ECE 241 Final Lab Report Fall 2016

Names: Joe Boutte & Nick Spiller

Objective: Build a thermostat using the Arduino, LM34 temperature to voltage sensor, and a MOSFET relay to control a computer fan. The thermostat is to have a temperature that can be set by the encoder, a clock set by the serial port, a temperature controlled fan, a time and temperature readout, and persistent temperatures using the Arduino EEPROM.

Hardware Added: For this project an LM34, temperature to voltage device, is used to measure the room temperature. A schematic for the inclusion of this device is included from the Texas Instrument datasheet below in Figure 1.

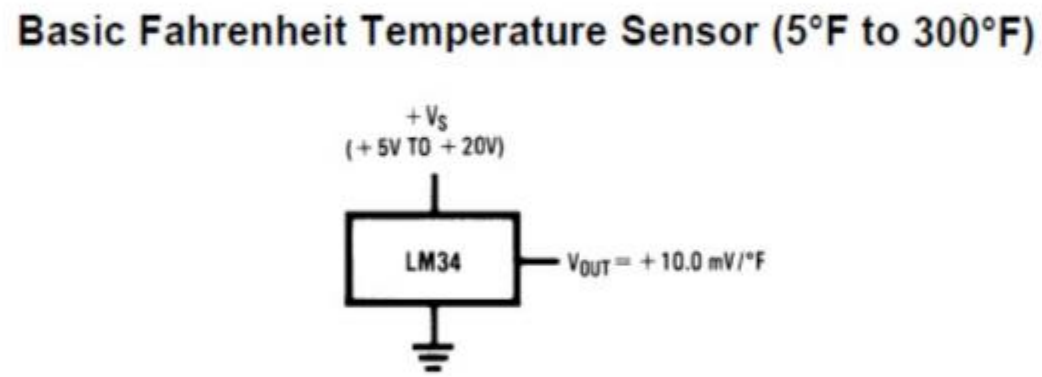


Figure 1. Schematic of LM34 circuit

The V_{OUT} in figure 1 is attached to pin A7 on the Arduino. The 5.0-volt output from the Arduino was used as the reference for the ADC, giving us a range of 0 to 500(5.0/ 10 mV) degrees Fahrenheit, way too large for our intended use as a thermostat.

The unit turns on a fan to emulate a cooling system. As the pins on the Arduino cannot provide sufficient power to run the fan, a separate power source was used (9-volt battery) to power the fan. The circuit of the board using a TIP120 relay is included in Figure 2.

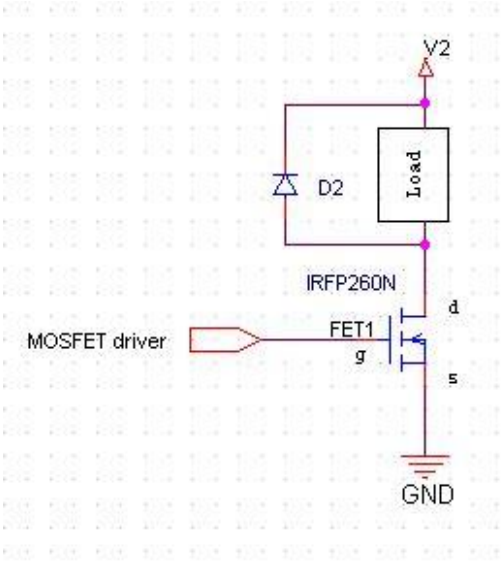


Figure 2. Relay Circuit

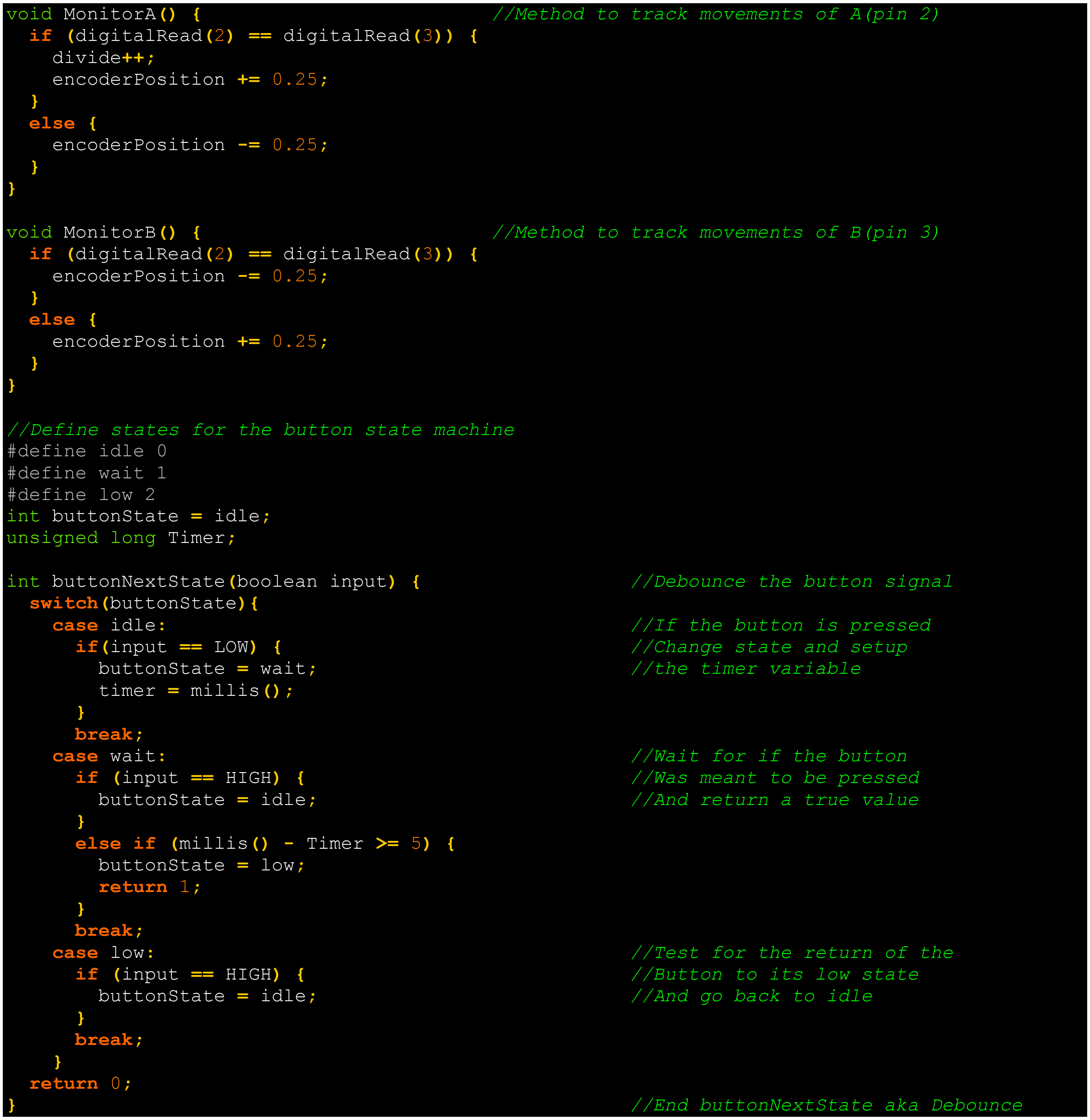
Software: Below is what the inputs do the thermostat followed by the code used in the project.

The encoder knob is used to set the temperatures of the system, first the daytime temperature then the nighttime temperature. After both temperatures are set, the system updates these values in the Arduino EEPROM using a library called EEPROMex which enables easier writing of data formats other than bytes to the Arduino EEPROM. The code for reading the encoder is included in Appendix A.

The serial ports are used for a system report and setting the time on the device. If 'R' or 'r' is sent to the Arduino, it sends back the temperatures that the system is programmed with, what is written to the EEPROM, if it is day, if the fan is on, and the current system time. To set the time, one enters 'S' or 's' followed by a space then the time, in military formatting and separated by ':'. The system reads this in and sets the time accordingly. The code used to interact with the serial ports is included in Appendix B.

The complete program code is included in Appendix C.

Appendix A:



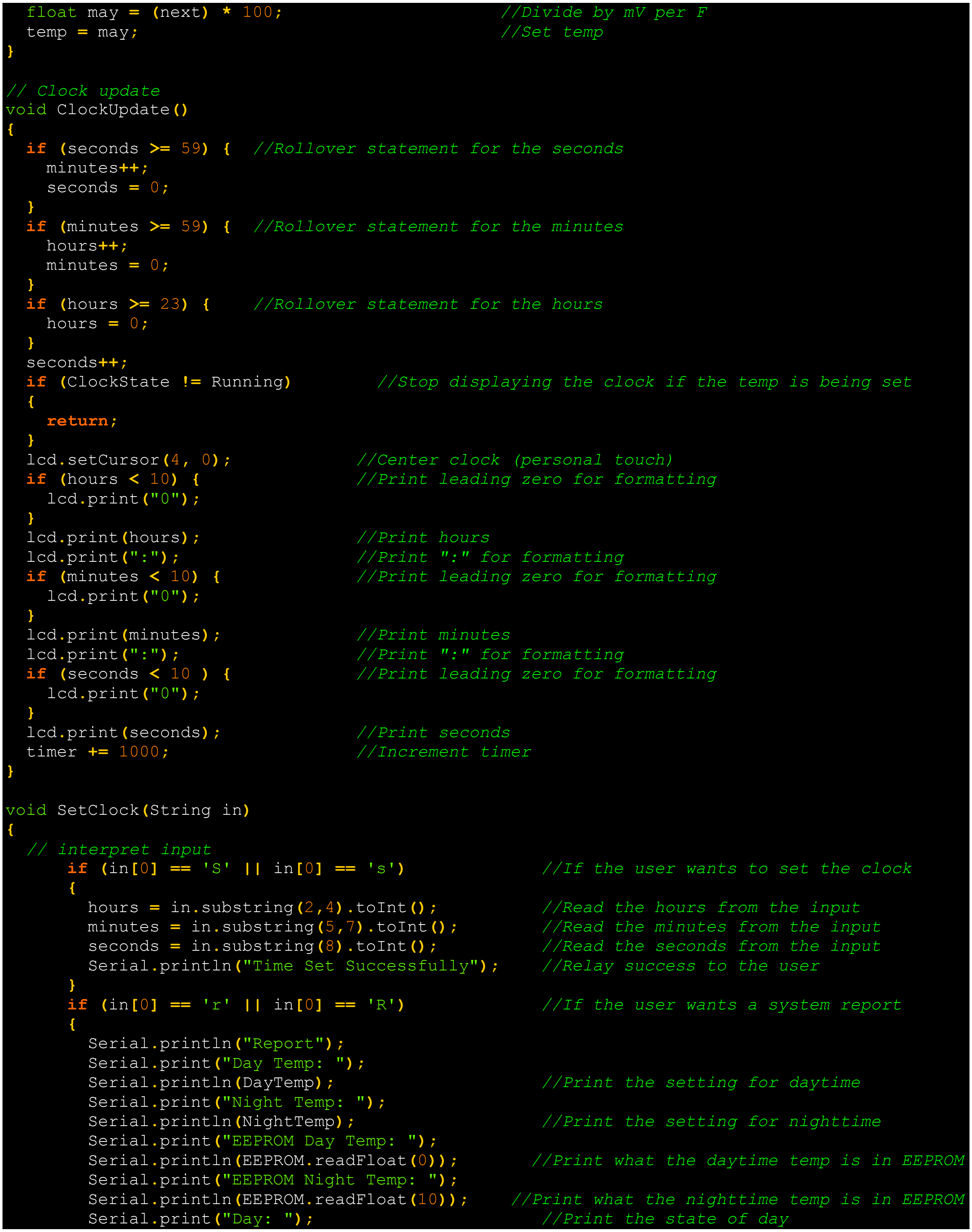
```
void MonitorA() {                         //Method to track movements of A(pin 2)
  if (digitalRead(2) == digitalRead(3)) {
    divide++;
    encoderPosition += 0.25;
  }
  else {
    encoderPosition -= 0.25;
  }
}

void MonitorB() {                         //Method to track movements of B(pin 3)
  if (digitalRead(2) == digitalRead(3)) {
    encoderPosition -= 0.25;
  }
  else {
    encoderPosition += 0.25;
  }
}

//Define states for the button state machine
#define idle 0
#define wait 1
#define low 2
int buttonState = idle;
unsigned long Timer;

int buttonNextState(boolean input) {                  //Debounce the button signal
  switch(buttonState){
    case idle:                                        //If the button is pressed
      if(input == LOW) {                              //Change state and setup
        buttonState = wait;                           //the timer variable
        timer = millis();
      }
      break;
    case wait:                                        //Wait for if the button
      if (input == HIGH) {                            //Was meant to be pressed
        buttonState = idle;                           //And return a true value
      }
      else if (millis() - Timer >= 5) {
        buttonState = low;
        return 1;
      }
      break;
    case low:                                         //Test for the return of the
      if (input == HIGH) {                            //Button to its low state
        buttonState = idle;                           //And go back to idle
      }
      break;
    }
  return 0;
}                                                     //End buttonNextState aka Debounce
```

Appendix B:

```
void SetClock(String in)
{
  // interpret input
      if (in[0] == 'S' || in[0] == 's')             //If the user wants to set the clock
      {
        hours = in.substring(2,4).toInt();          //Read the hours from the input
        minutes = in.substring(5,7).toInt();        //Read the minutes from the input
        seconds = in.substring(8).toInt();          //Read the seconds from the input
        Serial.println("Time Set Successfully");    //Relay success to the user
      }
      if (in[0] == 'r' || in[0] == 'R')            //If the user wants a system report
      {
        Serial.println("Report");
        Serial.print("Day Temp: ");
        Serial.println(DayTemp);                    //Print the setting for daytime
        Serial.print("Night Temp: ");
        Serial.println(NightTemp);                  //Print the setting for nighttime
        Serial.print("EEPROM Day Temp: ");
        Serial.println(EEPROM.readFloat(0));       //Print what the daytime temp is in EEPROM
        Serial.print("EEPROM Night Temp: ");
        Serial.println(EEPROM.readFloat(10));    //Print what the nighttime temp is in EEPROM
        Serial.print("Day: ");                      //Print the state of day
        if (day)
        {
          Serial.println("Yes");
        }
        else
        {
          Serial.println("No");
        }
        Serial.print("Fan: ");
        if (fan)                                    //Print the state of fan
        {
          Serial.println("On");
        }
        else
        {
          Serial.println("Off");
        }
        if (hours < 10) {               //Print leading zero for formatting
          Serial.print("0");
        }
        Serial.print(hours);            //Print hours
        Serial.print(":");              //Print ":" for formatting
        if (minutes < 10) {             //Print leading zero for formatting
          Serial.print("0");
        }
        Serial.print(minutes);          //Print minutes
        Serial.print(":");              //Print ":" for formatting
        if (seconds < 10 ) {            //Print leading zero for formatting
          Serial.print("0");
        }
        Serial.println(seconds);        //Print seconds
        Serial.println("To set: enter 's-' followed by the time in military format, separated
by a single character");
      }
} //End of SetClock
```

Appendix C:

```
// includes for systems on Arduino
#include <EEPROMex.h>
#include <LiquidCrystal.h>
LiquidCrystal lcd(11, 9, 5, 6, 7, 8);

// Variable that are to be updated.
// Declared as globals.
float DayTemp = 0.0;
float NightTemp = 0.0;
float curTemp;
float temp;
bool day;
bool fan = false;

// Clock variables
int hours = 0;
int minutes = 0;
int seconds = 0;
unsigned long timer;

// Update the Temperature based on Time
void TempTime()
{
  if (hours < 19 && hours > 5)                  //If it is daytime hours
  {
    curTemp = DayTemp;                          //Current temperature is the daytime
temperature
    day = true;                                 //Set day to true
  }
  else                                          //Otherwise it is night
  {
    curTemp = NightTemp;                        //Current temperature is the nighttime
temperature
    day = false;                                //Set day to false
  }
}

//Check if the fan needs to be turned on or off
void FanUpdate()
{
  if (temp > curTemp)                           //If it's hotter than desired
  {
    digitalWrite(12, HIGH);                     //Turn on the fan
    fan = true;                                 //Set fan to true

  }
  else                                          //Otherwise
  {
    digitalWrite(12, LOW);                      //Turn off the fan
    fan = false;                                //Set fan to false
  }
}

enum ClockStates { Running, Temp_Set };
ClockStates ClockState = Running;

//Read temperature from the sensor
void ReadTemp()
{
  int red = analogRead(A7);                     //Read the voltage through the ADC
  float next = (red / 1023.0) * 5.0;            //Convert to voltage
```



```
  float may = (next) * 100;                     //Divide by mV per F
  temp = may;                                   //Set temp
}

// Clock update
void ClockUpdate()
{
  if (seconds >= 59) {  //Rollover statement for the seconds
    minutes++;
    seconds = 0;
  }
  if (minutes >= 59) {  //Rollover statement for the minutes
    hours++;
    minutes = 0;
  }
  if (hours >= 23) {    //Rollover statement for the hours
    hours = 0;
  }
  seconds++;
  if (ClockState != Running)        //Stop displaying the clock if the temp is being set
  {
    return;
  }
  lcd.setCursor(4, 0);            //Center clock (personal touch)
  if (hours < 10) {               //Print leading zero for formatting
    lcd.print("0");
  }
  lcd.print(hours);               //Print hours
  lcd.print(":");                 //Print ":" for formatting
  if (minutes < 10) {             //Print leading zero for formatting
    lcd.print("0");
  }
  lcd.print(minutes);             //Print minutes
  lcd.print(":");                 //Print ":" for formatting
  if (seconds < 10 ) {            //Print leading zero for formatting
    lcd.print("0");
  }
  lcd.print(seconds);             //Print seconds
  timer += 1000;                  //Increment timer
}

void SetClock(String in)
{
  // interpret input
      if (in[0] == 'S' || in[0] == 's')             //If the user wants to set the clock
      {
        hours = in.substring(2,4).toInt();          //Read the hours from the input
        minutes = in.substring(5,7).toInt();        //Read the minutes from the input
        seconds = in.substring(8).toInt();          //Read the seconds from the input
        Serial.println("Time Set Successfully");    //Relay success to the user
      }
      if (in[0] == 'r' || in[0] == 'R')             //If the user wants a system report
      {
        Serial.println("Report");
        Serial.print("Day Temp: ");
        Serial.println(DayTemp);                    //Print the setting for daytime
        Serial.print("Night Temp: ");
        Serial.println(NightTemp);                  //Print the setting for nighttime
        Serial.print("EEPROM Day Temp: ");
        Serial.println(EEPROM.readFloat(0));       //Print what the daytime temp is in EEPROM
        Serial.print("EEPROM Night Temp: ");
        Serial.println(EEPROM.readFloat(10));    //Print what the nighttime temp is in EEPROM
        Serial.print("Day: ");                      //Print the state of day
        if (day)
        {
          Serial.println("Yes");
        }
        else
        {
          Serial.println("No");
        }
        Serial.print("Fan: ");
        if (fan)                                    //Print the state of fan
        {
          Serial.println("On");
        }
        else
        {
          Serial.println("Off");
        }
        if (hours < 10) {               //Print leading zero for formatting
          Serial.print("0");
        }
        Serial.print(hours);            //Print hours
        Serial.print(":");              //Print ":" for formatting
        if (minutes < 10) {             //Print leading zero for formatting
          Serial.print("0");
        }
        Serial.print(minutes);          //Print minutes
        Serial.print(":");              //Print ":" for formatting
        if (seconds < 10 ) {            //Print leading zero for formatting
          Serial.print("0");
        }
        Serial.println(seconds);        //Print seconds
        Serial.println("To set: enter 's-' followed by the time in military format, separated by a single character");
      }
} //End of SetClock

// Support functions for Display state and LCD
void ReadInState() {
  // Read in DisplayState
  if (EEPROM.readByte(150) != 202)          // This indicates that the
  {                               // EEPROM has NOT been written to.
    DayTemp = 75.0;
    NightTemp = 65.0;
  }
  else                            // if EEPROM has been written
  {                               // Read in other parameters.
    DayTemp = EEPROM.readFloat(0);
    NightTemp = EEPROM.readFloat(10);
  }
} // end of ReadInState

void WriteOutState()
{
  // Update the day and night temperatures, writing is necessary
  EEPROM.updateFloat(0, DayTemp);
  EEPROM.updateFloat(10, NightTemp);
  EEPROM.updateByte(150, 202);
}// end of WriteOutState

//Float for setting a temperature
float encoderPosition = 0.0;
int divide = 0;
```

```
void MonitorA() {                         //Method to track movements of A(pin 2)
  if (digitalRead(2) == digitalRead(3)) {
    divide++;
    encoderPosition += 0.25;
  }
  else {
    encoderPosition -= 0.25;
  }
}

void MonitorB() {                         //Method to track movements of B(pin 3)
  if (digitalRead(2) == digitalRead(3)) {
    encoderPosition -= 0.25;
  }
  else {
    encoderPosition += 0.25;
  }
}

//Define states for the button state machine
#define idle 0
#define wait 1
#define low 2
int buttonState = idle;
unsigned long Timer;

int buttonNextState(boolean input) {                  //Debounce the button signal
  switch(buttonState){
    case idle:                                        //If the button is pressed
      if(input == LOW) {                              //Change state and setup
        buttonState = wait;                           //the timer variable
        timer = millis();
      }
      break;
    case wait:                                        //Wait for if the button
      if (input == HIGH) {                            //Was meant to be pressed
        buttonState = idle;                           //And return a true value
      }
      else if (millis() - Timer >= 5) {
        buttonState = low;
        return 1;
      }
      break;
    case low:                                         //Test for the return of the
      if (input == HIGH) {                            //Button to its low state
        buttonState = idle;                           //And go back to idle
      }
      break;
    }
  return 0;
}                                                     //End buttonNextState aka Debounce

// put your setup code here, to run once:
void setup()
{
  // Setup the display
  lcd.begin(16, 2);
  lcd.clear();
  ReadInState();

  //Setup encoder functions
  pinMode(2, INPUT);
  pinMode(3, INPUT);
  attachInterrupt(digitalPinToInterrupt(2), MonitorA, CHANGE);
  attachInterrupt(digitalPinToInterrupt(3), MonitorB, CHANGE);

  // Setup clock variables
  hours = 0;
  minutes = 0;
  seconds = 0;
  timer = millis();
  Timer = millis();

  // Indicators of write process.
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  digitalWrite(13, LOW);
  digitalWrite(12, LOW);

  // Setup Serial port
  Serial.begin(9600);
} // End of setup

enum SystemStates { Normal, Day_Set, Night_Set };
SystemStates System = Normal;

// put your main code here, to run repeatedly:
void loop()
{
  switch (System)
  {
    case Normal:
      if (millis() - timer >= 1000)             //Do normal functions every second
      {
        ClockUpdate();                          //Update the clock
        TempTime();                             //Check if it has become day or night
        lcd.setCursor(4,1);                     //Move cursor
        ReadTemp();                             //Read in the temperature from the sensor
        if (temp < 100.0)                       //Formatting
        {
          lcd.print(" ");
        }
        lcd.print(temp);                        //Print the temperature
        if (temp > 100.0)                       //Formatting
        {
          lcd.print(" ");
        }
        lcd.print("F");                         //Fahrenheit
        FanUpdate();                            //Update if the fan needs to be on or off
      }
      // Check incoming serial data.
      if (Serial.available()) {
        SetClock(Serial.readString());
      }// end of Serial input.
      if (buttonNextState(digitalRead(4)))      //Read the encoder button
      {
        encoderPosition = DayTemp;              //Set value to be changed
        System = Day_Set;                       //Change states
        lcd.clear();                            //Clear LCD
        lcd.setCursor(2,0);                     //Formatting
        lcd.print("New Day Temp");              //Which temp is being set
        ClockState = Temp_Set;                  //Stop displaying the clock
      }
      break;
    case Day_Set:
      if (millis() - timer >= 1000)             //Every second
      {
        ClockUpdate();                          //Update the clock
        ReadTemp();                             //Read the temperature
        FanUpdate();                            //Update if the fan needs to be on or off
      }
      lcd.setCursor(6,1);                       //Move the cursor
      lcd.print(encoderPosition);               //Print the current setting of the day temp, 
changes in MonitorA and MonitorB
      if (buttonNextState(digitalRead(4)))      //Read the encoder button
      {
        DayTemp = encoderPosition;              //Set daytime temp to new value
        encoderPosition = NightTemp;            //Set the value to be changed
        System = Night_Set;                     //Change states
        lcd.clear();                            //Clear LCD
        lcd.setCursor(1,0);                     //Formatting
        lcd.print("New Night Temp");            //Which temp to be set
      }
      break;
    case Night_Set:
      if (millis() - timer >= 1000)             //Every second
      {
        ClockUpdate();                          //Update the clock
        ReadTemp();                             //Read the temperature
        FanUpdate();                            //Update if the fan needs to be turned on or 
off
      }
      lcd.setCursor(6,1);                       //Move the cursor
      lcd.print(encoderPosition);               //Print the current setting of the night 
temp, changes in MonitorA and MonitorB
      if (buttonNextState(digitalRead(4)))      //Read the encoder button
      {
        NightTemp = encoderPosition;            //Set nighttime temp to new value
        System = Normal;                        //Change states
        ClockState = Running;                   //Start displaying the clock again
        WriteOutState();                        //Update temp values in EEPROM
        lcd.clear();                            //Clear the LCD
      }
      break;
  }
} // end of loop
```