For just $2, convert any existing wired doorbell into a smart doorbell; using ESPHome and Home Assistant

By Franck Nijhof / In ESPHome, Home Assistant, Tutorials / August 4, 2019 / 14 Min read / 138 comments

This article is going to show you how to convert your (existing) wired doorbell (also works for battery-powered chimes) into a smart, WiFi-enabled, doorbell. The essential components needed sets you back about $2, yes, that is correct, just two bucks. No soldering is

The result of this DIY Smart Doorbell project on my test bench, ready to be installed

Integrating your doorbell into your smart home is a very logical step to take. Making your doorbell smart, allows you to do cool things with it, for example:

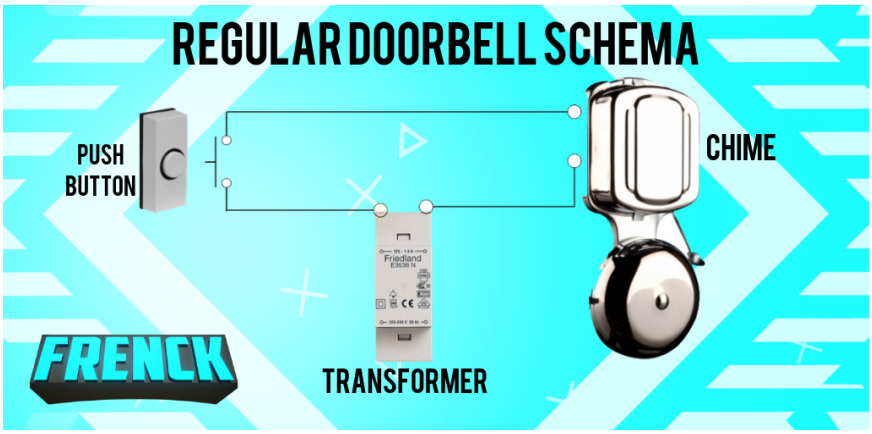
- Turn the chime/bell off after a specific time, when the kids or you went to bed. Also, turn it on again in the morning.
- Send out push notifications to your phone/tv /watch/smart speaker, on the doorbell button push.
- Take a snapshot from a front door camera, on the doorbell button push.
- Stream your front door camera to your TV, on the doorbell button push.
- Ring the doorbell continuously in case of an emergency (e.g., smoke detectors triggered).

I have provided some of these automations for Home Assistant, as an example, at the end of this article.

The method I provide is, of course, not the only

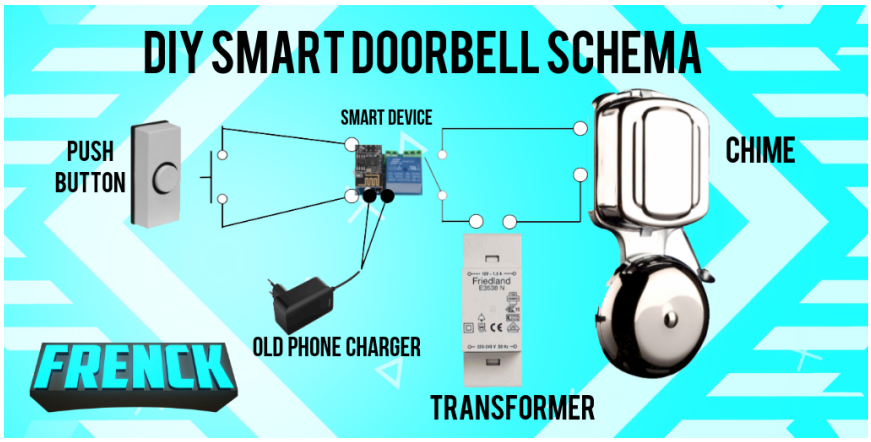
Table of contents

How does a doorbell work; Before and after.



Regular doorbell circuit schema

A pretty standard doorbell set up has a power supply, hooked up directly to the chime. Prevent the doorbell chime going continuously, the circuit is interrupted by the doorbell button, which acts as a switch. The circuit is only active/complete when the doorbell button is pushed. Hence, your chime turns on when someone pushes the doorbell button.



DIY Smart doorbell circuit schema

Home Assistant, without activating the chime.

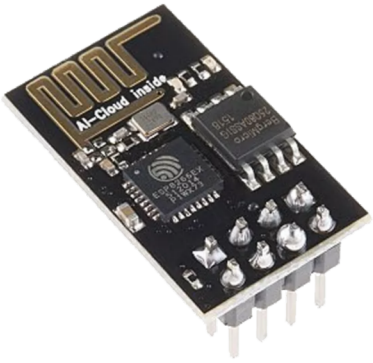
The device we are using is going to act as a sensor for the doorbell button push, and acts as a switch for the chime. Effectively, we are moving the activation of the chime from the doorbell button, to the new device. Our new device needs power, and since the one used consumes 5 volts. Any old USB phone charger you have, does the job just fine.

Smart doorbell stuff you'll need

To achieve this, you need some "stuff". The $2 price tag is based on the actual device we are going to create, however, if you don't have some of the needed tools, that would raise the total price. Nevertheless, the additional stuff is really cheap and are things that, in my humble opinion, every DIY home automator should have in their toolbox (for future projects).

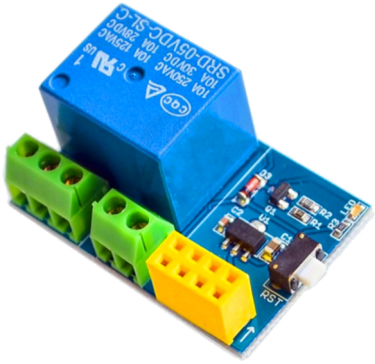
Based on the country you are in, additional shipping costs may apply. I live in the Netherlands, and I've paid 1,80EUR in total, including shipping. Checking out the different sites for the right price helps. I've noticed the prices for these little boards vary, but getting it for

chip called the ESP-01S and a small board with a relay on it.



ESP-01S

The ESP-01S is one of the smallest WiFi boards available, that works ESPHome and Home Assistant.



Relay module

This relay module board for the ESP-01(S), It comes pre-soldered, and the ESP will just slide on top of it.

Buy them in bundle

The nice thing about these two is that they fit together, are pre-soldered and often sold together. There is also

The board needs to be powered and requires 5 Volts. Lucky, most phone chargers provide those and I'm sure most of you have an old one around you could use.

That brings me to the shopping list of the main components needed:

An ESP-01S

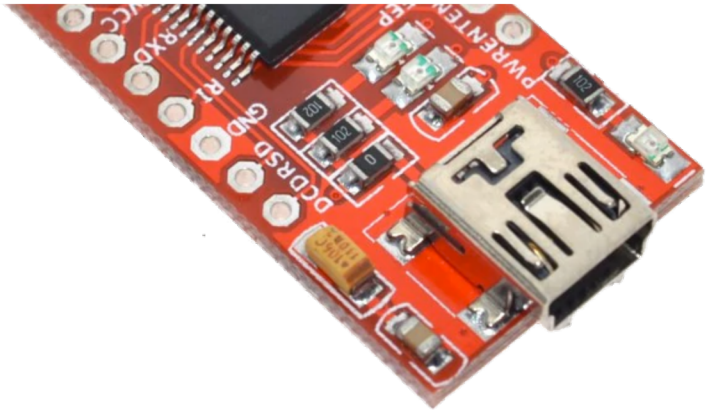
The relay module board for the ESP-01(S)

An old phone charger, delivering 5 Volts

Other tools and hardware

Besides the main components, you'll need some other things for building it as well. Without a doubt, you need some screwdrivers and a wire stripper. A pair of tweezers can also help put wires to place, and for modifying to the board later on.

The ESP01-S chip needs to be programmed. To do that, you'll need an FTDI adapter. If you have ever flashed ESP chips before (e.g., SonOff switches), you probably already have one. If not, you should get one! It is an essential tool to have when you are into home automation.



FTDI Adapter

An FT232RL FTDI USB To serial converter adapter
is a must a have
for any smart home enthusiastic to have in its
toolbox.

Click here to get one

Additionally, some wire is needed. I always have a bunch of Dupont cables in stock. They are really helpful in many many projects I've done. It allows you to wire up prototypes really quickly as well.

Dupont wires with different male/female connector combinations, is useful to have around. It allows for quick DIY prototypes, but useful in final projects as well.

your own Dupont wires using a wire crimper. It is way more flexible compared to pre-made wires.

Depending on where you are going to place the smart doorbell, you might be need a case as well. The resulting board is pretty small. Maybe it can already fit one of your electrical boxes. Another option is getting a small project box online, like this one:

this does the trick.

Needed software

Obviously, you need some software to pull this off. All software used is open source and free for you to use.

- Home Assistant
- ESPHome
- ESPHome Flasher

Home Assistant is my home automation hub of choice, and for this guide, I assume you are using it as well. However, this project can be used on other hubs as well (like Domoticz or OpenHAB). I've added a section at the end of the article for more information about this.

Modifying to the ESP-01S chip

To make this setup work, you would have to make a small modification to the ESP-01S chip. Reason for this modification is that you have to free up an additional pin for connecting your doorbell button. These input/output pins, are called GPIO pins.

Don't worry, this modification is easy. We need to bend the GPIO2 pin from the ESP-01S. This allows you to access the pin for connecting the doorbell button, while disconnecting it from the relay board.

The lost reset functionality is later be restored via a

the chip

This project uses ESPHome to create firmware for the ESP-01S chip. I'm not going to make a full tutorial on how to set up ESPHome since that is really well covered on their website. So, before continuing, make sure you've set up ESPHome.

- Getting Started with ESPHome (for Hass.io users)
- Getting Started with ESPHome (when not on Hass.io)

> For the rest of this guide, I'm going to assume you are running Hass.io. However, the ESPHome getting started guide for non-Hass.io users is extremely detailed, and you should be able to adapt easily.

Creating, building and downloading the firmware

Create a file called `doorbell.yaml`; for Hass.io users, create the file in the `/config/esphome` folder, so you end up with the file: `/config/esphome/doorbell.yaml`.

```
# WiFi connection, correct these
# with values for your WiFi.
wifi:
  ssid: !secret wifi_ssid
  password: !secret wifi_password

# Enable logging.
logger:

# Enable Home Assistant API.
api:

# Enable over-the-air updates.
ota:

# Enable Web server.
web_server:
  port: 80

# Sync time with Home Assistant.
time:
  - platform: homeassistant
    id: homeassistant_time

# Text sensors with general information.
text_sensor:
  # Expose ESPHome version as sensor.
  - platform: version
    name: Doorbell ESPHome Version
  # Expose WiFi information as sensors.
  - platform: wifi_info
    ip_address:
      name: Doorbell IP
    ssid:
      name: Doorbell SSID
    bssid:
      name: Doorbell BSSID
```

```
    name: Doorbell WiFi Signal
    update_interval: 60s

# Global to store the on/off state of the chime
globals:
  - id: chime
    type: bool
    restore_value: true
    initial_value: 'true'

# Exposed switches.
switch:
  # Switch to restart the doorbell.
  - platform: restart
    name: Doorbell Restart

  # Switch to turn on/off the chime.
  - platform: gpio
    id: relay
    inverted: true
    name: Doorbell Chime
    pin: GPIO0

  # Switch to turn on/off chime when
  # doorbell button is pushed.
  #
  # It creates a "virtual" switch based
  # on a global variable.
  - platform: template
    name: Doorbell Chime Active
    id: chime_active
    restore_state: false
    turn_on_action:
      - globals.set:
          id: chime
          value: 'true'
    turn_off_action:
```

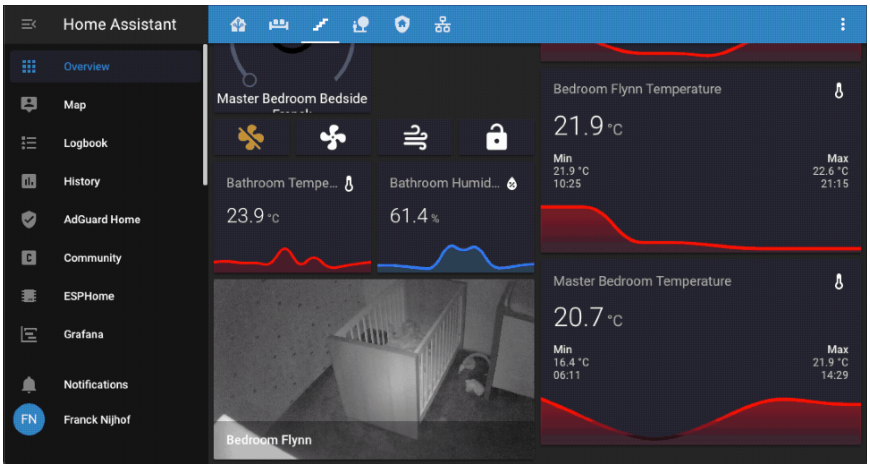
```yaml
  - platform: gpio
    id: button
    name: Doorbell Button
    pin:
      # Connected to GPIO on the ESP-01S.
      number: GPIO2
      mode: INPUT_PULLUP
      inverted: true
    filters:
      # Small filter, to debounce the button press.
      - delayed_on: 25ms
      - delayed_off: 25ms
    on_press:
      # Only turn on the chime when it is active.
      then:
        if:
          condition:
            - switch.is_on: chime_active
          then:
            - switch.turn_on: relay
    on_release:
      # On release, turn of the chime.
      - switch.turn_off: relay
```

doorbell.yaml hosted with ❤ by **GitHub** **view raw**

The above file shows a ESPHome project definition; the ESPHome project code, or also referred to as ESPHome YAML. The code describes the firmware allowing ESPHome to generate it. I've have tried to add as much additional text as possible to the above, to help you understand how it works.

On line 9 & 10, make sure to set your WiFi details or else

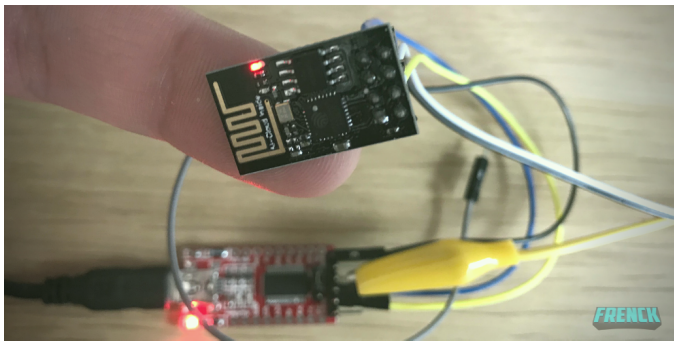
recording on how that works:



Doorbell ESPHome: Validate, compile and download

Uploading the firmware to the ESP-01S chip

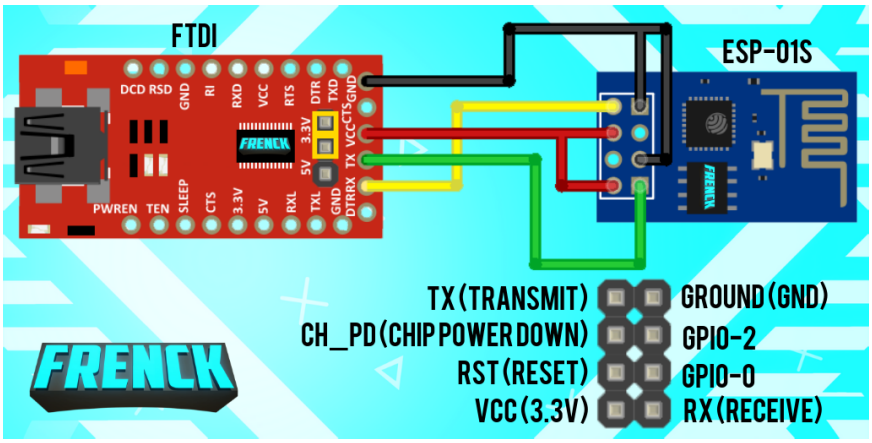
Now you have the



Flashing my ESP-01S for the smart doorbell

firmware (the downloaded `doorbell.bin`), you can start putting it onto the ESP-01S chip for use. This flashing procedure is the hardest part, and if you are new to this, please have some patience, it might need a

(OTA) via WiFi. Wire the FTDI & ESP-01S according to the following wire schematic:



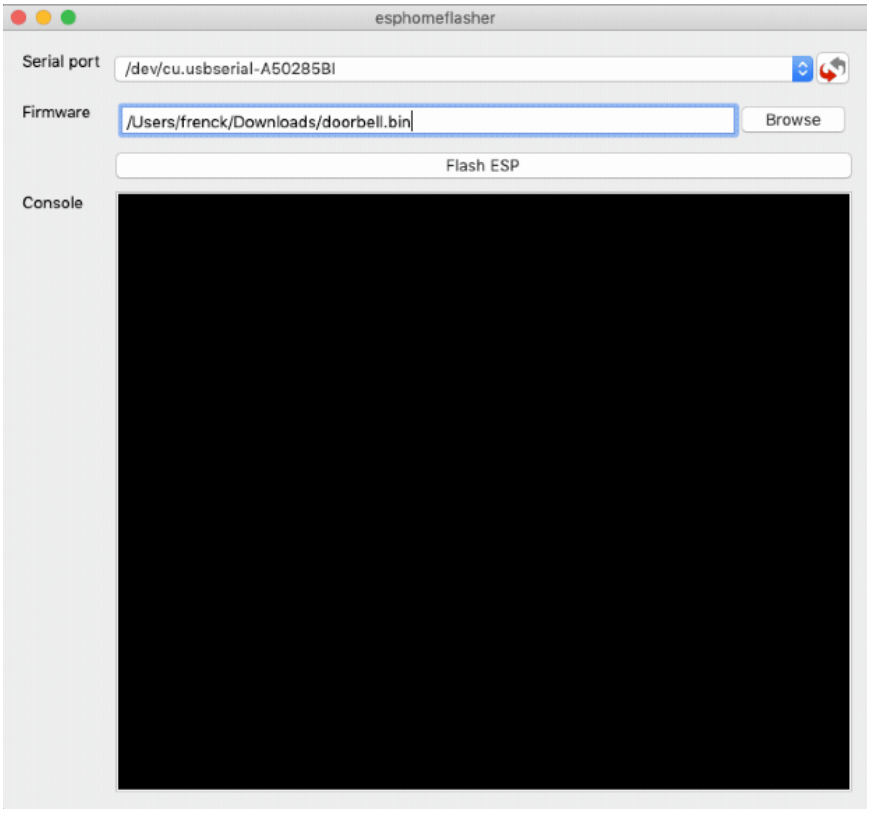
FTDI to ESP-01S Wiring Schema

> Make sure to double-check the wiring and set the FTDI to 3.3V. Even if you are an experienced flasher of ESP chips, please note, the VCC/3.3V and ground wires being double wired! Although, I've used a very common FTDI board in this schema, your FTDI might look different, ensure you wiring up correctly by verifying the names on your board.

Ready to flash? Good!

Plugin the FTDI adapter to your PC, and start-up the ESPHome Flasher tool. Yes, I know, it is possible to flash directly from ESPHome itself, however, if you are like

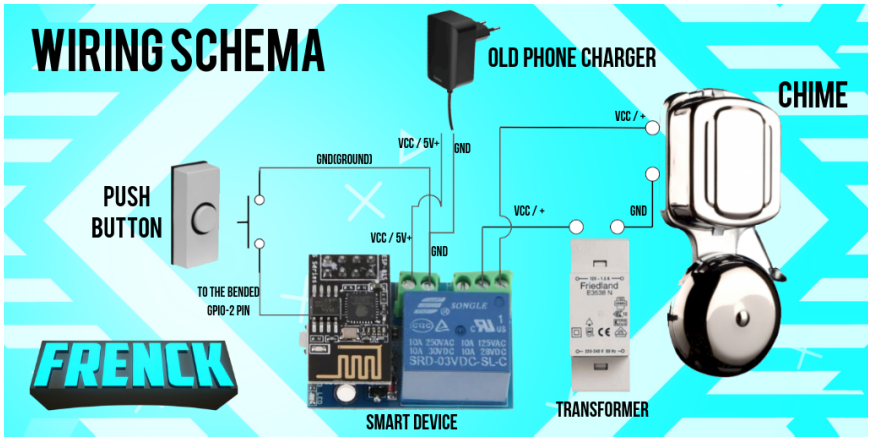
Select the available serial port, and load the downloaded `doorbell.bin` file, by clicking the “Browse” button. Start the flash procedure by clicking the “Flash ESP” button. The console shows you the progress and tells you when it finishes.



Flashing firmware using the ESPHome Flasher

If this process fails, please make sure to check your wiring. A common mistake is mixing up the TX/RX wires between the FTDI & the ESP-01S. They should be cross-connected (TX -> RX, RX -> TX).

correctly.



DIY Doorbell Wiring Guide

Please note, how the ground wire from the phone charger is shared with the push button. To hook up the push button to the bent GPIO-2 pin, I recommend using a Dupont wire, since you can slide it right onto the pin in that pretty tight space.

Wiring the GPIO-2 pin with a Dupont wire

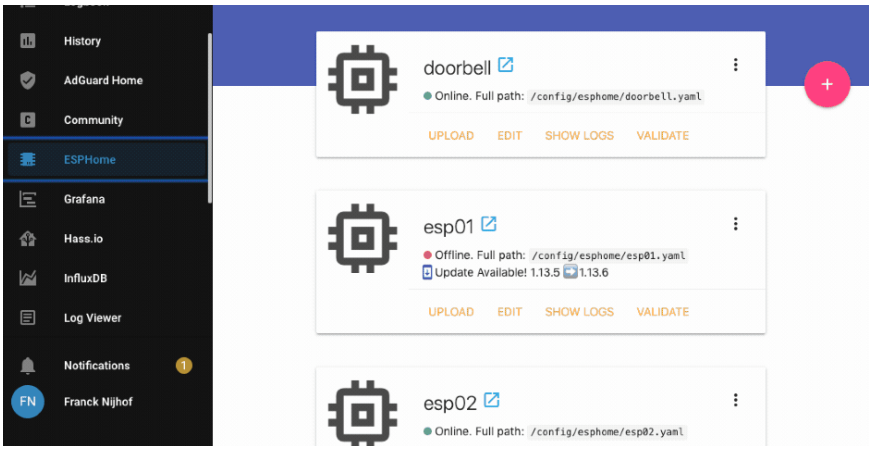
If your chime is a battery-powered one, the schema does not differ that much. Just act like the transformer is not in the above image. The two connections from the battery-powered chime, connect directly connect to the board (to the `COM` and `NO` labeled ports).

Done? Awesome! Pushing the doorbell button should now ring the chime already! So basically, you've now ended up with what you had already...

Now you can continue to configure your smart doorbell in Home Assistant, let the fun begin!

Integrating with Home Assistant

Welcome to the easier part of this guide. Integrating your new, self-created, smart doorbell with Home Assistant. Home Assistant will discovery it automatically. The only thing you need to do is to



Integrating the ESPHome DIY smart doorbell with Home Assistant

Home Assistant smart doorbell automations

At this point, you have a WiFi-enabled doorbell. Is it smart already? Not really...

Adding some automations to Home Assistant would make it really smart! There are many automations one could create with this. However, let me give you a couple of basic and useful examples.

Sending notifications to your phone

This little automation sends a notification to our phone

```yaml
#
automation:
  - alias: "Doorbell Notifications"
    trigger:
      platform: state
      entity_id: binary_sensor.doorbell_button
      to: 'on'
    action:
      - service: notify.mobile_app_frencks_iphone
        data:
          title: Doorbell
          message: Ding dong! Someone is at the door!
          data:
            attachment:
              content-type: jpeg
              url: http://tiptag.com.ar/wp-content/up
      - service: notify.mobile_app_ninja_iphone
        data:
          title: Doorbell
          message: Ding dong! Someone is at the door!
          data:
            attachment:
              content-type: jpeg
              url: http://tiptag.com.ar/wp-content/up
```

home-assistant-doorbell-notification.yaml hosted with ❤ by **GitHub** **view raw**

Disable the chime during the late hours

I have two kids. Nothing is more annoying than the sound of the doorbell chime about 15 minutes after you've put them into bed 😞. This little automation will

```yaml
automation:
  - alias: "Doorbell off after 8pm"
    trigger:
      platform: time
      at: "20:00:00"
    action:
      service: switch.turn_off
      entity_id: switch.doorbell_chime_active

  - alias: "Doorbell on after 7am"
    trigger:
      platform: time
      at: "07:00:00"
    action:
      service: switch.turn_on
      entity_id: switch.doorbell_chime_active
```

home-assistant-doorbell-chime.yaml hosted with ❤ by **GitHub** **view raw**

Streaming the front door camera when someone is at the door

This little automation is useful if you have a camera pointing at your front door. If you push the doorbell button, it will send out the camera stream of your front door camera to your living room TV 📺.

```yaml
---
# This automation triggers when the doorbell button i
# and out the front door camera stream to the tv.
#
```

```
14          entity_id: camera.front_door
15          media_player: media_player.living_room_tv
```

home-assistant-doorbell-camera.yaml hosted with ❤ by GitHub view raw

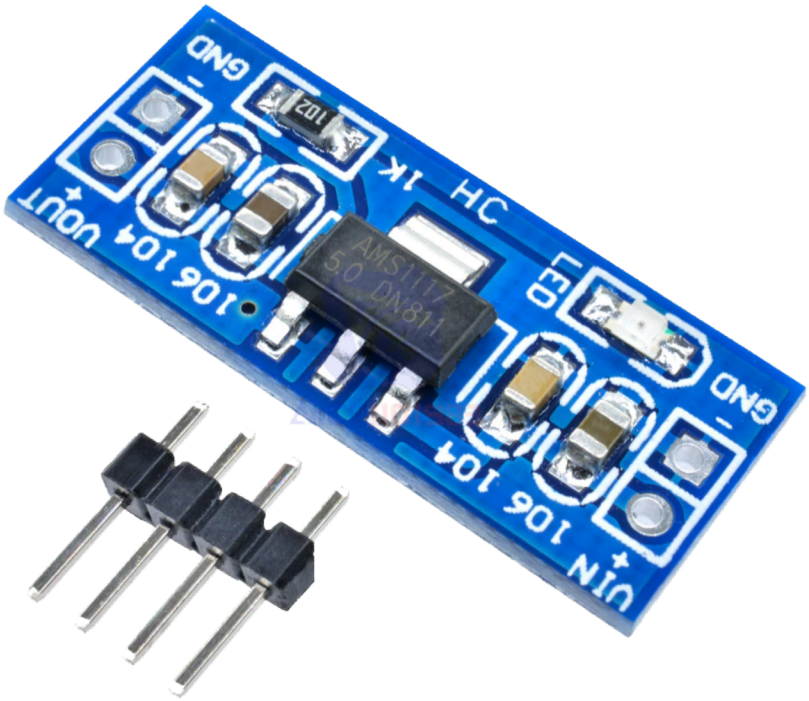
More smart doorbell automation ideas

- Disable the chime when you arm the alarm when home, indicating you went to bed.
- Use pressure sensors in your bed to turn off the chime when sleeping.
- Create a camera snapshot when someone is at the door, and send it out to your phone.
- Use the chime as an alarm bell as well, e.g., in case smoke or water leak is detected.
- Use facial recognition on your front door camera, to disable the doorbell for the ones you don't want to open the door for. E.g., an unknown person or your mother in law 😈
- Prevent over abuse of doorbell pushes, by disabling it for a couple of minutes after it was pushed.
- Hook the doorbell chime active switch into your voice assistant, allowing you to say "Hey Google, turn off the doorbell"

However, some are worth mentioning.

Using the chimes' power supply using a step-down

In my case, my doorbell chime is powered by 8 Volts (DC) from the transformer. While to voltage is too high for the board, I could have used a little regulator to "step-down" from the 8 Volts to the 5 Volts required by the board.



DC Voltage Regulator Step

These “step-down” are available for around a dollar on AliExpress. However, since I have enough older chargers I could use, and do have a power outlet available in my utility closet (where the doorbell wires are at); I saved myself the extra buck 😉.

Using multiple doorbell buttons (e.g., front- & backdoor)

In some homes, there are multiple doorbells. E.g., one for the front door, the other for the back door. Mostly they are connected to the same chime. There are three ways to use this guide in those cases:

1. Connect the doorbell push buttons in parallel to a single chip. This is fast and cheap, however, you cannot distinguish which button was pushed.
2. Connect two of those boards in parallel to the chime. You would need two boards and one chime, however, you can still distinguish the doorbell pushed.
3. Replace the ESP-01S and relay board with one of the bigger brothers. Those provide more GPIO

Are you not using Home Assistant? Really? Or maybe you prefer to use MQTT with OpenHAB, Node-RED or Domoticz. Well, you can use this project as well! ESPHome provides an MQTT interface.

For more information about using MQTT with ESPHome, refer to the ESPHome MQTT Client Component documentation.

This article is not about a smart doorbell...

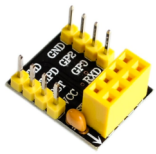
I hope you enjoy your DIY smart doorbell! However, now the time has come for me to come clean. This article was never about the doorbell! This article is about showing you how easy, fun, and cheap DIY smart home solutions are. ESPHome & Home Assistant are just amazing tools to allow everyone to jump in and create amazing things.

The ESP-01S is an amazingly cheap and tiny ESP8266-based board, that is rarely considered or discussed nowadays. Mostly you'll find articles and references to her big brothers; the ESP8266 and ESP32. However, if you need just one or two GPIO pins, it is a viable option.

the ESP-01S as well?

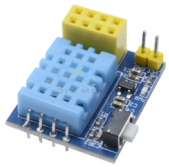
RGB LED module

This RGB led module goes for about a dollar and can control addressable LED strips like the WS2813.



Breakout module

This little board breaks out the pins from the ESP. Helpful if you want to be able to remove the ESP from your project.



DHT11 Module

The DHT11 module allows you to build a simple temperature and humidity sensor.

Did you build the doorbell? Nice! I love to see how you did!
Send me a picture of the result via Twitter!

../Frenck

Calibrating an ESPHome flashed power plug

June 12, 2019

In "ESPHome"

Introducing WireGuard VPN for Hass.io

September 1, 2019

In "Home Assistant"

Home Assistant Paulus invaded my live stream

May 15, 2019

In "Blog"

ABOUT THE AUTHOR

Franck Nijhof

Home Assistant enthusiast and contributor, Hass.io add-ons creator, IoT explorer, slightly assholic at first sight but actually a nice guy.

VIEW ALL POSTS