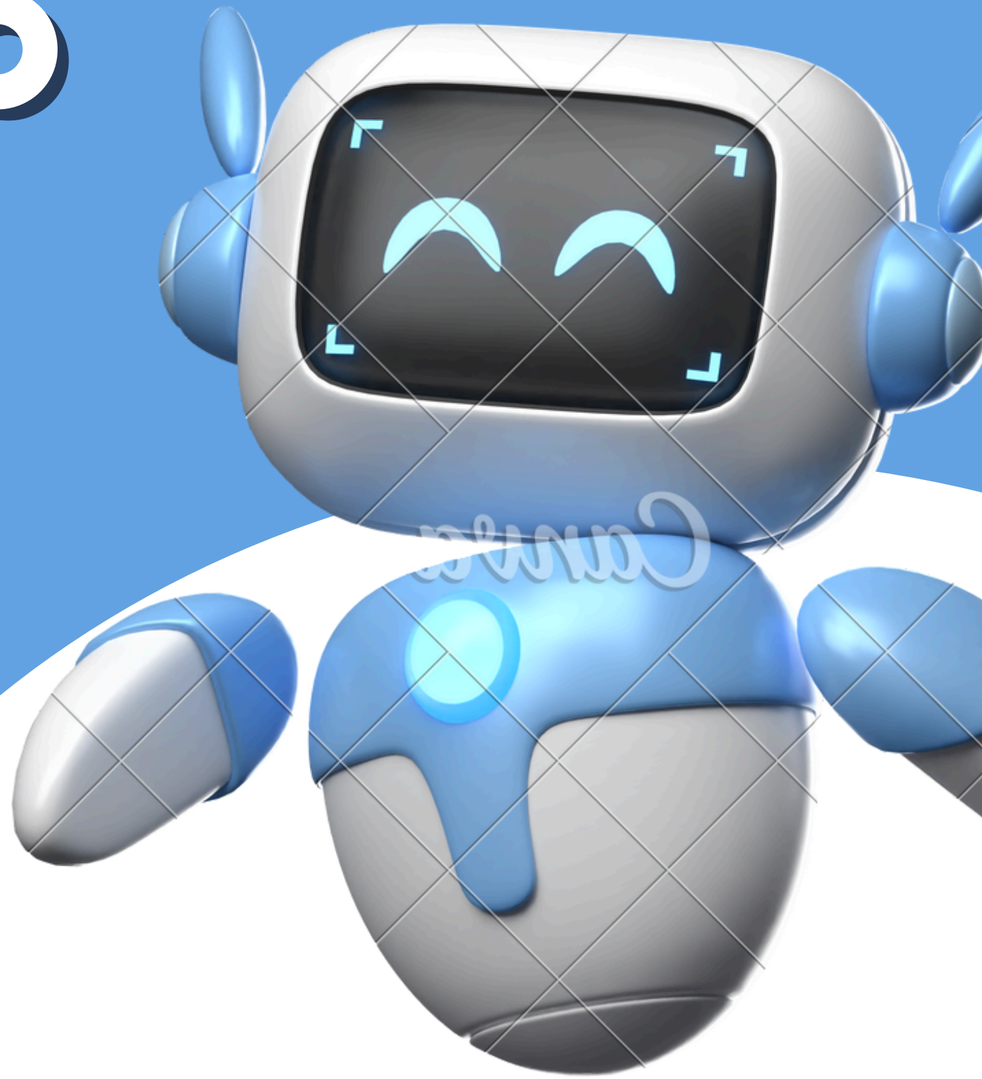
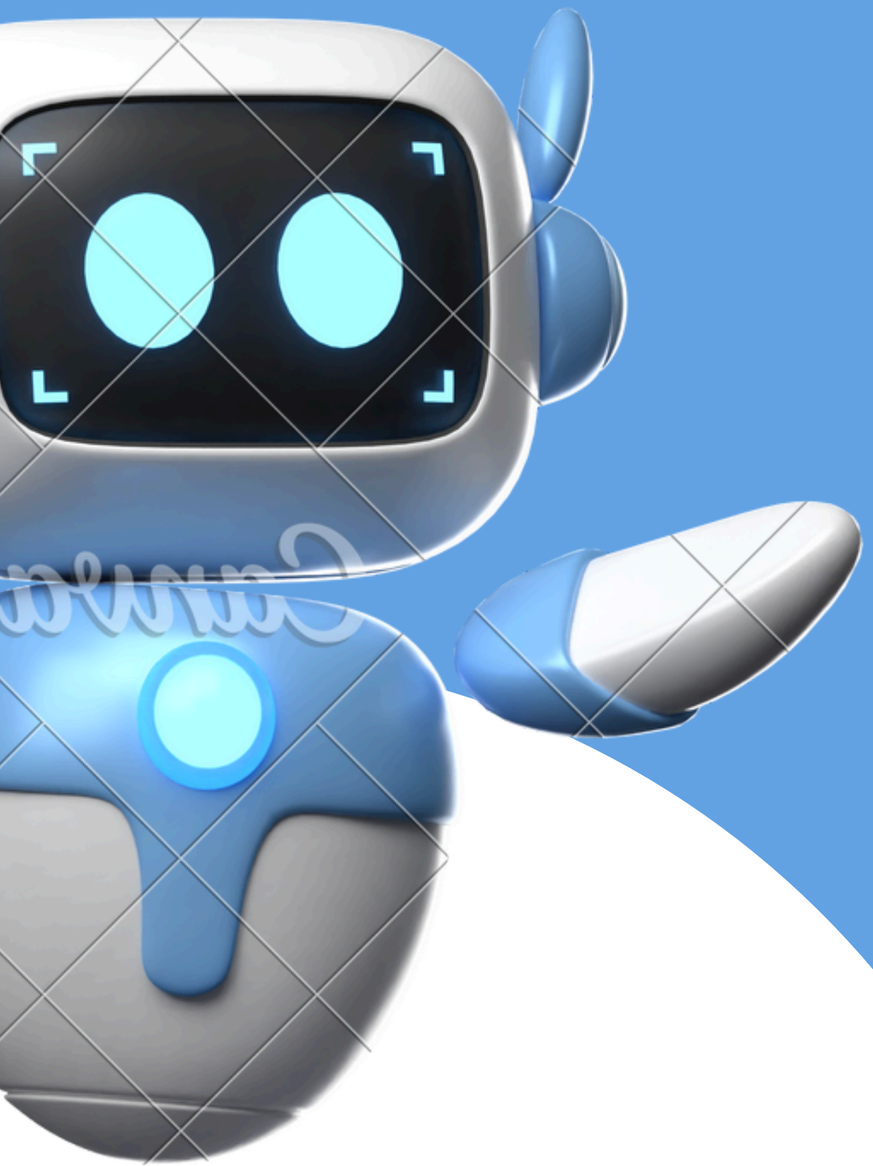
CARRO BOMBERO

ROBÓTICA II

- Leopoldo J. Contreras Melendrez
- Gustavo Huilla Ramos

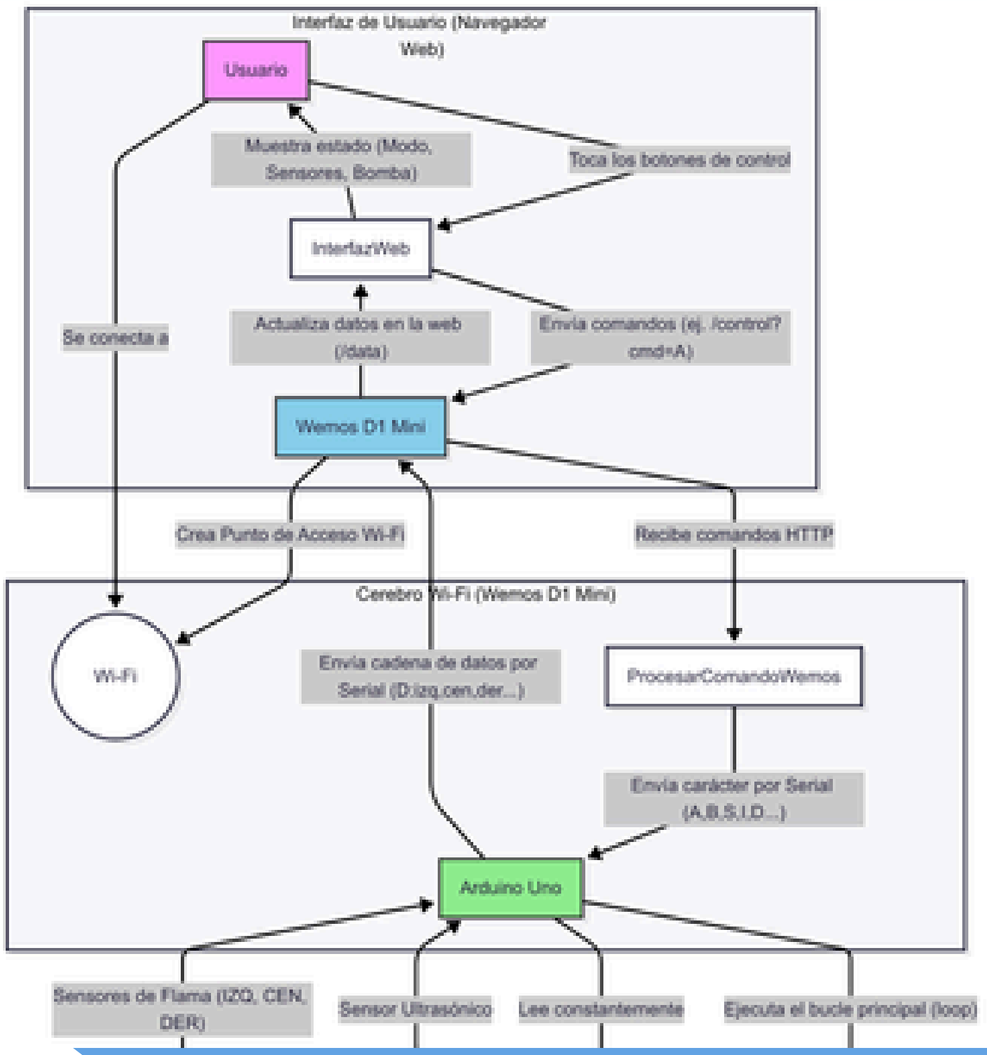


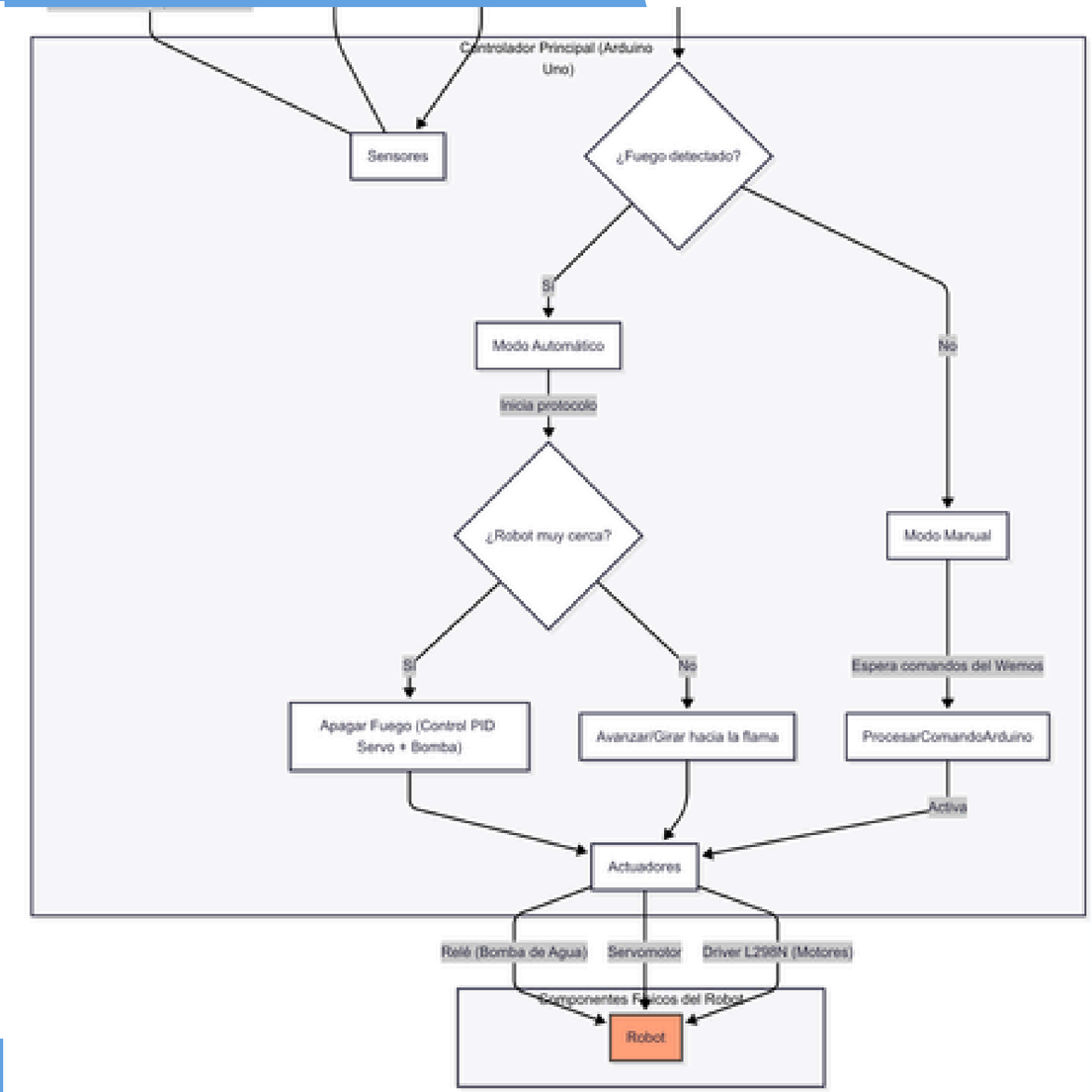


OBJETIVOS DEL PROYECTO

1. Implementar una forma mas rápida y facil de controlar el carrito a travez de una interfaz.
2. Mostrar los valores numéricos de los sensores en la Interfaz
3. Imprementar el modelo PD (Proporcional-Derivativo)

DIAGRAMA

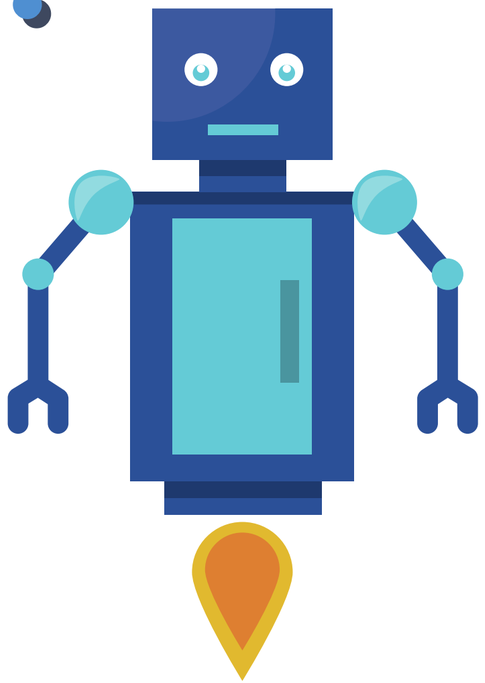

Interfaz de Usuario (Navegador Web)
Usuario
Muestra estado (Modo, Sensores, Bomba)
Toca los botones de control
InterfazWeb
Actualiza datos en la web (/data)
Envía comandos (ej. /control?cmd=A)
Se conecta a
Wemos D1 Mini
Crea Punto de Acceso Wi-Fi
Recibe comandos HTTP
Cerebro Wi-Fi (Wemos D1 Mini)
Wi-Fi
Envía cadena de datos por Serial (D:izq,cen,der...)
ProcesarComandoWemos
Envía carácter por Serial (A,B,S,I,D...)
Arduino Uno
Sensores de Flama (IZQ, CEN, DER)
Sensor Ultrasónico
Lee constantemente
Ejecuta el bucle principal (loop)


Controlador Principal (Arduino Uno)
Sensores
¿Fuego detectado?
Sí
No
Modo Automático
Inicia protocolo
¿Robot muy cerca?
Modo Manual
Sí
No
Espera comandos del Wemos
Apagar Fuego (Control PID Servo + Bomba)
Avanzar/Girar hacia la flama
ProcesarComandoArduino
Activa
Actuadores
Relé (Bomba de Agua)
Servomotor
Driver L298N (Motores)
Componentes Físicos del Robot
Robot

¿CÓMO FUNCIONA EL PD?

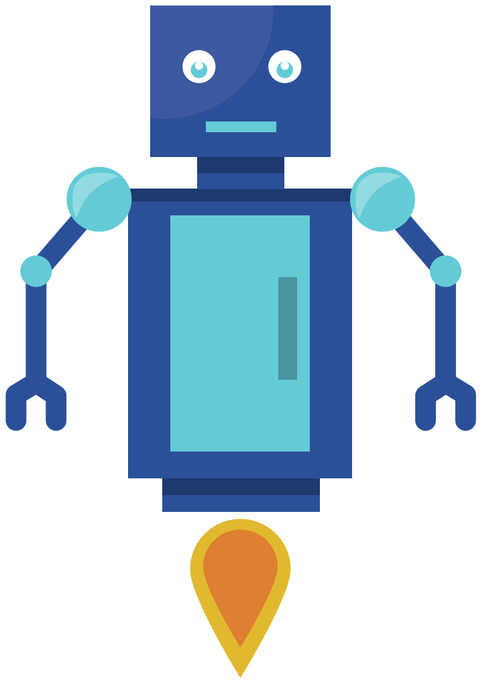
- P (Proporcional): Si el fuego está más cerca de un sensor que del otro, el servomotor gira con más fuerza hacia ese lado, Cuanto más grande sea la diferencia, más rápido se moverá.

- D (Derivativo): Esta parte sirve para que el robot no se pase de largo ni empiece a moverse de un lado a otro sin control.



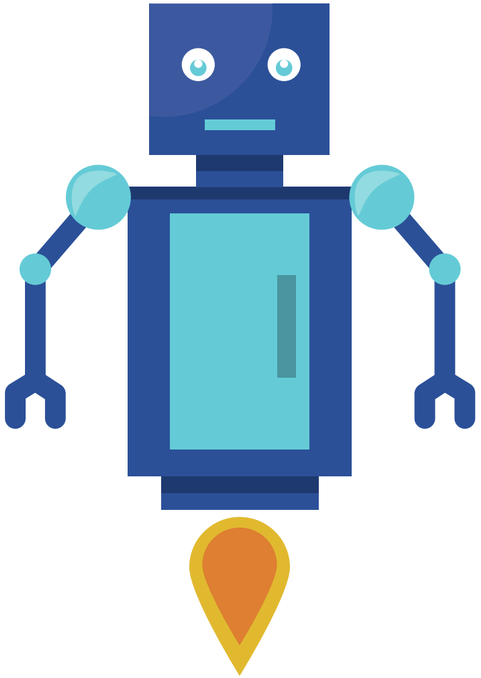
¿POR QUÉ ELEGÍ PD PARA CONTROLAR EL SERVO?

- Porque el robot necesita reaccionar rápido ante el fuego.
- El PD permite mover el servo de forma rápida pero sin pasarse.



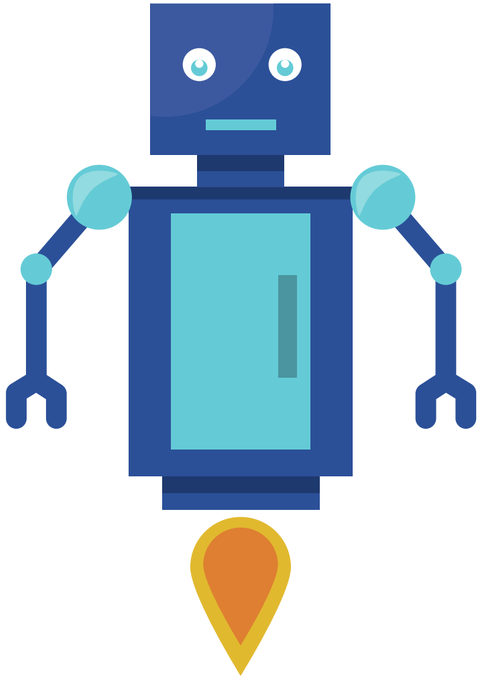
QUÉ HACEN KP Y KD?

- Kp = 0.05: sirve para que el robot sepa cuánto girar según el error. si es pequeño se vuelve lento.
- Kd = 0.04: Sirve para estabilizar el giro y evitar que se pase o se descontrole. Si es muy alto, se vuelve lento.
- Estos valores fueron ajustados por prueba y error para lograr un movimiento rápido pero estable.



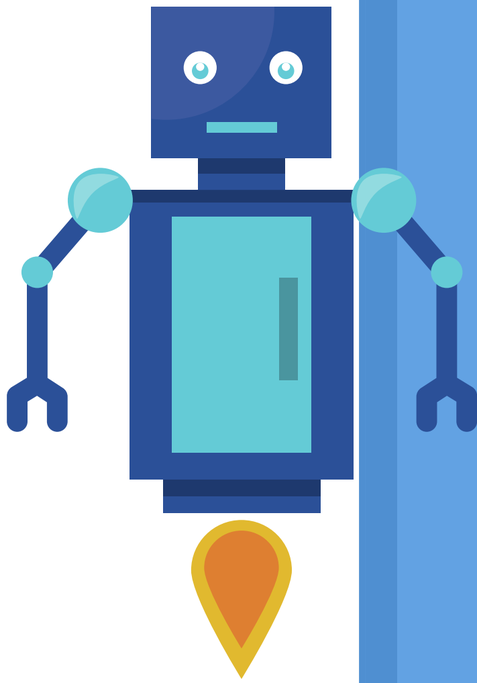
SEGURIDAD DEL SERVO

- “El ángulo está limitado a 60°–120° porque es el rango suficiente para cubrir el fuego desde los dos sensores laterales sin necesidad de girar más.”



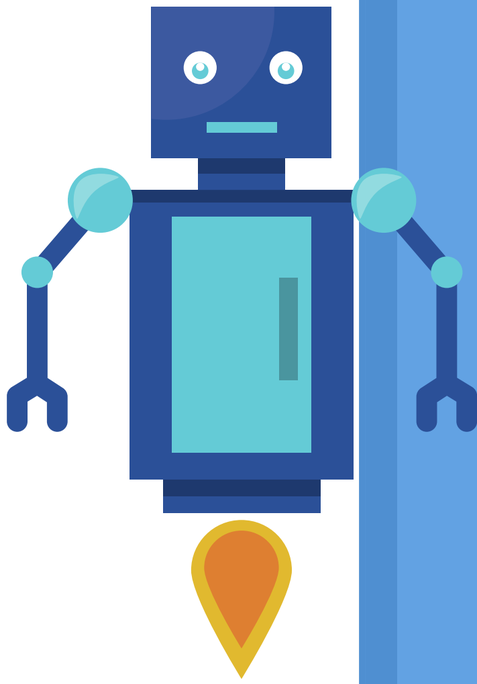
CODIGO DEL PD

```
// --- Lógica del Modo Automático ---
void seguirFuego() {
  // Si el robot está muy cerca, se detiene y activa el modo de extinción
  if (distancia_actual > 0 && distancia_actual < 20) {
    detener();
    apagarFuegoPD(); // Llamada a la nueva función con control PD
    return;
  }

  // Si los sensores detectan fuego muy cercano, activa el modo de extinción
  if (sensor_izquierdo < UMBRAL_CERCA || sensor_centro < UMBRAL_CERCA || sensor_derecho < UMBRAL_CERCA) {
    detener();
    apagarFuegoPD(); // Llamada a la nueva función con control PD
  } else if ((sensor_izquierdo < UMBRAL) || (sensor_centro < UMBRAL) || (sensor_derecho < UMBRAL)) {
    // Lógica para seguir el fuego
    if (sensor_centro < UMBRAL) adelante();
    else if (sensor_izquierdo < sensor_derecho) izquierda();
    else derecha();
  } else {
    detener();
  }
}
```



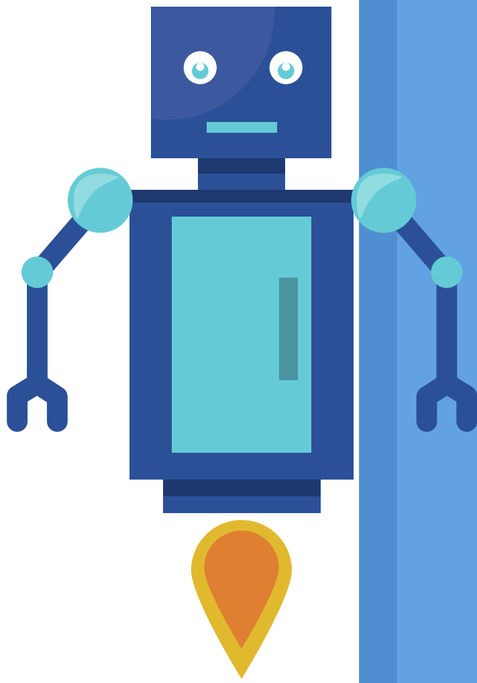
CODIGO DEL PD

```
//  Función de extinción con control PD ---
void apagarFuegoPD() {
  digitalWrite(BOMBA, HIGH);
  estado_bomba = "encendida";
  estado_servo = "movimiento";

  unsigned long tiempoInicio = millis();
  // Ejecuta el control PD durante 5 segundos para apuntar y extinguir
  while (millis() - tiempoInicio < 5000) {
    // Para una respuesta rápida del servo, leemos los sensores de nuevo aquí
    int val_izq = analogRead(IZQ);
    int val_der = analogRead(DER);

    // El error es la diferencia entre los sensores. Un valor más bajo indica más fuego.
    // Si val_izq < val_der, el fuego está a la izquierda -> error negativo.
    // El servo debe moverse a la izquierda (ángulo mayor).
    // La corrección de la posición será: pos_actual -= salida_pd
    error = val_izq - val_der;

    // Cálculo de la salida del controlador Proporcional-Derivativo
    salida_pd = (Kp * error) + (Kd * (error - error_previo));

    // Actualizar la posición del servo
    posicion_servo -= salida_pd;

    // Limitar la posición del servo a un rango seguro (ej. 60-120 grados)
    posicion_servo = constrain(posicion_servo, 60, 120);
```

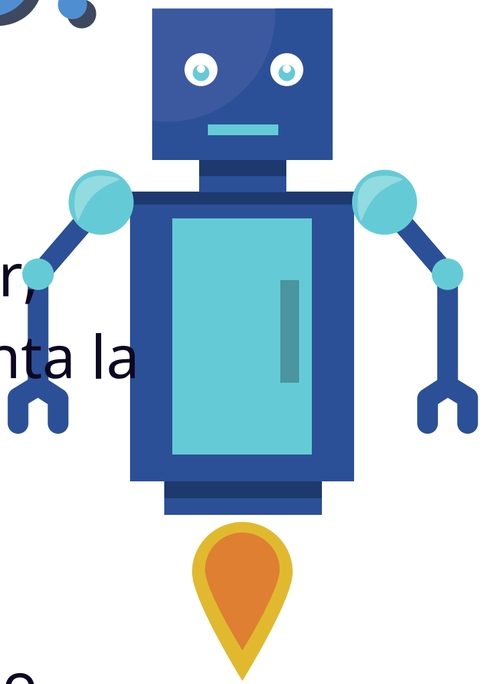


CODIGO DEL PD

```
203
204     // Enviar el comando al servo
205     servo.write(posicion_servo);
206
207     // Guardar el error actual para la siguiente iteración
208     error_previo = error;
209
210     delay(20); // Pequeña pausa para estabilizar el sistema y el ciclo de control
211   }
212
213   // Al terminar, apagar la bomba y centrar el servo
214   digitalWrite(BOMBA, LOW);
215   estado_bomba = "apagada";
216   estado_servo = "detenido";
217
218   // Regresar el servo a la posición central de reposo
219   posicion_servo = 90;
220   servo.write(posicion_servo);
221   error_previo = 0; // Reiniciar el error previo para la próxima activación
222 }
223
```



¿CÓMO FUNCIONA EL PID?



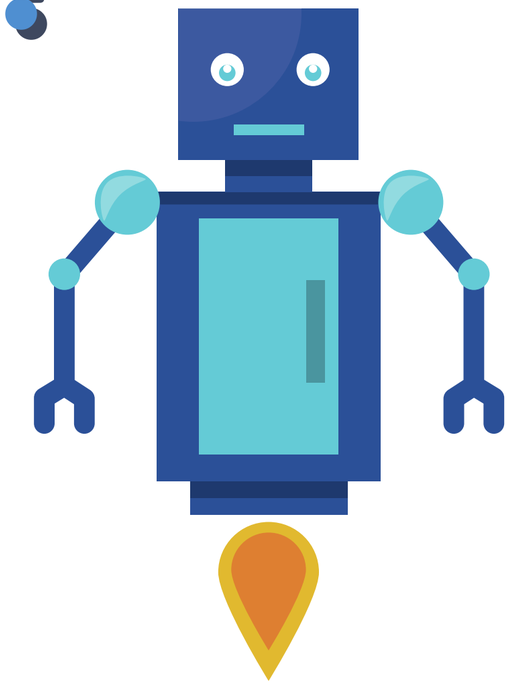
- Kp (Constante Proporcional): Definida como Kp = 0.07. Este término es responsable de la respuesta inicial del servomotor, corrigiendo el error presente. Un valor más alto de Kp aumenta la velocidad de reacción, pero si es demasiado elevado, puede causar que el servomotor oscile.

- Ki (Constante Integral): Definida como Ki = 0.001. Este término ayuda a corregir los errores acumulados a lo largo del tiempo, lo que es fundamental para eliminar el error estacionario, es decir, pequeños errores persistentes que el control proporcional por sí solo no puede corregir.

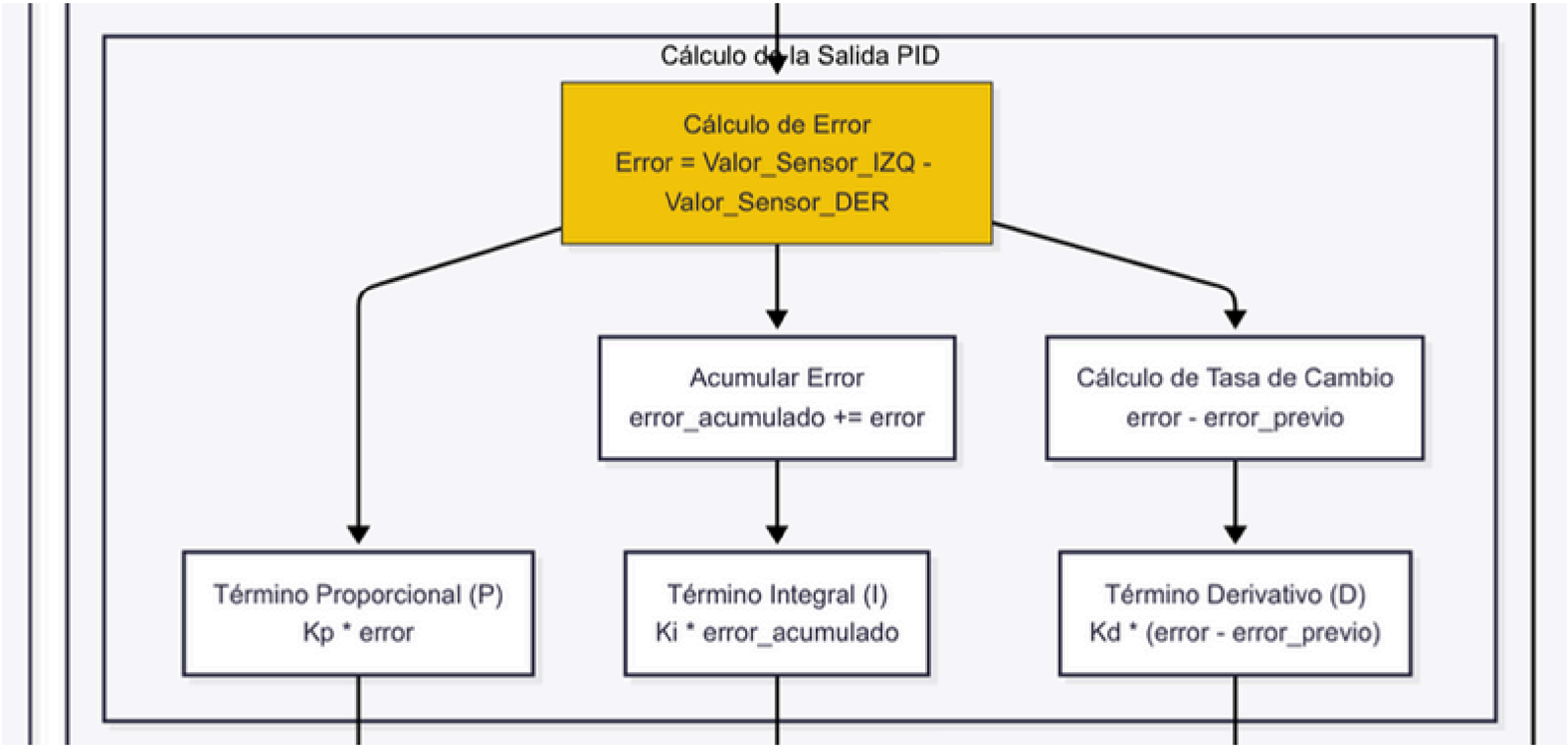
QUÉ HACEN KP, KD, KI?

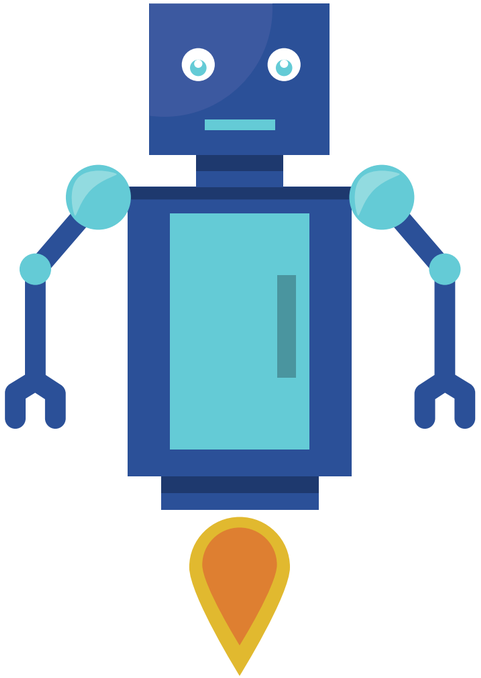
- Kp = 0.007: sensibilidad al error. Si es muy alto, oscila.
- Kd = 0.04: frena el movimiento. Si es muy alto, se vuelve lento.
- Ki = 0.001: predice el error.

- Estos valores fueron ajustados por prueba y error para lograr un movimiento rápido pero estable.



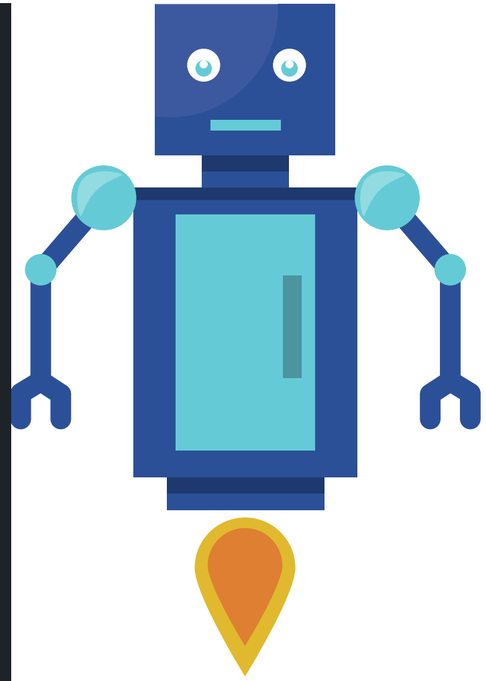
FLUJO


Cálculo de la Salida PID
Cálculo de Error
Error = Valor_Sensor_IZQ -
Valor_Sensor_DER
Acumular Error
error_acumulado += error
Cálculo de Tasa de Cambio
error - error_previo
Término Proporcional (P)
Kp * error
Término Integral (I)
Ki * error_acumulado
Término Derivativo (D)
Kd * (error - error_previo)



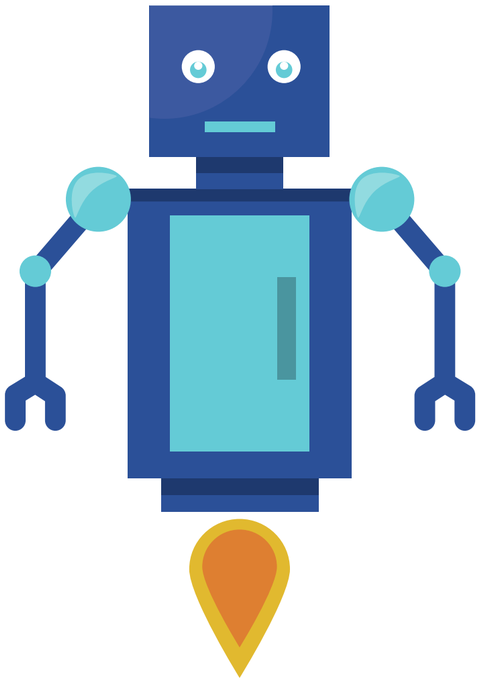
CODIGO DEL PID

```
// --- Lógica del Modo Automático ---
void seguirFuego() {
  // Si el robot está muy cerca, se detiene y activa el modo de extinción
  if (distancia_actual > 0 && distancia_actual < 20) {
    detener();
    apagarFuegoPID(); // Llamada a la nueva función con control PID
    return;
  }

  // Si los sensores detectan fuego muy cercano, activa el modo de extinción
  if (sensor_izquierdo < UMBRAL_CERCA || sensor_centro < UMBRAL_CERCA || sensor_derecho < UMBRAL_CERCA) {
    detener();
    apagarFuegoPID(); // Llamada a la nueva función con control PID
  } else if ((sensor_izquierdo < UMBRAL) || (sensor_centro < UMBRAL) || (sensor_derecho < UMBRAL)) {
    // Lógica para seguir el fuego
    if (sensor_centro < UMBRAL) adelante();
    else if (sensor_izquierdo < sensor_derecho) izquierda();
    else derecha();
  } else {
    detener();
  }
}
```



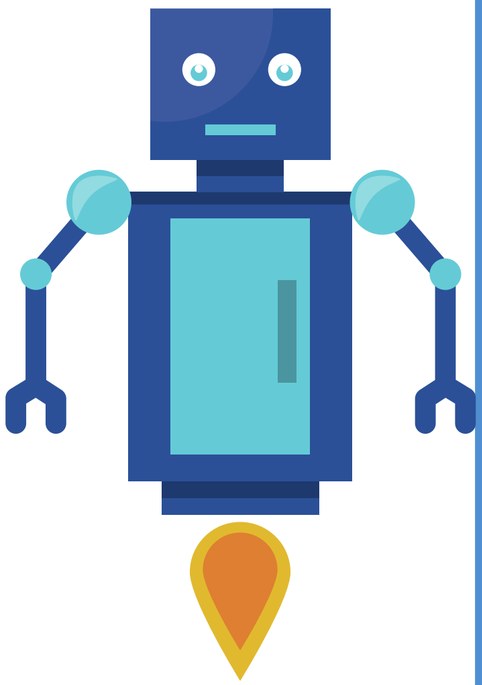
CODIGO DEL PID

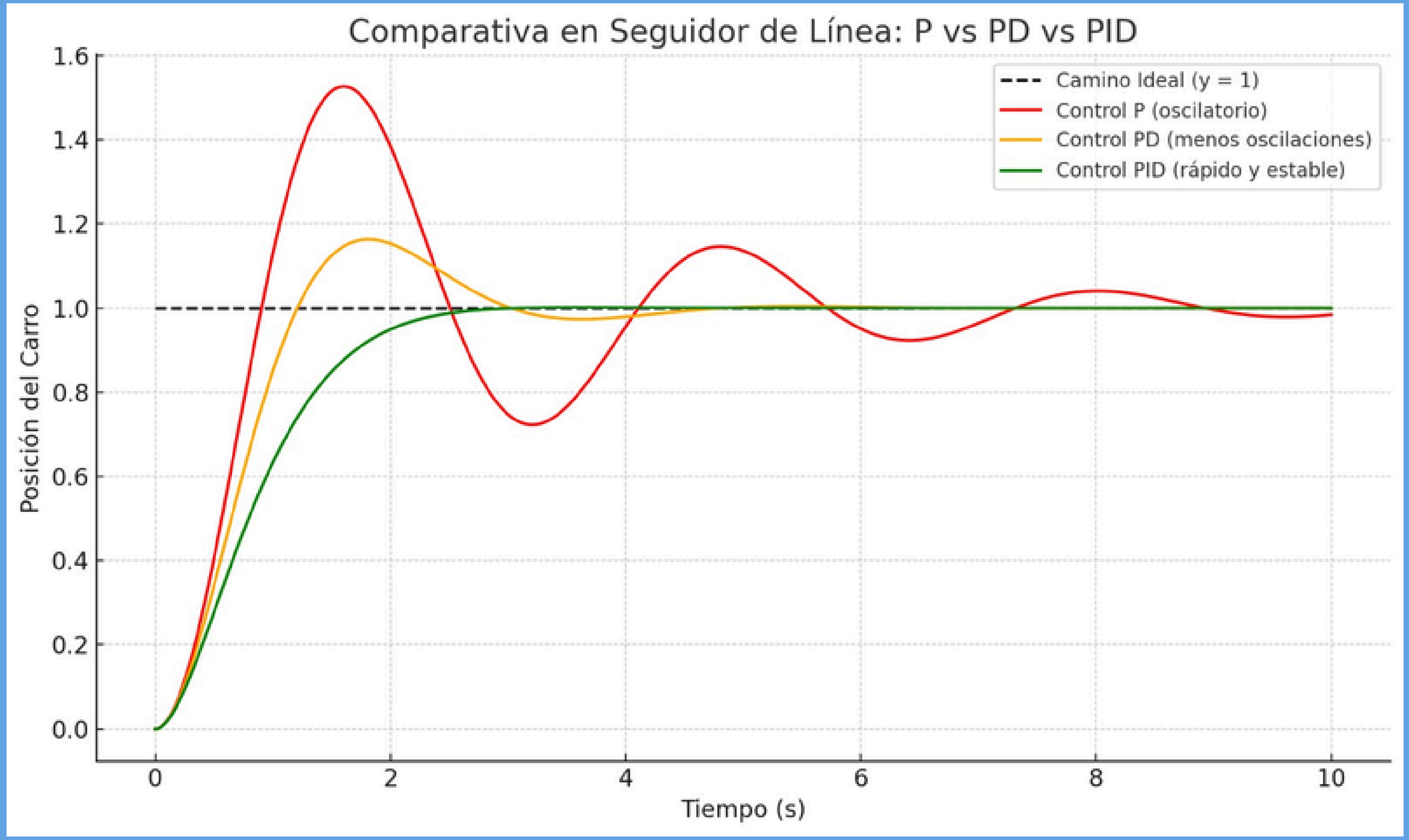
```
// --- Función de extinción con control PID ---
void apagarFuegoPID() {
  digitalWrite(BOMBA, HIGH);
  estado_bomba = "encendida";
  estado_servo = "movimiento";

  unsigned long tiempoInicio = millis();
  // Ejecuta el control PID durante 5 segundos para apuntar y extinguir
  while (millis() - tiempoInicio < 5000) {
    // Para una respuesta rápida del servo, leemos los sensores de nuevo aquí
    int val_izq = analogRead(IZQ);
    int val_der = analogRead(DER);

    // El error es la diferencia entre los sensores. Un valor más bajo indica más fuego.
    error = val_izq - val_der;

    // TÉRMINO INTEGRAL: Acumula el error a lo largo del tiempo.
    error_acumulado += error;

    // --- Cálculo de la salida del controlador PID ---
    salida_pid = (Kp * error) + (Ki * error_acumulado) + (Kd * (error - error_previo));

    // Actualizar la posición del servo
    posicion_servo -= salida_pid;

    // Limitar la posición del servo a un rango seguro (ej. 60-120 grados)
    posicion_servo = constrain(posicion_servo, 60, 120);
```



CODIGO DEL PID

```
    // Enviar el comando al servo
    servo.write(posicion_servo);

    // Guardar el error actual para la siguiente iteración (para el término derivativo)
    error_previo = error;

    delay(20); // Pequeña pausa para estabilizar el sistema y el ciclo de control
  }

  // Al terminar, apagar la bomba y centrar el servo
  digitalWrite(BOMBA, LOW);
  estado_bomba = "apagada";
  estado_servo = "detenido";

  // Regresar el servo a la posición central de reposo
  posicion_servo = 90;
  servo.write(posicion_servo);

  // Reiniciar errores para la próxima activación
  error_previo = 0;
  error_acumulado = 0;
}
```



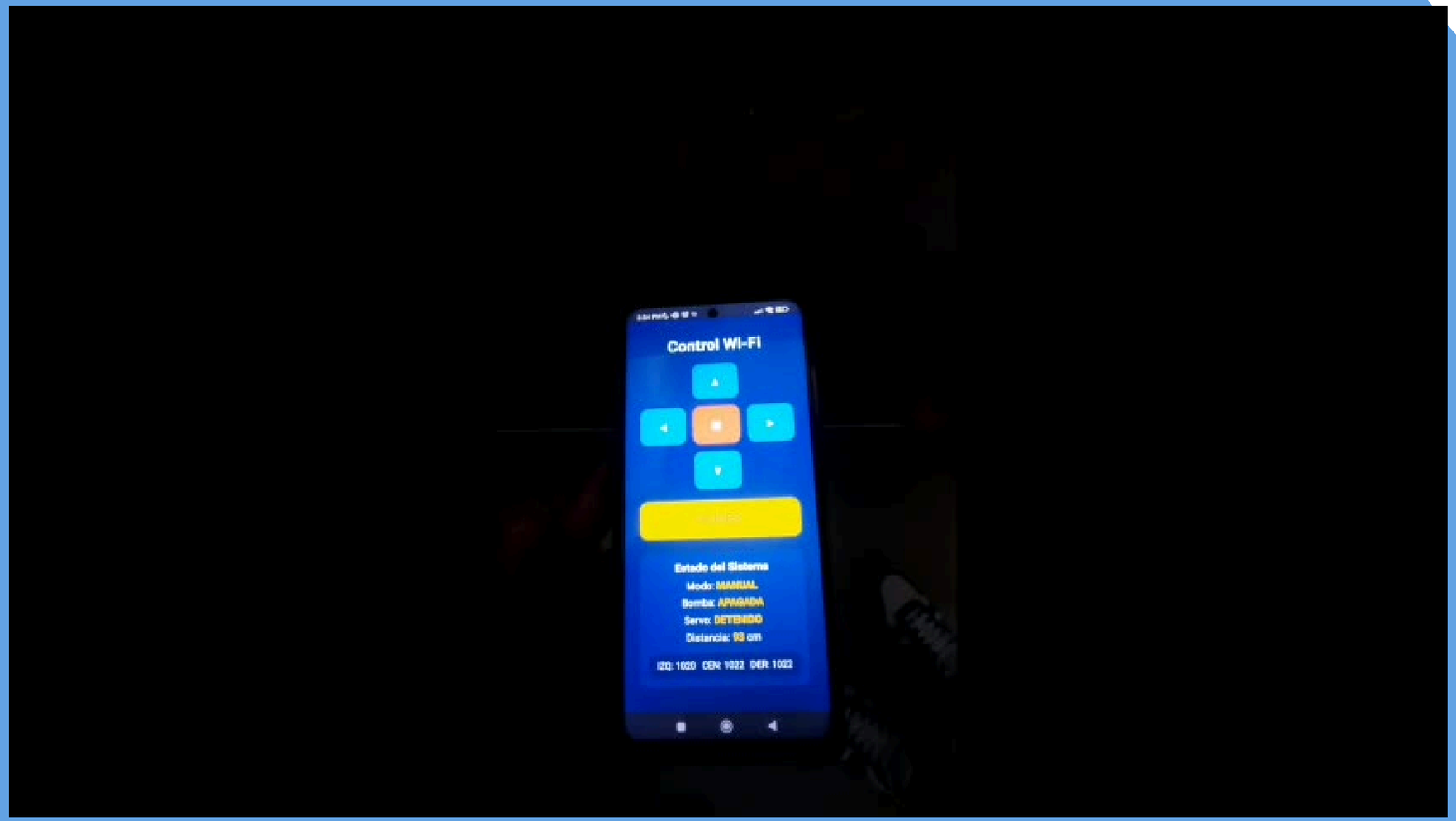

Comparativa en Seguidor de Línea: P vs PD vs PID
Camino Ideal (y = 1)
Control P (oscilatorio)
Control PD (menos oscilaciones)
Control PID (rápido y estable)
Posición del Carro
Tiempo (s)
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10

MODOS


Control Wi-Fi
▲
◄
■
►
▼
BOMBA
Estado del Sistema
Modo: MANUAL
Bomba: APAGADA
Servo: DETENIDO
Distancia: 9 cm
IZQ: 1014 CEN: 1015 DER: 1015


Control Wi-Fi
▲
◄
■
►
▼
BOMBA
Estado del Sistema
Modo: AUTO
Bomba: ENCENDIDA
Servo: DETENIDO
Distancia: 38 cm
IZQ: 346 CEN: 998 DER: 1011

FUNCIONAMIENTO



MUCHAS GRACIAS