Sustainability Project

Abstract:

This paper is about an electrical project themed sustainability. It describes and explains an electrical system that is used as a programmable interface for use with a digital thermostat. The system can be programmed to adjust the temperature setting of a thermostat at designated times throughout the day. That way, I can reduce the amount of time my air conditioning is running. The result is an overall reduction in power consumption.
**Introduction:**

This project, initially, was just going to consist of two simple circuits that interfaced with my digital thermostat to adjust temperature settings at different times throughout the day. In order to do that, it was necessary to code a clock in the micro-controller. Once the clock was coded, I needed a way to manually set the time of day. This required there be some type of clock interface, a way to read the current time and adjust it accordingly. To solve this problem I built a binary clock based roughly on a binary watch that I have. The end result is a binary clock that is easy to read and has two buttons, minutes and hours, that can be used for setting the time. The micro-controller keeps track of the current time with that clock and adjusts the thermostat settings at the designated times I programmed it for.

**List of Materials:**

- resistor $K\Omega \times 15$
- resistor $100\Omega \times 12$
- NPN type transistor $\times 12$
- Arduino Micro-controller
- LED $\times 12$
- battery 9V
- approx. 6ft of wire
- bread board $\times 3$

Note: Some resistors of other values were used to deal with the discrepancies in luminosity of not having all the same types of LEDs.
The Clock:

The clock is coded using a standard base ten number system, but displays in binary, using LEDs. There is a row of four yellow LEDs and a row of six red LEDs. The row of red LEDs displays the minutes and each individual LED represents the binary value. This is better illustrated in figure 1. The row of yellow LEDs displays the hours and again is given the respective binary values. The time is read by summing the binary values of the LEDs that are on in the hours and minutes rows respectively. Those summations are the standard time in minutes and hours.

Each LED is controlled by one digital output pin on the micro-controller. The LEDs are powered by a separate 9V battery. When the code calls for the LED to be turned on, the micro-controller puts a value of 5V on the pin, which goes to a transistor that turns the light on. Figure 2 illustrates the circuits for the LEDs, there are ten of these. They are connected this way so that no current is drawn from the micro-controller. In addition, the 9V source as opposed to the micro-controllers 5V output, provides for brighter LEDs.

Figure 1. Shows the binary values for each LED. The yellow row displays hours and the red row displays minutes.

Figure 2. A schematic drawing of the LEDs for the clock. There are a total of ten of these. Only 2 are shown.
To set the clock I provided two buttons. One button controls the minutes and one controls the hours. When the button is pressed and released quickly it adds a one to the minutes/hours. When the button is pressed and held, it adds one to the minutes/hours every 0.5 seconds until the button is released. For these circuits I used the analog inputs and set it up in such a way that when the input detects a value below threshold it reads this as an “on”. So when the button is depressed the analogue input is at ground. When the button is open, it is at 5V. I did it this way because after some testing it seemed to work the best with the analogue inputs. A schematic of this circuit is provided in figure 3.

![Figure 3. Schematic for the buttons used to set the clock.](image)

The code for this clock (and the thermostat) is attached. The code is commented and I believe illustrates more clearly how this was done, rather than a lengthy explanation. Briefly, the program uses the micro-controllers millisecond timer and counts milliseconds up to one second. Then it counts seconds up to one minute and resets the seconds, etc. For the buttons and setting, it just adds +1 to the current count and proceeds counting milliseconds in this way. Then for the thermostat it just uses an “if then” set up. For example, to set it to turn the temperature setting down 3 degrees at 5pm, it would just have something like “if hour==5 && minute==0, then digitalWrite(11, HIGH); delay(500); digitalWrite(11, LOW); delay(500),” where the last part would turn it on and off two more times. The code is fairly simply.
Thermostat:

I described above how the code for this works. This is how it is interfaced with my current digital thermostat. My thermostat has one button, that if pressed, raises the temperature one degree, and one button that lowers it one degree if pressed. I created a circuit that will “press” these buttons a designated number of times at a designated time of day. All it does is connect one wire to one side of the button and one wire to the other side of the button. When the digital output is turned on, its connected to a transistor that closes the circuits and bypasses the buttons. Figure 4 shows the schematic for this. There are two of these, one for plus degrees and one for minus degrees. Since I could not uninstall my thermostat at home, in my circuit for demonstration, I use LEDs to represent the bypass of these switches. I also programmed the micro-controller so it adds plus 3 degrees every thirty seconds and minus 3 degrees the other thirty seconds. I demonstrate that though with two separate LEDs that represent the process.

Conclusion:  This project turned out to be a bit larger than initially planned, but also turned out to be more interesting. While the prototype functions smoothly, the practicality is not quite reasonable, since the circuitry is no less than huge. This will probably not be going in my living room.
// This is the source code for the micro-controller

int second=0, minute=0, hour=0, pm=0; //start the time on 00:00:00
int valm=0, valh=0, ledstats, i;

void setup() { //set outputs and inputs
  pinMode(1, OUTPUT);pinMode(2, OUTPUT);pinMode(3, OUTPUT);pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);pinMode(7, OUTPUT);pinMode(8, OUTPUT);pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(11, OUTPUT);pinMode(12, OUTPUT);
  pinMode(0, INPUT);
}

void loop() {

  static unsigned long lastTick = 0; // set up a local variable to hold the last time we moved forward one second
  // (static variables are initialized once and keep their values between function calls)
  // move forward one second every 1000 milliseconds

  if (millis() - lastTick >= 1000) {
    lastTick = millis();
    second++;
  }

  // move forward one minute every 60 seconds
  if (second >= 60) {
    minute++;
    second = 0; // reset seconds to zero
  }

  // move forward one hour every 60 minutes
  if (minute >=60) {
    hour++;
    minute = 0; // reset minutes to zero
  }

  if (hour >=12) {
    hour=0;
    minute = 0; // reset minutes to zero
  }

  ledstats = digitalRead(0);  // read input value, for setting leds off, but keeping count
  if (ledstats == LOW) {

for(i=1;i<=13;i++){
digitalWrite(i, LOW);
}

//minutes units
if(minute == 1 || minute == 3 || minute == 5 || minute == 7 || minute == 9 || minute == 11 ||
minute == 13 || minute == 15 ||
minute == 17 || minute == 19 || minute == 21 || minute == 23 || minute == 25 || minute == 27 ||
minute == 29 || minute == 31 || minute == 33 || minute == 35 || minute == 37 || minute == 39 || minute == 41 || minute == 43 || minute == 45 || minute == 47 || minute == 49 ||
minute == 51 || minute == 53 || minute == 55 || minute == 57 || minute == 59) { digitalWrite(1, HIGH);} else {
digitalWrite(1,LOW);
}

if((minute >= 8 && minute <= 15) || (minute >= 24 && minute < 60) || (minute >= 40 && minute <= 47) || (minute >= 56 && minute < 60)) {digitalWrite(3, HIGH);} else {
digitalWrite(3,LOW);
}

if((minute >= 16 && minute <= 31) || (minute >= 48 && minute < 60)) {digitalWrite(5, HIGH);} else {
digitalWrite(5,LOW);
}
HIGH);} else

{digitalWrite(5,LOW);}
    if(minute >= 32 && minute < 60) {digitalWrite(6, HIGH);} else
{digitalWrite(6,LOW);}

// hour units
    if(hour == 1 || hour == 3 || hour == 5 || hour == 7 || hour == 9 || hour == 11)
{digitalWrite(7, HIGH);} else
{digitalWrite(7,LOW);}

    if(hour == 2 || hour == 3 || hour == 6 || hour == 7 || hour == 10 || hour == 11)
{digitalWrite(8, HIGH);} else
{digitalWrite(8,LOW);}

    if(hour == 4 || hour == 5 || hour == 6 || hour == 7 || hour == 12) {digitalWrite(9, HIGH);} else
{digitalWrite(9,LOW);}

    if(hour == 8 || hour == 9 || hour == 10 || hour == 11 || hour == 12) {digitalWrite(10, HIGH);} else
{digitalWrite(10,LOW);}

    }

// controlling the AC
// set up for every 30 seconds for demonstration
    if(second==58) {digitalWrite(11, HIGH);
    delay(500);
    digitalWrite(11, LOW);
    delay(500);
    digitalWrite(11, HIGH);
    delay(500);
    digitalWrite(11, LOW);
    delay(500);
    digitalWrite(11, HIGH);
    delay(500);
    digitalWrite(11, LOW);
    delay(500);
    digitalWrite(11, HIGH);
    delay(500);
    digitalWrite(11, LOW);
    delay(500);}

    if(second==28) {digitalWrite(12, HIGH);
    delay(500);
    digitalWrite(12, LOW);
    delay(500);
    digitalWrite(12, HIGH);}
delay(500);
digitalWrite(12, LOW);
delay(500);
digitalWrite(12, HIGH);
delay(500);
digitalWrite(12, LOW);
delay(500);}  

//setting the clock:
valm = analogRead(0); // add one minute when pressed
if(valm<800) {
    minute++;
    second=0;
    delay(250);
}

valh = analogRead(5); // add one hour when pressed
if(valh<800) {
    hour++;
    second=0;
    delay(250);
}