POLITÉCNICA

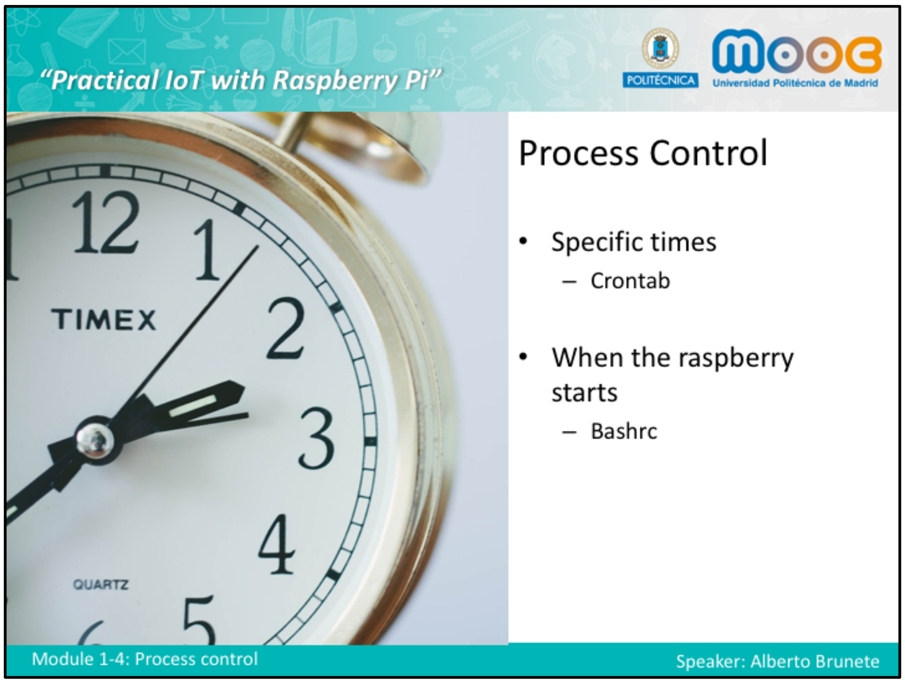
Universidad Politécnica de Madrid

"Practical IoT with Raspberry Pi"

Module 1-4: Process Control

Hello and welcome to the Module 1, fourth video, Process control. Here you will learn how to execute your program in a scheduled time, so you don't have to execute manually. It will execute automatically when you specify.

Video 1.4: crontab, bashrc



“Practical IoT with Raspberry Pi”

POLITÉCNICA Universidad Politécnica de Madrid

Process Control

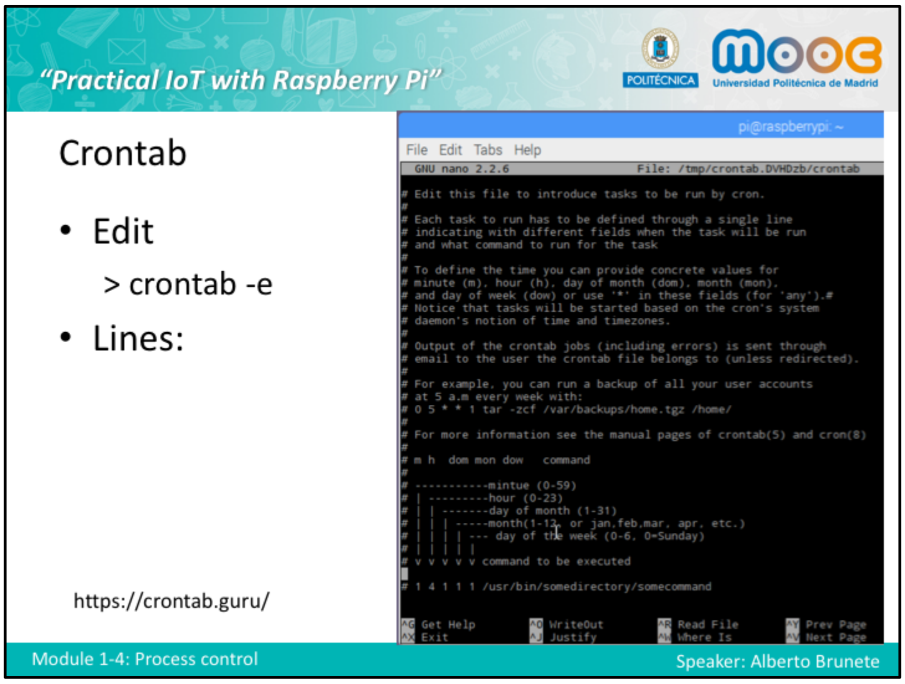
- Specific times
  - Crontab
- When the raspberry starts
  - Bashrc

Module 1-4: Process control

Speaker: Alberto Brunete

In remote operation, you are not present to execute you program. You want the program to be executed at specific times, or only one time when the Raspberry boots up. We will explain two tools to do this.

The first one is cron and crontab, used to run our program at specific time. And the second one is bashrc, used to run our program when the raspberry boots. Let’s star with cron and crontab.


"Practical IoT with Raspberry Pi"
POLITÉCNICA
MOOC
Universidad Politécnica de Madrid

Crontab

- Edit

  > crontab -e

- Lines:

https://crontab.guru/

pi@raspberrypi: ~
File Edit Tabs Help

```
  GNU nano 2.2.6                File: /tmp/crontab.DVHDzb/crontab

# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
#
# -----------mintue (0-59)
# | ---------hour (0-23)
# | | -------day of month (1-31)
# | | | -----month(1-12, or jan,feb,mar, apr, etc.)
# | | | | --- day of the week (0-6, 0=Sunday)
# | | | | |
# v v v v v command to be executed

# 1 4 1 1 1 /usr/bin/somedirectory/somecommand

^G Get Help     ^O WriteOut     ^R Read File    ^Y Prev Page
^X Exit         ^J Justify      ^W Where Is     ^V Next Page
```

Cron is a system daemon (a program that runs in the background) used to execute desired tasks at designated times.

A crontab file is a simple text file containing a list of commands meant to be run at specified times. It is edited using the crontab command. The commands in the crontab file (and their run times) are checked by the cron daemon, which executes them in the system background.

Each user has a crontab file. The cron daemon checks a user's crontab file regardless of whether the user is actually logged into the system or not.

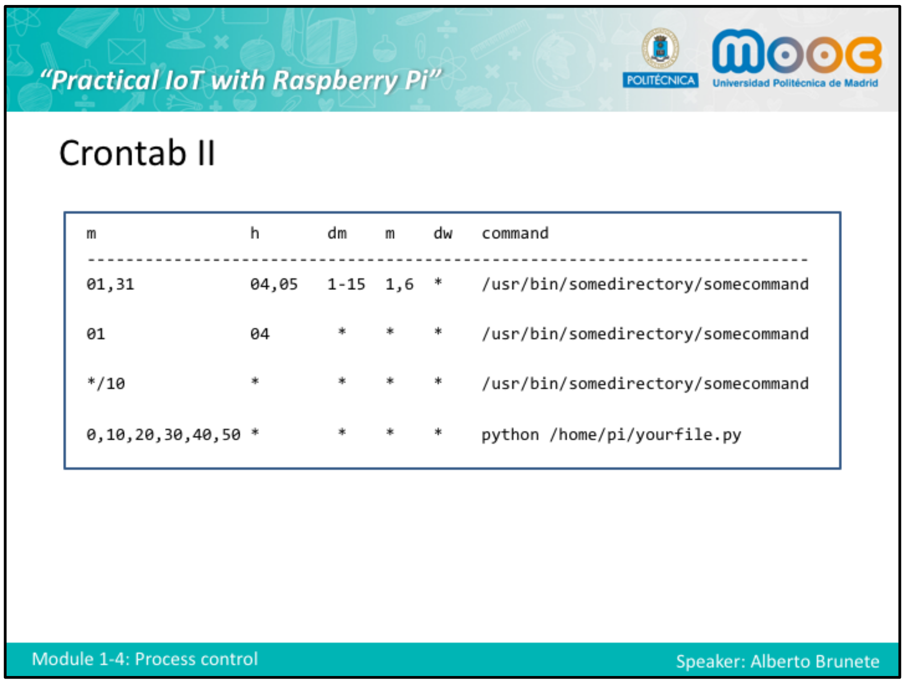
To use cron, add entries to your own user's crontab file. To edit the crontab file enter:
> crontab -e

Each line has five time-and-date fields, followed by a command, followed by a newline character. The fields are separated by spaces. The five time-and-date fields cannot contain spaces. The five time-and-date fields are as follows: minute (0-59), hour (0-23, 0 = midnight), day (1-31), month (1-12), weekday (0-6, 0 = Sunday).

For example, the line in the file will run /usr/bin/somedirectory/somecommand at 4:01am on January 1st plus every Monday in January.

Use the crontab guru to easily configure your scheduled tasks.

---
https://help.ubuntu.com/community/CronHowto


"Practical IoT with Raspberry Pi"

POLITÉCNICA

MOOC Universidad Politécnica de Madrid

Crontab II

```
m               h      dm    m    dw   command
---------------------------------------------------------------------------
01,31           04,05  1-15  1,6  *    /usr/bin/somedirectory/somecommand

01              04      *    *    *    /usr/bin/somedirectory/somecommand

*/10            *       *    *    *    /usr/bin/somedirectory/somecommand

0,10,20,30,40,50 *      *    *    *    python /home/pi/yourfile.py
```

Module 1-4: Process control

Speaker: Alberto Brunete

More examples.
Comma-separated values can be used to run more than one instance of a particular command within a time period. Dash-separated values can be used to run a command continuously.

In the first example, the program will run from the first day to the 15th day of January and June. At 4:01, 4:31, 5:01, 5:31.
01,31 04,05 1-15 1,6 * /usr/bin/somedirectory/somecommand

An asterisk (*) can be used so that every instance (every hour, every weekday, every month, etc.) of a time period is used. The second example will run all days of the years at 4:01.
01 04 * * * /usr/bin/somedirectory/somecommand

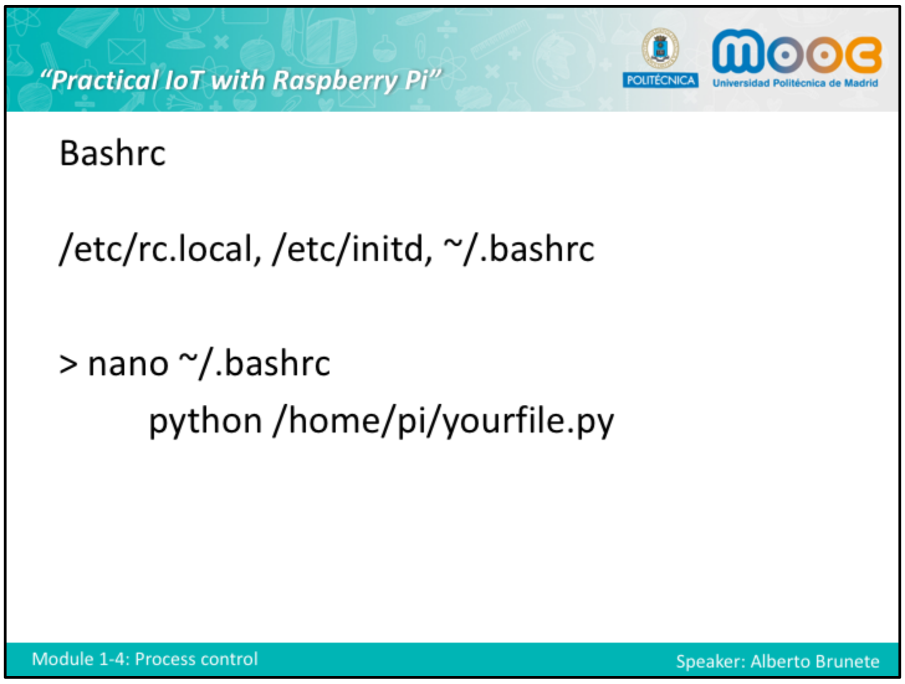
You may want to run a script some number of times per time unit. For example if you want to run it every 10 minutes use the third crontab entry (runs on minutes divisible by 10: 0, 10, 20, 30, etc.)
*/10 * * * * /usr/bin/somedirectory/somecommand

This is also equivalent to the more cumbersome

```
0,10,20,30,40,50 * * * * /usr/bin/somedirectory/somecommand
```

One you save the file, it is automatically configured, and your task will automatically run at the specified times.

Please make sure that your program terminates before it is executed next time.



"Practical IoT with Raspberry Pi"

POLITÉCNICA MOOC Universidad Politécnica de Madrid

Bashrc

/etc/rc.local, /etc/initd, ~/.bashrc

```
> nano ~/.bashrc
        python /home/pi/yourfile.py
```

Module 1-4: Process control

Speaker: Alberto Brunete

There are several solution to execute a file or script when the Raspberry boots. /etc/rc.local, /etc/initd, etc. We will use bashrc.

Please be very careful because if you execute something at the startup of the system, you may loose control of your raspberry, because no terminal may appear. With basrc you will not have that problem, because you can cancel the task executed by pressing ctrl-c.

The .bashrc file is a script that is executed whenever a new terminal session is started. This is what happens when you open a new terminal window or the raspberry boots up.

The .bashrc file itself contains a series of configurations for the terminal session. This includes setting up or enabling: colouring, completion, the shell history, command aliases and more.

The bashrc file starts with a dot, so it is hidden. To display it you should use ls .*

You can edit it with a text editor or in the terminal with the nano editor:

```
> nano ./.bashrc
```

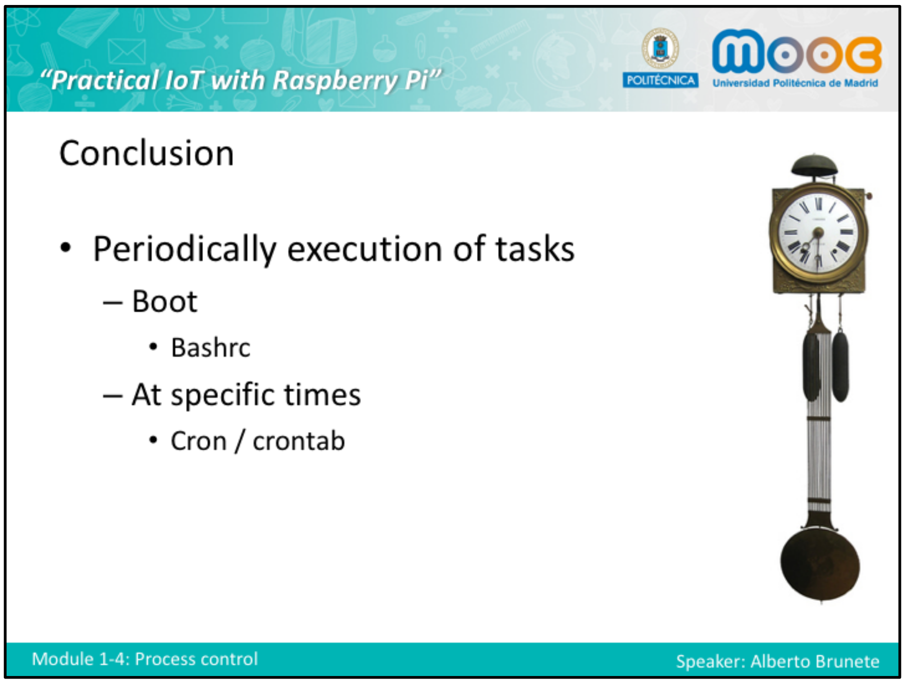
Include at the end the file you want to execute.

```
python /home/pi/yourfile.py
```

If you have to do several tasks, you can write an script.

Save the file, and next time the raspberry boots, your file will be executed.

By contrast a terminal session in **login mode** will ask you for user name and password and execute the ~/.bash_profile script. This is what takes place, for instance, when you log on to a remote system through SSH.



"Practical IoT with Raspberry Pi"

Conclusion

- Periodically execution of tasks
  - Boot
    - Bashrc
  - At specific times
    - Cron / crontab

Module 1-4: Process control

Speaker: Alberto Brunete

In this lesson we have seen how to execute our program at specific times: either at the booting time, with bashrc, or at specific times of the day, month or year, with crontab.

With this we finish our first module, I hope you have enjoyed it!