# Requirements

Raspberry Pi Zero W

## Raspberry Pi Setup

* Rasbian Linux RPi 4.9.35+
* Python v3.5+
* Pip3
* Flask
* ntpd

Python modules

* RPi.GPIO
* Flask
* APscheduler

# Installation

1. Copy the files to /home/pi/PragoPi
2. App.py and the templates directory with main.html should be copied
3. Create the system service to start the dashboard.
4. To test the aplication you can start it by

sudo python3 app.py

## PragoPi configuration

Several items can be configured in the application by editing the **app.py** file.

You can configure the

* application port (default 80)
* if \_\_name\_\_ == "\_\_main\_\_":
* scheduler = BackgroundScheduler()
* scheduler.add\_job(doImpulse, *trigger*='cron', *minute*='\*', *id*='minuter')
* scheduler.start()
* app.run(*host*='0.0.0.0', *port*=80, *debug*=False)

## Creating Raspberry Pi system services

The service for *PragoPi* needs to be created.

*PragoPi* is the main service providing the REST interface to query and control the time.

The service is started on Linux boot.

### PragoPi Service

1. Create file /usr/lib/systemd/system/pragopi.service

With content

[Unit]

Decription=Pragotron Watch Service

After=multi-user.target

[Service]

Type=simple

ExecStart=/usr/bin/python3 /home/pi/PragoPi/app.py > /home/pi/PragoPi/pragopi.l$

Restart=on-abort

[Install]

WantedBy=multi-user.target

2. Set the file properties with commands

sudo chmod 644 /lib/systemd/system/pragopi.service

chmod +x /home/pi/PragoPi/app.py

sudo systemctl daemon-reload

sudo systemctl enable pragopi.service

sudo systemctl start pragopi.service

### How to check the service status

sudo systemctl status pragopi.service

### How to check the service logs

sudo journalctl -f -u pragopi.service

Background information on how to create a service is from

<http://www.diegoacuna.me/how-to-run-a-script-as-a-service-in-raspberry-pi-raspbian-jessie/>

# APPENDIX A – HOW TO RUN A SCRIPT AS A SERVICE IN RASPBERRY PI - RASPBIAN JESSIE

[2 YEARS AGO](http://www.diegoacuna.me/how-to-run-a-script-as-a-service-in-raspberry-pi-raspbian-jessie/)**•**[IOT](http://www.diegoacuna.me/category/iot/)**,**[LINUX](http://www.diegoacuna.me/category/linux/)**•**[**8**](http://www.diegoacuna.me/how-to-run-a-script-as-a-service-in-raspberry-pi-raspbian-jessie/)

The [Raspberry Pi](https://www.raspberrypi.org/) is an incredible popular credit card size mini computer with awesome capabilities.  Despite of is mini size it behaves pretty much like a regular computer (it has an OS with a graphical interface, an ethernet port so you can surf the web, an HDMI output, etc.) but also it exposes several inputs/outputs (GPIO) to interact with the real world (sensors and cool tech gadgets).

A pretty common task when using this device, is to run some script (for example a python script) as a service in the operating system so it can start on boot, stop and restart using systemctl and more. In this post I'm going to explain how to set a little script as a service using Raspbian Jessie in a Raspberry Pi.

First of all, we are going to write a small python script which print "Hello World" every 60 seconds. This is going to be our service script (hello\_world.py):

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | #!/usr/bin/python    from time import sleep    try:      while True:          print "Hello World"          sleep(60)  except KeyboardInterrupt, e:      logging.info("Stopping...") |

You can execute it by python hello\_world.py. If you get boring reading so many hello worlds, press Ctrl+C (or Cmd+C on OSX) to stop it. Save this file as hello\_world.py in your home folder (home/pi/). Now we're going to define the service to run this script:

|  |  |
| --- | --- |
| 1  2 | cd /lib/systemd/system/  sudo nano hello.service |

The service definition must be on the /lib/systemd/system folder. Our service is going to be called "hello.service":

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | [Unit]  Description=Hello World  After=multi-user.target    [Service]  Type=simple  ExecStart=/usr/bin/python /home/pi/hello\_world.py  Restart=on-abort    [Install]  WantedBy=multi-user.target |

Here we are creating a very simple service that runs our hello\_world script and if by any means is aborted is going to be restarted automatically. You can check more on service's options in the next wiki: <https://wiki.archlinux.org/index.php/systemd>.

Now that we have our service we need to activate it:

|  |  |
| --- | --- |
| 1  2  3  4  5 | sudo chmod 644 /lib/systemd/system/hello.service  chmod +x /home/pi/hello\_world.py  sudo systemctl daemon-reload  sudo systemctl enable hello.service  sudo systemctl start hello.service |

For every change that we do on the /lib/systemd/system folder we need to execute a daemon-reload (third line of previous code). If we want to check the status of our service, you can execute:

|  |  |
| --- | --- |
| 1 | sudo systemctl status hello.service |

In general:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | # Check status  sudo systemctl status hello.service    # Start service  sudo systemctl start hello.service    # Stop service  sudo systemctl stop hello.service    # Check service's log  sudo journalctl -f -u hello.service |

REFERENCES

1. <https://wiki.archlinux.org/index.php/systemd>
2. <https://www.digitalocean.com/community/tutorials/understanding-systemd-units-and-unit-files>
3. <https://coreos.com/os/docs/latest/getting-started-with-systemd.html>