#include <SPI.h>

int local;

unsigned long OneSecondTimer;

float Temperature;

float tempThermoCouple;

float maxDayTemp = 70;

float maxNightTemp = 70;

int addDayDegree = 0;

int addNightDegree = 10;

float tempAdjust = 0.0;

#include <EEPROM.h>

#include<LiquidCrystal.h>

LiquidCrystal LcdDriver(11,9,5,6,7,8); //establishes specified pins as the pins used by the display

unsigned long SecondsTimer;

int INTERVAL;

int Hours, Minutes, Seconds;

int clockState;

int encoderPosition = 0;

int adjustState = 1;

enum ButtonState

{

Idle,

Wait,

Low

};

ButtonState button;

unsigned long Time;

//This function sets up the pins to use as input/output

void SW\_SpiInitialize()

{

pinMode (A3, OUTPUT);

pinMode (A5, OUTPUT);

pinMode (A4, INPUT);

}

//This function pushes bits high or low for using the pins

int SW\_Spil6( void )

{

int tempHold = 0;

PORTC &= ~0x08;

PORTC &= ~0x20;

for (int k = 15; k >= 0; k--)

{

PORTC |= 0x20;

//Checks if PINC is LOW

if(PINC & 0x10)

{

bitSet(tempHold, k);

}

PORTC &= ~0x20;

}

PORTC |= 0x08;

return tempHold;

}

//This function is used to update the clock on the LCD

void UpdateClock()

{

//Checks if seconds passed is < 59

if (Seconds < 59)

{

Seconds++;

}

//else, sets seconds to 0

else

{

Seconds = 0;

//Checks if minutes passed is < 59

if (Minutes < 59)

{

Minutes++;

}

//else, minutes is set to 0

else

{

Minutes = 0;

//Checks if hours < 23

if (Hours < 23)

{

Hours++;

}

//Otherwise, hours is reset

else

{

Hours = 0;

}

}

}

}

//This creates a function which is used for sending the clock to the LCD

void SendClock()

{

//If the hours is within the range from {1, 9}

if (Hours < 10 && Hours >= 0)

{

LcdDriver.print("0"); //prints "0" at location/cursor position

}

LcdDriver.print(Hours); //prints Hours; This works because if Hours is > 10, it will print 2 dig instead of one, voiding the if above

LcdDriver.print(":"); //prints the semicolon between hours and minutes

//Checks if the minutes is within range {1, 9}

if (Minutes < 10 && Minutes >= 0)

{

LcdDriver.print("0"); //prints "0" at location/cursor position

}

LcdDriver.print(Minutes); //prints the Minutes

LcdDriver.print(":"); //prints the semicolon between minutes and seconds

//checks if seconds is between {1, 9}

if (Seconds < 10 && Seconds >= 0)

{

LcdDriver.print("0"); //prints "0" at location/cursor position

}

LcdDriver.print(Seconds); //always prints seconds

}

//This function sends the Day temp to the LCD

void SendDay()

{

LcdDriver.setCursor(9,0); //sets the cursor at first row, at 9th column/cell

LcdDriver.print("D:"); //prints D at cursor location

LcdDriver.print(maxDayTemp); //prints the maxDayTemp

}

//This function send the Night temp to the LCD

void SendNight()

{

LcdDriver.setCursor(9,1); //sets the cursor at the second row, at the 9th column/cell

LcdDriver.print("N:"); //Prints N at cursor location

LcdDriver.print(maxNightTemp); //prints the masNightTemp

}

//This function sends the entire screen; consolidates the send to screen function

void ConsolidateScreen()

{

UpdateClock(); //Calls update clock

LcdDriver.setCursor(0,0); //sets cursor at first row, zeroth cell

SendClock(); //calls send clock

SendDay(); //calls send day; sends the day temperature

LcdDriver.setCursor(0,1); //sets cursor at second row, zeroth cell

LcdDriver.print("T:");// LcdDriver.print( "Temp = "); //prints "T:" for the temperature/thermocouple

LcdDriver.print(tempThermoCouple); //prints the value of tempThermoCouple at location

SendNight(); //Sends the night temperature

}

//This function reads the Temperature by calling SW\_Spi16

float Read\_SPI\_Temperature()

{

int TempData = SW\_Spil6(); //calls SW\_Spi16(); stores return val into temperary int

TempData = (TempData / 8 ) & 0x0fff; //sets the value of tempdata

Temperature = 0.25 \* (float) TempData; //casts tempdata to a float, and convert to celsius

Temperature = 1.8\*Temperature + 32.0; //converst temperature to Farenheit

return Temperature; //returns the Temperature

}

//This function is the state machine for the button state; accepts an int param "input"

int ButtonNextState( int input )

{

switch( button )

{

//The idle state; sets button state to wait & pin 13 high

case Idle:

//Chekcs if input is equal to 0

if (input == 0)

{

Time = millis(); //sets time to millis();

button = Wait; //sets state to Wait

digitalWrite(13, HIGH); //sets pin 13 to HIGH

}

break; //breaks out of switchcase

//The wait state; sets button state to idle or low

case Wait:

//Checks if input is = to 1

if (input == 1)

{

button = Idle; //sets button state to idle

}

//Checks if the time elapsed is 5 ms; input != 1

else if (millis() - Time >= 5 )

{

button = Low; //Sets button state to low

digitalWrite(13, LOW); //sets pin 13 to low

return 1; //returns value of 1

}

break; //breaks out of switch case

//The low state; sets button state to idle if input is 1

case Low:

//Checks if input == 1

if (input == 1)

{

button = Idle; //sets button state to idle

}

break; //breaks out of switch case

}

return 0; //returns 0

}

//This function is used for the encoder position, and pin 2 and 3

void MonitorA()

{

//checks if output of pin 2 and 3 is same

if (digitalRead(2) == digitalRead(3))

{

encoderPosition++; //increments the encoder position

//if encoderPosition is >= 2

if(encoderPosition >= 2)

{

tempAdjust += 0.5; //increments tempAdjust by .5

encoderPosition = 0; //sets encoder position to 0

}

}

//if output of pin 2 and 3 is !equal

else

{

encoderPosition--; //decrements the encoder position

//if encoderPosition <= -2

if(encoderPosition <= (-2))

{

tempAdjust -= 0.5; //decrements etmpAdjust at interval of .5

encoderPosition = 0; //sets encoder position to 0

}

}

}

//This function is used for manipulating the tempAdjust var

void MonitorB()

{

//checks if the output of pin 2 and 3 is equal

if (digitalRead(2) == digitalRead(3))

{

encoderPosition--; //decrements encoder position

//if ep <= -2

if(encoderPosition <= (-2))

{

tempAdjust -= 0.5; //decrements at interval of .5

encoderPosition = 0; //set ep to 0

}

}

//pin 2 and 3 output is !equal

else

{

encoderPosition++; //increments ep

//if ep >= 2

if(encoderPosition >= 2)

{

tempAdjust += 0.5; //increments at interval of .5

encoderPosition = 0; //sets ep to 0

}

}

}

//This function is used for adjusting the night temperature

void AdjustNightTemp()

{

//checks if the temp == the night temp

if (tempThermoCouple < (maxNightTemp + 5) && tempThermoCouple > ( maxNightTemp - 5))

{

digitalWrite(A0, LOW); //Pushes AO Low

digitalWrite(A1, LOW); //Pushes A1 Low

}

//checks if the temperature has varied by +5 degrees for night

if (tempThermoCouple >= maxNightTemp + 5)

{

digitalWrite(A0, HIGH); //Pushes bit A0 HIGH

digitalWrite(A1, LOW); //Pushes Bit A1 low

}

//checks if the temp has varied by -5 degrees for night

if (tempThermoCouple <= maxNightTemp - 5)

{

digitalWrite(A1, HIGH); //Pushes bit A1 HIGH

digitalWrite(A0, LOW); //Pushes bit A0 LOW

}

}

//This function is used for adjusting the day temperature

void AdjustDayTemp()

{

//checks if the day temp == to the temp thermo couple

if (tempThermoCouple < (maxDayTemp + 5) && tempThermoCouple > (maxDayTemp - 5))

{

digitalWrite(A0, LOW); //Pushes A0 low

digitalWrite(A1, LOW); //Pushes A1 high

}

//checks if the day temp as varied by +5 (hotter)

if (tempThermoCouple >= maxDayTemp + 5)

{

digitalWrite(A0, HIGH); //Pushes AO HIGH

}

//checks if the day temp has varied by -5 (colder)

if (tempThermoCouple <= maxDayTemp - 5)

{

digitalWrite(A1, HIGH); //Pushes A1 HIGH

}

}

//This function runs once; sets up the program

void setup()

{

SW\_SpiInitialize(); //Calls the initializing function for SPI

OneSecondTimer = millis(); //preps the timer

INTERVAL = 1000; //creates a variable to use for intervals

LcdDriver.begin(16,2); //makes the size of the display

LcdDriver.setCursor(0,1); //sets the cursor on line 1, cell 1

SecondsTimer = millis(); //sets SecondsTimer to millis()

Hours = 23; //first display hours

Minutes = 59; //first display minutes

Seconds = 55; //first display seconds

pinMode (2, INPUT); //makes pin 2 an input

pinMode (3, INPUT); //makes pin 3 an input

attachInterrupt (digitalPinToInterrupt(2) ,MonitorA, CHANGE);

attachInterrupt (digitalPinToInterrupt(3) ,MonitorB, CHANGE);

EEPROM.get(addDayDegree, maxDayTemp);

EEPROM.get(addNightDegree, maxNightTemp);

pinMode (A0, OUTPUT);//make sure blue led for cooling

pinMode (A1, OUTPUT);//make sure red led for heating

}

//This function runs continually; runs bulk of program

void loop()

{

while(clockState == 0)

{

while(Hours > 6 && Hours < 22)

{

if(millis() - SecondsTimer >= INTERVAL)

{

tempThermoCouple = Read\_SPI\_Temperature();

ConsolidateScreen();

SecondsTimer += INTERVAL;

AdjustDayTemp();

}

if( ButtonNextState( digitalRead(4)))

{

tempAdjust = Seconds;

clockState = 1;

break;

}

}

while(Hours < 7 || Hours > 21)

{

if(millis() - SecondsTimer >= INTERVAL)

{ //insures the clock updates every second

tempThermoCouple = Read\_SPI\_Temperature();

ConsolidateScreen();

SecondsTimer += INTERVAL;

AdjustNightTemp();

}

if( ButtonNextState( digitalRead(4)))

{

tempAdjust = Seconds;

clockState = 1;

break;

}

}

}

while(clockState == 1){

while(adjustState == 1){

LcdDriver.setCursor(0,0);

SendClock();

Seconds = tempAdjust;

if(tempAdjust < 0 || tempAdjust > 59){

tempAdjust = 0;

}

if( ButtonNextState( digitalRead(4))) {

tempAdjust = Minutes;

adjustState = 2;

}

}

while(adjustState == 2){

LcdDriver.setCursor(0,0);

SendClock();

Minutes = tempAdjust;

if(tempAdjust < 0 || tempAdjust >59){

tempAdjust = 0;

}

if( ButtonNextState( digitalRead(4))) {

tempAdjust = Hours;

adjustState = 3;

}

}

while(adjustState == 3){

LcdDriver.setCursor(0,0);

SendClock();

Hours = tempAdjust;

if(tempAdjust < 0 || tempAdjust >23){

tempAdjust = 0;

}

if( ButtonNextState( digitalRead(4))) {

adjustState = 4;

tempAdjust = maxDayTemp;

}

}

while(adjustState == 4){

maxDayTemp = tempAdjust;

LcdDriver.setCursor(9,0);

SendDay();

if( ButtonNextState( digitalRead(4))){

EEPROM.put(addDayDegree, maxDayTemp);

adjustState = 5;

tempAdjust = maxNightTemp;

}

}

while(adjustState == 5){

LcdDriver.setCursor(9,1);

SendNight();

maxNightTemp = tempAdjust;

if( ButtonNextState( digitalRead(4))){

EEPROM.put(addNightDegree, maxNightTemp);

adjustState = 6;

}

}

while(adjustState == 6){

adjustState = 1;

clockState = 0;

SecondsTimer = millis();

break;

}

}

}