In this project, I am using an Arduino board to show forecast of the weather from Google Weather service on a VGA screen. The display is obviously graphical one, with icons and some colorful text. It is one of those gadgets I wanted to have for some time now – A weather forecasting display that is there just like the wall clock. Yes I know you got nicer looking things on your angry-birds machine... I simply enjoy doing it myself. Had some challenges to overcome, like Arduino has only 2KB of RAM limitation, pretty small for parsing XML data…

As you can see in the pictures, the display shows all sorts of weather information and time (analog and digital clock). Of course you can do even nicer things if you were using some Android 7" device or something, but for me it would not be as fun as building this one.

**Features**

* Current and Forecast weather information graphically displayed
* Time display from the internet, using NTP protocol
* Does not require any PC to be involved
* Indoor temperature and humidity display
* Easy to connect and assembly, only Ethernet cable and power
* Supports DHCP for network connection
* Supports both VGA screens and small 3.2" screens
* Daylight-Saving support through a user button

Some more features include change of background color between night and day for the clocks and change background color of the weather data areas depending on temperatures.

Full assembly instructions are in word file inside the ZIP file with code and diagrams

**What is required to build this gadget?**

1. Arduino Ethernet Pro (<http://www.sparkfun.com/products/10536>) or Arduino with Ethernet Shield
2. 4D Systems display (uLCD-32PT) <http://www.sparkfun.com/products/10089> or display adaptor (uVGA II) <http://www.sparkfun.com/products/10329>
3. Arduino I/O headers socket
4. 5x Jumper wires (only 4 required for operation), note the display and display adaptor have male pins and Arduino typically takes female headers.
5. 2GB or less micro-SD (only needs about 150KB)
6. Power Supply
7. Ethernet Cable
8. DHT-22 Temperature and humidity sensor (optional, only if you want indoor temp)
9. Push button (optional)
10. 2x 10K Ohm resistor (optional, only if you want indoor temp)

**Also for building the gadget, you will need the following**

1. FTDI board, better use 5Volts one like this <http://www.sparkfun.com/products/9716> one.
2. PC for programming
3. A micro-SD reader device
4. Soldering Iron

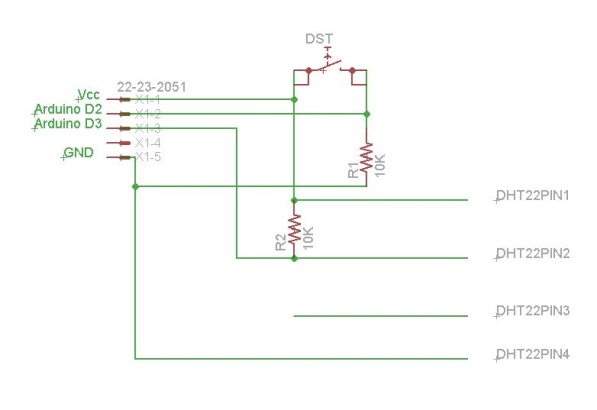
**Where to buy?**

All of the items can be purchased at Sparkfun or your favorite electronic components store. As for the display or display adaptor, they come pre-loaded with one of the firmware types SGC or GFX. It doesn't really matter which one you buy, as converting from one firmware flavor to the other is very simple task performed by a tool on the PC.

**How to assembly?**

Please read this thoroughly as there could be notes on changes that you might want to do.

1. After purchasing the goods, download software to your PC including:
   1. Arduino IDE (I use version 022)
   2. 4D Systems tools for programming their adaptors or displays from <http://www.4dsystems.com.au/prod.php?id=149>
   3. Arduino libraries including: NewSoftwareSerial, Arduino Time Library, Ethernet Library
2. Connect your display or display adaptor to the FTDI (Vcc to 5V, GND to GND, Rx to Tx and Tx to Rx, DTR on the FTDI to Reset pin) and the FTDI to USB on your PC
3. Upload the right firmware onto the LCD using **PmmC Loader** tool from 4D Systems (you need the GFX version)
4. Open the 4D systems Workshop tool; open the program file in the code\4D subdirectory of the ZIP file in this publication. Follow the steps as also shown in the picture for this tool to:
   1. Select the right display or adaptor model
   2. Select the right COM port (the one the FTDI generates)
   3. Select destination as "**Flash**" and not "Ram" (don't forget this!)
   4. Compile and Load the program to the display/adaptor
5. Place all the necessary Arduino libraries in the right location (if do not exist already)
6. Connect your Arduino to PC
7. Edit the Arduino files to match your time zone and place
   1. Edit the file NTPAndTime.cpp and change variable **timeZoneSeconds** to reflect the difference of your time zone from GMT. If you are west to GMT then use negative values.
   2. Change the variable **dsSeconds** to reflect the standard change from your time-zone
   3. Edit the file NetArduino.pde and change the variable **weatherDataGetString** to match you location (change Tel-Aviv to be your city). I recommend checking in the browser that it works by typing: <http://www.google.com//ig/api?weather=yourcity> in your browser address field. You should see some XML reply in return if that works.
8. Open the Arduino sketch NetArduino.pde, compile and load the program to your Arduino
9. Connect your micro-SD reader to your PC with the micro-SD in it
10. Save all the files from the **microSD** subdirectory into the root directory of the micro-SD
11. Safely remove the micro-SD from the reader and place it in the display or display adaptor
12. Disconnect the display and Arduino from PC
13. Connect the display or display adaptor to your Arduino as following:
    1. Vcc of the display/adaptor to +5V of Arduino
    2. GND of the display/adaptor to one of the Arduino GND connections
    3. Connect the Digital I/O pin 5 of Arduino to Rx pin (3rd pin from the right on the adaptor when VGA connector is on the top)
    4. Connect the Digital I/O pin 4 of Arduino to Tx pin (2nd pin from the right on the adaptor)
14. Build a small circuit for the push button (see optional DHT sensor circuit too for schema) as following:
    1. Push button connects to Vcc on one side and to resistor of 10K ohm on the other side.
    2. Other side of the resistor connected to ground
    3. Connect Arduino pin 2 to resistor/push-button junction
15. Connect the Ethernet cable to the Arduino
16. Connect the VGA display to the adaptor if you chose to use adaptor
17. Connect power supply to Arduino
18. Enjoy the weather ☺

If you also want the indoor temperature and humidity, follow these steps:

1. Build a small circuit with DHT22, Vcc goes to Arduino Vcc, GND to the Arduino GND and a 10K Ohm resistor from Vcc to the data pin.
2. Connect the data-pin of DHT22 to pin 3 of the Arduino

**Are you using Celsius or Fahrenheit?**

This program displays temperatures in Celsius. If however you are using Fahrenheit degrees, here are the changes you need to do (pretty simple)

1. Edit the 4GL display program file and change the function **ShowCelsiusSymbol** and change putstr("C"); to putstr("F");
2. Change the thresholds of temperatures for Hot and Cold indicators (TEMP\_THHOT and TEMP\_THCOLD constants) to what you think is good for you (89 and 68 is the equivalents of my current Celsius settings)
3. Edit the Arduino file NetArduino.pde and change the function **\_ProcessWeatherXMLLine** so that instead of the line:

if ((location = matchXmlTag(ptr, PSTR("<temp\_c data="))) != 0)

You will have

if ((location = matchXmlTag(ptr, PSTR("<**temp\_f** data="))) != 0).

That is take the XML tag temp\_f instead of temp\_c

1. Change the function **Far2Cel** in the same file to basically return t; instead of all the lines in there.
2. Compile and load the programs to the devices

**How it works?**

The Arduino is connecting to the Internet do download both time information (using NTP) and weather information from Google Weather service. Currently the Arduino stores this information in RAM and shows it on its default serial port (so if Arduino is connected to your PC you can see the text running). The weather information is polled once every minute and time is polled about once every 5 minutes.

Arduino sends this data over simple protocol of serial to the display that copies the data to internal memory, and show it graphically on the screen. As the display has a high-level programming language, it draws the screens and display images reflecting the weather condition. This is my first ever 4GL program, so it is likely that I did not do everything in most efficient way, or maybe even the wrong way. ANY feedback is welcomed.

Between updates to time, the display uses internal timer based interrupt that works every second to advance the time.

On VGA screens (the display resolution is 800x600), we have plenty of room to show all the information, so the display only flips the Analog and Digital clocks. On the smaller screens the program flips 4 pages: Digital clock, current weather, forecast weather and analog clock. Do note that the forecast on the VGA display will show 3 days of forecast while on the 3.2" display (resolution of 320x240), only 2 days forecast will be displayed.

The communication protocol between Arduino and the display is very simple one. I use frames, where each frame passes only single member data. Frame starts with '[' character and ends with ']' character. All data is transferred in ASCII mode. I will not detail the entire protocol, but here are some examples: City name is part of the Information section so if the city is Tel-Aviv, then the Arduino will send the display the following data: [ICTel-Aviv]

If for another example, the forecast for today high temperatures is to be 32 degrees, the Arduino will send the display the following string (0 means the entry in the forecast array, and member 0 is the forecast for today in the Google weather service): [FH032] similarly if the icon to be displayed for tomorrow condition is rain.gif then the Arduino will send [FI1rain.gif]. Simple isn't it?

The datasheet of the display is saying on some baud rates there are expected errors, so I wanted to avoid writing any CRC code, and chose the baud rate to be 31250, in which no errors are expected. If you want to play with the display using some terminal you can see the baud rate setting line in the **setup()** function and comment and uncomment the right one.

**To Do and Notes**

This program is presented as v0.95 on purpose. While it is running for several days already and I solved many annoying little bugs, there could very well be more. The 4GL program is my first ever program in this language. It is very likely to be sub-optimal and perhaps even with problems that were not found yet. The Arduino program could be improved, but it is fairly clean code for a gadget like this.

I am not a graphical designer and I will agree with anyone who will say it can be better presented. For me the current layout and graphical display is more than enough. Knowing your weaknesses is good right? If you have done something nicer would appreciate if you will share.

The implementation for setting (toggle) the day-light-savings (DST) on and off is very naive. I am thinking of doing that with a button that connects to the 4D display and not with Arduino as it is now. That will make the work with touch-screen a better fit. If you happen to write this, send me back the code please.

When using LCD with the touch option, it is possible to also use the touch to setup location and perhaps few more things, not only DS. As I have no intentions to leave Tel-Aviv in the near future, I did not do it. Hey, remember this is a hobby not a product.

Specifically I know I downloaded many images, including snow.gif (last snow in Tel-Aviv was in 1950). However, it is possible that there are conditions I do not know about. Since the API of Google Weather is not documented, I honestly don't know if there are more. If you come across a missing one, here are the steps required to add an additional icon (see also pictures for using the tools):

1. Get the missing icon from Google by following these steps (see picture too):
   1. Open your browser and point it to Google service by typing <http://www.google.com//ig/api?weather=yourcity> in your browser
   2. See missing icon should be listed there in the XML in an entry similar to this: <icon data="**/ig/images/weather/foobar.gif**"/> with icon name instead of foobar of course.
   3. Download the image by typing the following in the browser: <http://www.google.com/ig/images/weather/foobar.gif>, right click on the image and save it to the folder with the images (**microSD** folder).
   4. Launch the 4D systems tool called Graphics Composer, open the file **goowea.gcs**
   5. Push the Add button below the list of image files and add your new icon file
   6. Save the file using the save button
   7. Write down where the new icon file was added (before what and after what other file)
   8. Push the build button (looks like a chip with red legs)
   9. Exit the tool
2. Connect the micro-SD reader to your PC and save the folder of **microSD** on it again
3. Modify the 4D display program to know about the new icon and place the name in exactly the same place of order as saved from step 1.g above. So if the new icon name is **foobar.gif** then edit the file WeatherScreen.4dg in the 4D directory (assuming foobar.gif is the name and placed between coudy.gif and haze.gif. Change the lines:

byte i\_Cloudy "cloudy.gif",0

byte i\_Haze "haze.gif",0

To:

byte i\_Cloudy "cloudy.gif",0

byte i\_FooBar "foobar.gif",0

byte i\_Haze "haze.gif",0

And the lines

word IconNames

i\_coRain, i\_coSnow, i\_coStorm, i\_Cloudy, i\_Haze, i\_mCloudy, i\_mSunny, …

To

word IconNames

i\_coRain, i\_coSnow, i\_coStorm, i\_Cloudy, **i\_FooBar**, i\_Haze, i\_mCloudy, i\_mSunny, …

1. Send me the data, please