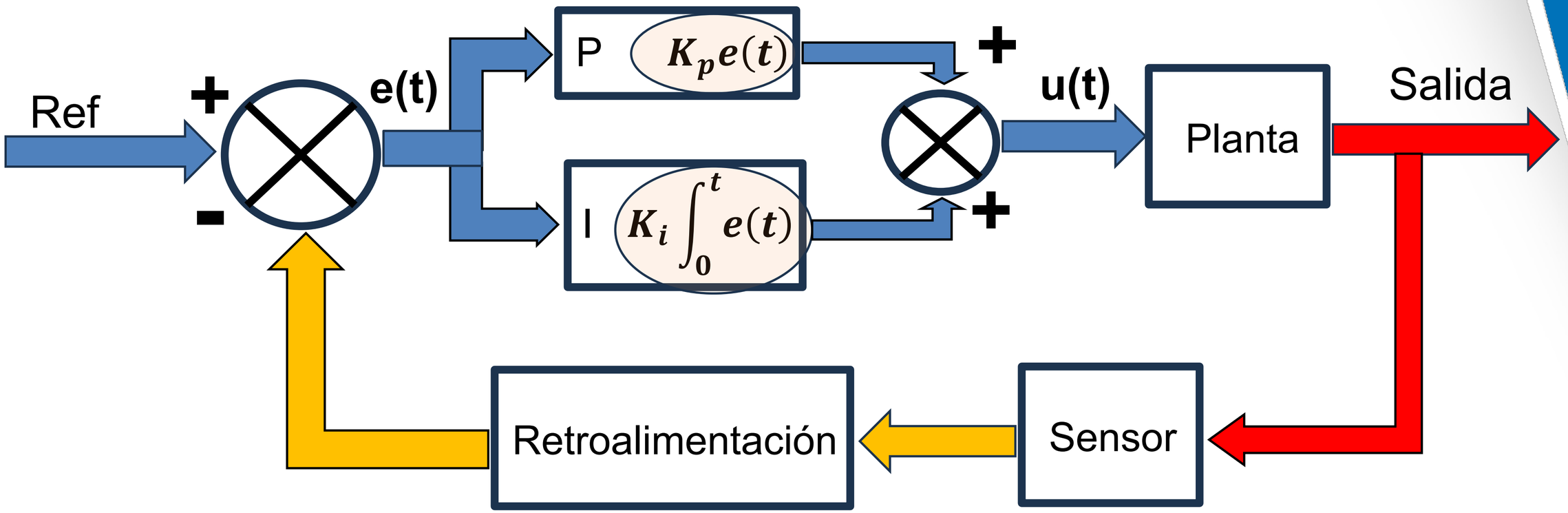


Controladores: Controladores ‘P’ proporcional



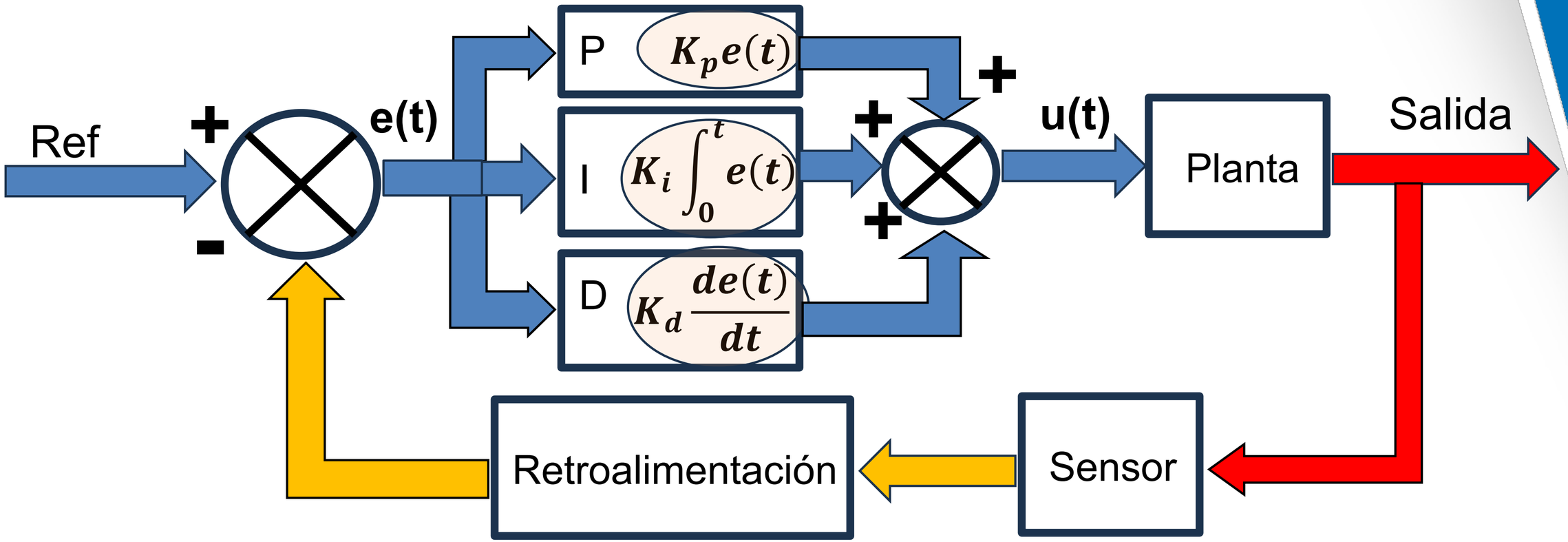
- **Controlador P:** Hace referencia al “presente”
- El problema del Offset (Error de estado estacionario)

Controladores: Controladores ‘PI’ proporcional-Integral



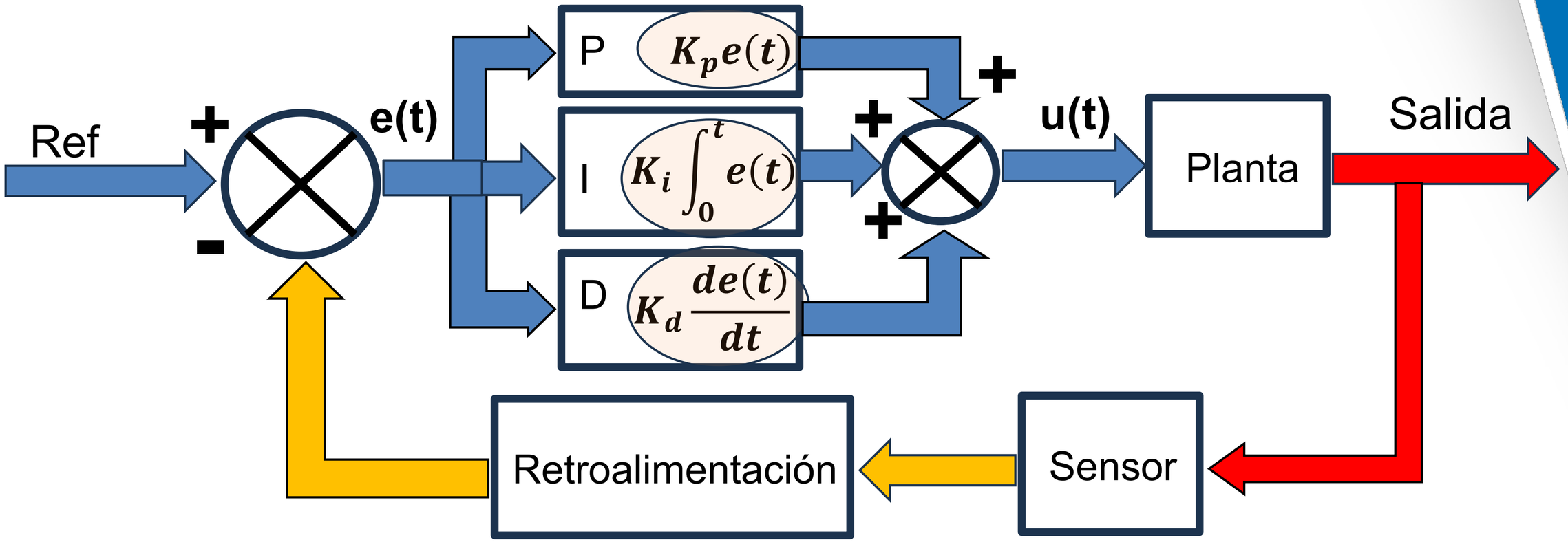
- **Controlador I:** Hace referencia al “pasado”
- Acumula error para eliminarlo. Elimina el offset
- Al acumular error, genera sobresaltos

Controladores: Controladores ‘PID’ proporcional-Integral-Derivativo



- **Controlador D:** Hace referencia al “futuro”
- Genera amortiguamiento y estabilidad. Predice el comportamiento
- Es muy sensible al ruido de los sensores

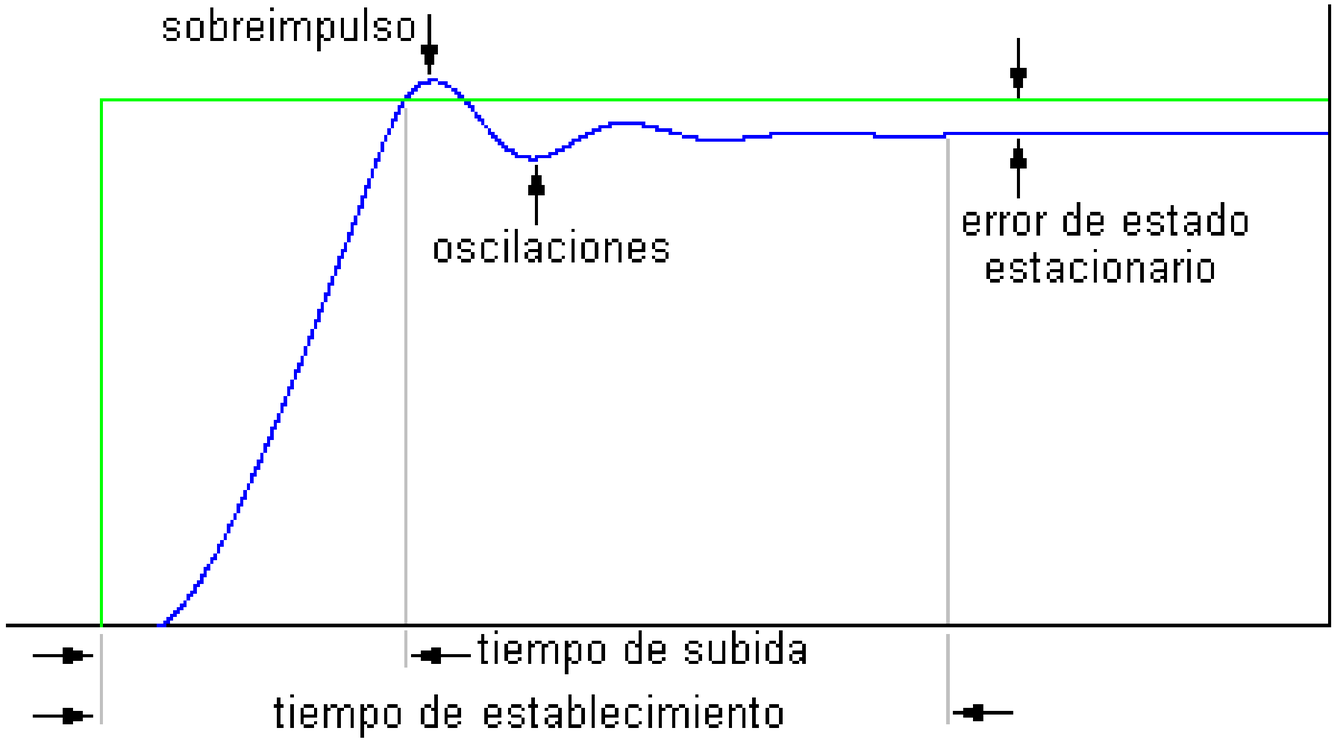
Controladores: Controladores ‘PID’ proporcional-Integral-Derivativo



$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{d(e(t))}{dt}$$

Controladores: Calibración del controlador

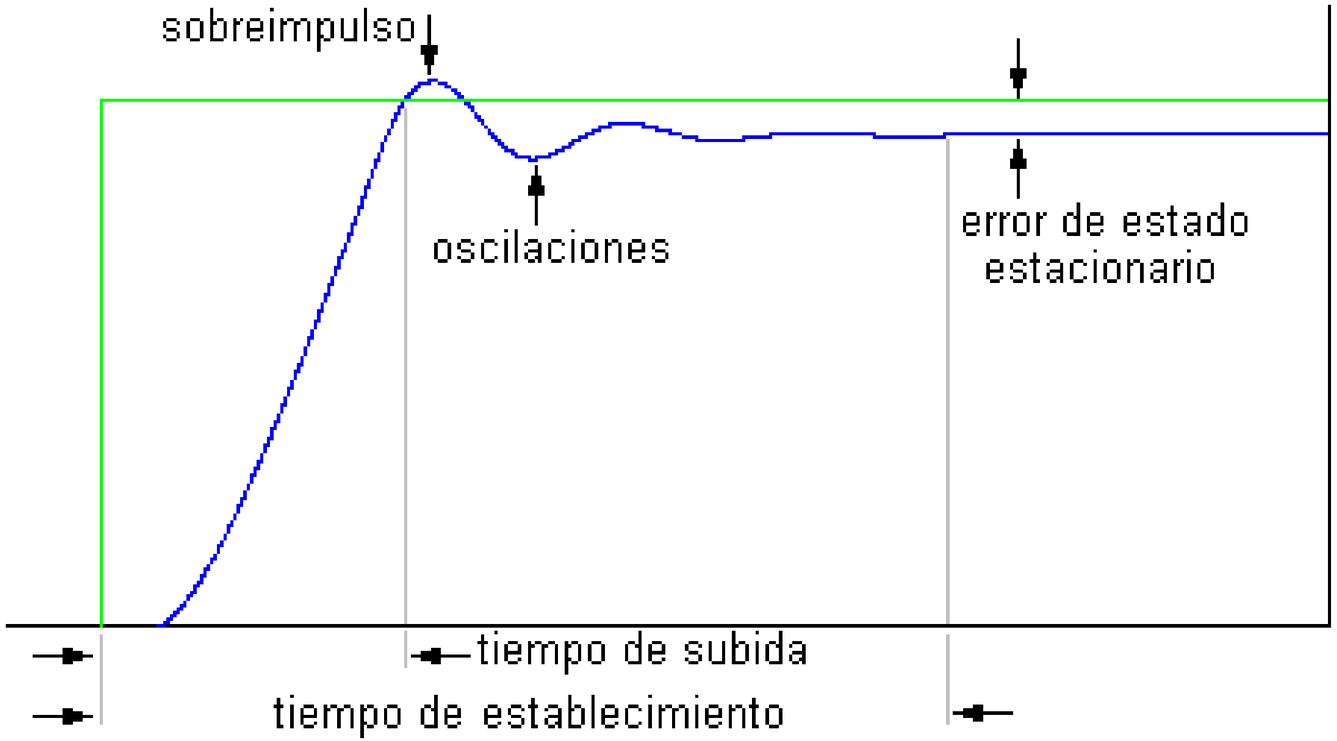
$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{d(e(t))}{dt}$$



- P ⟹ Velocidad
- I ⟹ Precisión
- D ⟹ Estabilidad

Controladores: Calibración del controlador

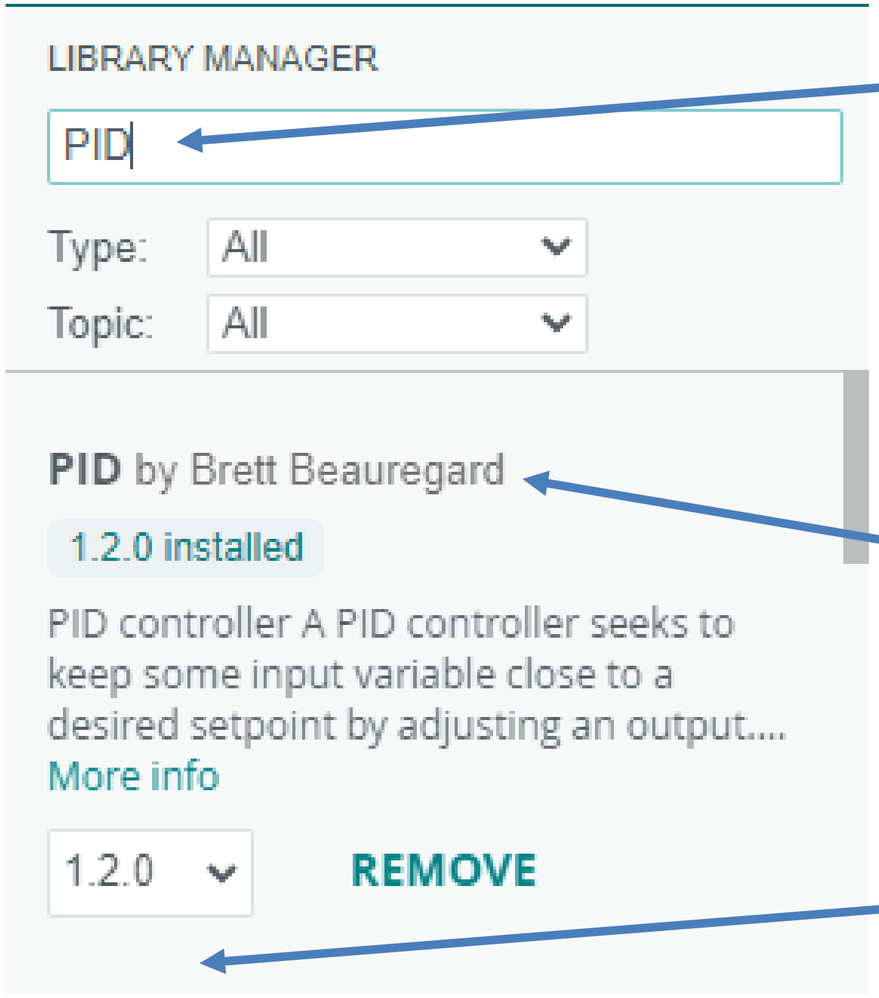
$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{d(e(t))}{dt}$$



1. Poner Ki y Kd en cero
2. Modificar Kp hasta que oscile de forma continua.
3. Reducir Kp a la mitad
4. Subir solo Ki, para eliminar el error estacionario
5. Subir Kd (solo si es necesario) para frenar sobreimpulsos

Controladores: Implementación con Arduino

1. Tools/Manage Libraries



2. Buscar PID

3. Seleccionar esta opción

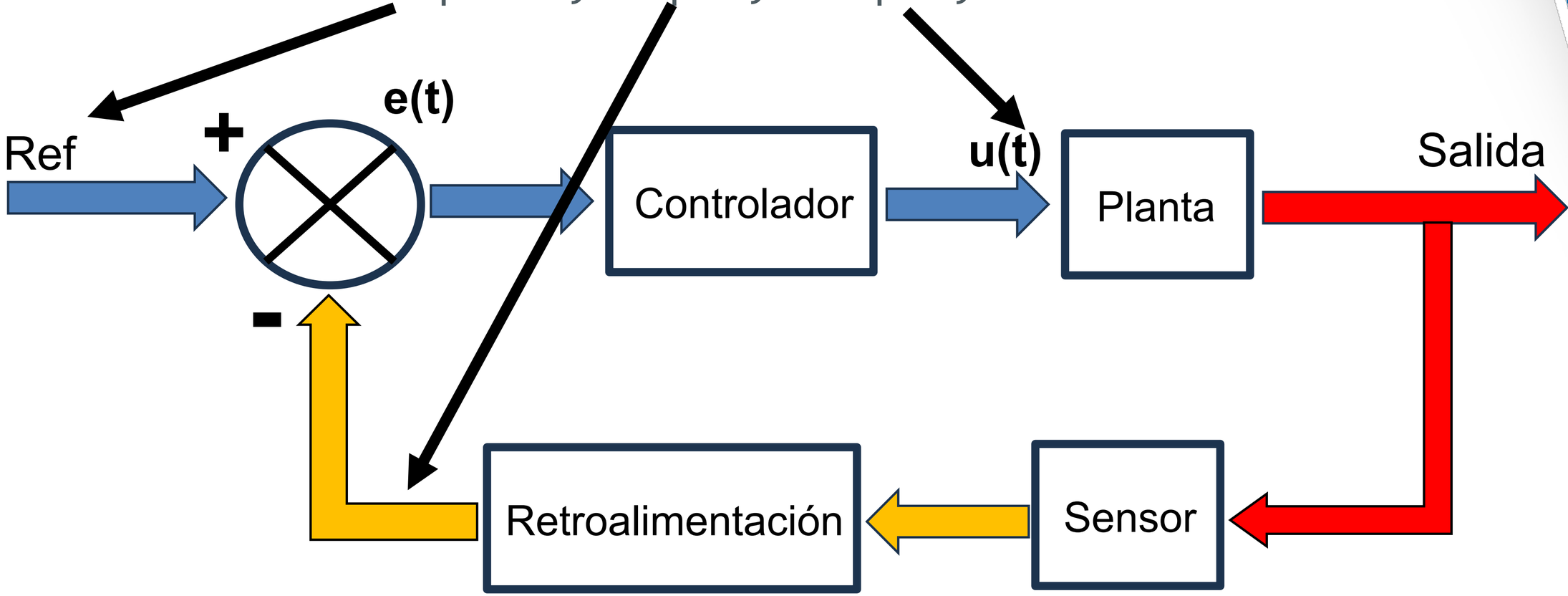
4. Instalar

Controladores: Implementación con Arduino

```
#include <PID_v1.h>
```

Variables que se deben conectar

```
double Setpoint, Input, Output;
```



Controladores: Implementación con Arduino

```
#include <PID_v1.h>
```

Variables que se deben conectar

```
double Setpoint, Input, Output;
```

Especificar los valores de configuración del controlador

```
double Kp=2, Ki=5, Kd=0;
PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, Salida);
```

Salida

- DIRECT: La salida aumenta, si la señal de referencia aumenta
- REVERSE La salida disminuye si la señal de referencia aumenta

Controladores: Implementación con Arduino

```
#include <PID_v1.h>

   double Setpoint, Input, Output;
   double Kp=2, Ki=5, Kd=0;
   PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, Salida);

 void setup() {
     Input =velocidad;
     Setpoint=20;
     myPID.SetMode(MOD0);
 }
```

Variable a la entrada

Valor de referencia

MODO

- **MANUAL:** Valor de defecto (apagado)
- **AUTOMATIC:** Inicia el controlador y los cálculos

Controladores: Implementación con Arduino

```
#include <PID_v1.h>
    double Setpoint, Input, Output;
    double Kp=2, Ki=5, Kd=0;
    PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, Salida);

void setup() {
    Input =velocidad;
    Setpoint=20;
    myPID.SetMode(MOD0);
}
```

Carga el nuevo valor medido del actuador

Realiza los cálculos y obtiene la salida

Carga la señal de salida en la planta

```
void loop()
{
  Input = analogRead(PIN_INPUT);
  myPID.Compute();
  analogWrite(PIN_OUTPUT, Output);
}
```