Sketch for Home Alert

The sketch is not large in terms of the line count, but it almost exhausts the Uno's available flash memory thanks to all the included libraries. There is lots of room for memory optimisation, but since I am at the prototyping stage, that's a project for another day. This code is available on Github.

Here is the sketch, with embedded comments:

```
#include <SPI.h>
#include <DMD.h>
#include <TimerOne.h>
#include "Arial_black_16.h"
```

Inclusions required for the DMD

```
#include <stdlib.h>
#include <Ethernet.h>
```

Inclusions required for the Ethernet Shield

```
#include <Wire.h>
#include "RTClib.h"
RTC_DS1307 rtc;
```

Inclusions required for the real-time clock, and declare the clock object (rtc).

```
#include "DHT.h"
```

Inclusions required for the DHT sensor, and definition of its data pin.

```
//Fire up the DMD library as dmd
#define DISPLAYS_ACROSS 1
#define DISPLAYS_DOWN 1
DMD dmd(DISPLAYS_ACROSS, DISPLAYS_DOWN);
```

Define the dimensions of the DMD and start it up. We only use a single DMD here.

```
#define DHTTYPE DHT22
#define DHTPIN 2
DHT dht(DHTPIN, DHTTYPE);
```

Define the type of DHT used (DHT22), its data pin, and initialise the sensor object (dht).

```
#define HW_ID        "1"
#define WEBSITE      "alert.arduinosbs.com"
#define WEBPAGE      "/get_message/"
#define IDLE_TIMEOUT_MS  3000
```

HW_ID is the ID of the device. It's ok to have multiple devices with the same ID, however remember that they will all be displaying the same message.

The host address and path to the web service that serves the message to be displayed on the DMD. If the service takes over 3 seconds to respond, give up.

```
byte mac[] = { 0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x02 };
IPAddress ip;
EthernetClient client;
```

Set the MAC address for the Ethernet Shield. Declare the ip address and Ethernet client objects.

```
char text_buffer[10];
```

A buffer used by the dtostr function, used to process the text that appears in the DMD.

```
boolean        reading      = false; //TRUE while the GET request is being received
String         get_request  = "";    //Holds the GET request
```

These are used by the GET response parser. I explain how this works in detail in Arduino SbS Lecture 38.

```
void ScanDMD()
{
  dmd.scanDisplayBySPI();
}
```

This function is called by the timer interrupt every 5ms. The call to scanDisplayBySPI refreshes the DMD. The interrupt is set in the setup function.

```
void setup(void)
```

Start the serial port so we can see messages from the sketch.

```
{
   Serial.begin(9600);
   Timer1.initialize( 5000 );
   Timer1.attachInterrupt( ScanDMD );
   dmd.clearScreen( true );

   dht.begin();
   Wire.begin();
   rtc.begin();

   if (! rtc.isrunning()) {
      rtc.adjust(DateTime(__DATE__, __TIME__));
   }

   if (Ethernet.begin(mac) == 0) {
   }
   Serial.print("My IP address: ");
   ip = Ethernet.localIP();
   for (byte thisByte = 0; thisByte < 4; thisByte++) {
     // print the value of each byte of the IP address:
     Serial.print(ip[thisByte], DEC);
     Serial.print(".");
   }
   Serial.println();
}

void loop(void)
{
   setup_DMD();
   make_get_request();
   process_response();
   display_environmental_info();
   show_time();
}

void setup_DMD(){
   dmd.clearScreen( true );
   dmd.selectFont(Arial_Black_16);
}
```

Initialise the Timer1 object to trigger an interrupt ever 5msec. Then, attached this interrupt to the ScanDMD function. This function will be called every 5msec and will refresh the LEDs on the DMD.

Clear the DMD.

Start the DHT device.

Start the I2C interface, and start the real-time clock device.

Check that the real-time clock is running. If it isn't, it means that it's time and date are not set, so set it based on the system time and date. The real-time clock device is discussed in detail in Arduino SbS Lecture 48 and 49.

Start the Ethernet Shield. An IP address will be leased from the local DHCP server, and it will be shown in the serial monitor. The Ethernet shield is discussed in detail in Arduino SbS Lecture 33 and 34.

Each loop starts by calling these 5 functions in sequence.

1. Setup the DMD
2. Make a HTTP request for a new message from the web service
3. Process the HTTP response from the web service and display it to the DMD
4. Display temperature and humidity info
5. Show the time for minute, then the date as scrolling text.

The setup_DMD function. Clears the screen, then sets the text font to Arial_Black_16.

```
void display_marquee(String &message)
{
    char buffer[message.length()+1];
    message.toCharArray(buffer, message.length()+1);
    dmd.clearScreen( true );
    dmd.drawMarquee(buffer,message.length(),
                          (32*DISPLAYS_ACROSS)-1,0);
    long start=millis();
    long timer=start;
    boolean ret=false;
    while(!ret){
      if ((timer+30) < millis()) {
        ret=dmd.stepMarquee(-1,0);
        timer=millis();
      }
    }
}
```

The **drawMarquee** function starts drawing the scrolling text onto the DMD. The text is read from an array of char. The **drawMarquee** function is also initialised with the length of the text message, and with the position from where the message is to appear. In this example, scrolling will start from the right side of the DMD, i.e. the the LED at the 31st column and 0th row. Change this to 0,0 to see what happens.

This structure will update the DMD 30msec after the last update. If you change the 30 to a higher number, the scrolling will slow down and vice-versa. The call to function **stepMarquee** is what does the actual update. The parameters control the direction of the scroll. With (-1,0), the text will move one LED to the left and 0 vertically. If you want to move the text towards the right, use (1,0). It is worth experimenting with these parameters in conjunction with those in **drawMaquee** to create different scrolling effects.

```
void show_time(){
    DateTime now = rtc.now();
    String hour;
    String minute;
    byte byteHour = now.hour();
    byte byteMinute = now.minute();
```

Get the time/date from the real-time clock. Store the hour and minute in byte variables to process later.

```
    if (byteHour < 10)
    {
      hour = "0";
      hour += String(byteHour,DEC);
    } else
    {
      hour = String(byteHour,DEC);
    }

    if (byteMinute < 10)
    {
      minute = "0";
      minute += String(byteMinute,DEC);
    } else
    {
      minute = String(byteMinute,DEC);
    }
```

Format the hour and minutes with leading zeros if they are less than 10.

```
    dmd.drawChar(  0,   3, hour[0], GRAPHICS_NORMAL );
    dmd.drawChar(  7,   3, hour[1], GRAPHICS_NORMAL );
    dmd.drawChar( 17,   3, minute[0], GRAPHICS_NORMAL );
    dmd.drawChar( 25,   3, minute[1], GRAPHICS_NORMAL );
```

Draw the text for hour and minutes. The **drawChar** function receives an LED position for the character we wish to print (column, row), then a single char value, and then the graphics mode. These modes are supported:

- GRAPHICS_NORMAL
- GRAPHICS_INVERSE
- GRAPHICS_TOGGLE
- GRAPHICS_OR
- GRAPHICS_NOR

```
for (int i=0;i<31;i++)
  {
    dmd.drawChar( 15,  3, '.', GRAPHICS_OR     );
    delay( 1000 );
    dmd.drawChar( 15,  3, '.', GRAPHICS_NOR    );
    delay( 1000 );
  }
}
```

This loop blinks a dot on and off, one second each, for 60 seconds in total, to show passage of time. You could also use writePixel to blink a single LED instead, with the same parameters.

```
void display_environmental_info(){
   byte b;
   DateTime now = rtc.now();
```

This function will display the time, date, temperature and humidity on the DMD

Get the current time and date from the real-time clock.

```
   String month;
   String day;
   String year;
   byte byteMonth = now.month();
   byte byteDay = now.day();
   year = String(now.year(),DEC);
```

Split the date to its components (day, month, year).

```
   if (byteMonth < 10)
   {
     month = "0";
     month += String(byteMonth,DEC);
   } else
   {
     month = String(byteMonth,DEC);
   }

   if (byteDay < 10)
   {
     day = "0";
     day += String(byteDay,DEC);
   } else
   {
     day = String(byteDay,DEC);
   }
```

If current month or day is smaller than 10, add a leading zero character to make the printout balanced.

```
   String marqueeText;
   marqueeText = day;
   marqueeText += '/';
   marqueeText += month;
   marqueeText += '/';
   marqueeText += year;
   marqueeText += ' ';
   marqueeText += dtostrf(dht.readTemperature(), 2, 2, text_buffer);
   marqueeText += '*';
   marqueeText += 'C';
   marqueeText += ' ';

   marqueeText += ' ';
   marqueeText += 'H';
```

This string will contain the message that will appear on the DMD.

The **dtostrf** function formats the temperature value that is returned by the DHT into a nicely formatted string that looks like this: 23.10.

Notice that I am only “adding” a single char at a time into the String object.

```
   marqueeText += ':';
   marqueeText += dtostrf(dht.readHumidity(), 2, 0, text_buffer);
   marqueeText += '%';
   display_marquee(marqueeText);
}
```

Call the **display_marquee** function which will display the string passed to it in the DMD.

```
void process_response(){
  unsigned long lastRead = millis();
  while (client.connected() && (millis() - lastRead < IDLE_TIMEOUT_MS)) {
    boolean currentLineIsBlank = false;
    get_request           = "";
    while (client.available()) {
       char c = client.read();
       if(reading && c == '\n')
        { reading = false;
          parseGetRequest(get_request);
          break;
        }

        if(reading){
          get_request += c;
         }
       if (reading && c=='\n')
       {
          break;
       }
       if (c == '\n' && currentLineIsBlank)  {
         reading = true;
       }
       if (c == '\n') {
         currentLineIsBlank = true;
       }
       else if (c != '\r') {
         currentLineIsBlank = false;
       }
    }
  }
  client.stop();
}
```

Once the body of the HTTP response has been capture, it is passed onto the **parseGetRequest** function where it will be parsed.

This function processes the HTTP response received from the web service. A detailed discussion that explains how it works is available in Lecture 38 of Arduino SbS.

```
void parseGetRequest(String &str) {
  Serial.print(F("Parsing this string:"));
  Serial.println(str);
  int   buzzer_state    = str[0] - '0';

  if (buzzer_state == 1)
      tone(3, 1000, 1000);

  String new_str = str.substring(1);
  display_marquee(new_str);
}
```

Parses the body of the HTTP response that was received from the web service.

Shows a message to the monitor. The F() function results in storing the string in flash memory instead of RAM.

The first character of the HTTP response from the web service will be either “1” or “0”. If it is a “1”, the buzzer will be activated. This character is obtained from **str[0]** (index 0 of the **str** object). We subtract the ASCII value of character “0” in order to get the actual numerical value of 1 or 0 rather than the ASCII value.

If buzzer_state is 1, then activate the buzzer, which is connected to digital pin 3, at 1KHz, for 1 second.

str.substring(1) will get all characters after the first (index 1) until the end of the string. This substring will be passed to the **display_marquee** function which will display it to the DMD.

```
void make_get_request(){
    if (client.connect(WEBSITE, 80)) {
     Serial.println(F("connected"));
     client.print(F("GET "));
     client.print(WEBPAGE);
     client.print(HW_ID);
     client.print(F(" HTTP/1.1\r\n"));
     client.print("Host: ");
     client.print(WEBSITE);
     client.print(F("\r\n"));
     client.println();
    } else {
     Serial.println(F("connection failed"));
    }
  }
```

This function will create an HTTP GET request to the web service. It is described in detail in Lecture 38 of Arduino SbS.

The main responsibility of this sketch is to make the Arduino a consumer of a web service. The web service is a simple web site with two end-points, one for a person to access via a web browser and submit a text string that they wish to display on the DMD, and another one where the Arduino will access in order to retrieve that text string.