Información Relevante del Proyecto: Sistema de Control de Tanques

Diseñado y Desarrollado por el Ing. Juan David Sandoval Valencia, Ingeniero en Electrónica

El presente documento detalla la arquitectura de hardware, el tipo de sensores y actuadores, y el funcionamiento del código implementado en el sistema de control de tanques, enfatizando la selección de componentes y el control de procesos.

1. Plataforma de Desarrollo: Arduino Mega 2560

El proyecto se basa en la plataforma **Arduino Mega 2560**, la cual es ideal para este tipo de aplicaciones debido a su gran número de pines digitales (54), pines analógicos (16) y múltiples puertos seriales. Esta capacidad de pines es crucial para gestionar la cantidad de sensores, actuadores y periféricos presentes en el sistema de control de tanques.

2. Pines y Componentes Clave del Sistema

Se ha definido la siguiente asignación de pines y configuración de componentes:

2.1. Salidas PWM para Control de Colores (Bombas/Válvulas)

Estos pines, compatibles con la funcionalidad PWM del Arduino Mega 2560, se emplean para el control proporcional de la cantidad de cada líquido de color. Esto se logra mediante la modulación de la velocidad de bombas o la apertura gradual de válvulas, permitiendo un control preciso de la dosificación.

- **Cian (C):** Pin Digital 3 (PWM)
- **Magenta (M):** Pin Digital 5 (PWM)
- **Amarillo (Y):** Pin Digital 6 (PWM)
- **Negro (K):** Pin Digital 9 (PWM)
- **Blanco (W):** Pin Digital 10 (PWM)

2.2. Entradas Analógicas para Sensores de Nivel

Los sensores de nivel son de tipo **analógico** y se conectan a las entradas analógicas del Arduino Mega 2560. El microcontrolador utiliza su **Conversor Analógico-Digital (ADC)** interno para transformar la señal de voltaje continua del sensor en un valor

digital (0-1023). Este valor es posteriormente escalado a un porcentaje de nivel (0-100%) para una interpretación más intuitiva.

- **Nivel Cian (C):** Pin Analógico A0
- **Nivel Magenta (M):** Pin Analógico A1
- **Nivel Amarillo (Y):** Pin Analógico A2
- **Nivel Negro (K):** Pin Analógico A3
- **Nivel Blanco (W):** Pin Analógico A4

2.3. Entradas Analógicas para Sensores de Temperatura

Los sensores de temperatura también son de tipo **analógico**. Su funcionamiento es análogo al de los sensores de nivel: la señal de voltaje analógica es leída por el **ADC** del Arduino Mega 2560 y convertida a un valor digital. Este valor digital se escala a grados Celsius mediante un factor de conversión (0.48828125 V/bit, asumiendo un sensor de 10mV/°C y una referencia de 5V para el ADC), permitiendo la medición precisa de la temperatura en cada tanque.

- **Temperatura Cian (C):** Pin Analógico A5
- **Temperatura Magenta (M):** Pin Analógico A6
- **Temperatura Amarillo (Y):** Pin Analógico A7
- **Temperatura Negro (K):** Pin Analógico A8
- **Temperatura Blanco (W):** Pin Analógico A9

2.4. Salidas Digitales para Calentadores (Resistencias Calefactoras)

Estos pines digitales controlan el estado de las resistencias calefactoras. Es crucial señalar que, aunque se utiliza la función `analogWrite()` en el código para el control PID, los pines específicos asignados para los calentadores (2, 4, 7, 8, 11) **son pines PWM en el Arduino Mega 2560**. Esto permite un control de potencia modulado (0-255) a las resistencias, lo cual es fundamental para la implementación del control PID.

- **Calentador Cian (C):** Pin Digital 2 (PWM)
- **Calentador Magenta (M):** Pin Digital 4 (PWM)
- **Calentador Amarillo (Y):** Pin Digital 7 (PWM)
- **Calentador Negro (K):** Pin Digital 8 (PWM)
- **Calentador Blanco (W):** Pin Digital 11 (PWM)

2.5. Salidas Digitales para Alarmas

Estos pines controlan indicadores visuales (LEDs) y un indicador auditivo (buzzer) para alertar sobre niveles de líquido bajos en los tanques.

- **LED Alarma Cian (C):** Pin Digital 12
- **LED Alarma Magenta (M):** Pin Digital 22
- **LED Alarma Amarillo (Y):** Pin Digital 23
- **LED Alarma Negro (K):** Pin Digital 24
- **LED Alarma Blanco (W):** Pin Digital 25

- **Zumbador (Buzzer):** Pin Digital 26

2.6. Componentes del Agitador

El sistema incorpora un agitador con control de velocidad y activación manual.

- **Control de Velocidad del Agitador:** Pin Digital 27 (PWM) - Permite ajustar la velocidad del motor del agitador.
- **Botón del Agitador:** Pin Digital 28 (Entrada con pull-up interno) - Permite la activación manual del agitador.

2.7. LED Integrado

- **LED de la Placa:** Pin Digital 13 - Utilizado para feedback visual general y en la función `blinkLED`.

3. Funcionamiento del Código de Arduino (Arduino Mega 2560)

El código implementado por el Ing. Juan David Sandoval gestiona de manera integral el sistema de control de tanques.

3.1. `setup()`: Inicialización del Sistema

La función `setup()` se ejecuta una sola vez al inicio y configura todos los pines como entradas o salidas, según su función. Se inicializa la comunicación serial para el monitoreo y control, y se establece el tiempo inicial para los cálculos PID.

3.2. `loop()`: Bucle Principal de Operación

La función `loop()` se ejecuta repetidamente y contiene la lógica principal del sistema:

- **`processSerialCommands()`:** Monitorea la comunicación serial para recibir comandos externos. Estos comandos pueden incluir la configuración de los niveles de dosificación de color (CMYKW) o la modificación de los setpoints de temperatura para cada tanque.
- **`readSensors()`:** Llama a `sendSensorData()` cada 5 segundos para leer y transmitir las temperaturas y niveles actuales de todos los tanques al puerto serial.
- **`controlTemperature()`:** Esta es la función central para la regulación de temperatura. Para cada tanque, lee la temperatura actual del sensor y utiliza un algoritmo **PID** para calcular la potencia necesaria para el calentador.
- **`checkLevels()`:** Verifica el nivel de líquido en cada tanque. Si el nivel cae por debajo del `LEVEL_LOW` (20%), activa el LED de alarma correspondiente y el zumbador para alertar sobre la condición de bajo nivel.
- **`checkAgitatorButton()`:** Monitorea el estado del botón del agitador. Mientras el botón está presionado, activa el motor del agitador a una velocidad predefinida

(AGITATOR_SPEED = 200) mediante PWM. Cuando el botón se suelta, el agitador se desactiva.
- **delay(100):** Introduce un pequeño retardo para estabilizar las lecturas y el procesamiento.

3.3. Funciones Clave

- **sendSensorData():** Calcula y envía las temperaturas y niveles actuales al monitor serial. La conversión de la lectura analógica del sensor a grados Celsius se realiza multiplicando el valor digital por 0.48828125.
- **getLevel(int pin):** Lee el valor analógico de un pin de nivel y lo mapea a un porcentaje de 0 a 100.
- **computePID(...):** Esta es la implementación del controlador Proporcional-Integral-Derivativo (PID).
- **Error (Proporcional):** error = setpoint - input; Calcula la diferencia entre la temperatura deseada (setpoint) y la temperatura actual (input). Un error grande genera una salida de control grande.
- **Error Acumulado (Integral):** *cumError += error * elapsedTime; Suma los errores a lo largo del tiempo. Ayuda a eliminar el error en estado estacionario (offset) y a compensar perturbaciones persistentes.
- **Tasa de Cambio del Error (Derivativo):** double rateError = (error - *lastError) / elapsedTime; Calcula la velocidad a la que cambia el error. Esto permite anticipar el comportamiento futuro del error y amortiguar oscilaciones.
- **Cálculo de la Salida:** output = Kp * error + Ki * (*cumError) + Kd * rateError; La salida del PID es una combinación ponderada de los tres términos, controlada por las ganancias Kp, Ki y Kd.
- **Restricción de la Salida:** output = constrain(output, 0, 255); La salida del PID se limita al rango 0-255, adecuado para el control PWM de los calentadores.

4. Control de Actuadores y Componentes de Potencia

Se destaca la importancia de los componentes de potencia intermedios, ya que el Arduino Mega 2560 no puede alimentar directamente cargas de alta corriente como resistencias calefactoras o motores.

4.1. Resistencias Calefactoras: Uso de MOSFETs

Para el control de las resistencias calefactoras, se debe emplear un **MOSFET de Canal N (N-Channel) de Nivel Lógico (Logic-Level)** por cada resistencia. Este tipo de MOSFET actúa como un interruptor electrónico de alta eficiencia.

- **Función:** El MOSFET permite que una fuente de alimentación externa (de mayor corriente y/o voltaje) suministre energía a la resistencia, bajo el control de la señal PWM de bajo voltaje del Arduino Mega 2560.
- **Ventajas del Logic-Level:** Se activa completamente (satura) con los 5V de salida del Arduino, lo cual es crucial para minimizar la disipación de calor en el MOSFET.
- **Conexión Típica del MOSFET (N-Channel, Low-Side Switching):**
- El **Gate (G)** del MOSFET se conecta al pin PWM del Arduino (ej., cHeaterPin).

- El **Drain (D)** del MOSFET se conecta a un terminal de la resistencia calefactora. El otro terminal de la resistencia se conecta al polo positivo de la fuente de alimentación externa.
- El **Source (S)** del MOSFET se conecta a la tierra común (GND) del sistema (Arduino y fuente de alimentación externa).
- **Es fundamental que la tierra (GND) del Arduino Mega 2560 esté conectada a la tierra (GND) de la fuente de alimentación externa** para asegurar el correcto funcionamiento del MOSFET.
- **MOSFETs Recomendados:**
- **IRLZ44N:** MOSFET de canal N de nivel lógico. Soporta hasta 55V (VDS) y aproximadamente 47A (ID), con una baja resistencia en estado ON (RDS(on) de aproximadamente 0.022Ω a VGS=5V). Es robusto y ampliamente disponible.
- **FQP30N06L / P30N06LE:** Otra excelente opción de canal N de nivel lógico, con VDS de 60V, ID de 30A y RDS(on) de aproximadamente 0.035Ω a VGS=5V.
- **Módulos MOSFET para Arduino:** Para simplificar el cableado, se recomienda el uso de módulos pre-ensamblados que contengan MOSFETs de nivel lógico, como los basados en el IRLZ44N, que ya incluyen las resistencias de protección y bornes de conexión.
- **Disipación de Calor:** Si la corriente consumida por las resistencias es considerable (superior a 1-2 amperios), el MOSFET deberá ser acompañado de un **disipador de calor** para evitar el sobrecalentamiento y asegurar una operación confiable.

4.2. Motor del Agitador: Driver de Motor

El motor del agitador, al ser una carga inductiva y requerir una corriente mayor a la que un pin del Arduino puede proporcionar, necesitará un **driver de motor** externo.

- **Tipo de Driver:** Aunque el código actual solo controla la velocidad en un sentido con PWM, el driver puede ser un **Módulo MOSFET** (similar al de los calentadores, si solo se controla la velocidad en una dirección) o un **Puente H** si se quisiera implementar control de dirección en el futuro.
- **Función:** El driver recibe la señal PWM del Arduino Mega 2560 y utiliza la corriente de una fuente de alimentación externa para energizar el motor, controlando su velocidad.