

The DoorBell Project v1.2

A doorbell showcase demonstrating wireless configuration and upgrading of embedded Arduino devices by using Captive portal, Hotspot and Arduino OTA.

Copyright: the GNU General Public License version 3 (GPL-3.0) by Eric Kreuwels, 2017



Contact: Eric.kreuwels@gmail.com

Contents

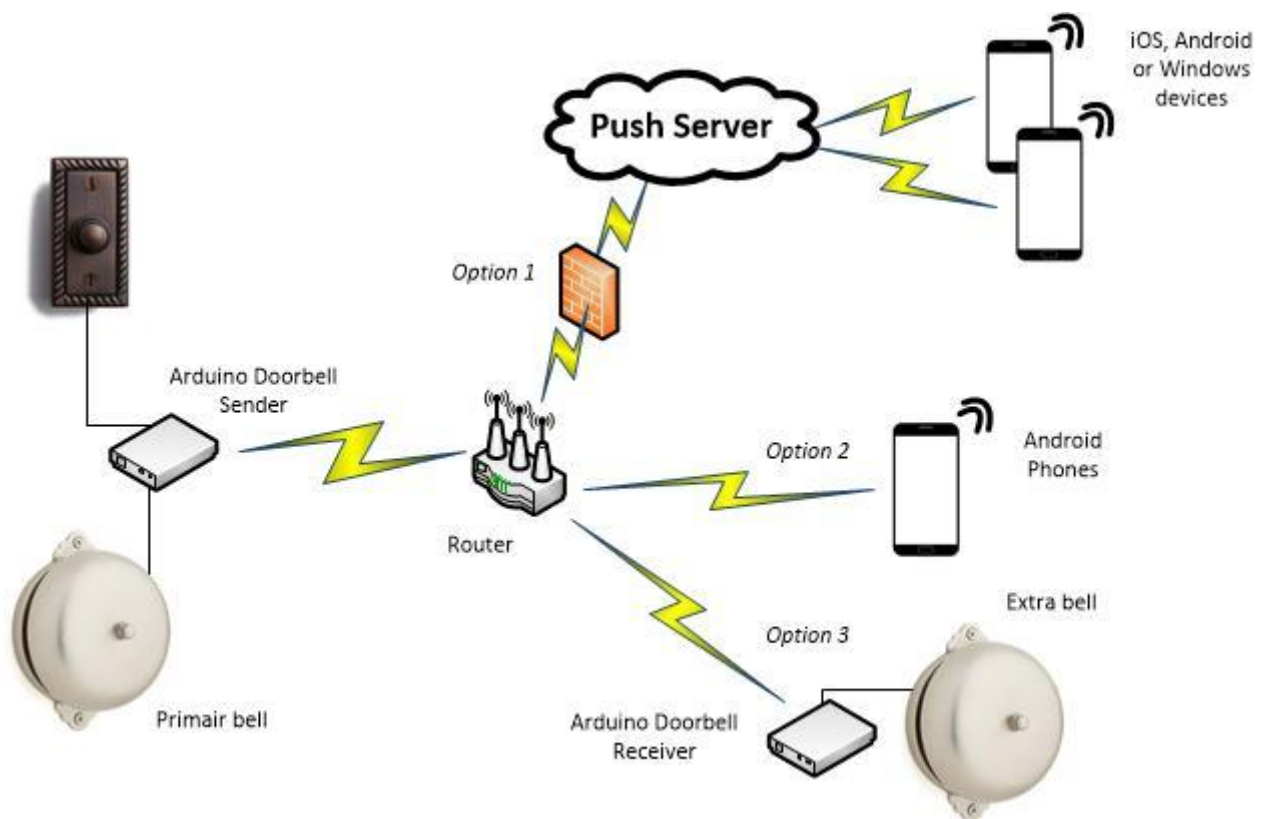
The DoorBell Project v1.2.....	1
Introduction:.....	2
The Arduino device:.....	3
The (re-)Config button:.....	3
The three modes (Configuration, Operational & OTA):	3
Prerequisites.....	5
Wiring schematics	5
Using a push Server:	8
How does it work?.....	8
Example using PushBullet and PushingBox:	8
The dedicated Android smartphone app:	10

Introduction:

The doorbell project consists of an Arduino with on board Wi-Fi (tested on a Wemos D1 R2). This device acts as a *sender* that monitors the doorbell button. When the doorbell button is pushed, the *sender* broadcasts this event via Wi-Fi. Multiple receivers can monitor these broadcasts. Receivers are either another Arduino board that is configured as *receiver* or mobile devices

Mobile devices can react on a Push Notification or for Android run a dedicated Doorbell app. The dedicated Android app only works when connected to the local Wi-Fi network, Push notifications need a push server (e.g. PushingBox) and can be received by a wide ranges of free apps for i.e. Android Windows or iOS based devices (e.g. PushBullet) . These notifications will arrive anywhere.

Picture Basic setup:



The Arduino device:

The Arduino board supports switching between three modes; a configuration mode (softAP), OTA mode for upgrading firmware and the operational mode (STA).

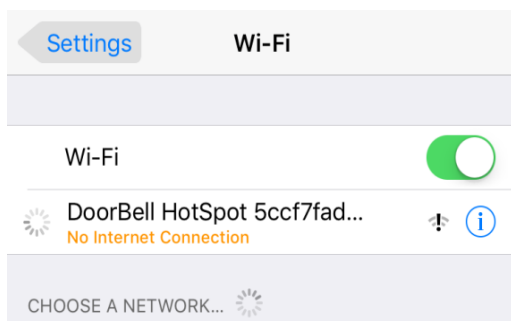
The (re-)Config button:

With this button board modes can be switched.

- Press between 1-3 seconds to toggle between configuration and operational mode (this is a dynamic switch, without restarting the board), or to leave OTA mode (will restart the board).
- Press the button for 10 seconds or longer to switch into OTA mode. After either a successful flash the board will automatically restart into the previous mode (either operational or configuration mode).

The three modes (Configuration, Operational & OTA):

- Initially the board starts up in the **configuration mode**. The signal LED will be constantly ON. Here the board acts as a captive hotspot with a webserver for entering configuration settings. When you connect a smart phone with this hotspot the board can be configured. When the configuration is saved (it persists all settings in EEPROM to survive reboots), the board switches to **operational mode** where it acts as a *Sender* or *Receiver* (as configured)
Step 1: Go to the WiFi settings of your smartphone to connect with the Hotspot. Ignore the message that there is no internet connection.
Step 2: If the configuration page doesn't open automatically (typical issue on iPhone) open the page manually via the browser and navigate to address 10.10.10.1:



Note: UDP port numbers cannot be reserved.
Try other values if you encounter issues,
i.e. 54250 instead of the default 49152.

10.10.10.1

Doorbell configuration page

Enter the credentials for connecting the WLAN to be used for broadcasting Door Bell events:

WLAN SSID

WLAN Password

Enter the listen port used by the client apps (default=49152, advised range 49152..65535):
UDP Listen port

Define the device type:
A **Sender** monitors a bell button to broadcast such event
Receivers monitor these broadcasts to trigger a relay
Device type

Optionally configure a Push Server:
The example values are for the free service PushingBox (use GET to enable). To enable a push server need to configure the HTTP command type into GET or POST.
Push mode

Server name

Push Command

- In the **operational mode**, The signal LED will shortly flash once per five seconds. In this mode the board either monitors the doorbell push button (configured as *sender*) or listen to doorbell rang broadcasts from a *sender* (configured as a *receiver*). In both modes a doorbell relay can be triggered as well. With a “re-config” button, the board can be switched back into **configuration mode**

Note: The next chapter gives an example how to use a Push Server

- **OTA mode:** The signal LED will flicker quickly. In this mode the Arduino board can be flashed with new firmware via the Arduino IDE with a connection over Wi-Fi.
 - First switch the board into OTA (press the reconfig button for at least 10 seconds to enter OTA).
 - Then (re-)start your Arduino IDE. Under the Ports, an extra entry, besides the COM entries, should occur with the Name Doorbell_OTA and an IP address. The firmware can be uploaded to this new network port.

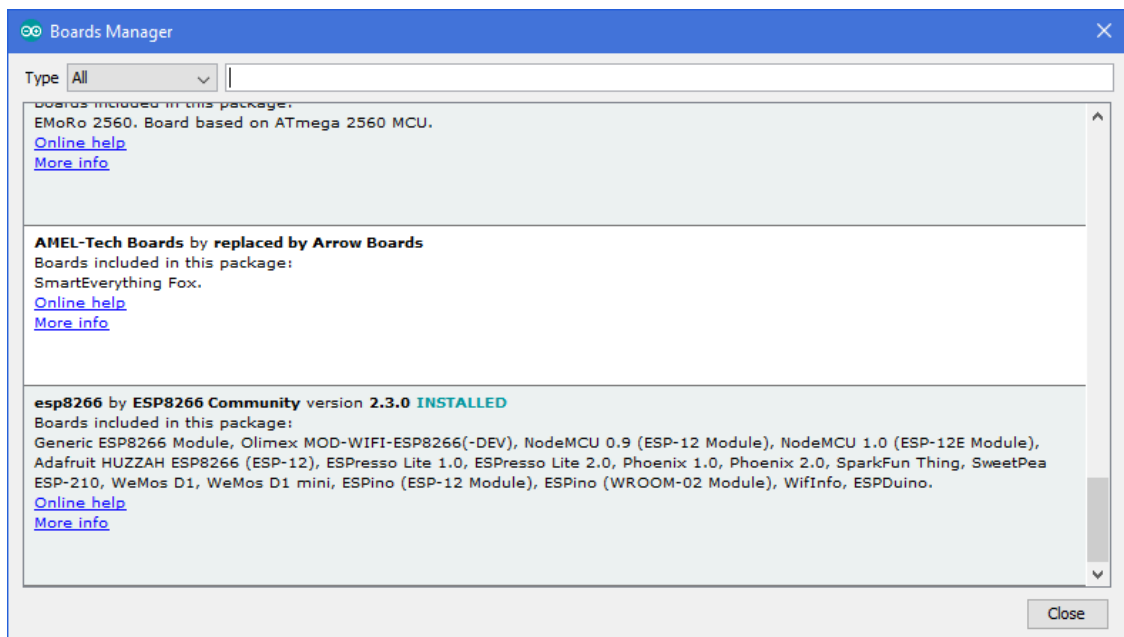
PS1: OTA from the Arduino IDE requires that Python 2.7.x is installed with the option to include Python into the Path was enabled (is by default off in the Python setup)

PS2: When the board is connected with the PC (via USB for programming the board) it can also be fully configured via the serial monitor as well. Type Help in the Send box of the monitor to get all the commands.

Prerequisites

Please note the Arduino code was designed and tested on a Wemos D1-R2 board. I used the Arduino 1.8.3. with the ESP8266 board library version 2.3.0 and Python 2.7.13.

- For using OTA upgrades, Python 2.7.x needs to be installed. Enable the option to including Python into PATH. This option is not enabled by default!
- To install the ESP library enter the following URL under **Preferences->additional Board Manager**
URLs: http://arduino.esp8266.com/stable/package_esp8266com_index.json.
- Then open the Board manager under **Tools->board 'xyz'->Board Manager**. In this menu scroll down and right click the ESP library to install.
- Once installed successfully it should look like:



Note: Changes needed when using similar Arduino boards:

- The Wemos D1-R2 pinning will be different
Note: even the older Wemos D1-R1 pinning is different
- There are some ESP specifics like using the EEPROM. To write to the EEPROM on the ESP8266 additional commands are needed compared to a standard Arduino board (begin() to enable reading/writing and commit() to save what is written)
- The ESP8266 has a strange watchdog timer. When the board reboots, the behavior depends on the input values of 3 digital IO's.

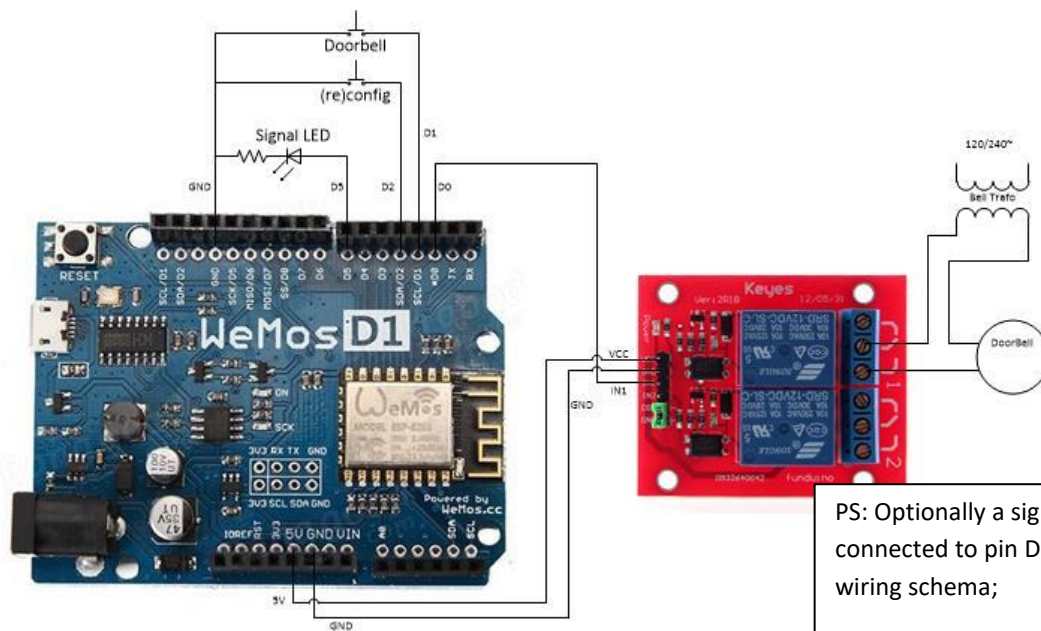
Wiring schematics

Both the wiring for *Sender* or the *Receiver* mode are very similar. The only difference is that a Receiver doesn't connect with a Door Bell push button (it needs to be trigger by broadcasts from a *Sender*).

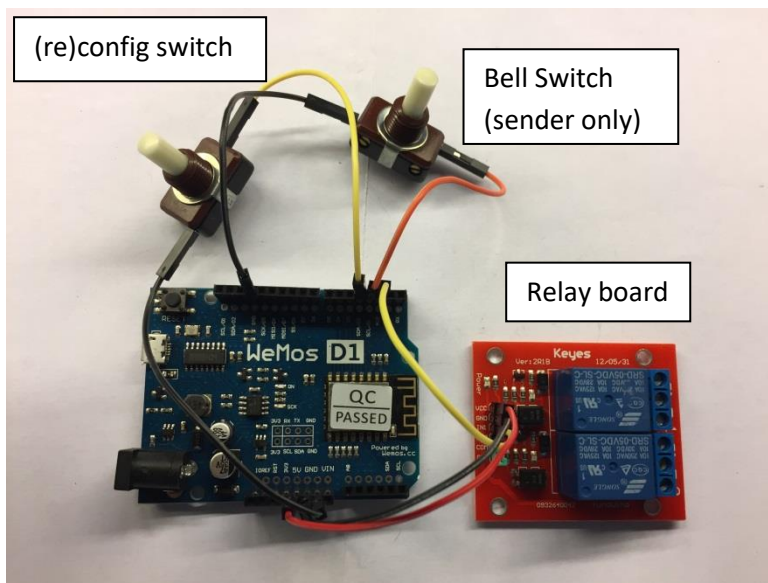
Note: for the *Sender* an alternative wiring is possible with some pro's and con's as explained later.

The basic Sender and Receiver

Schematics:



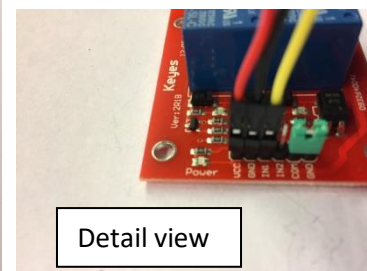
A real example picture:



PS: Optionally a signaling LED can be connected to pin D5 as shown in the wiring schema;

This Signal LED will be ON in configuration mode, or flash shortly every 5 seconds when in operational mode

(Don't forget to put this LED in series with a resistor of 330 Ohm or more).



Detail view

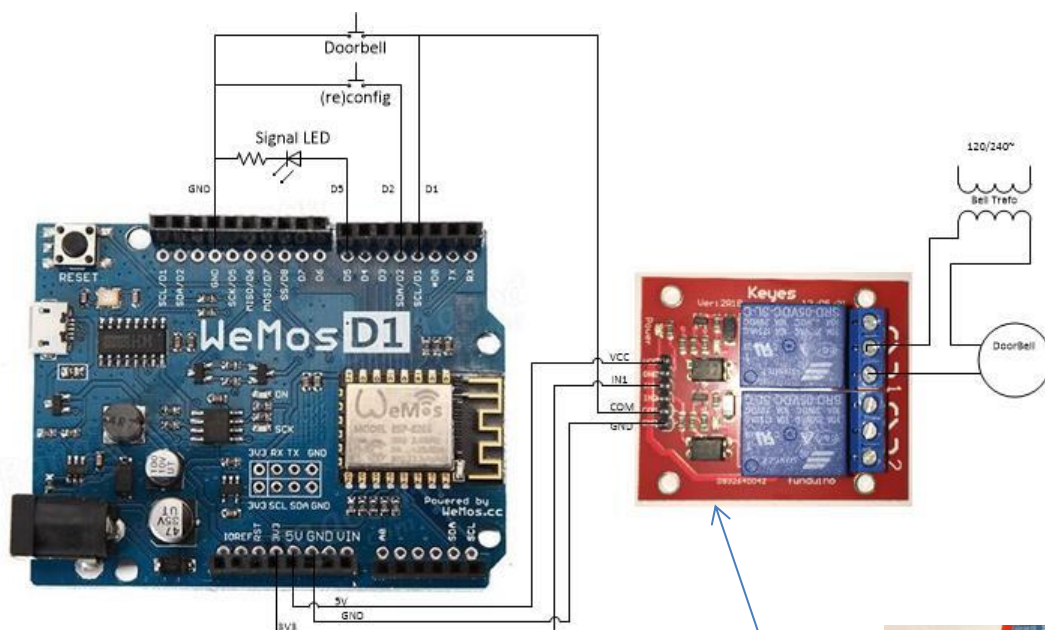
Alternative wiring for the Sender

The basic wiring is most simple and works with basically every relay suitable for Arduino. The only drawback is that the bell relay is activated for a fixed duration of ca 0.6 seconds, but that will not work if the board malfunctions.

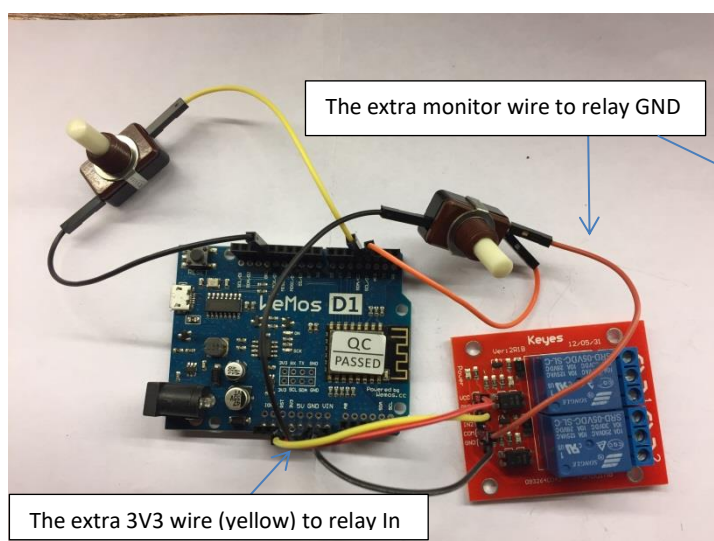
An alternative wiring is possible where the Arduino board monitors the relay, instead of controlling it. Although this wiring is a little more complicated and requires a more specific type of relay board, the bell will ring even if the board doesn't work.

Important: the relay board has to support splitting the ground on the input part and the output/relay part (as the keys board used in the pictures)

This enables to power the IN of the relay board with the 3.3V of the Arduino board and switch the ground of the input part of the relay board. On D1 of the Arduino board we monitor the voltage on the doorbell switch. The schematics (*Sender option 2*):

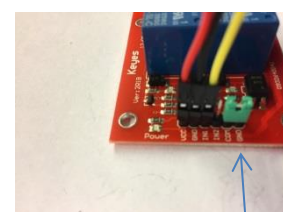


A real example picture:

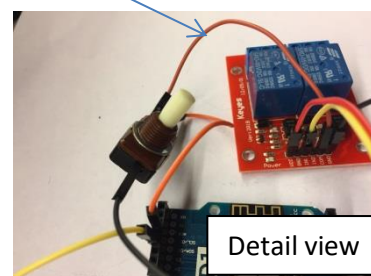


The extra monitor wire to relay GND

The extra 3V3 wire (yellow) to relay In



Remove the GND – COM jumper



Detail view

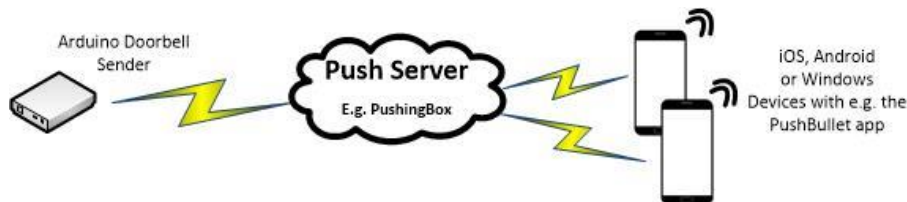
Using a push Server:

Credits: I based this on the Push server example demonstrated by Clement Storck:

<http://makezine.com/projects/notifying-doorbell-with-pushingbox/>


How does it work?

Push servers like pushingBox.com can be triggered via a HTTP request with an unique ID. This request will initiate a push Notification to any mobile device you like. These push notification can be received with various Apps, like PushBullet:



Example using PushBullet and PushingBox:

Note: The simplest way to use these services is to logon with e.g. your Google account.

Step 1: install  PushBullet on one or more mobile devices (select app in app/play store)

Step 2: Go to an account on <https://www.pushbullet.com/>. Create an access token under Settings. Copy this token. It will be something like `o.QSrOntg.....`


Step 3: Open an account on <https://www.pushingbox.com> and create a service for PushBullet under my Services. To create this service you have to enter the token created in step 2:



pushingbox

Dashboard My Services My Scenarios Settings

Services

 **Pushbullet**
Push notification service for iOS, Android, Windows, Firefox and Chrome

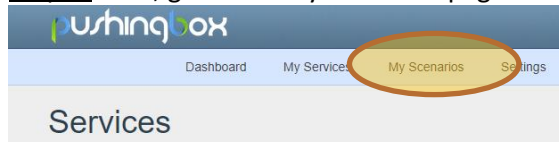
Select this service

Name of your Pushbullet configuration:
Pushbullet Clement

Access token:
your token

Device token (optional):
ignore this one

Step 4: Now, go to the My Scenarios page.



pushingbox

Dashboard My Services My Scenarios Settings

Services

Create a scenario called something like "push doorbell notification". Click on Create scenario. Click on Add an Action. Add an action using the service created in step 3.

The new scenario will appear in the list below with an unique Device ID.

Step 5: Test your new scenario:

Scenario name	DeviceID			
Demo scenario	v6CC...	Test	Manage	Delete
Push doorbell notification	vFD7...	Test	Manage	Delete

When you press test, a notification has to appear on the devices where you installed PushBullet in step 1. If not, review what is wrong.

Step 6: Copy the Device ID:

Scenario name	DeviceID			
Demo scenario	v6CC...	Test	Manage	Delete
Push doorbell notification	vFD7...	Test	Manage	Delete

Step 7: Open the configuration page of your Arduino (press the reconfigure button and open the hotspot. Manually go to 10.10.10.1 if the page doesn't open automatically)

The Doorbell Arduino device supports generic configuration of HTTP based Push servers:

1. Just enter the right server address,
2. Enable the HTTP push notifications request by selecting either GET or POST.
3. Enter the right content for your unique HTTP request

The defaults are already prepared for usage with PushingBox.com:

1. Enable PushingBox requests by selecting GET (instead of Disabled)
2. Replace <yourDeviceID> with your device ID copied in step 4:

Optionally configure a Push Server:
The example values are for the free service PushingBox (use GET to enable). To enable a push server need to configure the HTTP command type into GET or POST.

Push mode: Disabled ▼

Server name: api.pushingbox.com

Push Command: /pushingbox?devd=<yourDeviceID>

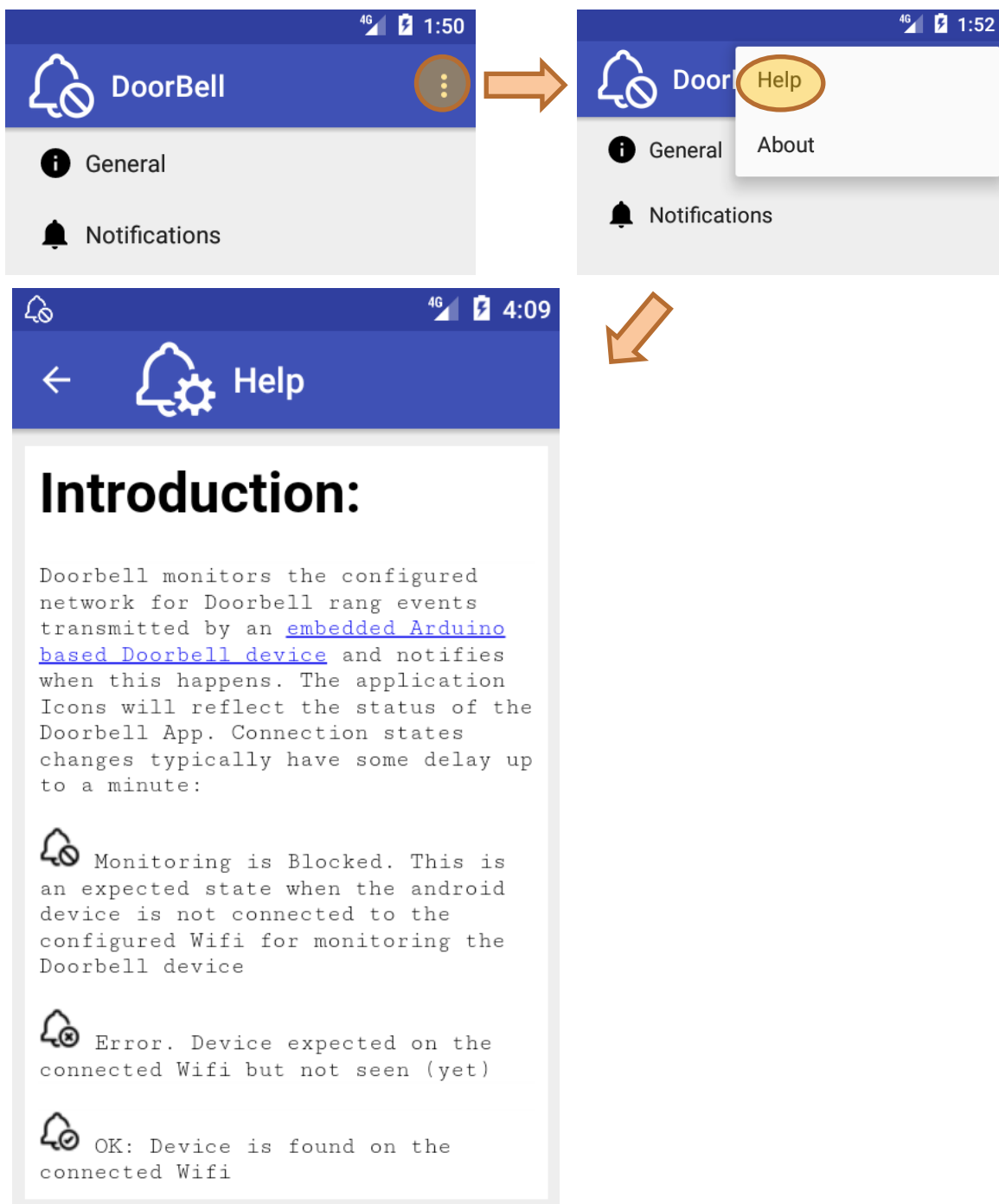
The dedicated Android smartphone app:

The smartphone app supports Android 4.2 (JELLY_BEAN / API 17) and up. The app is not officially released (costs money). So please download it from my personal website:

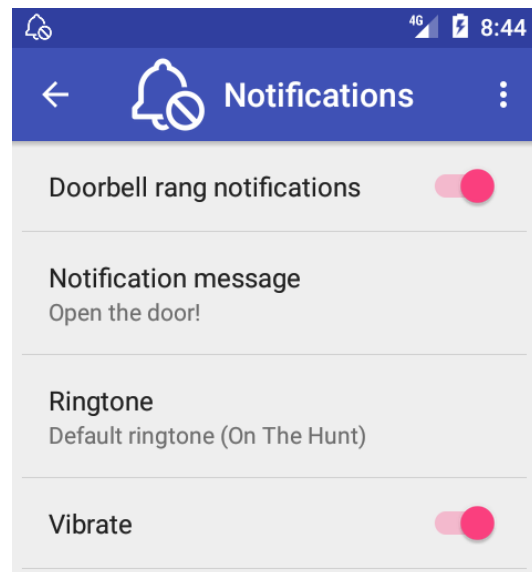
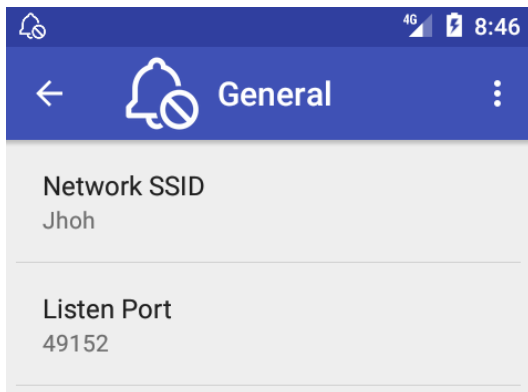
<http://www.kreuwels.com/DoorBell101beta-release.apk>. Testing revealed it runs solid on both Android 5 and 7 releases, but tends to stop now and then on Android 6.

The app is intended to monitor one Arduino based Doorbell sender device. When a doorbell rang event is received a configurable notification (ringtone and/or vibration) will be launched.

Additionally a sticky notification represents the connection status. This notification needs to be sticky to keep the background monitoring service alive. To preserve the phone battery, the background monitoring service will go into idle mode when there is no connection with the Wi-Fi configured network. The DoorBell app contains a help page with further instructions:



The screenshots below are examples of the **general** and **notifications configuration** screens. The general page contains the settings to connect with the *Sender* (Arduino device), the Notification page to tweak the desired notifications settings:



When the *Doorbell rang* notifications are enabled, these notifications can be removed by swiping them to the left. The *status notifications* of the Doorbell app are *sticky*. You can't remove them but when tapped it will open the Doorbell app. Example of these notifications:

