Arduino Nano Code for HVAC Protection Circuit
Version 0.1, December 2025

File: ArduinoHVACController.ino, inside subdirectory ArduinoHVACController
(Per Arduino integrated development environment requirements)

```
//////////////////LIBRARIES TO INCLUDE////////////////////////////////
// include the library for the Liquid Crystal display
#include <LiquidCrystal.h>    // standard lib for liquid crystal
#include <Arduino.h>          // basic Arduino functions
#include <avr/wdt.h>          // Allows us to have a watchdog timer


//////////////ARDUINO  OUTPUTS//actual board//////////////////////////
#define HVACPOWERED   6 //  D6  shows when circuit is up and allows power to HVAC
                    //  send HIGH to LED & resistor;
#define DELAYOUTPUT   7 //  D7  shows during the time that delay is active
                    //  sends HIGH to LED and resistor;
#define NANOLED       13//  D13 is onboard LED
#define ACPWRINPUT    A0 // A0 allows reading isolated AC power input
#define DELAYCONTROL  A1 // A1 allows reading delay request during SETUP
#define RELAYOUTPUT   5 //  Goes to Solid State Relay to control AC
#define DELAYTIME     100 //  use 1500 for 3 minutes
#define ERRORBOUND    20  //  error bound on change in 4 measurements




//////////////////////STATES////////////////////////////////

#define ACGONE        0
#define ACGOOD        1




/*****************************LOGIC
************************************************

SETUP
Setup turns off the transmitter power by setting ALC to approx -4 VDC
Setup turns off the ON light
Setup turns off the DELAY light

*************************************************************************************
**********/
```

```
//////////////////////////STRUCTURES & VARIABLES ////////////////////////////////////
unsigned long   current_milliseconds;        // the current number of milliseconds since SEND output
int             delay_count;                 // the desired delay in terms of count
                                             // approx 500 counts / minute
int             lockoutcounter=DELAYTIME;    // downcounter checking power good
int             refvoltagereading[200];      // reference voltages
int             normalpeakreading;
int             normalaveragereading;
int             failuremode;  // track how it glitched
                        // 1 = low average 1/4 cycle
                        // 2 = low average, 1/2 cycle
                        // 3 = high average, 1/2 cycle
                        // 4=  low peak voltage, 1/2 cycle
                        // 5 = high peak voltage, 1/2 cycle
                        // "cycle" = 16msec, full 60 hz cycle.  We are using fullwave rectified
                        // so we get 120Hz ripple --> we check 8 msec equivalent to half cycle
                        // of original AC

LiquidCrystal lcd(8, 9, 10, 11, 12, 13);  // The GLG board -- sequencer, rotator controller, arduino
winkeyer etc

////////////////// SET UP ROUTINE (EXECUTED ONCE ON STARTUP ////////////////////////////////////////
void setup() {
  int i;  // general purpose counter
  int j;  // another

  pinMode(RELAYOUTPUT,  INPUT);  // Don't allow this any ability to write until needed!
  pinMode(ACPWRINPUT,   INPUT);
  pinMode(HVACPOWERED, OUTPUT);
  pinMode(DELAYOUTPUT, OUTPUT);
  pinMode(NANOLED, OUTPUT);       // onboard LED

  analogReference(DEFAULT); // sets 5V as the top of reference

  //digitalWrite(RELAYOUTPUT, LOW); // turn OFF the relay output to de-energize the relay
  digitalWrite(HVACPOWERED,LOW);  // turn off that LED
  digitalWrite(DELAYOUTPUT,HIGH); // turn ON the delay LED
  digitalWrite(NANOLED,HIGH);    //  turn ON the Nano LED

  cli();
  wdt_reset();
  /*
    WDTCSR configuration:
    WDIE = 1: Interrupt Enable
    WDE = 1 :Reset Enable
    See table for time-out variations:
    WDP3 = 0 :For 1000ms Time-out
    WDP2 = 1 :For 1000ms Time-out
    WDP1 = 1 :For 1000ms Time-out
```

```
   WDP0 = 0 :For 1000ms Time-out
  */
  // Enter Watchdog Configuration mode:
  WDTCSR |= (1 << WDCE) | (1 << WDE);
  // Set Watchdog settings:
  WDTCSR = (1 << WDIE) | (1 << WDE) | (0 << WDP3) | (1 << WDP2) | (1 << WDP1) | (1 <<
WDP0);
  // trying to set it for 2 seconds
  sei();  // R-enable the interrups

  //Serial.begin(115200);   // this sets up the USB to give us diagnostic info
                        // at baud rate 115200 (fast)

  lcd.begin(16, 2);       // initialize the lcd display



  lcd.setCursor(0,1);     // set cursor to 0th column, 1st row (2nd line on ours)
  Lcd_Clear_Line();
  lcd.setCursor(0,1);

  lcd.setCursor(0,0);     // set the cursor at 0th column, 0th row
  Lcd_Clear_Line();
  lcd.setCursor(0,0);
  lcd.print(F("HVAC Protection"));
  MyDelay(2000);
  lcd.setCursor(0,0);     // set the cursor at 0th column, 0th row
  Lcd_Clear_Line();
  lcd.setCursor(0,0);
  lcd.print(F("KX4Z Sequencer"));
    MyDelay(2000);    // Delay so this can be READ
    wdt_reset();
  lcd.setCursor(0,0);     // set the cursor at 0th column, 0th row
  Lcd_Clear_Line();
  lcd.setCursor(0,0);
  lcd.print(F("Version 0.1"));
  MyDelay(2000);    // Delay so this can be READ
  wdt_reset();
  lcd.setCursor(0,0);     // set the cursor at 0th column, 0th row
  Lcd_Clear_Line();
  lcd.setCursor(0,0);
  lcd.print(F("Setup ACdetect"));
  MyDelay(100);    // Delay so this can be READ

// Approximately 80 measurements is one period (for 120Hz ripple)
for(i=0;i<200;i++){
  refvoltagereading[i] = analogRead(ACPWRINPUT);
}
```

```
wdt_reset();


for(i=0; i<10; i++) {
  normalpeakreading = 0;
  normalaveragereading = 0;

  for(j=i*8; j<(i*8)+75; j++){
    if(refvoltagereading[j] > normalpeakreading) normalpeakreading= refvoltagereading[j];
    normalaveragereading = normalaveragereading + refvoltagereading[j];
  }
  normalaveragereading = normalaveragereading/75;
  wdt_reset();
  lcd.setCursor(0,0);
  Lcd_Clear_Line();
  lcd.setCursor(0,0);
  lcd.print(F("Avg:"));
  lcd.setCursor(5,0);
  lcd.print(i);
  lcd.setCursor(10,0);
  lcd.print(normalaveragereading);
  lcd.setCursor(0,1);
  Lcd_Clear_Line();
  lcd.setCursor(0,1);
  lcd.print(F("Pk:"));
  lcd.setCursor(6,1);
  lcd.print(normalpeakreading);
  wdt_reset();
  MyDelay(1000);
  wdt_reset();
}


/*
for(i=0; i<200;i++){
   lcd.setCursor(0,0);     // set the cursor at 0th column, 0th row
  Lcd_Clear_Line();
  lcd.setCursor(0,0);
  lcd.print(F("V:"));
  lcd.setCursor(4,0);
  lcd.print(i);
  lcd.setCursor(9,0);
  lcd.print(refvoltagereading[i]);
  wdt_reset();
  MyDelay(100);
  wdt_reset();
}
*/
```

```
// Need to turn off all the current systems and measure the offset voltages...
//Serial.print("PowerLevel:  ");
//Serial.println(analogRead(A0));
/*
lcd.setCursor(0,1);
Lcd_Clear_Line();
lcd.setCursor(0,1);
sprintf(buffer,"Send:  ");
lcd.print(buffer);
lcd.print(analogRead(A0) );
*/

wdt_reset();
// Read the analog delay control to set up the lockoutcounter
                    // 1 min approx equal 500 counts

  delay_count = 100 + analogRead(DELAYCONTROL) * 4;
  lockoutcounter = delay_count;

digitalWrite(NANOLED, LOW);  // turn OFF LED until
while(lockoutcounter>0) {
  //MyDelay(100);  // wait 100 msec
  wdt_reset();

  if(PowerCheck()==1){
    lockoutcounter--;
    digitalWrite(NANOLED, HIGH);  // we found power
    lcd.setCursor(0,0);     // set the cursor at 0th column, 0th row
    Lcd_Clear_Line();
    lcd.setCursor(0,0);
    lcd.print(F("SETUP:AC!"));
    lcd.setCursor(11,0);
    lcd.print(delay_count);
    lcd.setCursor(0,1);     // set the cursor at 0th column, 0th row
    Lcd_Clear_Line();
    lcd.setCursor(0,1);
    lcd.print(lockoutcounter);
    lcd.setCursor(8,1);
    lcd.print(F("HVAC OFF"));
    wdt_reset();
  }
  else {
    lockoutcounter=delay_count;
    digitalWrite(NANOLED,LOW);   // no power
    lcd.setCursor(0,0);     // set the cursor at 0th column, 0th row
    Lcd_Clear_Line();
    lcd.setCursor(0,0);
    lcd.print(F("No AC"));
```

```
    lcd.setCursor(11,0);
    lcd.print(delay_count);
    lcd.setCursor(0,1);     // set the cursor at 0th column, 0th row
    Lcd_Clear_Line();
    lcd.setCursor(0,1);
    lcd.print(lockoutcounter);
    wdt_reset();
   }
  }  // end of lockout counter loop

  // If we got here, we had 1 minutes of constant good power!
  lcd.setCursor(0,0);     // set the cursor at 0th column, 0th row
  Lcd_Clear_Line();
  lcd.setCursor(0,0);
  lcd.print(F("SetupGood"));
  wdt_reset();
  MyDelay(500);    // Delay so this can be READ
  wdt_reset();
  pinMode(RELAYOUTPUT,  OUTPUT);  // Now we need to use it!
  digitalWrite(RELAYOUTPUT, HIGH); // turn ON power to the HVAC
  digitalWrite(HVACPOWERED,HIGH);  // turn off that LED
  digitalWrite(DELAYOUTPUT,LOW); // turn OFF the delay LED
  digitalWrite(NANOLED, HIGH);  // indicate good setup
  lcd.setCursor(0,0);     // set the cursor at 0th column, 0th row
  Lcd_Clear_Line();
  lcd.setCursor(0,0);


}  //end of SETUP

////////////////////////////READ POwer LEVEL////////////////////////////////
int PowerCheck() {
int i, n, powertop, powerbottom, powerreading;
powertop = 0;   // initialize as not present
powerbottom= 0; // initialize as not present
int reading[80];   // readings of voltage in 0-1023

int average=0;
int peak=0;

for(i=0;i<75;i++){
  reading[i] = analogRead(ACPWRINPUT);
  if (reading[i] > peak) peak = reading[i];
  average = average + reading[i];
  if(i==30) {
    if(average<1500) {    // really LOW LOW VOLTAGE!!
      failuremode = 1; // low average 1/4 cycle through
      return(0);
      }
```

```
    } // end of check for low voltage halfway through half cycle
  }  // end of 75 reading loop
average = average/75;
wdt_reset();
//  C R I T E R I A  ////////////////////////////////////
//  Set 6% limits on peak voltage =   24 up, 24 down
//  Set 6% limits on average voltage =  14 up, 14 down
//  note earlier check on absence of voltage for significant portion....

if (average < 236) {
  failuremode= 2;
  return (0);   // Originally 240
}
if (average > 270){
  failuremode= 3;
  return (0);   // Originally 265
}
if (peak<376){
  failuremode =4;
  return(0);         // originally 390 (very tight!)
}
if (peak>434){
  failuremode=5;
  return(0);         // originally 410 (very tight!)
}
// if it didn't fail, then it passed!!
return(1);


}  // end of PowerCheck routine

//////////////////////////////READ DELAY SETTING////////////////////////////////////
void readdelaysetting() {
// read the 0-5V setting of the delay setting potentiometer
//int delayinputsetting;

//delayinputsetting = analogRead(DELAYINPUT);  // reads 0-1023 for 0-250 mSec delay
//delay_milliseconds = delayinputsetting/4   ;  // int from 0 to about 250
return;
}



//////////////////////////LOOP ROUTINE ////////////////////////////////////////////
// Meat of the program: executed over and over again
void loop() {
  char  buffer[20];
  // put your main code here, to run repeatedly:
  // Turn on Power to HVAC
```

```
pinMode(RELAYOUTPUT,  OUTPUT);
digitalWrite(RELAYOUTPUT, HIGH);  // turn ON Solid State Relay
digitalWrite(HVACPOWERED,HIGH);   // turn off that LED
digitalWrite(DELAYOUTPUT,LOW);    // turn OFF the delay LED
digitalWrite(NANOLED, HIGH);      // turn ON the Nano LED
lcd.setCursor(0,0);     // set the cursor at 0th column, 0th row
//Lcd_Clear_Line();
//  lcd.setCursor(0,0);
 lcd.print(F("PowerGood   "));
 wdt_reset();
lcd.setCursor(8,1);     // set the cursor at 0th column, 0th row
lcd.print(F("HVAC ON "));
wdt_reset();



wdt_reset();      // Reset the watchdog timer -- it it doesn't get reset within 2 seconds,
                  // it will restart the software
if(PowerCheck()==1) {
  //lcd.setCursor(0,0);     // set the cursor at 0th column, 0th row
  //Lcd_Clear_Line();
  //lcd.setCursor(0,0);
  //lcd.print(F("PowerGood"));
  wdt_reset();
 }

else {
  // Detected a power failure!
  // Quickly turn everything off and start counting down good power
  // Circuit likely to die and go through startup again...
  digitalWrite(RELAYOUTPUT,LOW); // turn OFF the relay output to de-energize the relay
  MyDelay(2);   // time to turn it off
  pinMode(RELAYOUTPUT,  INPUT);  // now make it impossible to turn back on!
  // Hoping this will stop microprocessor dying mistakes....
  digitalWrite(HVACPOWERED,LOW);  // turn off that LED
  digitalWrite(DELAYOUTPUT,HIGH); // turn ON the delay LED
  digitalWrite(NANOLED, LOW)    ; // indicate power loss detected
  lcd.setCursor(0,0);     // set the cursor at 0th column, 0th row
  Lcd_Clear_Line();
  lcd.setCursor(0,0);
  lcd.print(F("PwrGlitch"));
  lcd.setCursor (11,0);
  lcd.print(failuremode);  // print the first failure detected
  lcd.setCursor(0,1);
  Lcd_Clear_Line();
  lcd.setCursor(8,1);
  lcd.print(F("HVAC OFF"));
  wdt_reset();
```

```
  lockoutcounter = delay_count;  // reset the lockout counter
    while(lockoutcounter>0) {
//     MyDelay(100);  // wait 100 msec
      wdt_reset();

      if(PowerCheck()==1){
      lockoutcounter--;
      digitalWrite(NANOLED, HIGH);  // we found power
     // lcd.setCursor(0,0);     // set the cursor at 0th column, 0th row
     // Lcd_Clear_Line();
     // lcd.setCursor(0,0);
     // lcd.print(F("AC Found"));
      lcd.setCursor(0,1);     // set the cursor at 0th column, 1th row
     // Lcd_Clear_Line();
      lcd.setCursor(0,1);
      lcd.print(lockoutcounter);
      wdt_reset();
    }
    else {
      lockoutcounter=delay_count;
      digitalWrite(NANOLED,LOW);   // no power
      lcd.setCursor(0,0);     // set the cursor at 0th column, 0th row
      Lcd_Clear_Line();
      lcd.setCursor(0,0);
      lcd.print(F("PwrGlitch"));
      lcd.setCursor(11,0);
      lcd.print(failuremode);  // show how it failed
      lcd.setCursor(0,1);     // set the cursor at 0th column, 0th row
      Lcd_Clear_Line();
      lcd.setCursor(0,1);
      lcd.print(lockoutcounter);

      lcd.setCursor(8,1);
      lcd.print(F("HVAC OFF"));
      wdt_reset();

    }  // end of ELSE

    } // END OF while


  }   // eLSE
  // If we get here, power was good and OK to turn it back on,
  // Which will occur on the next LOOP


}  // END OF LOOP

//////////////////////////supporting subroutines ////////////////////////////////////
```

```
////////////////////////////////////////////////////////////////////////////////


//////////////////////////CLEAR LINE/////////////////////////////////////////////////
void Lcd_Clear_Line()
{
  lcd.print(F("                "));  // should be exactly 16 spaces
}
//////////////////////////PRINT TO THE SCREEN ////////////////////////////////////////
void  lcd_display(char *s1,char *s2, int dtime)
{
  // dtime= milliseconds to delay
  lcd.setCursor(0,0);     // set the cursor at 0th column, 0th row
  // make sure the strings are null terminated after the 16th character.  1st char = *s1
  *(s1+15) = 0;
  *(s2+15) = 0;
  lcd.print(s1);
  lcd.setCursor(0,1);     // set cursor to 0th column, 1st row (2nd line on ours)
  lcd.print(s2);
  // call the delay function, while handling the watchdog
  wdt_reset();      // Reset the watchdog timer -- it it doesn't get reset within 2 seconds,
                    // it will restart the software
  MyDelay(dtime);
  wdt_reset();      // Reset the watchdog timer -- it it doesn't get reset within 2 seconds,
                    // it will restart the software
}
//////////////////////MICROSECOND DELAY NON BLOCKING ////////////////////////////////
void MyMicroSecondsDelay(int udelay)
{
  int microseconds;
  unsigned long current_microseconds, next_microseconds;

  current_microseconds = micros();
  next_microseconds = current_microseconds + (unsigned long) udelay;
    while(micros()<next_microseconds);
  return;

}

////////////////////////////MILLISECOND DELAY NON BLOCKING//////////////////
void MyDelay(int msec)
{

  unsigned long local_current_milliseconds, next_milliseconds, intermediate_milliseconds;
  int x;
  int thousands;
  wdt_reset();
```

```
  local_current_milliseconds = millis();
  next_milliseconds = local_current_milliseconds + (unsigned long) msec;
  thousands = msec/1000;              // integer division
  if(thousands>=1) {
      for(x=1;x<= thousands; x++){
          intermediate_milliseconds = local_current_milliseconds + (unsigned long) (x*1000);
          while (millis() < intermediate_milliseconds);
          wdt_reset();
          }
    }
  // now finish out the remainder, which should be less than 1000 milliseconds
   while (millis()<next_milliseconds);
   wdt_reset();
  return;
}
```