#include <math.h>

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

LiquidCrystal\_I2C lcd (0x27, 16,2);

int relayPin = 10; // Transformer serial port

int PWM\_A = 3;

int PWM\_B = 11;

int DIR\_A = 12;

int DIR\_B = 13;

int BRAKE\_A = 9;

int BRAKE\_B = 8;

int ZL\_relay1 =1

int ZL\_relay2 = 2 ;

int ntc\_1 = A0; // Set the reactor temperature sensor IO port

int ntc\_2 = A1; // Set the mold temperature sensor IO port

int shuiweiPin = A2; // water level sensor IO port

int renewButton = 7; // total button

int alarmLed = 4; // reaction part alarm indicator

int Led1 = 5; // Reaction part work indicator

int Led2 = 6; // Concentrated crystallization part of the work indicator

int sw; // water level

int b = 1; // b, c is used as a digital switch

int a = 1;

int Go;

float val [9]; // array of temperature values

float celsius

float celsius\_sum = 0;

unsigned long startTime, nowTime, interval;

/ \* Reactor with Mold Temperature Read \* /

float ntcRead\_1 ()

{

 float hq = 5.0 \* (analogRead (ntc\_1)) / 1024;

 float x = 30 \* hq / (5.0 - hq);

 float hs = log (x / 10);

 float temp = 1 / (hs / 3950 + 1 / 298.15) -273.15;

 return temp;

}

float ntcRead\_2 ()

{

 float hq = 5.0 \* (analogRead (ntc\_2)) / 1024;

 float x = 30 \* hq / (5.0 - hq);

 float hs = log (x / 10);

 float temp = 1 / (hs / 3950 + 1 / 298.15) -273.15;

 return temp;

}

float readTemp\_1 () // reactor temperature read

{

    for read (int i = 0; i <= 8; i ++) // read nine temperature values ​​in one second and get the temperature average

   {

      val [i] = ntcRead\_1 ();

      delay (100);

      celsius = val [i] \* (5.0 / 1023 \* 100);

      celsius\_sum = celsius + celsius\_sum;

    }

        celsius\_sum = celsius\_sum / 9.0;

        delay (100);

        return celsius\_sum

}

float readTemp\_2 () // Concentrate the crystallization temperature read

{

   for (int i = 0; i <= 8; i ++)

   {

      val [i] = ntcRead\_2 ();

      delay (100);

      celsius = val [i] \* (5.0 / 1023 \* 100);

      celsius\_sum = celsius + celsius\_sum;

    }

        celsius\_sum = celsius\_sum / 9.0;

        delay (100);

        return celsius\_sum

}

void MotorA\_run () // Create a motor run stop function

{

  analogWrite (PWM\_A, 180);

  digitalWrite (DIR\_A, HIGH);

  digitalWrite (BRAKE\_A, LOW);

}

void MotorA\_stop ()

{

  digitalWrite (DIR\_A, HIGH);

  digitalWrite (BRAKE\_A, HIGH);

}

void MotorB\_run ()

{

  analogwrite (PWM\_B, 180);

  digitalWrite (DIR\_B, HIGH);

  digitalWrite (BRAKE\_B, LOW);

}

void MotorB\_stop ()

{

  digitalWrite (DIR\_B, HIGH);

  digitalWrite (BRAKE\_B, HIGH);

}

void MotorC\_run ()

{

  digitalWrite (relayPin, LOW);

}

void MotorC\_stop (){

 digitalWrite(relayPin,HIGH) ;

/ \* Water level read \* /

int readShuiwei ()

{

  int swVal;

  swVal = analogRead (shuiweiPin);

  return swVal;

}

/ \* Reaction part operation status judgment \* /

void Open ()

{

     if (digitalRead (renewButton) == HIGH)

       {

         Go = 1;

       }

     else

      {

        Go = 0;

      }

}

/ \* Temperature control system \* /

void tempControl\_1 () // reaction temperature control

{

  float tempvalue = readTemp\_1 ();

  if (tempvalue> = 31.5) // The optimum temperature is 30

  {

    digitalWrite (ZL\_relay1, LOW);

  }

  else if (tempvalue <= 28.5)

  {

    digitalWrite (ZL\_relay1, HIGH);

  }

}

void tempControl\_2 ()

{

  float tempvalue = readTemp\_2 ();

  if (tempvalue> = 11.5)

  {

   digitalWrite (ZL\_relay2, HIGH);

  }

  else if (tempvalue <= 8.5)

  {

   digitalWrite (ZL\_relay2, LOW);

  }

}

        / \* Main program \* /

void setup ()

{

  lcd.init (); // LCD initialize, turn on the backlight, clear the screen

  delay (20);

  lcd.init ();

  delay (20);

  lcd.backlight ();

  lcd.clear ();

  lcd.setCursor (0,0); // locate the first row of the first column

  lcd.print ("Welcome");

  lcd.setCursor (5,1); // locate the second row of the fifth column

  lcd.print ("Start Soon");

  pinMode (relayPin, OUTPUT);

  pinMode (PWM\_A, OUTPUT); / / set the interface output mode

  pinMode (DIR\_A, OUTPUT);

  pinMode (BRAKE\_A, OUTPUT);

  pinMode (DIR\_B, OUTPUT);

  pinMode (PWM\_B, OUTPUT);

  pinMode (BRAKE\_B, OUTPUT);

  pinMode (renewButton, INPUT);

  pinMode (Led1, OUTPUT);

  pinMode (Led2, OUTPUT);

  pinMode (alarmLed, OUTPUT);

  digitalWrite (alarmLed, HIGH);

}

void loop ()

{

  float tempvalueA = readTemp\_1 ();

  float tempvalueB = readTemp\_2 ();

  delay (1500);

  lcd.clear ();

  lcd.setCursor (0,0); // display the temperature value in the first row of the first column of the LCD

  lcd.print ("TempA:");

  lcd.print (tempvalueA);

  lcd.setCursor (1,1);

  lcd.print ("TempB:");

  lcd.print (tempvalueB);

  tempControl\_2 (); // Crystallization temperature control starts

  if (a == 1)

  {

    startTime = millis (); // record the time to enter the working system

  }

     if (b)

      {

         Open (); / / determine the current operating status

      }

  if (Go) // does not initially have an alarm, so after two seconds, the device starts. But then every time the alarm is triggered, the device will work when the alarm is released manually (by pressing the button).

   {

    a = 0; // make "startTime" only record the moment when you just started working

    b = 0;

    tempControl\_1 (); // reaction temperature control starts

    MotorA\_run (); // Reaction part motor A work, warning light extinguishes, reaction part green light 1 lights up

    digitalWrite (alarmLed, LOW);

    digitalWrite (Led1, HIGH);

    nowTime = millis (); // record the current moment

    if (nowTime> = startTime)

    {

      interval = nowTime - startTime;

    }

    else

    {

      interval = nowTime + 33720000 - startTime; // because the system chronograph millis () will be reset to 33720000ms

    }

    if (interval> = 5000) // When the reaction time reaches 5 seconds after the predetermined period, the reaction part stops working and the diversion starts

      {

        MotorA\_stop (); // Reaction part motor downtime

        digitalWrite (Led1, LOW); // The reaction part is off and the green indicator is off

        MotorB\_run (); // Diversion motor starts

        digitalWrite (alarmLed, HIGH); / / reaction of a shutdown, red LED alarm issued to remind the liquid

        delay (3000); // diversion time

        MotorB\_stop (); // diversion motor B stops working

        Go = 0;

        a = 1;

        b = 1;

      }

   }

   else

   {

    MotorA\_stop (); // The reaction part is shut down and the indicator light is off

    digitalWrite (Led1, LOW);

   }

     / \* Crystallization work \* /

 sw = readShuiwei (); // read the water level

 (sw> = 300) // When the reaction part reacts and completes the diversion, the crystallization part starts to work and stops when the water level drops below 300

   {

     digitalWrite (Led2, HIGH); // Crystallization work, green light is on

     MotorC\_run (); // Concentrate the crystallization part pressure pump operates (via relay)

   }

   else

   {

    digitalWrite (Led2, LOW);

    MotorC\_stop (); // Concentrated crystallization pressure pump Shuts down (via relay)

   }

}