

1 Introducció

En aquest projecte es desenvolupa una aplicació *Internet of Things* client que es comunica amb un servidor connectat a una base de dades. A més a més s'implementaran clients *Android* i *Web* executant-se sobre navegador en la part final del projecte. El material que s'ha d'adquirir es troba a <http://soft0.upc.edu/~francesc/pbe/MaterialIoT.html>

El projecte es desenvolupa sota la metodologia CDIO—Conceive Design Implement Operate—, d'àmplia acceptació en l'ensenyament de les enginyeries. Es valorarà la simplicitat, elegància i claredat de les implementacions.

Primer cal configurar la *Raspberry Pi 3B+*. S'ha de baixar la imatge *Raspbian Stretch* de https://downloads.raspberrypi.org/raspbian_latest, descomprimir-la i gravar-la a la *microSD*. Una còpia local de baixada més ràpida es troba a <http://soft0.upc.edu/~francesc/pbe/2019-06-20-raspbian-buster-full.zip>. Les instruccions es troben a <https://www.raspberrypi.org/documentation/installation/installing-images/linux.md>.

La baixada i descompressió de l'imatge s'ha de fer al directori `/tmp` ja que és gran i no hi cap a la zona. Es recomana:

- Comprovar que l'arxiu `.zip` té la signatura `sha256` correcta.
- Fer servir la comanda `dd` amb les opcions `status` i `conv`. Si es fa en els ordinadors del Lab la comanda `dd` no disposa de l'opció de `status`.
- Comprovar que les signatures `sha256` de l'arxiu `.img` i del dispositiu `/dev/sdX` són iguals. S'ha de fer immediatament després d'haver gravat i comptabilitzant únicament el nombre de sectors gravats.

A continuació modifiqueu els arxius de configuració adients del directory `/etc` per aconseguir una configuració *headless*—connexió per `ssh` via `wifi`—. La informació es troba a <https://caffinc.github.io/2016/12/raspberry-pi-3-headless/>. Aquest article està una mica desfasat. És important tenir en compte que per que funcioni `ssh` en les últimes imatges, `wpa_supplicant.conf` s'ha de crear a l'arrel de la partició `boot` i en l'arranc es copia automàticament a `/etc/wpa_supplicant`. En els ordinadors del Lab no disposeu de privilegis per modificar arxius de la partició `root` de la tarjeta `microSD`, per tant ho haureu de fer en el vostre portàtil.

Hi ha dues xarxes WIFI al Lab S-103:

Docencia ENTEL seguretat: oberta. El problema de fer servir aquesta per connectar els RPi és que assigna IP dinàmica (10.0.100.xxx) i llavors no et pots connectar per `ssh` ja que no coneixes la IP—veure a continuació com solventar-ho—. L'altra restricció es que no permet comunicació amb dispositius connectats—no es poden fer proves client/servidor—

Un exercici interessant a fer consisteix en visualitzar pel display la IP. S'ha de buscar un arxiu de configuració de `dhcpcd` i programar un shell script que invoqui un petit programa—Python3/Ruby— que visualitzi la IP.

PBE_2018 seguretat: oberta, DHCP del rang de l'aula: 10.0.30.150..10.0.30.189. Cada alumne disposa de dues adreces IP—per ETH i WIFI— que es configuren estàticament amb `routers=10.0.30.1` i `domain_name_servers=10.0.2.6`. Es la recomanada.

Una altra possibilitat consisteix en connectar per ethernet la RPi3 amb un portàtil amb connexió a internet ([compartir internet](#))

Podeu configurar la RPi amb dues xarxes WIFI: PBE_2018 i la de casa. Per tal de no anar modificant els arxius de configuració cada vegada que es canvia de xarxa, és molt interessant configurar de tal forma que es detecti quina xarxa és present i es connecti automàticament. A casa podeu treballar connectant un monitor HDMI i un teclat.

Per treballar amb *RPi* farem servir `ssh` amb username `pi` i password `raspberry`. El servidor gràfic serà `vnc`. Aquí teniu una guia: <https://www.realvnc.com/en/connect/docs/raspberry-pi.html>

2 Primer puzzle

Programeu en Python3/Ruby un programa que imprimeixi per consola el uid—user identifier— de la tarjeta o clauer *Mifare S50* o de la targeta UPC—Infineon Mifare classic 1K—. El uid s'ha d'imprimir en *hexadecimal* i majúscules, p.e. **45CD2A5B**. El *puzzle 1* s'usarà en el *puzzle 2*.

Convé seguir la següent plantilla:

Listing 1: python

```
class Rfid...:  
    ...  
    # return uid in hexa str  
    def read_uid(self):  
        ...  
  
    if __name__ == "__main__":  
        rf = Rfid...()  
        uid = rf.read_uid()  
        print(uid)
```

Listing 2: ruby

```
class Rfid...  
    ...  
    # return uid in hexa str  
    def read_uid  
        ...  
    end  
end  
  
if __FILE__ == $0  
    rf = Rfid...new  
    uid = rf.read_uid  
    puts uid  
end
```

Haureu d'instal·lar les biblioteques Python3/Ruby per RPi adients per cada perifèric. Cada alumne disposa d'un RFID diferent:

Itead PN532 NFC module la documentació es troba a https://www.itead.cc/wiki/ITEAD_PN532_NFC_MODULE. Es programarà per I2C

Elechouse PN532 NFC module la documentació es troba a http://www.elechouse.com/elechouse/images/product/PN532_module_V3/PN532_%20Manual_V3.pdf. Es programarà per UART

MFRC-RC522 module És un disseny xinés molt econòmic i popular. La documentació original de desenvolupament—en xinés— es troba a <https://yq.aliyun.com/articles/59111>—s'ha de navegar des de un lloc sense firewalls que no bloquegin part del contingut. Sembla ser que des del Lab S103 no es visualitza correctament—. És difícil de trobar en altres llocs aquesta informació tècnica de desenvolupament, sobre tot l'esquemàtic. Afortunadament el que ens cal és algun dels múltiples tutorials de funcionament que es troben fàcilment. Es programarà per SPI

Mifare USB card reader La descripció del lector es troba a: https://www.amazon.es/dp/B01BV3MSX6/ref=asc_df_B01BV3MSX654071104/?tag=googshopes-21&creative=24526&creativeASIN=B01BV3MSX6&linkCode=df0&hvdev=c&hvnetw=g&hvqmt=. Es un lector que funciona com un teclat

ACR122U USB NFC reader la pàgina de producte es troba a: <https://www.acs.com.hk/en/products/3/acr122u-usb-nfc-reader/>

i un alumne per grup disposa d'un display LCD 20x4:

<https://www.amazon.com/keyestudio-Display-Module-Arduino-Raspberry/dp/B01799UUGS>. Ha de programar en Python3/Ruby un programa que llegeixi en una sola crida de lectura—a ser possible— un string multilínia per la consola i el visualitzi en el display.

S'haurà d'entregar un document individual de màxim 2 fulls imprès amb:

- configuracions realitzades
- biblioteques i paquets instal·lats
- problemes trobats
- codi ben formatat

2.1 Per aprofundir més en la programació d'LCDs

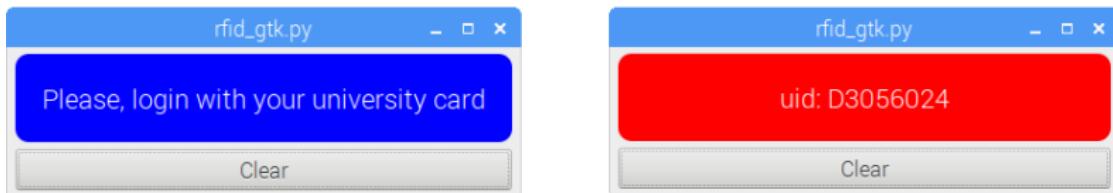
Busqueu dos drivers de nucli Linux: `lucidm/lcdi2c` i `gorskima/hd44780-i2c`. Estudieu la seva funcionalitat. Aquests drivers tenen l'avantatge que es poden usar des de qualsevol llenguatge:

- Quin és millor?
- El driver `gorskima/hd44780-i2c` no permet posicionar el cursor. Estudieu el seu codi, en particular `hd44780-dev.c`—no és gaire difícil—. Modifiqueu bàsicament la funció `hd44780_handle_esc_seq_char` per admetre comandes de posicionament ANSI: `Esc[<row>;<column>H`, on `row` va de `1..ROWS` i `column` de `1..COLUMNS`
- Programeu el `puzzle1` amb els dos drivers

3 Segon puzzle

Ara programareu una versió gràfica del puzzle anterior, amb les biblioteques `PyGObject` i `Ruby-GNOME2/gtk3`. Hauríeu de fer servir el `puzzle1` com a biblioteca.

La finestra principal està formada per un `Label` que demana login amb la targeta RF—carnet UPC o Mifare— i un `Button` de *Clear*. Quan es llegeix la targeta, apareix l'uid en hexadecimal:

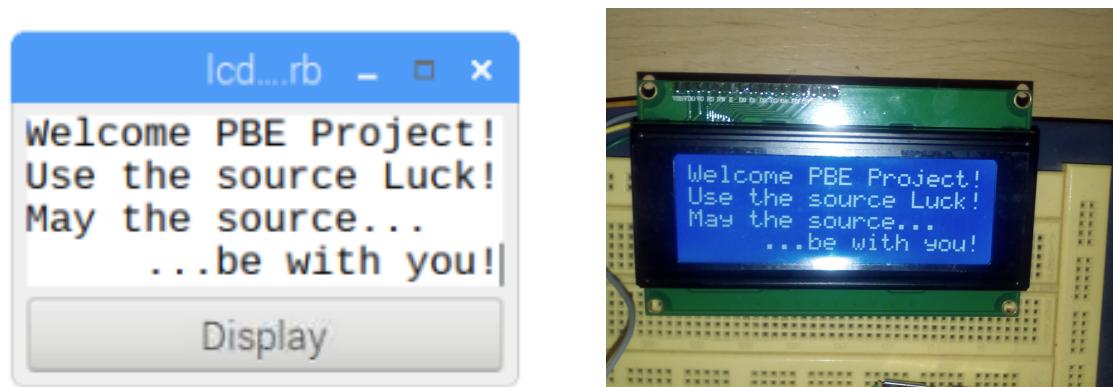


Una consideració d'implementació important a tenir en compte és que `read_uid` és bloquejant i el tractament d'events en entorns gràfics **NO** s'ha de bloquejar. Una discussió de la problemàtica es pot trobar a https://en.wikipedia.org/wiki/Event_dispatching_thread. Una bona documentació de la mateixa problemàtica aplicada a `PyGObject`—els conceptes també serien vàlids per `Ruby`— la podeu trobar a <https://pygobject.readthedocs.io/en/latest/guide/threading.html>. Això vol dir que `read_uid` s'haurà d'executar en un thread auxiliar. El fet que s'hagi d'executar el codi bloquejant en un thread auxiliar comporta un altre problema, que és que no es pot executar codi gràfic en aquest thread—les biblioteques gràfiques no són reentrants. Una motivació es pot trobar en [aquest article](#).

També pot succeir que algunes biblioteques de lectors facin consulta—polling—, es a dir testegin la presència de la targeta RF sense bloquejar. S'haurà de programar una solució adequada en aquest cas.

La biblioteca `ruby mfrc522` i d'altres que fan servir el mòdul `ffi` pot donar errors en `runtime` amb `Gtk`. La solució consisteix en reinstal·lar la `gemma ffi` i totes les biblioteques que la facin servir.

La programació del display consistira en un `TextView` on s'introduirà l'string multilínia i un `Button` de *Display* que al clicar visualitzarà l'string en el display:

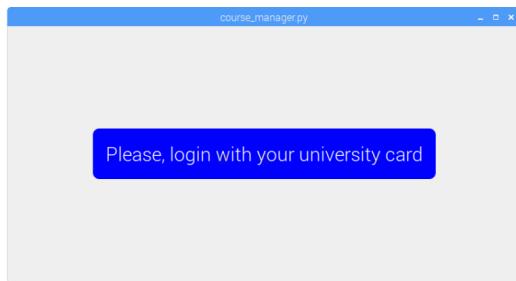


Les biblioteques gràfiques `Gtk3` són estàndard en `Raspbian`, força riques i s'estilitzen com en una aplicació web, amb `CSS`. Tracteu de fer servir l'estilitzat per aconseguir una app atractiva visualment.

4 Critical Design Review (CDR)

Es vol dissenyar una aplicació Client/Servidor semblant a *Atenea*. L'alumne pot consultar tres taules/documents: els treballs—*tasks*— a entregar, els horaris—*timetables*— o les notes—*marks*— obtingudes. El client és una aplicació gràfica RPi3 programada en Python3/Ruby que es connecta amb el Servidor programat en PHP/NodeJS(Javascript) que accedeix a una Base de Dades MySQL/MongoDB amb les taules/documents anteriors. Més endavant es dissenyaran clients ANDROID i WEB.

L'alumne s'autentifica amb la targeta RF:



El seu uid s'envia al servidor i es cerca en una taula interna **students** de la BD formada per parells (**name**, **student_id**). **student_id** és el uid. Quan l'estudiant s'autentifica, es visualitza el seu nom en el display:



Si el uid no existeix en la taula **students** es mostra un missatge d'error que un cop confirmat torna al principi

A continuació es demana la consulta—query— que es vol realitzar a les taules/documents de la BD. La taula pot ser: **tasks**, **timetables** o **marks**. Les columnes de **tasks** són: **date**—data d'entrega—, **subject**—assignatura— i **name**—nom del treball—. Les columnes de **timetables** són: **day**—dia de la classe—, **hour**—hora—, **subject**—assignatura— i **room**—aula—. Les columnes de **marks** són: **subject**—assignatura—, **name**—nom de l'examen o treball—, **mark**—nota— i columnes internes que es refereixen a l'estudiant. També existeix la taula interna **students** que únicament es fa servir per autentificació

La sintaxi d'una *query* és: **table?constraint&constraint&...&constraint**. Una constraint és una restricció sobre les columnes. Exemples:

timetables llista horaris a partir del dia/hora actual

tasks?date[gte]=now&limit=1 llista el primer treball a partir de la data actual

tasks?date=2018-10-22 llista els treballs amb data 2018-10-22

timetables?limit=1 llista la següent classe

timetables?day=Fri&hour[gt]=08:00 llista la primera classe després de divendres a les 8h

marks?mark[lt]=5 llista totes les notes suspeses

marks?subject=PBE llista les notes de PBE

La paraula reservada **now** es pot fer servir allà on sigui raonable per indicar el dia, hora o data actual. Cada taula té un ordenament per defecte raonable:

The screenshot shows a window titled "course_manager.py" with a "Logout" button in the top right corner. A search bar contains the text "timetables". Below it is a table titled "timetables" with the following data:

day	hour	subject	room
Tue	08:00:00	TD	A4-105
Tue	10:00:00	PSAVC	A4-105
Tue	11:00:00	DSBM	A4-105
Tue	12:00:00	RP	A4-105
Wed	08:00:00	Lab PBE	C4-S10
Thu	08:00:00	PBE	A4-105
Thu	10:00:00	TD	A4-105
Thu	12:00:00	PSAVC	A4-105
Fri	08:00:00	RP	A4-105
Fri	10:00:00	TD	A4-105
Fri	11:00:00	DSBM	A4-105
Mon	08:00:00	Lab RP	D3-006
Mon	10:00:00	PSAVC	A4-105
Mon	12:00:00	Lab DSBM	D3-006

El servidor retorna els horaris a partir de la següent classe

The screenshot shows a window titled "course_manager.py" with a "Logout" button in the top right corner. A search bar contains the text "tasks". Below it is a table titled "tasks" with the following data:

date	subject	name
2018-09-30	AST	Práctica 1
2018-10-08	PBE	Project Plan
2018-10-10	AST	Práctica 2
2018-10-22	PBE	Requirement Specifications
2018-11-25	PBE	Critical Design Report
2018-12-05	AST	Práctica 3
2018-12-08	AST	Práctica 4
2018-12-13	PBE	Final Report

Els treballs s'ordenen per data d'entrega

The screenshot shows a window titled "course_manager.py" with a "Logout" button in the top right corner. A search bar contains the text "marks". Below it is a table titled "marks" with the following data:

subject	name	mark
AST	control teoria	7.5
AST	Lab1	3.2
AST	Lab2	4.7
AST	final	8.6
PBE	puzzle1	2.8
PBE	puzzle2	8.4
PBE	control	7.3
PBE	CDR	8
PBE	FR	9

Les notes s'ordenen per assignatura

La comunicació amb el servidor es farà a traves del protocol http GET. Aquesta comunicació no es pot fer en el thread principal i s'ha de fer en un thread auxiliar. El servidor enviarà els resultats en format JSON

Quan un usuari entri en sessió, s'engegarà un timer que passat un temps d'inactivitat, sortirà de sessió.

Plantegeu el projecte de forma gradual, de més simple amb funcionalitat bàsica, a més detallat. Experimenteu amb l'estilitzat CSS

4.1 El Servidor

El servidor es programarà en PHP o NodeJS. La Base de Dades serà MySQL i opcionalment per els que programin en NodeJS podran triar MongoDB. La idea és programar servidors d'aplicacions http simples sense usar biblioteques/frameworks.

La idea bàsica consisteix en traduir *query* en sentències SQL—per MySQL— o objectes Javascript—per MongoDB—. El més pràctic és convertir *query* en un objecte. Pot ajudar la funció `parse_str` en PHP o be parsejar l'string directament. En NodeJS les biblioteques no són tan complertes com en PHP. Es poden trobar còpies de `parse_str` per NodeJS i també formatejadors de `date`—busqueu la biblioteca `date format`—

Per PHP es pot fer servir el servidor de desenvolupament `php -S`. No cal que estigui integrat en un servidor http genèric tipus Apache

Per executar servidor i Base de Dates hi han tres opcions:

- instal·lar PHP/NodeJS i MySQL/MongoDB en un ordinador propi. L'avantatge és l'aprenentatge associat. Precisa d'un treball de configuració
- fer servir el servei de màquines virtuals de l'ETSETB Ravada. Connecteu-vos a <https://rvd1.upc.edu>. La màquina virtual és pbe-telem-QXXXX on Q és el quadrimestre (P o T) i XXXX és l'any (2019, 2020, etc). Haureu de configurar PHP/NodeJS i MySQL/MongoDB.

Funciona un servei de proxy per accedir als serveis configurats. P.e. si s'arrenca el servidor `php` pel port 55555, el script és `course_manager.php` i el path, `/timetables?day=Mon`, amb la comanda de terminal `php -S pbe-telem-T2019-josep-sanchez.local:55555`, la url d'accés serà: `http://rvd1.upc.edu/proxy/pbe-telem-T2019-josep-sanchez:55555/course_manager.php/timetables?day=Mon`

Si s'arrenca el servidor `node` pel port 44444, i el path és `/tasks?subject=PBE`, amb la comanda de terminal `node course_manager.js 44444`, la url d'accés serà: `http://rvd1.upc.edu/proxy/pbe-telem-T2019-josep-sanchez:44444/tasks?subject=PBE`

- fer servir serveis de *allotjament—hosting*— tant pel servidor com per la base de dades. S'en poden trobar de gratuïts. Consisteix en registrar-se i configurar el compte i la Base de Dades. La desavantatge és que algunes característiques del llenguatge i/o Base de Dades poden estar limitades o no ser gratuïtes

Totes tres són interessants de provar. Si m'hagués de decantar per una triaria la primera—s'aprèn més—

4.2 Clients Web/Adroid

Programeu un client Web/Android amb la mateixa funcionalitat que el client RPi. L'autentificació es farà p.e. amb `login/password` el `name/student_id` de la taula `students`.

El client Android s'ha de programar amb **Android Studio**. Per el client Web no s'ha de fer servir cap framework. La idea és la mateixa que en el CDR: entendre el funcionament bàsic d'una aplicació web a partir únicament de la biblioteca estàndard en runtime.

El client Web es pot programar, apart de HTML i CSS, amb Javascript i tècniques AJAX—`XMLHttpRequest`— en el client o be amb PHP en el servidor. Convé fer servir tècniques d'implementació que separin al màxim la presentació—HTML/CSS— del codi—Javascript/PHP—.

Algunes pantalles del client web:



Algunes pantalles del client android:

Welcome to Course Manager

host:port/path

username

uid password

LOGIN

Welcome: [Francesc Oller](#)

LOGOUT

[timetables](#)

SEND

timetables

day	hour	subject	room
Mon	08:00:00	Lab RP	D3-006
Mon	10:00:00	PSAVC	A4-105
Mon	12:00:00	Lab DSBM	D3-006
Tue	08:00:00	TD	A4-105
Tue	10:00:00	PSAVC	A4-105
Tue	11:00:00	DSBM	A4-105
Tue	12:00:00	RP	A4-105
Wed	08:00:00	Lab PBE	C4-S10
Thu	08:00:00	PBE	A4-105
Thu	10:00:00	TD	A4-105
Thu	12:00:00	PSAVC	A4-105
Fri	08:00:00	RP	A4-105