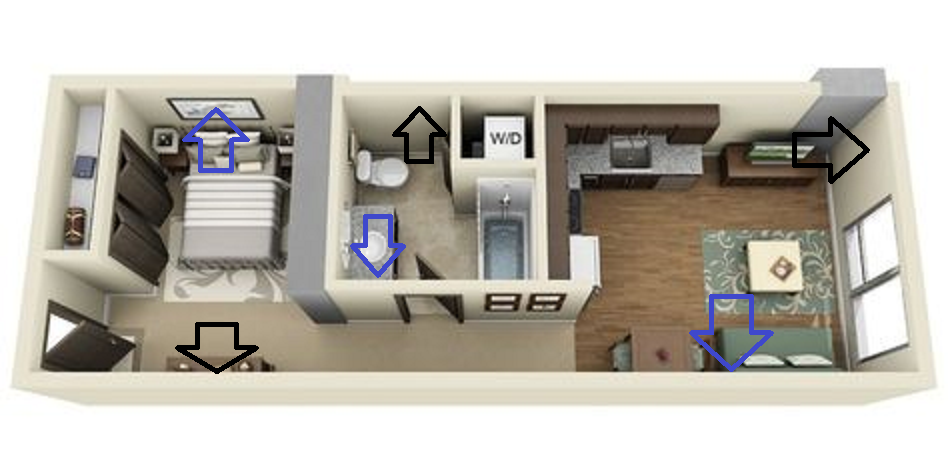
**Control de temperatura**

Para el desarrollo de este proyecto se tomó como referencia un departamento de dos ambientes que cuenta con un sistema de control de temperatura centralizado.

Locación donde se utilizará el sistema:  
  
  
  
  
  
En la imagen siguiente, las flechas negras indican la ubicación de los actuadores y las flechas azules la de los sensores.  
  


Para el modelado matemático se tomo en cuenta que un sistema derivativo (D) o PID representaría variaciones muy amplias debido que en los momentos donde se activa el actuador se produce un cambio repentino de temperatura, por lo tanto, el error es considerablemente grande en ese periodo de tiempo. Teniendo esto en cuenta, se procedió al desarrollo de un sistema proporcional e integral (PI). El código para el control del sistema es el siguiente:

import numpy as np

import matplotlib.pyplot as plt

class CasaInteligente:

    def \_\_init\_\_(self, temperatura\_inicial, K, tau):

        self.temperatura\_actual = temperatura\_inicial

        self.temperatura\_deseada = temperatura\_inicial

        self.historial\_temperaturas = [temperatura\_inicial]

        self.K = K

        self.tau = tau

        self.dt = 1.0 / 60  # Paso de tiempo en horas (1 minuto)

        self.error\_integral = 0

        self.error\_anterior = 0

    def actualizar\_temperatura(self, nueva\_temperatura):

        self.temperatura\_actual = nueva\_temperatura

        self.historial\_temperaturas.append(nueva\_temperatura)

    def controlador\_pi(self, Kp, Ki):

        error = self.temperatura\_deseada - self.temperatura\_actual

        self.error\_integral += error \* self.dt

        u = Kp \* error + Ki \* self.error\_integral

        self.error\_anterior = error

        return u

    def simular\_dia(self, temperatura\_deseada, Kp, Ki):

        self.temperatura\_deseada = temperatura\_deseada

        for \_ in range(24 \* 60):  # Simular durante 24 horas en intervalos de 1 minuto

            u = self.controlador\_pi(Kp, Ki)

            dydt = (self.K \* u - self.temperatura\_actual) / self.tau

            nueva\_temperatura = self.temperatura\_actual + dydt \* self.dt

            self.actualizar\_temperatura(nueva\_temperatura)

    def graficar\_temperaturas(self):

        plt.plot(np.arange(len(self.historial\_temperaturas)) / 60, self.historial\_temperaturas, label="Temperatura")

        plt.axhline(self.temperatura\_deseada, color="r", linestyle="--", label="Temperatura deseada")

        plt.xlabel("Hora")

        plt.ylabel("Temperatura (°C)")

        plt.title("Control de temperatura en una casa inteligente")

        plt.legend()

        plt.ylim(21,23)

        plt.xlim(0,1)

        plt.show()

def main():

    casa = CasaInteligente(27, K=10, tau=3)

    casa.simular\_dia(22, Kp=10, Ki=7)  #90,0.1,0

    casa.graficar\_temperaturas()

if \_\_name\_\_ == "\_\_main\_\_":

    main()

El gráfico del comportamiento queda de la siguiente manera:  
  
Gráfico

Descripción generada automáticamente

Como se ve en el gráfico, la curva es considerablemente suave como para no percibir cambios abruptos de temperatura.