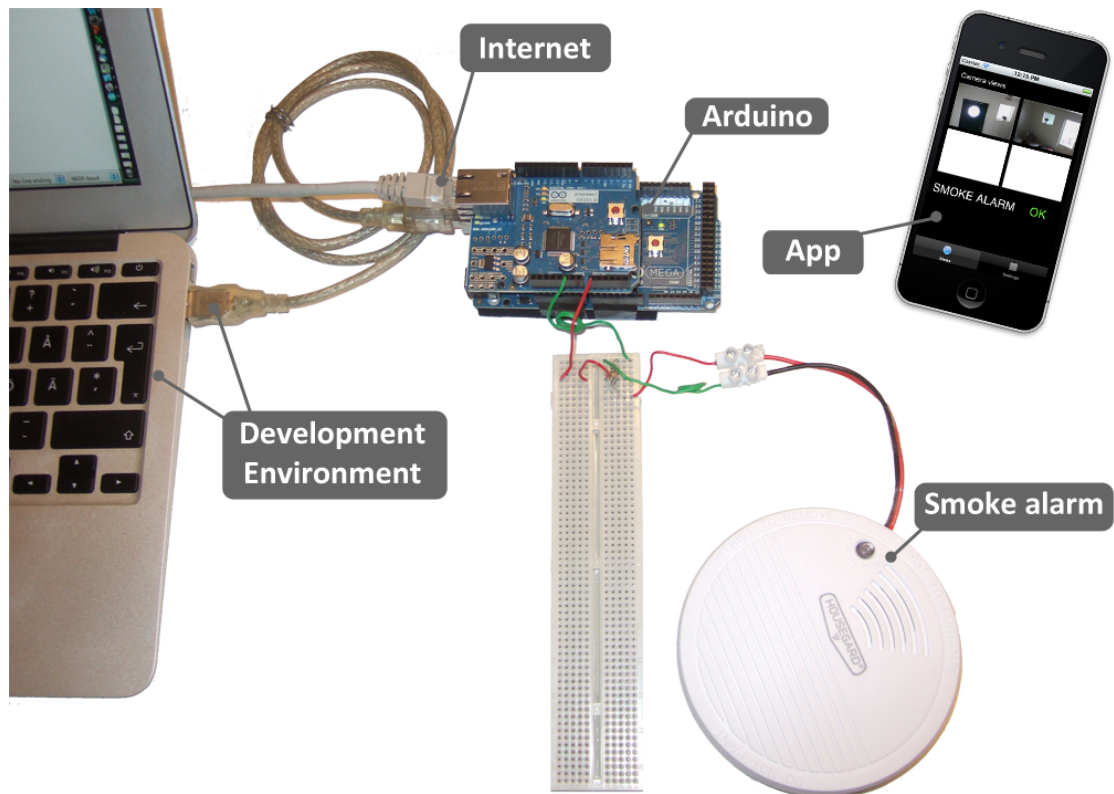


Network based smoke alarm - using Arduino and iPhone



Introduction

How cool would it not be to check the status of your smoke alarm through an mobile app over Internet?! I would say very cool and totally necessary! So in this document I will describe the technical solution of my network based smoke alarm.

The technical solution is rather simple. When smoke hits the smoke alarm (detector), the smoke alarm will send out voltage to its buzzer. That voltage will be read by an analog input pin on the Arduino. The Arduino will post the value to a web server (running PHP/MySQL). In the other end of the system, an iPhone app will continuously read the values and notify the user if a high voltage value is found.

Table of contents

INTRODUCTION	2
TABLE OF CONTENTS	3
PREREQUISITES	4
CREDITS	4
EQUIPMENT	4
SYSTEM OVERVIEW	5
BREADBOARD VIEW	5
BUILD INSTRUCTIONS	6
SETUP THE HARDWARE	6
SETUP THE SOFTWARE	6
MODIFY THE SMOKE ALARM	7
CONNECT THE SMOKE ALARM TO THE ARDUINO THROUGH A COUPLING PLATE	7
TEST THE MODIFIED SMOKE ALARM	7
MODIFY THE ANALOGREADSERIAL	8
DEVELOP THE WEB SITE	9
DEVELOP AN APP	9
POSSIBLE IMPROVEMENTS	11
REFERENCES	11

Prerequisites

All build steps are not described in detail (some are just mentioned briefly). So this document requires that the reader have a basic knowledge of electronics and software development. Don't forget to ask www.google.com when you need help.

Credits

Thanks to my employer HiQ [5] that gave me time to finalize this project and also thanks to Jakob Selbing who supported with electric skills.

Equipment

All the equipment I used in this experiment is listed below. Some parts can be changed to similar solutions. E.g. if you don't have a Mac, you can use a PC and create an Android app instead.

1. Arduino Mega 2560 rev3,
<http://www.kjell.com/sortiment/el/elektronik/mikrokontroller/arduino/arduino-mega-2560-rev3-p87861>
2. Arduino Ethernet Shield rev3,
<http://www.kjell.com/sortiment/el/elektronik/mikrokontroller/arduino/arduino-ethernet-shield-rev3-p87864>
3. Coupling plate,
<http://www.kjell.com/sortiment/el/elektronik/elektroniklab/kopplingsplatta-lodfri-p36251>
4. Cables for coupling plate,
<http://www.kjell.com/sortiment/el/elektronik/elektroniklab/bygelsladdar-140st-p36254>
5. Thin wires that can be used for soldering
6. Housegard HG-102 smoke alarm,
http://www.teknikmagasinet.se/db.pl?tf=product_link.html&link=&artnr=570477&title=Brandvarnare%20optisk&
7. Computer with Arduino development environment,
<http://arduino.cc/en/Main/Software>
8. Computer with Apple Xcode development environment (in my case MacBook Air), <http://www.apple.com/se/macbookair/specs.html>
9. Two 5,6K ohm resistors, e.g.
<http://www.clasohlson.com/se/Motst%C3%A5ndssats/Pr363840000>
10. RJ-45 Ethernet cable
11. USB cable for Arduino programming (upload of application)
12. Soldering iron
13. Solder
14. Optional: multimeter
15. Web server with PHP & MySQL

System overview

The remote controlled smoke alarm system consists of four main parts, microcontroller, smoke alarm, web server and an smoke alarm app (for iPhone).

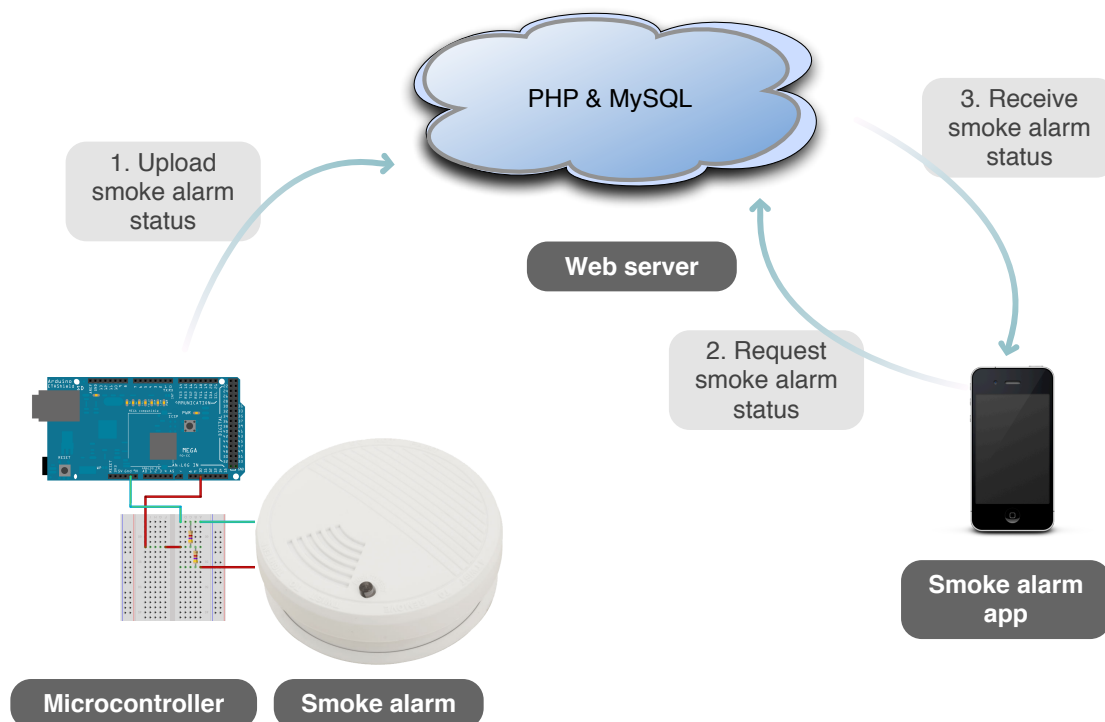


Figure 1 - Overview of the remote controlled smoke alarm.

When smoke hits the detector in the smoke alarm, the IC circuit will send out ~10 volt on pin 7. The Arduino that is connected to the smoke alarm on pin 7 and its ground will detect that the voltage raises from 0 to 5 volt. Since the Arduino only can handle 0-5 volts, two resistors on the coupling board will reduce the voltage from 10 to 5 volt.

When the Arduino reads the voltage on the analog input pin 0, it will send the value to the web server with a HTTP GET request message. The web server running PHP receives the request message and stores the information in a MySQL database.

The phone with the smoke alarm app continuously polls the web server for the smoke alarm status, if it receives a status value that indicate smoke (>0 volt) it will notify the user.

Breadboard view

The ground (GND) and analog input 0 (A0) pin is used on the Arduino. On the smoke alarm the IC (BL59S10) I/O pin (number 7) and ground is used.

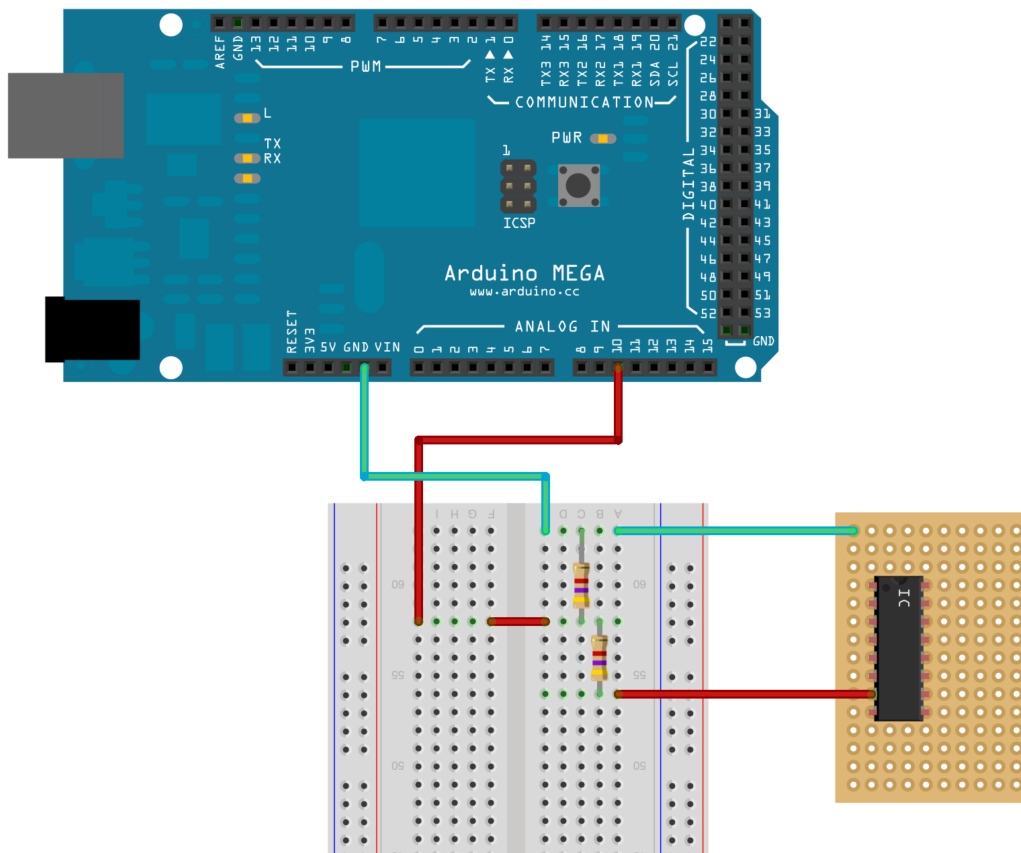


Figure 2 - Breadboard view.

Build instructions

The instructions below should be read in chronological order.

Setup the hardware

- Put the Arduino Ethernet Shield rev3 on top of the Arduino Mega 2560 rev3 (the two units is from now called "the Arduino").
- Connect the Arduino with a RJ-45 Ethernet cable to Internet and a USB cable to the computer.

Setup the software

- Download and install the Arduino development environment from <http://arduino.cc/en/Main/Software>.
- Install Apache webserver with PHP, MySQL and phpMyAdmin or create an account at a web hotel that provide such services.
- Download and install Xcode. Xcode is free as long as you do not need to run the app on real hardware. There is a very good inbuilt iPhone and iPad simulator in Xcode that can be used for this purpose. Xcode can be

downloaded for free from Mac OS X App Store (or
<https://developer.apple.com/xcode/>).

Modify the smoke alarm

- Open up the smoke alarm (HG-102) and take out the circuit board.
- Verify that the 9 volt battery is not connected to the circuit board.
- Solder a thin (red) wire on the dot in front of the IC's pin 7 and a thin (black) wire on the ground, see Figure 3 below.

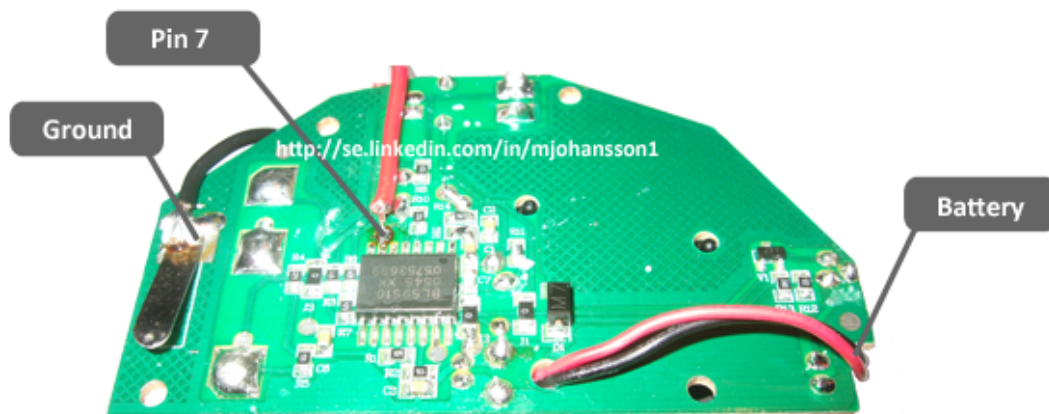


Figure 3 - Modification of the smoke alarm circuit board.

- Do not connect the red wire on pin 7 directly to the Arduino since it has almost 10 volt, which is more than the Arduino is made for. If you have a multimeter you can now add the 9 volt battery and measure the voltage on the newly soldered red and black wire when you press the test button. It should be around 10 volt. When the test button is un-pressed the multimeter should measure 0 volt.

Connect the smoke alarm to the Arduino through a coupling plate

- Connect the newly soldered ground (black) wire and pin 7 (red) wire to the coupling plate as shown in Figure 2.
- Add two resistors at 5,6 K ohm as shown in the same figure. Since the output voltage from the smoke alarm circuit is around 10 volt, it must be reduced for the Arduino which only can handle 0-5 volt. This is made by using a voltage divider (see [2] for more information).
- Connect the other wires as shown in the figure. The wires to the Arduino are connected to the ground (GND) and analog input 0 (A0) pins.

Test the modified smoke alarm

- It is now time to test the modifications.
- Open the Arduino development environment and from the File menu select Examples > 01.Basics > AnalogReadBasics. This example is a good starting point for this development since it continuously reads and displays the voltage on analog input pin 0 (A0).

- Start and configure the Arduino development environment (see [3] for more information).
- Go to menu Tools > Board and select Arduino Mega 2560 or Mega ADK
- Go to menu Tools > Serial Port and select something like/dev/tty.usbmodem (can differ).
- Open an example program from menu File > Examples > 01.Basics > AnalogReadSerial.
- Upload the code to the Arduino and show the Serial Monitor (menu Tools > Serial Monitor).
- Press the test button on the smoke alarm circuit and verify that the Arduino detects the voltage that should be around 4-5 volt.

Modify the AnalogReadSerial

- Save the AnalogReadSerial as something else (e.g. FireAlarmReporter).
- The FireAlarmReport needs to be modified to report the voltage to the web server using the Ethernet library [4]. Sorry, but I will not go into details here so you have to learn by your self how to use it.
- In this experiment, the FireAlarmReporter sends HTTP GET requests to <http://complexit.se/arduino/rval.php?sid=1&v=4>, where *rval.php* is the PHP file that will save the data to the MySQL server, *sid* is an own-defined ID for this "fire alarm reporter unit" and *v* is the voltage value.

Code snippet that can be useful:

```
char url_host[] = "www.complexit.se";
EthernetClient client;
int sensor_id = 1;
int sensor_value = 0;

int reportSensorStatus() {
  Serial.println("Reporting status...");
  size_t printed = client.print("GET ");
  client.print("/arduino/rval.php?");
  client.print("sid=");
  client.print(sensor_id);
  client.print("&v=");
  client.print(sensor_value);
  client.println(" HTTP/1.0");
  client.println("Host: www.complexit.se"); //Needed for shared hosts
  client.println();
  Serial.print("Sent ");
  Serial.print(printed);
  Serial.println(" bytes.");
}
```

The code snippet above can be used to report the value (*sensor_value*) to the web server. Save the return value from `analogRead(A0)` into *sensor_value* before invoking the `reportSensorStatus` method.

As you can see in the code I have added the comment "//Needed for shared hosts". That line is required if your webserver's ip number is shared with other websites (common for web hotels). The line will tell the web server which one its websites that want to receive the HTTP GET request method.

Develop the web site

Now when the Arduino can send the smoke alarm voltage values with HTTP GET request messages, it is time to fetch those messages. That will be done with PHP and MySQL.

- I use three files on the webserver: rval.php (register values), gval.php (get values) and cfg.php (webserver configuration). The cfg.php file is included from the rval.php and gval.php files.
- The following code is used for receiving the variables in the HTTP GET request:

```
$sensor_id = $_GET['sid'];  
$sensor_value = $_GET['v'];
```

The sensor_id and sensor_value is then saved into a table.

- To create the table that will store the smoke alarm values, I recommend phpMyAdmin (<http://www.phpmyadmin.net/>).
- The gval.php is the file that the app will read, so it takes a sensor id as input and returns the sensor's value to the app. The data sent to the app is in my solution in XML format. A code snippet from my gval.php is:

```
$sensor_id = $_GET['sid'];  
echo get_response_message_xml(get_sensor_status($sensor_id));
```

The sensor id is saved to sensor_id in the same way as in rval.php. The get_sensor_status is a method that looks into the database for the latest voltage value for the given sensor_id. The get_response_message_xml is a method that presents a given input value in a XML format. That XML format is also known by the app.

Develop an app

Since I have a Mac and an iPhone, the decision of making an iPhone app for this experiment wasn't so difficult. If you have a Mac but no iPhone you can test your app in the iPhone/iPad simulator. If you don't have a Mac, this can be done in Eclipse by using the Android SDK [6].

- Open Xcode and start a new project of type iOS Application > Tabbed Application. On the first tab I placed the status of the smoke alarm and two web camera views. The second tab can be good for adding and removing sensors or other devices you might want to monitor.
- In the first tab (FirstViewController.m) that is automatically generated by Xcode, I placed a NSTimer that continuously polls the gval.php file and then updates an UILabel with the sensor status. The UILabel will show "ALARM!" or "OK", depending of the received value from the gval.php.
- The XML data is downloaded by the download method that uses the NSURLConnection class. A short code snippet that might help:

```
-(NSData *)download:(NSString *)server {
    /* ... code removed... */
    NSURL *serverUrl = [NSURL alloc] initWithString:server];
    serverUrlRequest = [NSMutableURLRequest
    requestWithURL:serverUrl];

    NSData *rawData = [NSURLConnection
    sendSynchronousRequest:serverUrlRequest
    returningResponse:&resp error:&err];
    if(!rawData) {
        NSLog(@"Check Internet connection.");
        return nil;
    }

    httpResponse = (NSHTTPURLResponse *)resp;
    switch ([httpResponse statusCode]) {
        case 200:
            {
                /* ... code removed ... /
                return rawData;
            }
            break;
        default:
            {
                NSLog(@"Unhandled statusCode=%d",
                [httpResponse statusCode]);
                break;
            }
    }
    return nil;
}
```

The download method returns NSData* on success or nil on failure. The NSData* is then (outside the download method) converted to an UTF-8 encoded NSString*.

- To get the actual smoke alarm voltage value, I implemented a XML parser that is compatible with the NSXMLParser class. It takes the UTF-8 NSString and returns the smoke alarm status.
- When all your code are in place, just press Command + R to run your app on either a real iPhone (require to buy developer license) or in the iPhone simulator.
- When I did this I could not resist to also add my IP cameras to the app. I added two DCS-932L web cameras. If the app says "ALARM!" I can now also verify that visually in real time.

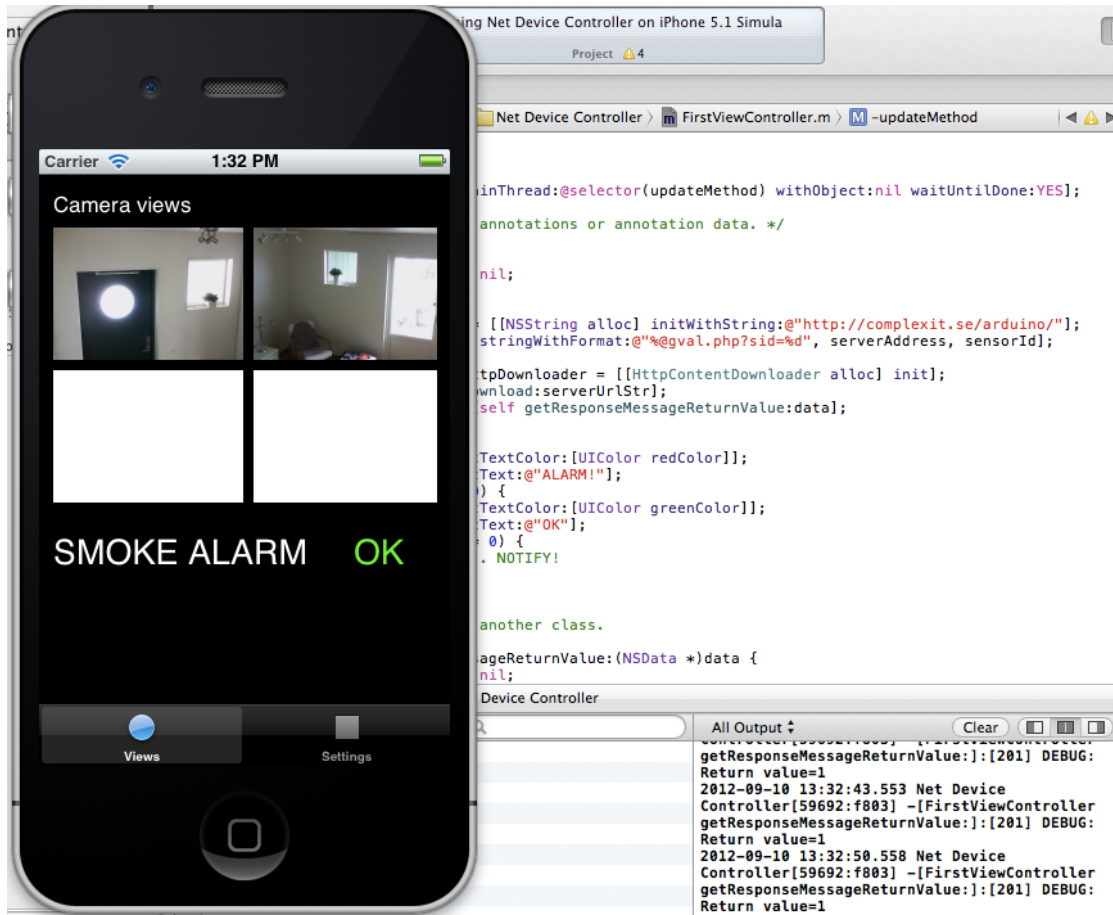


Figure 4 - The smoke alarm app in an iPhone simulator.

Possible improvements

My impression of the smoke alarm detector HG-102 is that it is not so sensitive. It requires, in my opinion, rather much smoke before it starts to alert. If I will make this again I would choose another smoke alarm detector.

It is also overkill to have an Arduino Mega 2560 rev3 for this purpose. There are cheaper alternatives like Arduino Uno that is also compatible to the Arduino Ethernet Shield rev3.

It is also possible to connect additional wires to the smoke alarm in order to control the buzzer. That can be useful if you want to remote test the smoke alarm. Or why not joke with the persons in the room...

References

- [1] BL59S10 data sheet, URL: <http://html.alldatasheet.com/html-pdf/164341/BELLING/BL59S10/294/1/BL59S10.html>
- [2] Voltage divider, URL: http://en.wikipedia.org/wiki/Voltage_divider
- [3] Getting started w/ Arduino on MacOSX, URL: <http://arduino.cc/en/Guide/MacOSX>
- [4] Arduino Ethernet library, URL: <http://arduino.cc/en/Reference/Ethernet>
- [5] HiQ, <http://www.hiq.se/>

By Mikael Johansson,
<http://www.linkedin.com/in/mjohansson1>
2012-09-08

[6] Android SDK, <http://developer.android.com/sdk>

Updated 9/11/12 10:57 AM