Scaricare Win32 diskImager

<https://sourceforge.net/projects/win32diskimager/>

Scaricare Raspbian Buster with desktop (Image with desktop based on Debian Buster)

<https://www.raspberrypi.org/downloads/raspbian/>

formattare SD in FAT 32 con SD CARD FORMATTER <https://www.sdcard.org/downloads/formatter/>

Preparare la SD con Win32 diskImager (selezionare il file IMG relativo al raspbian)

Aggiornare il S.O

sudo apt -get update

sudo apt -get upgrade

yes the gl driver wont work on 4.4.y kernel update, what you can do is

~~sudo BRANCH=next rpi-update~~ non serve!

Inserire la SD, caricare il sistema operativo, ruotare lo schermo di 180 gradi

<https://www.raspberrypi-spy.co.uk/2017/11/how-to-rotate-the-raspberry-pi-display-output/>

**Step 1 – Edit Config.txt**

Start by editing the config.txt file :

sudo nano /boot/config.txt

Add one of the following lines to the bottom of the file :

display\_rotate=0

display\_rotate=1

display\_rotate=2

display\_rotate=3

0 is the normal configuration. 1 is 90 degrees. 2 is 180 degress. 3 is 270 degrees.

If you are using the Official Raspberry Pi touch screen you can use “lcd\_rotate” rather than “display\_rotate”.

Save the file by using CTRL-X, Y then ENTER.

DISABILITARE BLUETOOTH (velocizza avvio)

Sempre nell’editor del config.txt, aggiungere le seguenti righe:

# Disable Bluetooth

dtoverlay=disable-bt

**Step 2 – Reboot**

Then reboot using :

sudo reboot

When the Pi restarts the display should be rotated.

Installare python

<https://github.com/instabot-py/instabot.py/wiki/Installing-Python-3.7-on-Raspberry-Pi>

# Install Python

Install the required build-tools:

apt install libffi-dev libbz2-dev liblzma-dev libsqlite3-dev libncurses5-dev libgdbm-dev zlib1g-dev libreadline-dev libssl-dev tk-dev build-essential libncursesw5-dev libc6-dev openssl git

Download, build and install Python 3.6.5 (the most recent release is available on the [official website](https://www.python.org/downloads/source/" \t "_blank)):

wget <https://www.python.org/ftp/python/3.7.6/Python-3.7.6.tar.xz>  
tar xf Python-3.7.6.tar.xz  
cd Python-3.7.6  
./configure --enable-optimizations  
make -j -l 4  
sudo make altinstall

These steps took me around 4 hours on a Raspberry Pi 2 B.

f you're using bash, open your bashrc and

export PATH=$PATH:/home/pi/ Python-3.7.6

alias python=’/home/pi/ Python-3.7.6/python’

python –version per verificare

**echo $PATH per verificare**

<https://linuxconfig.org/how-to-change-from-default-to-alternative-python-version-on-debian-linux>

Once finished, open ~/.bashrc file using nano:

sudo nano ~/.bashrc

and add new alias on the top of the file to change your default python executable:

alias python3='python3.6'

You can also usepython by changing the previous command to alias python='python3.6'. Once done, exit nano and source your .bashrc file:

. ~/.bashrc

Confirm the changes by checking your default python3 version:

python3 -V

The previous command should give you python3.6.5.

# Install pip

Sudo apt install python3-pip -🡪 non ci dovrebbe essere bisogno

To download and install pip run the following:

curl -O https://bootstrap.pypa.io/get-pip.py  
sudo python3.6 get-pip.py

And now you can install packages for Python 3.6 with pip-3.6!

<https://stackoverflow.com/questions/25981703/pip-install-fails-with-connection-error-ssl-certificate-verify-failed-certi>

installare KIVY seguendo la guida e aggiungere le path:

export PATH=$PATH:/home/pi/ .local/bin

da python controllare la home dir di kivy

import kivy  
    print(kivy.kivy\_home\_dir)

ed editare il config.ini

If you are using the official Raspberry Pi touch display, you need to configure Kivy to use it as an input source. To do this, edit the file ~/.kivy/config.ini and go to the [input] section. Add this:

mouse = mouse

mtdev\_%**(**name**)**s = probesysfs**,**provider=mtdev

hid\_%**(**name**)**s = probesysfs**,**provider=hidinput

INSTALLARE ATLAS (per matplotlib)

**sudo apt**-**get install** **libatlas**-**base**-**dev**.

Installare matplotlib

pip install matplotlib

sudo raspi-config e configurare Locale (tutti i locale)+ OpenGL FAKE KMS

change GPU memory 256

<https://stackoverflow.com/questions/52553499/kivy-ui-is-very-slow-on-a-rpi>

<https://stackoverflow.com/questions/31786122/raspberry-pi-python-kivy-extremely-slow-with-sudo>

problemi Opengl (risolvere)

<https://github.com/anholt/linux/issues/24#issuecomment-207720976>

<https://www.raspberrypi.org/forums/viewtopic.php?f=66&t=140865&p=936901>

<https://github.com/anholt/linux/issues/24>

You can set paths permanently in two places  
  
1. /etc/bash.bashrc for everyone  
2 ~/.bashrc for each user  
  
Add your script paths there.  
  
For python there's also:  
/usr/lib/python\*/site.py  
where the sys.path get set  
  
To make a python program

**Code:**[**Select all**](https://www.raspberrypi.org/forums/viewtopic.php?t=62150)

print "hello world"

run automatically you have to add a line

**Code:**[**Select all**](https://www.raspberrypi.org/forums/viewtopic.php?t=62150)

#!/usr/bin/python

print "hello world"

Then make the file executable with chmod 755 hellow.py or chmod r+x hellow.py

* ~~EDITARE IL FILE CONFIG.TXT~~

~~Inserimento del pulsante di accensione fra GPIO\_3 (pin 5) e massa, così da avere un unico pulsante di accensione~~

~~sudo nano /boot/config~~

~~inserire la seguente riga:~~

~~dtoverlay=gpio-shutdown,gpio\_pin=3~~

~~salvare il file (ctrl+o) e riavviare raspberry~~

~~ora occorre cambiare i pin di default della interfaccia I2C:~~

~~You can implement a software I2C bus on any pair of spare GPIO. You will need to add resistor pull-ups to 3V3 on any spare GPIO you use. For reference pins 3 and 5 have 1k8 pull-ups to 3V3.~~

~~You create the software bus by adding a configuration entry in /boot/config.txt. For details see i2c-gpio in /boot/overlays/README.~~

~~Name: i2c-gpio~~

~~Info: Adds support for software i2c controller on gpio pins~~

~~Load: dtoverlay=i2c-gpio,<param>=<val>~~

~~Params: i2c\_gpio\_sda GPIO used for I2C data (default "23")~~

~~i2c\_gpio\_scl GPIO used for I2C clock (default "24")~~

~~i2c\_gpio\_delay\_us Clock delay in microseconds~~

~~(default "2" = ~100kHz)~~

~~bus Set to a unique, non-zero value if wanting~~

~~multiple i2c-gpio busses. If set, will be used~~

~~as the preferred bus number (/dev/i2c-<n>). If~~

~~not set, the default value is 0, but the bus~~

~~number will be dynamically assigned - probably~~

~~3.~~

~~E.g. to create a software bus on (Broadcom numbered) GPIO 5 and 6 add the following line to /boot/config.txt. You will need to reboot for the change to take effect.~~

~~dtoverlay=i2c-gpio,i2c\_gpio\_sda=5,i2c\_gpio\_scl=6~~

~~A new bus /dev/i2c-3 will be created. You may use all the standard I2C calls to use the bus.~~

~~NEL MIO CASO:~~

**~~dtoverlay=i2c-gpio,i2c\_gpio\_sda=22,i2c\_gpio\_scl=27, bus = 3~~**

DISABILITARE SCREEN SAVER:

- opzione zero:

sudo raspi-config, cercare la voce di menu blank screen, disabilitare!!!!

-opzione1: installare xscreensaver:

sudo apt-get install xscreensaver

una volta installato, dal terminale lanciare il comando:

sudo xscreensaver e dalla finestra settings disabilitare screen saver

OPZIONE 2: (meglio)

Aprire il terminale e digitare il comando:

sudo nano /etc/xdg/lxsession/LXDE-pi/autostart

inserire le seguenti righe nel file di testo e salvare (ctrl+o)

@xset s off

@xset -dpms

@xset s noblank

OPZIONE 3:

If you don't want to install the screensaver you'll need to edit a config file.  
  
Open LXTerminal to get to a command line.  
First make a copy of the config file, in case you mess it up.

**Code:**[**Select all**](https://www.raspberrypi.org/forums/viewtopic.php?t=138353)

sudo cp /etc/lightdm/lightdm.conf /etc/lightdm/lightdm.conf.orig

Once you have a backup copy then use "nano" to edit it.

**Code:**[**Select all**](https://www.raspberrypi.org/forums/viewtopic.php?t=138353)

sudo nano /etc/lightdm/lightdm.conf

Use the keyboard arrow keys to move the cursor and scroll down the file until you get to "[SeatDefaults]". Make sure it's the **SeatDefaults** entry with the brackets around it. Then add a line underneath so it looks just like this..

**Code:**[**Select all**](https://www.raspberrypi.org/forums/viewtopic.php?t=138353)

[SeatDefaults]

xserver-command=X -s 0 -dpms

To exit nano and save press <CTRL> and x, then press y to confirm, then press <ENTER>  
You'll need to either restart the desktop or reboot for it to take effect, it only reads the config file when the desktop starts.

**Installazione dei linguaggi (LOCALES)**

Se non eseguito in fase di installazione:

aprire terminale

sudo raspi-config

andare su localisation options 🡪 Change locale 🡪 scegliere la lingua (o tutte le lingue) da aggiungere

Per controllare aprire terminale e digitare: locale -a

COLLEGAMENTO E CONFIGURAZIONE RTC

Inserire RTC DS3231 fra i pin 1-3-5-7

ABILITARE L’INTERFACCIA I2C DALLE IMPOSTAZIONI

## Wiring your RTC module to the Raspberry Pi

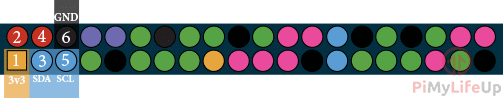
On your **RTC Module**, you should find at least four connections. Some RTC circuits may come with more, but we only need the following four for it to work with the Raspberry Pi: **VCC**/**5V**/**Vin** (IC Power-supply pin), **SDA** (Serial Data Line), **SCL** (Serial Clock Line) and **GND** (Ground power-supply pin)

You can either connect these lines directly to your Raspberry Pi or connect it [to a breadboard](https://go.pimylifeup.com/LSWdzn/amazon/breadboard) and then to the Raspberry Pi. For this tutorial, we utilized the **Pi RTC PCF8523** from Adafruit which plugs in directly over the first six pins which significantly simplifies the process of setting up an **RTC** (**Real Time Clock**) module.

However, wiring up a normal **PCF8523**, **DSL1307** and a **DS3231** isn’t a complicated process, following our guide below you should have everything connected to in no time.

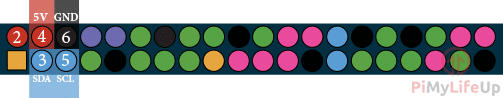
### **DS3231 & PCF8523**

* **Vin** connects to **Pin 1**
* **SDA** connects to **Pin 3**
* **SCL** connects to **Pin 5**
* **GND** connects to **Pin 6**



### **DS1307**

* **Vin** connects to **Pin 4**
* **SDA** connects to **Pin 3**
* **SCL** connects to **Pin 5**
* **GND** connects to **Pin 6**



## Configuring the Raspberry Pi for I2C

Before we begin setting up and utilizing our RTC on the Raspberry Pi, we first have to make use of the **raspi-config** tool to configure our Raspberry Pi for use with I2C.

**1.** Let’s begin this tutorial by ensuring our [Raspberry Pi is entirely up to date](https://pimylifeup.com/update-raspbian/); this ensures that we will be utilizing all the latest software available.

sudo apt-get update

sudo apt-get upgrade

**2.** With the Raspberry Pi now entirely up to date we can now run its configuration tool to begin the process of switching on I2C. Run the following command to launch the configuration tool.

sudo raspi-config

**3.** This command will bring up the configuration tool; this tool is an easy way to make a variety of [changes to your Raspberry Pi’s configuration](https://pimylifeup.com/raspi-config-tool/). Today, however, we will only by exploring how to enable the I2C interface.

Use the **arrow keys** to go down and select “**5 Interfacing Options**“. Once this option has been selected, you can press **Enter**.

**4.** On the next screen, you will want to use the **arrow keys** to select “**P5 I2C**“, press **Enter** once highlighted to choose this option.

**5.** You will now be asked if you want to enable the “**ARM I2C Interface**“, select **Yes** with your **arrow keys** and press **Enter** to proceed.

**6.** Once the raspi-config tool makes the needed changes, the following text should appear on the screen: “**The ARM I2C interface is enabled**“.

However, before I2C is genuinely enabled, we must first restart the Raspberry Pi. To do this first get back to the terminal by pressing **Enter** and then **ESC**.

Type the following command into the terminal on your Raspberry Pi to restart it.

sudo reboot

**7.** Once the Raspberry Pi has finished restarting we need to install an additional two packages, these packages will help us tell whether we have set up I2C successfully and that it is working as intended.

Run the following command on your Raspberry Pi to install **python-smbus** and **i2c-tools**:

sudo apt-get install python-smbus i2c-tools

**8.** With those tools now installed run the following command on your Raspberry Pi to detect that you have correctly wired up your RTC device.

sudo i2cdetect -y 1

If you have successfully wired up your RTC circuit, you should see the **ID #68** appear. This id is the address of the **DS1307**, **DS3231** and the **PCF85231** RTC Chips.

Once we have the Kernel driver up and running the tool will start to display **UU** instead, this is an indication that it is working as intended.

## Setting up the Raspberry Pi RTC Time

With I2C successfully setup and verified that we could see our RTC circuit then we can begin the process of configuring the Raspberry Pi to use our RTC Chip for its time.

**1.** To do this, we will first have to modify the Raspberry Pi’s boot configuration file so that the correct Kernel driver for our RTC circuit will be successfully loaded in.

Run the following command on your Raspberry PI to begin editing the **/boot/config.txt** file.

sudo nano /boot/config.txt

**2.** Within this file, you will want to add one of the following lines to the bottom of the file, make sure you use the correct one for the RTC Chip you are using. In our case, we are using a **PCF8523**.

**DS1307**

dtoverlay=i2c-rtc,ds1307

**PCF8523**

dtoverlay=i2c-rtc,pcf8523

**DS3231**

dtoverlay=i2c-rtc,ds3231

Once you have added the correct line for your device to the bottom of the file you can save and quit out of it by pressing **Ctrl + X**, then **Y** and then **Enter**.

**3.** With that change made we need to restart the Raspberry Pi, so it loads in the latest configuration changes.

Run the following command on your Raspberry Pi to restart it.

sudo reboot

**4.** Once your Raspberry Pi has finished restarting we can now run the following command, this is so we can make sure that the kernel drivers for the RTC Chip are loaded in.

sudo i2cdetect -y 1

You should see a wall of text appear, if **UU** appears instead of **68** then we have successfully loaded in the Kernel driver for our RTC circuit.

**5.** Now that we have successfully got the kernel driver activated for the RTC Chip and we know it’s communicating with the Raspberry Pi, we need to remove the **fake hwclock**package. This package acts as a placeholder for the real hardware clock when you don’t have one.

Type the following two commands into the terminal on your Raspberry Pi to remove the fake-hwclock package. We also remove hwclock from any startup scripts as we will no longer need this.

sudo apt-get -y remove fake-hwclock

sudo update-rc.d -f fake-hwclock remove

**6.** Now that we have disabled the **fake-hwclock** package we can proceed with getting the original hardware clock script that is included in Raspbian up and running again by commenting out a section of code.

Run the following command to begin editing the original RTC script.

sudo nano /lib/udev/hwclock-set

**7.** Find and comment out the following three lines by placing **#** in front of it as we have done below.

**Find**

if [ -e /run/systemd/system ] ; then

exit 0

fi

**Replace With**

#if [ -e /run/systemd/system ] ; then

# exit 0

#fi

Once you have made the change, save the file by pressing **Ctrl + X** then **Y** then **Enter**.

## Syncing time from the Pi to the RTC module

Now that we have our RTC module all hooked up and Raspbian and the Raspberry Pi configured correctly we need to synchronize the time with our RTC Module. The reason for this is that the time provided by a new RTC module will be incorrect.

**1.** You can read the time directly from the RTC module by running the following command if you try it now you will notice it is currently way off our current real-time.

sudo hwclock -D -r

**2.** Now before we go ahead and sync the correct time from our Raspberry Pi to our RTC module, we need to run the following command to make sure the time on the Raspberry Pi is in fact correct. If the time is not right, make sure that you are [connected to a Wi-Fi](https://pimylifeup.com/setting-up-raspberry-pi-wifi/) or Ethernet connection.

date

**3.** If the time displayed by the date command is correct, we can go ahead and run the following command on your Raspberry Pi. This command will write the time from the Raspberry Pi to the RTC Module.

sudo hwclock -w

**4.** Now if you read the time directly from the RTC module again, you will notice that it has been changed to the same time as what your Raspberry Pi was set at. You should never have to rerun the previous command if you keep a battery in your RTC module.

sudo hwclock -r

You should hopefully now have a fully operational RTC module that is actively keeping your Raspberry Pi’s time correct even when it loses power or loses an internet connection. I hope you have enjoyed this [fun Pi project](https://pimylifeup.com/category/projects/) and will put it to good use.

Risorse in caso di problemi:

https://raspberrypi.stackexchange.com/questions/59295/can-we-use-other-gpio-pins-as-i2c

https://raspberrypi.stackexchange.com/questions/38186/how-to-install-real-time-clock-rtc-on-raspbian/38304#38304

https://www.abelectronics.co.uk/kb/article/30/rtc-pi-on-a-raspberry-pi-raspbian-stretch

**Inserire pulsante di spegnimento:**

COLLEGARE IL PULSANTE FRA I PIN GPIO27 – GROUND (13-14) (non serve pull-up esterna)

Il rele di abilitazione alimentazione andrà collegato invece al GPIO22(15-16)

L’uscita parte a 0, poi si porta a 1 quando rasp è acceso. Torna a 0 DOPO lo spegnimento sicuro

dtoverlay=gpio-shutdown,gpio\_pin=23,active\_low=1,gpio\_pull=up

dtoverlay=gpio-poweroff,gpiopin=24,active\_low="y"

**Abilitare protocollo 1-wire sul pin GPIO25**

To enable the one-wire interface you need to add the following line to /boot/config.txt, before rebooting your Pi:

1. dtoverlay=w1-gpio

or

1. dtoverlay=w1-gpio,gpiopin=x

if you would like to use a custom pin (default is BCM4, as illustrated in pinout herein).

Alternatively you can enable the one-wire interface on demand using raspi-config, or the following:

1. sudo modprobe w1-gpio

Newer kernels (4.9.28 and later) allow you to use dynamic overlay loading instead, including creating multiple 1-Wire busses to be used at the same time:

1. sudo dtoverlay w1-gpio gpiopin=4 pullup=0 # header pin 7
2. sudo dtoverlay w1-gpio gpiopin=17 pullup=0 # header pin 11
3. sudo dtoverlay w1-gpio gpiopin=27 pullup=0 # header pin 13

once any of the steps above have been performed, and discovery is complete you can list the devices that your Raspberry Pi has discovered via all 1-Wire busses (by default BCM4), like so:

1. ls /sys/bus/w1/devices/

n.b. Using w1-gpio on the Raspberry Pi typically needs a 4.7 kΩ pull-up resistor connected between the GPIO pin and a 3.3v supply (e.g. header pin 1 or 17). Other means of connecting 1-Wire devices to the Raspberry Pi are also possible, such as using i2c to 1-Wire bridge chip

COLLEGARE RESISTENZA DA != K fra GPIO25 e 3.3v, collegare il sensore e aprire il config:

sudo nano /boot/config.txt

aggiungere la riga

1. dtoverlay=w1-gpio,gpiopin=25

Nota: se si è attivato il 1-Wire dal menu, occorre aprire il config, trovare la riga e modificarla poiché di default il 1-Wire è attivato su GPIO 4 che è occupato dal connettore RTC

**CONFIGURAZIONE SENSORI TEMPERATURA:**

Qui la guida adafruit completa:

<https://learn.adafruit.com/adafruits-raspberry-pi-lesson-11-ds18b20-temperature-sensing/hardware>

qui info più dettagliate:

<https://www.waveshare.com/wiki/Raspberry_Pi_Tutorial_Series:_1-Wire_DS18B20_Sensor>

per verificare i dispositivi 1-wire:

lsmod | grep w1

IMPORTANTE!!!!!! Aggiungerle il caricamento dei moduli kernel al boot:

sudo nano /etc/modules

**aggiungere:**

w1-gpio

w1-therm

Or adding these commands to the python code:

os.system('modprobe w1-gpio')

os.system('modprobe w1-therm')

INVERTIRE LO SCHERMO: NON FUNZIONA

<https://www.raspberrypi.org/forums/viewtopic.php?t=219333>

SETTAGGIO RISOLUZIONE:

cd /boot

sudo nano config.txt

cercare la voce “FrameBuffer” e modificare la risoluzione: 800 x 480

<https://raspberrypi.stackexchange.com/questions/72118/set-display-resolution-for-official-7-touchscreen-display>

**SETTAGGIO USCITA CUFFIE: 🡪NON SERVE PIU’**

andare su sudo-raspi-config, selezionare advanced options, quindi headphones

**SETTAGGIO VNC:**

**seguire le istruzioni da questi link:**

raspi-config e settare VNC :  
<https://www.techcoil.com/blog/how-to-enable-vnc-server-on-raspbian-stretch-with-raspi-config/>

link per creare un account privato su VNC.clud

<https://www.realvnc.com/en/raspberrypi/#sign-up>

in generale seguire le istruzioni:

<https://www.raspberrypi.org/documentation/remote-access/vnc/>

Occorre registrarsi e convalidare una mail valida.

installare real vnc su smartphone e/o su altri dispositivi. Il collegamento da remoto viene effettuato con la username e password utilizzati per l’accesso su raspberry (user = pi; password= propria password

**CONFIGURAZIONE SCHEDA AUDIO USB ESTERNA:**

Acquistare un modulo di questo tipo (GENERALPLUS…importante)



Inserirlo in una presa USB

Digitare sul terminale:

$ aplay -L

Verificare che vi sia una riga del tipo:

default:CARD=Device

segnarsi il nome.

Verificare il “card number”: 0 è il device interno del raspberry, 1 dovrebbe essere la scheda audio esterna.

amixer -c 1 info

appare una descrizione con indicato se si tratta di device usb esterno etc.

cerchiamo il controllo dell’audio (numid)

$ amixer -c 1 controls

numid**=**3,iface**=**MIXER,name**=**'Mic Playback Switch'

numid**=**4,iface**=**MIXER,name**=**'Mic Playback Volume'

numid**=**7,iface**=**MIXER,name**=**'Mic Capture Switch'

numid**=**8,iface**=**MIXER,name**=**'Mic Capture Volume'

numid**=**9,iface**=**MIXER,name**=**'Auto Gain Control'

numid**=**5,iface**=**MIXER,name**=**'Speaker Playback Switch'

numid**=**6,iface**=**MIXER,name**=**'Speaker Playback Volume'

numid**=**2,iface**=**PCM,name**=**'Capture Channel Map'

Ci interessa il numid numero 6 perché è il playback volume.

Nel file config sezione [AUDIO] inseriamo:

card\_num = 1  
numid = 6

ORA SETTIAMO IL GAIN AL MASSIMO DALL’USCITA AUDIO:

Da terminale digitare:

$ alsamixer

Quindi tasto F6, selezionare il controller USB e coi tasti freccia mettere il volume al massimo.

Regolare il trimmer del LM386 a seconda del miglior rapporto segnale/rumore

Risorse:

<https://die-antwort.eu/techblog/2017-12-raspberry-pi-usb-audio-interface-command-line/>

<https://learn.adafruit.com/usb-audio-cards-with-a-raspberry-pi/instructions>

CREARE AUTOSTART ENTRY con .desktop file

<https://www.itechfy.com/tech/auto-run-python-program-on-raspberry-pi-startup/>

creare directory autostart da terminale:

***mkdir /home/pi/.config/autostart***

usare editor nano per creare file desktop:

***nano /home/pi/.config/autostart/dougho.desktop***

copiare le seguenti righe, salvare con CTRL+X e riavviare

***[Desktop Entry]***

***Type= Application***

***Name= PiCube***

***Exec= python3  /home/pi/Dough-O/main.py***