Что такое цифровой двойник?

Цифровой двойник в производственных комплексах и технологических процессах – это виртуальная модель, которая точно воспроизводит реальный производственный объект или процесс. Эта концепция включает в себя следующие ключевые аспекты:

1. **Точное Отражение Реальности**: цифровой двойник создается на основе реальных данных о физическом объекте или процессе. Он детально воспроизводит все характеристики, структуру, процессы и динамику работы.
2. **Интеграция с Данными в Реальном Времени**: цифровой двойник постоянно синхронизируется с реальным объектом, получая данные через датчики, системы мониторинга и другие источники. Это позволяет отслеживать текущее состояние объекта и проводить анализ в реальном времени.
3. **Анализ и Оптимизация**: с помощью цифрового двойника можно анализировать производственные процессы, предсказывать неполадки, оптимизировать работу оборудования и процессов. Это способствует повышению эффективности и снижению затрат.
4. **Планирование и Тестирование**: Цифровые двойники используются для моделирования и тестирования изменений в производственном процессе, внедрения новых технологий или изменений в конструкции оборудования, не влияя на реальный процесс.
5. **Обучение и Тренировка**: они также могут служить обучающим инструментом для персонала, позволяя проводить тренировки на виртуальных моделях, не подвергая риску реальное оборудование.

**Слайд № \_\_. Блок-схема в Матлаб**

На представленной блок-схеме системы управления, которая, используется для моделирования работы Цифрового двойника производственных комплексов и технологических процессов. Схема включает в себя следующие ключевые элементы:

* **PID-контроллер** (Пропорционально-Интегрально-Дифференцирующий контроллер), настраиваемый с использованием коэффициентов KP, KI, KD для управления процессом.
* Входные сигналы для **PID-контроллера**, такие как заданное значение (set point) и измеренное значение (process variable), которые образуют сигнал ошибки (E(t)).
* **Устройства измерения**, обеспечивающие обратную связь для контроллера, и блоки сатурации, которые ограничивают выходной сигнал контроллера.
* **Трехпозиционные реле** для управления потоками газа и воздуха, а также для определения соотношения газ-воздух.
* **Актуаторы** для подачи газа и воздуха, а также датчики потока для мониторинга текущих уровней.
* Блоки для расчета **коэффициента расхода** и **технического воздушного потока**.
* Интеграция с **MATLAB - OPC** (OLE for Process Control), платформой для обмена данными между автоматизированными системами и MATLAB.

В нижней части схемы присутствует описание OPC, указывающее на то, что это технология для обмена данными между различными автоматизированными системами, включая ПЛК (программируемые логические контроллеры) и MATLAB, который может работать с данными OPC через специальные интерфейсы.

Эта блок-схема используется для обучения и тестирования различных сценариев управления в безопасной симулированной среде, что позволяет оптимизировать процессы и системы управления до их реального применения на производстве.

**Слайд с визуализацией с кодесис**

На разработанной схеме показана пользовательская интерфейсная панель системы визуализации, которая используется для управления и мониторинга процессов в рамках цифрового двойника производственного оборудования или технологического процесса. Панель содержит следующие элементы:

1. **Графики**:
   * Слева находится график, предназначенный для отображения текущего значения температуры, уровня задание в миллиамперах.
   * Справа находится график, который показывает установленное значение температуры в процентах и градусах Цельсия.
2. **Инструменты управления и навигации**:
   * Под графиками находятся панели с кнопками для управления отображением данных на графиках, такие как перемещение влево, вправо, увеличение масштаба, уменьшение масштаба, обновление данных и поиск.
3. **Переключатели источников данных**:
   * В нижней части панели есть переключатель, который позволяет выбирать между различными источниками данных, такими как «OPC Matlab», «Ручное управление» и «Сброс регулятора».
4. **Настройки PID-регулятора**:
   * Также в нижней части панели присутствуют поля для ввода параметров PID-регулятора: коэффициенты KP, KI и KD, которые, влияют на работу системы управления.
5. **Параметры трендов и пределы управления**:
   * Внизу справа расположены поля для ввода параметров линий тренда и пределов управления, такие как TLL (нижний предел управляющего сигнала) и THN (верхний предел управляющего сигнала), а также для настройки соответствующих процентных значений (обозначены как "%f").
6. **Индикаторы**:
   * По центру панели находятся индикаторы текущего состояния системы, в том числе уровень задания по температуре, текущая температура и уровень подачи энергии. Они помечены различными цветами: зеленым, черным и синим соответственно.

Эта панель представляет собой инструмент для визуального мониторинга и управления параметрами цифрового двойника, что позволяет оператору в реальном времени отслеживать и корректировать работу системы.

**Блок схема программы управления**

На блок-схеме представлена программа для управления технологическим параметром температурой, на основе ПИД-регулирования, реализованная на языке контроллеров функциональных блоков (CFC), соответствующего стандарту МЭК 61131-3. Рассмотрим каждый функциональный блок по порядку:

1. **Scaling\_mA\_act**:
   * Этот блок отвечает за преобразование входного аналогового сигнала (X\_mA), который представляет текущее значение температуры, из диапазона mA\_LL - mA\_HH (нижний и верхний пределы токового сигнала) в нормализованный диапазон TLL - THH (нижний и верхний пределы температуры).
2. **Переключатель (switch)**:
   * Позволяет выбирать между автоматическим режимом управления (act\_temp), когда система сама поддерживает температуру, и ручным заданием (zadanie), когда оператор сам устанавливает требуемое значение.
3. **PID\_Controller**:
   * Является основным блоком управления, использующим алгоритм ПИД-регулирования для поддержания температуры. Он получает отклонение (E\_t) от желаемого значения температуры и корректирует управляющий сигнал (Y), используя настраиваемые коэффициенты KP, KI и KD.
4. **Temp\_out**:
   * Выходной сигнал ПИД-регулятора, который затем будет использоваться для управления технологическим процессом.
5. **Scaling\_U\_mA**:
   * Этот блок преобразует управляющий сигнал из нормализованного вида обратно в аналоговый токовый сигнал (mA), который может быть использован управляющими устройствами.
6. **LIN\_TRAFO**:
   * Блоки линейного преобразования (обозначены номерами 1, 6 и 8) служат для преобразования входных и выходных сигналов из одного диапазона в другой, например, из диапазона измерения датчика в диапазон управления исполнительного механизма.
7. **mA**:
   * Выходной сигнал блока Scaling\_U\_mA, который представляет собой токовый сигнал, отправляемый к исполнительному механизму.
8. **Scaling\_mA\_proc**:
   * Преобразует аналоговый сигнал в процентное значение (от 0 до 100%), что может быть полезно для визуализации или дальнейшего управления в системах, где используются процентные значения.
9. **sig\_procent**:
   * Выходной сигнал блока Scaling\_mA\_proc, который представляет собой процентное значение управляющего сигнала, готового к отображению на интерфейсе пользователя или использованию в других целях управления.

Каждый блок взаимодействует с другими блоками для обеспечения точного и стабильного управления температурой, что является критически важным для поддержания оптимальных условий в производственных процессах. Переменные в этих блоках могут иметь типы данных Real (для представления числовых значений) и Bool (для логических операций и переключений между режимами управления).

**Описание архитектуры**

Представлена схема архитектуры системы управления производственным процессом или комплексом. В центре расположены четыре основных компонента системы:

1. **Главная ПО** - основное программное обеспечение, которое управляет всем процессом.
2. **ПО объекта управления (CODESYS)** - программное обеспечение для управления конкретным объектом или процессом.
3. **LOCAL Server** - локальный сервер, который выполняет роль центрального узла, обеспечивающего коммуникацию между различными частями системы.
4. **MATLAB** - программное обеспечение, используемое для технических расчётов и моделирования, играет роль симулятора объекта.

Связанные с этими четырьмя элементами компоненты:

* **OPC Server** - сервер, необходимый для обмена данными между устройствами и программным обеспечением. Этот компонент связывает главное ПО и ПО объекта управления с MATLAB и базой данных.
* **Data Base SQL** - база данных SQL, которая служит для хранения данных, получаемых и обрабатываемых системой.
* **Сервер для проекта** - отдельный компонент, предположительно используемый для развертывания проекта или для управления проектными данными.

На периферии схемы расположены элементы, которые взаимодействуют с пользователями:

* **UI** (Пользовательский интерфейс) - интерфейс для взаимодействия с системой, возможно, для мониторинга и управления процессами.
* **Инженер** - человек, отвечающий за настройку и обслуживание системы.
* **Оператор** - пользователь или оператор, работающий непосредственно с системой.

Каждый из этих элементов взаимодействует друг с другом по определённым правилам и протоколам, формируя интегрированную систему для автоматизации и управления производственными процессами.

**Код**

# Импортируем необходимые библиотеки

import numpy as np

# Определяем класс для ПИД-регулятора

class PIDController:

def \_\_init\_\_(self, kp, ki, kd, set\_point):

"""

Инициализация ПИД-регулятора с коэффициентами и заданным значением

kp, ki, kd - коэффициенты пропорциональной, интегральной и дифференциальной составляющих

set\_point - желаемое значение управляемой величины

"""

self.kp = kp

self.ki = ki

self.kd = kd

self.set\_point = set\_point

self.integral = 0

self.prev\_error = 0

def update(self, measured\_value, dt):

"""

Обновление ПИД-регулятора

measured\_value - измеренное значение управляемой величины

dt - временной шаг

"""

# Рассчитываем ошибку

error = self.set\_point - measured\_value

# Рассчитываем составляющие ПИД-регулятора

self.integral += error \* dt

derivative = (error - self.prev\_error) / dt

# Рассчитываем выходное значение

output = self.kp \* error + self.ki \* self.integral + self.kd \* derivative

# Обновляем предыдущую ошибку

self.prev\_error = error

return output

# Создаем экземпляры ПИД-регуляторов для температуры и соотношения газ-воздух

pid\_temperature = PIDController(kp=2.0, ki=0.0, kd=1.0, set\_point=100) # Задаем желаемую температуру в градусах

pid\_gas\_air\_ratio = PIDController(kp=1.0, ki=0.1, kd=0.05, set\_point=0.5) # Задаем желаемое соотношение газ/воздух

# Переменные для симуляции процесса

current\_temperature = 90 # Текущая температура

current\_gas\_air\_ratio = 0.6 # Текущее соотношение газ/воздух

dt = 0.1 # Временной шаг (100 мс)

# Функция симуляции управления процессом

def control\_process():

# Симуляция работы системы управления в течение 10 секунд

for \_ in range(int(10/dt)):

# Получаем выходные сигналы ПИД-регуляторов

temp\_control\_signal = pid\_temperature.update(current\_temperature, dt)

gas\_air\_control\_signal = pid\_gas\_air\_ratio.update(current\_gas\_air\_ratio, dt)

# Имитация воздействия управляющих сигналов на процесс

# Здесь должен быть код, который взаимодействует с реальными устройствами или симулятором процесса

# Например, изменение температуры и соотношения газ/воздух в зависимости от управляющих сигналов

# current\_temperature += temp\_control\_signal \* коэффициент\_мощности

# current\_gas\_air\_ratio += gas\_air\_control\_signal \* коэффициент\_регулировки

# Печать текущих значений для мониторинга

print(f"Температура: {current\_temperature}, Сигнал регулирования температуры: {temp\_control\_signal}")

print(f"Соотношение газ/воздух: {current\_gas\_air\_ratio}, Сигнал регулирования соотношения газ/воздух: {gas\_air\_control\_signal}")

# Запускаем процесс управления

control\_process()

В этой программе:

* **PIDController** - класс, инкапсулирующий логику работы ПИД-регулятора.
* **pid\_temperature** и **pid\_gas\_air\_ratio** - два экземпляра класса **PIDController**, один для управления температурой, другой для управления соотношением газ-воздух.
* **control\_process** - функция, симулирующая процесс управления, в которой происходит чтение текущих значений температуры и соотношения газ-воздух, их коррекция с помощью ПИД-регуляторов и вывод текущих значений для мониторинга.