Introduction

The ***MyDoorOpener*** project has attracted a surprisingly high number of enthusiasts over the years. Last year we introduced a simplified documentation and assembly process to the masses and reached thousands of people. That initiative brought the project accessibility to a whole new level and the project's reach went beyond the experienced electronics hobbyists.

Today, we have even better news ... that process got even simpler! Some of the parts involved in our prior assembly process got discontinued over time but newer and simpler parts got available on the market. Therefore, today we are thrilled to present you with a revisited and even simpler assembly process.

Again, the objective of this assembly process is to provide an easier way to build the project, particularly for people with more limited knowledge of general electronics.

This doesn't mean you don't need to know anything about electronics ... You still need to have some basic understanding. What it does mean is that you can get around with a simplified components selection/purchase process. For many, this is great news, which makes the project a whole lot more accessible.

This revisited simplified approach has pros and cons.

Pros
Single source for components purchase
Simplified components selection process
Simplified assembly procedure
No (or minimal) soldering required

Cons
Final assembly may not be as robust as when soldering techniques are used
Must deal with limited industry available components, enclosure, etc

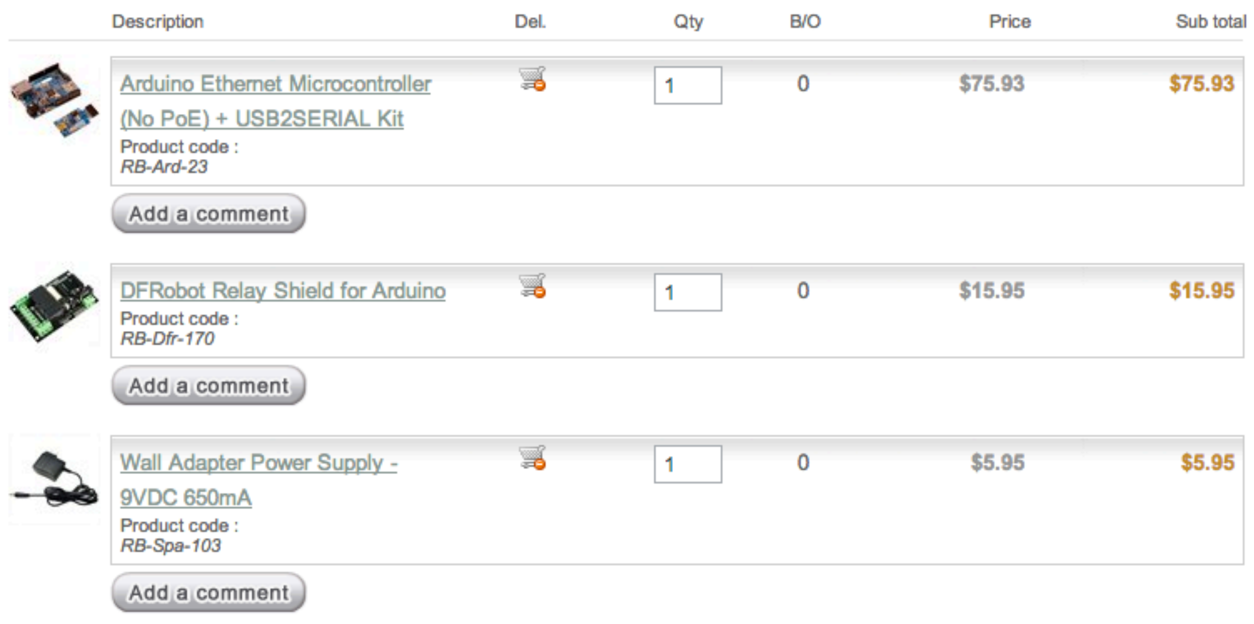
Components Supplier

The components for this assembly process can be ordered directly from the following electronics parts suppliers (we are not affiliated with any of them in any way) :

Suppliers
http://www.adafruit.com/
http://www.dfrobot.com
http://www.robotshop.com/

Components List

Following is the list of components that should be ordered to build the ***MyDoorOpener*** project (this setup can control up to 4 doors/devices) :



Description	Del.	Qty	B/O	Price	Sub total
Arduino Ethernet Microcontroller (No PoE) + USB2SERIAL Kit Product code : *RB-Ard-23*		1	0	$75.93	$75.93
DFRobot Relay Shield for Arduino Product code : *RB-Dfr-170*		1	0	$15.95	$15.95
Wall Adapter Power Supply - 9VDC 650mA Product code : *RB-Spa-103*		1	0	$5.95	$5.95

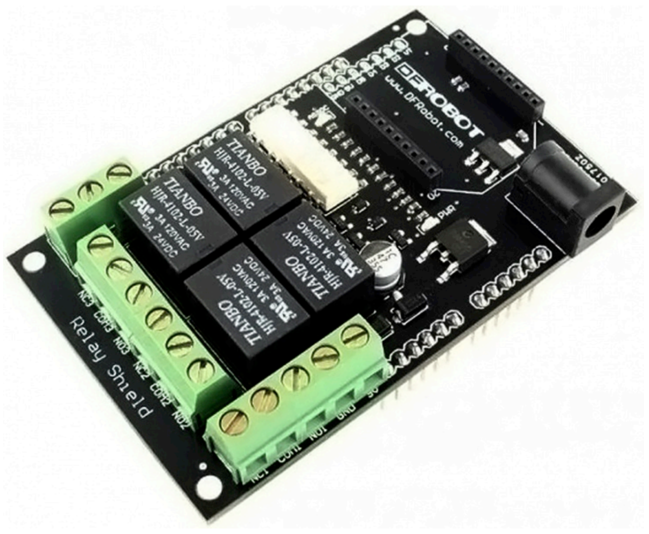
The prices above can give you an indication of rough costing for the project, but those prices may obviously change over time. Same applies for the actual part numbers and their availability. We can't guarantee anything, but we'll try our best to keep this document up to date (please report any discrepancy you find when purchasing your components).

The Hardware

The Arduino Ethernet board is a 2-in-1 board that combines an Arduino micro-controller board and an Ethernet shield, all on a single board. Alternatively, you could use a separate Arduino Uno board combined with an Arduino Ethernet shield.

REF: Arduino Ethernet

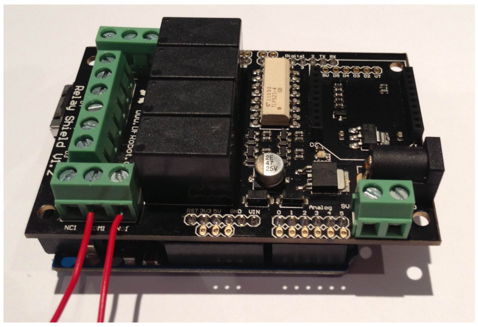
The DFRobot Relay shield offers 4 controllable relays on a single shield. It also has screw connecting blocks for easy attachment of wires. Alternatively, you could use the similar SeedStudio relay shield, although it hasn't been tested officially with MyDoorOpener.



REF: DFRobot Relay Shield

The Assembly Process

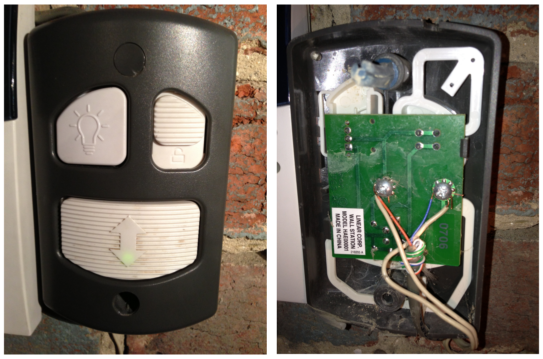
1. Assembly required for opening and closing of door/device.



Connect one of the DFRobot relays to your existing garage door opener wall button. This will require the installation of 2 low voltage wires going from one of the terminal blocks labelled COMx and NOx, to your existing garage door wall mount button station (the ‘x’ in COMx and NOx is a value between 1 and 4).

When you open up your garage wall mount button station, you’ll typically find 2 screws onto which you’ll attach the other ends of the 2 low voltage wires coming from the relay block.

It doesn’t matter which terminal from the wall mount button station gets linked with COMx or NOx, for as long as the entire setup can form a closed loop whenever the relay gets triggered.

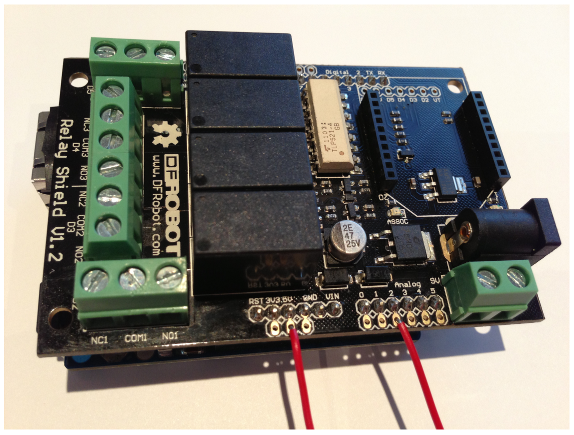


Repeat the above procedure for each door or device you want controlled (up to 4), using a distinct COMx/NOx terminal block for each door/device.

Following is the pin# used for each of the relays on the Relay Shield:

Device #	Pin for relay
1 (operated via relay @ COM1/NO1)	Digital pin #2
2 (operated via relay @ COM2/NO2)	Digital pin #3
3 (operated via relay @ COM3/NO3)	Digital pin #4
4 (operated via relay @ COM4/NO4)	Digital pin #5

2. Assembly required for status monitoring (opened/closed) of door/device.



Connect the DFRobot Relay Shield to your status sensing device, typically installed on your door/device. This will require the installation of 2 low voltage wires going from the shield's GND pin and one of the shield's analog pins. Have those 2 wires form a closed loop, going thru your status sensing device.

When connecting the 2 wires on the DFRobot Relay Shield, you will need to solder them onto the board (or fasten them using silicone or any other fastener you're comfortable working with).

We recommend using the following pin configuration for your status sensing device(s):

Device #	Pins for sensor
1 (operated via relay @ COM1/NO1)	Analog pin #2, GND
2 (operated via relay @ COM2/NO2)	Analog pin #3, GND
3 (operated via relay @ COM3/NO3)	Analog pin #4, GND
4 (operated via relay @ COM4/NO4)	Analog pin #5, GND

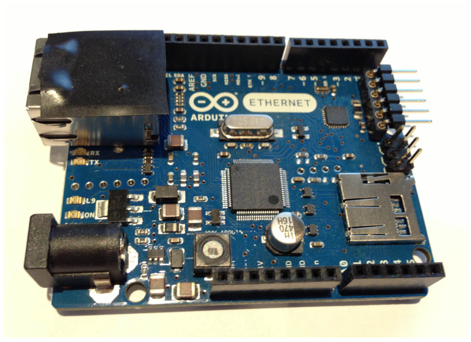
Your status sensing device should be configured so that it connects the GND to the analog pin when the door is closed. When the door gets opened, your status sensing device should disconnect the GND from the analog pin. This is how MyDoorOpener knows the door/device is opened or closed. So, GND attached to analog pin means the door is closed and a broken link means the door is opened.

You can use any status sensing device you wish, for as long as it respects the above rules. Typically, you'd use contact sensors which should be readily available from your local hardware store:

Typical alarm contact sensors should work great. You can also Google around "contact sensor" or "garage contact sensor" to find good candidates for your project.

Repeat the above procedure for each door or device you want monitored, using a distinct analog pin for each door/device. The GND pin should be shared for all sensors.

3. Connecting the Arduino Ethernet board and DFRobot Relay Shield together.

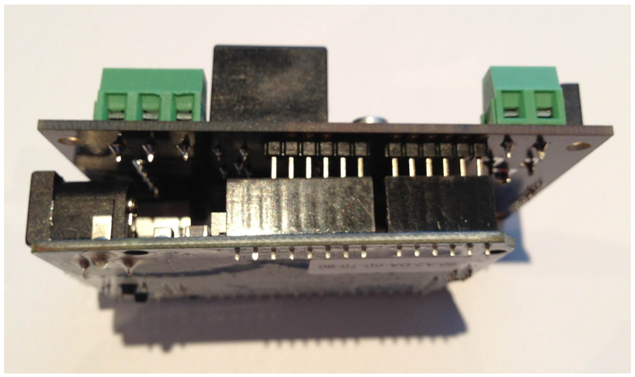


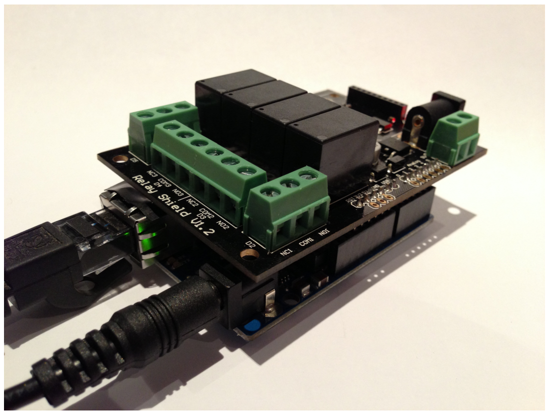
The DFRobot Relay Shield design has pins underneath it, that, if not properly protected, could produce a short-circuit when connected onto the Arduino Ethernet board.

Therefore, you should place some black electrical tape on top of your Arduino's Ethernet connector to prevent such short-circuit from occurring.

Snap the DFRobot Relay Shield onto the Arduino Ethernet board.

The height of the ethernet connector on the Arduino Ethernet board prevents the relay shield from evenly snapping onto the Arduino, but that shouldn't prevent things from working properly.





Connect the Arduino controller to your network/router using a standard RJ45 network cable. If you want to attach your Arduino to a wireless network, you can do so using a TP-LINK nano router in client mode (or any other similar products). Using such product will make your Arduino Ethernet, Wi-Fi enabled.

Connect the Arduino controller to the power supply and have it connected to an AC wall outlet. Don't use a 9v battery or USB to power your hardware. Always use an AC wall adapter with enough power to drive the relays.

This concludes the hardware assembly process.

Software Configuration

1. Download the Arduino IDE software from the Arduino web site, based on the operating system you are using :

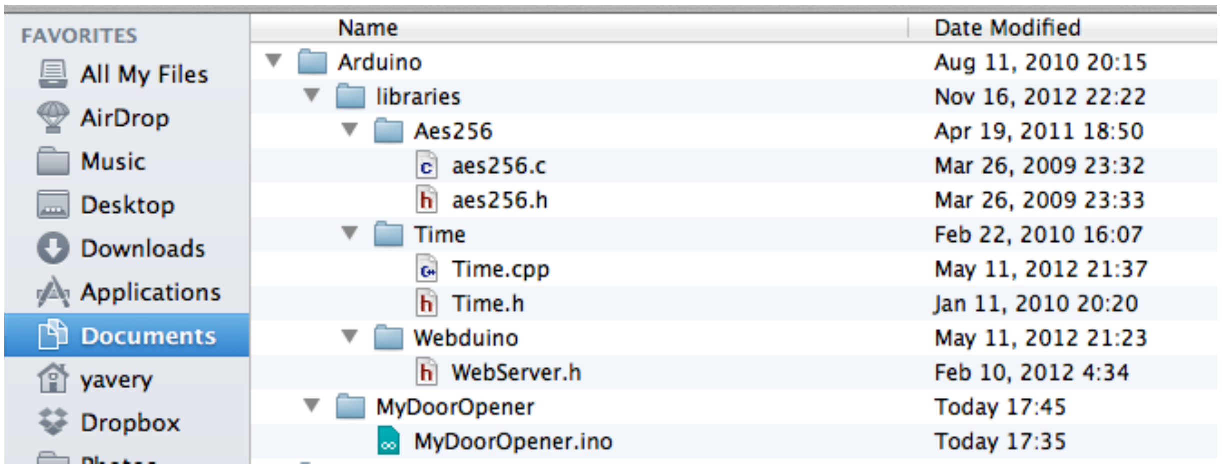
Operating System	Arduino Download
Windows	http://arduino.googlecode.com/files/arduino-1.0.3-windows.zip
Mac OSX	http://arduino.googlecode.com/files/arduino-1.0.3-macosx.zip
Linux 32bit	http://arduino.googlecode.com/files/arduino-1.0.3-linux32.tgz
Linux 64bit	http://arduino.googlecode.com/files/arduino-1.0.3-linux64.tgz

* Note that the version currently supported is **version 1.0.3**. Using any other version of the Arduino IDE is not officially supported.

2. Download the MyDoorOpener project ZIP file :

   https://github.com/yanavery/MyDoorOpener-Arduino/zipball/master

3. Have the files extracted from the ZIP file so that the directory and file structure is similar to the following :

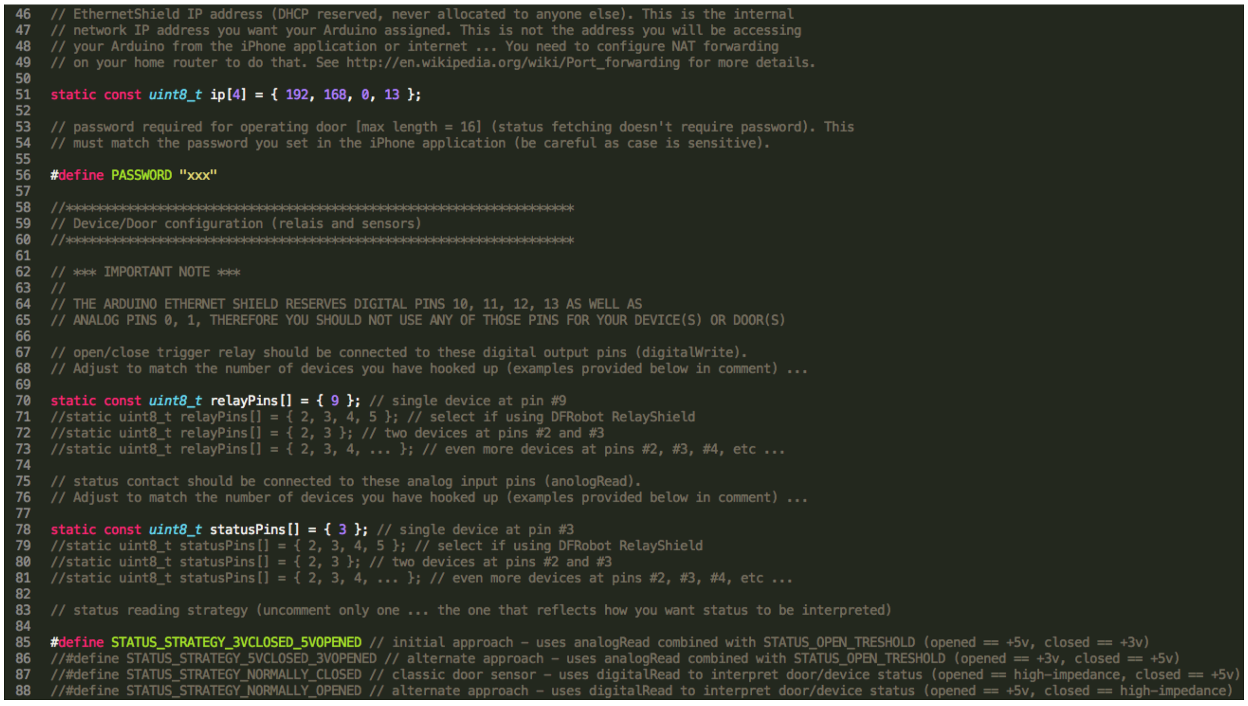


Name	Date Modified
Arduino	Aug 11, 2010 20:15
libraries	Nov 16, 2012 22:22
Aes256	Apr 19, 2011 18:50
aes256.c	Mar 26, 2009 23:32
aes256.h	Mar 26, 2009 23:33
Time	Feb 22, 2010 16:07
Time.cpp	May 11, 2012 21:37
Time.h	Jan 11, 2010 20:20
Webduino	May 11, 2012 21:23
WebServer.h	Feb 10, 2012 4:34
MyDoorOpener	Today 17:45
MyDoorOpener.ino	Today 17:35

Respecting the location of libraries is important. If you don’t, you will get compiler warnings and/or errors when compiling the project with the Arduino IDE.

4. Open the MyDoorOpener.ino file inside the Arduino IDE. Edit the following lines to match your specific configuration (and save) :

Line #	Item	Description
~ 51	IP Address	Change to match the network IP address you want your Arduino to be accessible from. You will need to make sure this network IP address is reserved and never assigned to any other devices on your internal network. This is not the address you will be accessing your Arduino from the iPhone application. You need to configure NAT forwarding on your home router to do so. See http://en.wikipedia.org/wiki/Port_forwarding for more details.
~ 56	Password	Change to the password you want to protect your Arduino with. This password needs to match the password you will be entering in the iPhone application.
~70	Relay Pins	Comment out the default relay pins configuration line (the one that specifies pin #9 only) and uncomment the one that specifies pins **#2, #3, #4 and #5** instead.
~78	Status Pins	Comment out the default status pins configuration line (the one that specifies pin #3 only) and uncomment the one that specifies pins **#2, #3, #4 and #5** instead. **Important: From that array remove any pins that you will not be using.**
~85	Status Strategy	Comment out the default ***STATUS_STRATEGY_3VCLOSED_5VOPENED*** status strategy and uncomment the ***STATUS_STRATEGY_NORMALLY_CLOSED*** instead.



```
46  // EthernetShield IP address (DHCP reserved, never allocated to anyone else). This is the internal
47  // network IP address you want your Arduino assigned. This is not the address you will be accessing
48  // your Arduino from the iPhone application or internet ... You need to configure NAT forwarding
49  // on your home router to do that. See http://en.wikipedia.org/wiki/Port_forwarding for more details.
50
51  static const uint8_t ip[4] = { 192, 168, 0, 13 };
52
53  // password required for operating door [max length = 16] (status fetching doesn't require password). This
54  // must match the password you set in the iPhone application (be careful as case is sensitive).
55
56  #define PASSWORD "xxx"
57
58  //************************************************************************
59  // Device/Door configuration (relais and sensors)
60  //************************************************************************
61
62  // *** IMPORTANT NOTE ***
63  //
64  // THE ARDUINO ETHERNET SHIELD RESERVES DIGITAL PINS 10, 11, 12, 13 AS WELL AS
65  // ANALOG PINS 0, 1, THEREFORE YOU SHOULD NOT USE ANY OF THOSE PINS FOR YOUR DEVICE(S) OR DOOR(S)
66
67  // open/close trigger relay should be connected to these digital output pins (digitalWrite).
68  // Adjust to match the number of devices you have hooked up (examples provided below in comment) ...
69
70  static const uint8_t relayPins[] = { 9 }; // single device at pin #9
71  //static uint8_t relayPins[] = { 2, 3, 4, 5 }; // select if using DFRobot RelayShield
72  //static uint8_t relayPins[] = { 2, 3 }; // two devices at pins #2 and #3
73  //static uint8_t relayPins[] = { 2, 3, 4, ... }; // even more devices at pins #2, #3, #4, etc ...
74
75  // status contact should be connected to these analog input pins (anologRead).
76  // Adjust to match the number of devices you have hooked up (examples provided below in comment) ...
77
78  static const uint8_t statusPins[] = { 3 }; // single device at pin #3
79  //static uint8_t statusPins[] = { 2, 3, 4, 5 }; // select if using DFRobot RelayShield
80  //static uint8_t statusPins[] = { 2, 3 }; // two devices at pins #2 and #3
81  //static uint8_t statusPins[] = { 2, 3, 4, ... }; // even more devices at pins #2, #3, #4, etc ...
82
83  // status reading strategy (uncomment only one ... the one that reflects how you want status to be interpreted)
84
85  #define STATUS_STRATEGY_3VCLOSED_5VOPENED // initial approach - uses analogRead combined with STATUS_OPEN_TRESHOLD (opened == +5v, closed == +3v)
86  //#define STATUS_STRATEGY_5VCLOSED_3VOPENED // alternate approach - uses analogRead combined with STATUS_OPEN_TRESHOLD (opened == +3v, closed == +5v)
87  //#define STATUS_STRATEGY_NORMALLY_CLOSED // classic door sensor - uses digitalRead to interpret door/device status (opened == high-impedance, closed == +5v)
88  //#define STATUS_STRATEGY_NORMALLY_OPENED // alternate approach - uses digitalRead to interpret door/device status (opened == +5v, closed == high-impedance)
```

5. ***Optionally***, you can turn *ON* notifications from MyDoorOpener. By default all notifications are turned *OFF*. So if you don't care for notifications, you can skip this section altogether and move on to the next section of this document.

   MyDoorOpener currently supports 2 types of notifications:

Watchdog Open Notification	As soon as a door/device gets opened, a timer is started and after a configurable amount of time, a notification gets fired, unless the door/device gets closed before the timer reaches its expiration.
Open Notification	As soon as a door/device gets opened, a notification gets fired.

   Each notification type can be turned *ON* or *OFF* individually. You can get notified for a single type or for both. In all cases, notifications get fired whether the door/device gets opened thru the iPhone application or by any other means. For now, notifications are global, therefore, if turned *ON*, all sensors are observed. You currently can't specify which sensors are observed and which aren't.

   To turn *ON* "Watchdog Open Notifications", uncomment the following line (near line ~100). You can also change the default value (5) to the number of minutes you want:

   ```
   #define NOTIFICATIONS_WATCHDOG_MINUTES  5
   ```

   To turn *ON* regular "Open Notifications", uncomment the following line (near line ~103):

   ```
   #define NOTIFICATIONS_OPEN
   ```

... Notifications (continued)

MyDoorOpener currently supports 3 notification broadcasting mechanisms:

Push Notifications	Will send a push notification to your iPhone. Requires the purchase of the Prowl iPhone application.
SMS Notifications	Will send an SMS message to your mobile phone, using your mobile carrier's SMTP to SMS gateway.
Email Notifications	Will send an email message to a configurable email address, typically using your ISP's SMTP server.

Each notification broadcasting mechanism can be turned *ON* or *OFF* individually. You can get notified using a single mechanism or all of them, for as long as you configure them appropriately.

To turn *ON* "Push Notifications", uncomment the following line (near line ~110) and provide the required Prowl API key for your iPhone device (near line ~116).

```
#define PUSH_NOTIFICATIONS

static const char prowlApiKey[] = "paste-your-prowl-provided-api-key-here";
```

To turn *ON* "SMS Notifications", uncomment the following line (near line ~134), provide the required smtp address for your mobile (near line ~139) and smtp server (near line ~153).

```
#define SMS_NOTIFICATIONS

static const char smtpToForSms[] = "your-mobile-number@your-carrier-gateway.com";

static const char smtpServerName[] = "smtp-server.your-isp.com";
```

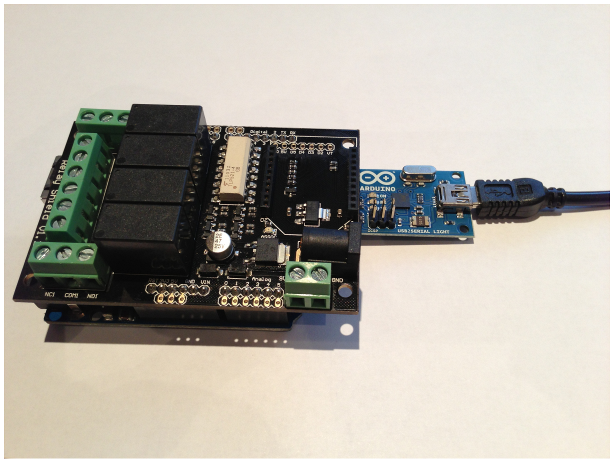
To turn *ON* "Email Notifications", uncomment the following line (near line ~148), provide the required smtp server (near line ~153) and email address (near line ~159).

```
#define SMTP_NOTIFICATIONS

static const char smtpServerName[] = "smtp-server.your-isp.com";

static const char smtpToForEmail[] = "your.address@gmail.com";
```

6. Compile the MyDoorOpener.ino project from within the Arduino IDE, using the **Sketch -> Verify/Compile** menu option. Make sure there are no compiler warnings or errors.

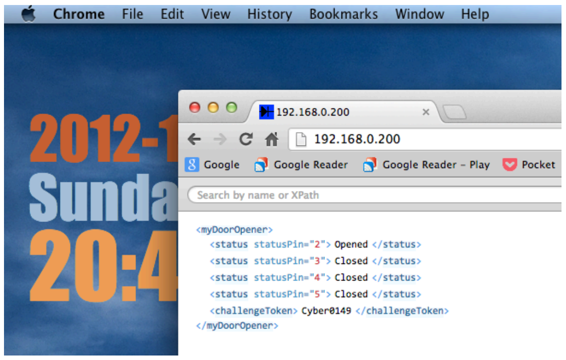


7. Connect your Arduino to your computer using a standard USB cable and the **USB2SERIAL module.** Upload the project, using the **File -> Upload** menu option from within the Arduino IDE.

8. Once the upload is completed, you can unplug the Arduino from your computer. You will also need to either “reset” the Arduino (using the reset button on the board) or unplug the AC power from the Arduino for a few seconds. This will have the MyDoorOpener program initiated.

This concludes the software configuration process.

Network Configuration



```
<myDoorOpener>
  <status statusPin="2"> Opened </status>
  <status statusPin="3"> Closed </status>
  <status statusPin="4"> Closed </status>
  <status statusPin="5"> Closed </status>
  <challengeToken> Cyber0149 </challengeToken>
</myDoorOpener>
```

1. You should first test your installation by trying to access your Arduino with a web browser from within the same internal network. To do so, fire up your favorite web browser and type in the IP address you configured your Arduino to listen to, from step #4 of the software configuration process.

   You should see a screen similar to the one on the left if things are working properly (your web browser must be able to display XML in order for this to work).

2. You should now configure your home router to do NAT port forwarding to your Arduino so that you can access it from the internet, using your home router's ISP assigned public IP address instead. This is required because the IP address assigned in step #4 of the software configuration process is an internal IP address which is not visible from the internet.

   The Arduino is listening on port 80 (default) so you will need to input a NAT forwarding rule that will forward to that port, off the IP address you configured in step #4 of the software configuration process.

   How NAT port forwarding works is specific from router to router, therefore it is beyond the scope of this document to cover these specifics. You can either look at your home router manual or Google around if you aren't sure how this can be achieved with your particular home router.

3. Another aspect you might want to look into is dynamic DNS. Since your home router most probably gets assigned an IP address by your ISP, it is subject to change at any time. In order to be shielded from such change, it is recommended to use a name rather than an IP address. The name will map to an IP address and with dynamic DNS services, that name to IP mapping will get maintained whenever your ISP allocates a new IP address to your home router.

   Again, the details on how dynamic DNS works and the intricacies for the many dynamic DNS service providers out there is beyond the scope of this document. You can Google around to get more details on how such service works, the many service providers that offer this service, as well as how it can be configured with your particular home router.

This concludes the network configuration process.

iPhone Configuration

1. Download the iPhone application from the Apple iTunes store : http://itunes.apple.com/app/mydooropener/id359774310

2. Once installed on your iPhone, open the application and click the small configuration gear located at the top right of the main screen.

3. Fill-in the configuration fields as follows :

Controller

Field Name	Description
URL	The URL used to connect to your Arduino from the internet. This is not the internal IP address for your Arduino, but a name or IP address that is visible from the internet. Typically, this will be a URL with the name or IP address of your home router on which you'll have configured a NAT forwarding rule that will forward to your Arduino. This URL must have the form http://xxx.yyy.zzz:port/ where xxx.yyy.zzz is either a DNS name or an IP address. The port number is optional.
Password	The password for your Arduino, as configured in the software configuration process. This must match as-is, and is case sensitive.

Doors & Devices

Field Name	Description
Relay Pin Number	Choose the relay pin number based on the relay block used: COM1/NO1 = Relay pin #2 COM2/NO2 = Relay pin #3 COM3/NO3 = Relay pin #4 COM4/NO4 = Relay pin #5
Status Pin Number	Choose the status pin number based on the designated analog pin number attached to your sensor for that door/ device.

We hope this document was helpful is setting up your **MyDoorOpener** project.

If you have questions or if you find inaccuracies in this document, please drop us a note at support@mydooropener.com .

Thanks for using MyDoorOpener and for your continued support!