2021

Auto-Biblio 2.0

MEMORIA

HICHAM NAF I SERGI VILLENA

FUNDACIÓ EDUARD SOLER | AVINGUDA EDUARD SOLER, 1, 17500 RIPOLL, GIRONA

Contingut

Auto-Biblio 2.0

Aquesta és la continuació del projecte Auto-Biblio, l'objectiu és continuar on ho vam deixar l'any anterior i aconseguir fer-lo funcional.

Introducció:

L'objectiu principal del nostre projecte és automatitzar la biblioteca de la Fundació Eduard Soler per tal de poder fer un préstec sense haver de disposar de personal.

Per disposar d'un control absolut dels llibres que entren i surten del recinte utilitzem:

- una **Raspberry Pi 4**, un ordenador en miniatura on hi ha la base de dades i els codis que fan possible la seva operació
- la tecnologia d'identificació **RFID**
- un lector de codis 1D (codis de barres convencionals) i 2D(codis QR)
- una pantalla tàctil per a interactuar amb l'usuari.

La base de dades que utilitzem és MySQL, on hi han les taules que porten a terme els registres. En el nostre projecte utilitzem 3 taules:

- La primera anomenada **Usuaris**, on registrem el codi QR de la targeta dels alumnes, els seus noms, els correus i un número d'identificació.
- La segona taula s'anomena **Llibres**, es registra el nom del llibre, el seu codi de barres, el codi RFID i un número d'identificació.
- L'última taula anomenada **Registres** on s'introdueix la ID de l'usuari i la ID del llibre que s'ha registrat prèviament i la data de quan s'ha registrat.

Per a fer els programes utilitzem el llenguatge **Python**, que és el més utilitzat en **Raspberry**.

En la pantalla tàctil es mostra una Interfície Gràfica, intentem que sigui amigable per a l'usuari, per tant la fem el màxim de senzilla possible.

Objectius Actuals:

Aquest any ens hem proposat 3 objectius:

Hardware

El primer objectiu és trobar una antena adequada per fer el sistema de seguretat que servirà per anar detectant els llibres que surten de la biblioteca. Una càmera de seguretat que ens permet fotografiar si surt un llibre sense registrar. I un lector de codi de barres per poder escanejar els llibres i les targetes dels alumnes.

Software

Una vegada aconseguit el material, millorar la Interfície Gràfica d'Usuari, incorporar el lector de barres, crear un programa de seguretat (que és l'encarregat de controlarà els llibres que surten i entren a la biblioteca) juntament amb l'antena **RFID** i la càmera de seguretat, i crear un programa que avisi als usuaris per correu electrònic quan s'acosta el temps límit de retorn.

També crearem un programa que serveixi per registrar nous usuaris i llibres i esborrar si no han d'estar més a la base de dades. Aquest programa ha de ser senzill d'utilitzar.

Instalació

Finalment portar a terme la instal·lació dels equips a la biblioteca. Hem de tenir en compte on anirà col·locada la **Raspberry**, l'antena **RFID**, la càmera de seguretat i els llibres.

Caractéristiques etiquetes i Antena RFID:

L'antena ha de poder detectar les etiquetes que aniran enganxades als llibres que passin per la porta de la biblioteca, això vol dir que ha de poder detectar una etiqueta a un mínim de 3 metres.

El mercat ens ofereix una gran varietat de etiquetes, cada una d'elles treballa a una freqüència diferent. Les podem agrupar en tres grups de freqüències:

- **Low Frequency(LF)**: Treballen a una freqüència d'entre 30 kHz a 300 kHz, ens proporciona una distancia de lectura de 10cm i té una velocitat de lectura menor a les etiquetes HF, però no són sensibles a les ones de radiofreqüència.
- **High Frequency(HF)**: Treballen a una freqüència d'entre 3 a 30 MHz, ens proporcionen una distancia de lectura d'entre 10 cm a 1 m.
- **Ultra High Frequency(UHF)**: Treballen entre 300 MHz a 3 GHz, ens proporciona una distancia de lectura fins a 12 m, té la velocitat de transferència de dades més elevada, però també és la més sensible a les interferències.

Degut a les exigències del nostre projecte ens hem decantat per utilitzar la **freqüència UHF** ja que ens permeten un distancia de lectura òptim de lectura i són les que s'ajusten més al nostre pressupost.

Però aquesta freqüència té la desavantatge de ser molt sensible a les interferències, és per això que treballa a un sota un estàndard, per no interferir amb altres tipus d'ona, com les de la ràdio. A part els cossos sòlids o masses líquides (com pot ser el nostre cos) poden impedir la lectura de la etiqueta.

Una solució que evita aquests inconvenients és la **freqüència HF**, però té una distancia de treball molt curta, aquesta distància és directament proporcional amb la mida de l'antena que utilitza (principalment el bobinat) i el cost del sistema, com més gran sigui el bobinat, més distància pot detectar, però és més car.

Per poder posar-la al costat de la porta hauríem de fer un bobinat que tingui les mateixes dimensions que la distància, és a dir que hauria de fer un metre i mig d'alçada aproximadament, com les barreres que podem trobar en els supermercats. Aquestes barreres tenen un cost molt elevat, per això descartem l'ús de la **freqüència HF**.

1Antena RFID

Trobem una antena que compleix aquesta condició, pot treballar en 5 freqüències diferents, entre 860-928 MHz.

Region
China 2 920.125~924.875MHz
China 1 840.125~844.875MHz
FCC 902~928MHz
Europe 865.1-867.9MHz

2 Normativa

A Europa hem de treballar amb les freqüències 865,1 MHz i 867,9 MHz per normativa i evitar problemes d'interferències amb altres tipus d'ones.

Té un distancia entre 4 i 15 metres, es pot modificar la distancia modificant la potència de l'antena, els dB.

Lector de codis de barres:

Per adquirir les dades dels usuaris i la referencia dels llibres utilitzem un lector de codis 1D i 2D

El lector ha de ser capaç de llegir els codis de barres dels llibres i els codis QR de les targetes dels alumnes.

3 Lector codis 1D i 2D

No ha set difícil trobar un lector que compleixi amb aquestes característiques, l'hem trobat fàcil a més té connector USB i les dades les transmet com un teclat.

Càmera de seguretat:

La càmera haurà d'estar posicionada en un lloc on la foto que faci es vegi la cara de la persona, és a dir que haurà d'estar al costat de la porta, depenent d'on la posem pot estar lluny del nostre dispositiu central, la **Raspberry** per aquest motiu hem decidit comprar una antena que vagi connectada a la wifi i podem agafar la imatge connectant-nos a la càmera, així no ens hem de preocupar per la distància del cablejat.

Amb aquest tipus de càmera ens hem trobat amb un problema que és que no podem agafar la imatge directament de la càmera per motius de seguretat, ja que si algú és connectes a la teva wifi podria gravar el que veu la càmera sense cap problema, per aquest motiu les càmeres que funcionen per wifi venen amb una aplicació, el que fan és enviar la imatge als servidors de la companyia i després te la torna a enviar a l'aplicació, no podem agafar la imatge directament connectant-nos a la IP de la càmera.

Programes:

Programa 1- Interfície Gràfica

Hem creat una Interfície Gràfica utilitzant ***TKinter**, aquest programa és la que es mostrarà en la pantalla de la **Raspberry**. Volem que la interfície sigui amigable per al usuari, així que l'hem creat el màxim de senzilla possible, per a no generar confusió.

Aquesta interfície conté:

- Dos botons
    - **Agafar Llibre**: per registrar els llibres a la base de dades quan algun usuari se'ls vulgui endur.
    - **Tornar Llibre**: per tornar els llibres a la biblioteca borant el registre de la base de dades.
- Un cartell central on es mostra el procés i els passos a seguir.

El primer botó serveix per registrar els llibres a la base de dades, quan un usuari prem aquest botó, la primera instrucció que ens apareix per la pantalla és escanejar la targeta de l'usuari, això ho fem amb el lector de barres, en aquest cas escanegem el codi QR de la targeta, seguidament el programa va a buscar a la base de dades el codi introduït per identificar la persona, si no el troba (com per exemple: no està registrat el codi o s'ha escanejat malament), per pantalla es mostra un missatge dient que hi ha un error i tornarà a l'inici.

Si el codi llegit es troba a la base de dades, la següent operació és escanejar el codi de barres del llibre, llavors el programa genera una columna amb la ID del llibre i la ID de l'usuari.

Quan el programa ens demani escanejar hi hem ficat un temps d'espera de 30 segons, si per exemple l'usuari marxa deixant-lo en espera, transcorreguts aquests segons torna directament a la pantalla d'inici, volíem ficar un botó per cancel·lar

l'operació, però era molt complicat, per aquest motiu hi hem incorporat un límit de temps en el seu lloc.

L'altre botó que ens serveix per a tornar el llibre és molt senzill, només s'ha d'escanejar el llibre que es vulgui tornar i s'esborrarà de la base de dades.

*Més informació -Annex1

Programa 2 Seguretat.

Aquest programa s'encarrega de llegir els llibres que surten per la biblioteca, l'antena està llegint les ID's de les etiquetes constantment, si detecta que un llibre no està registrat, la càmera de seguretat fa una fotografia i la guarda en una carpeta a la Raspberry, el nom de la fotografia té la data en què s'ha fet la fotografia amb l'hora i el nom del llibre que ha detectat que no està registrat.

També envia un correu electrònic al responsable de la biblioteca avisant-lo de quin llibre ha sortit i quan. El responsable si haurà de connectar remotament a la Raspberry i mirar la imatge que s'ha guardat.

Anex 2

Programa 3 Correu.

El programa serveix per avisar la persona que s'ha endut el llibre que s'acosta el temps límit que s'ha posat per a retornar el llibre, el programa envia un correu electrònic per informar-los. També quan se supera aquest temps límit si no s'ha retornat el llibre, envia un correu al responsable de la biblioteca avisant-lo.

Anex3

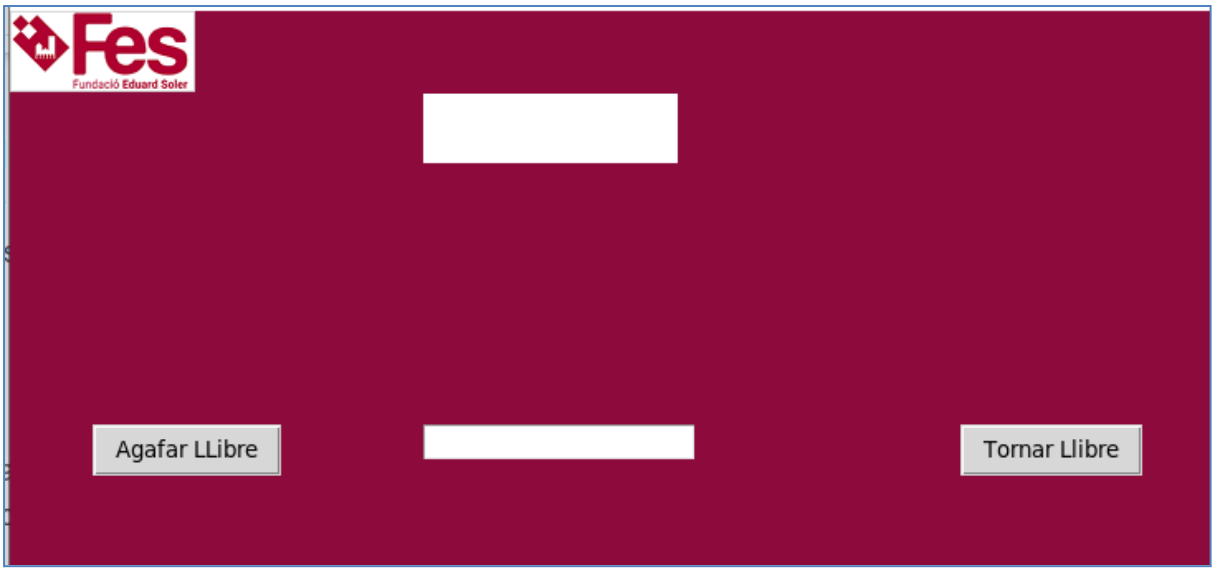
Conclusions

El següent pas és trobar una antena que detecti les etiquetes, les condicions són que treballa a la mateixa freqüència, que detecta a una distància de 3 metres i que no tingui un cost elevat.Té un preu de 128,8 € que és un preu assequible en comparació al sistema HF, estem parlant d'un preu superior als 1500 €,

El programa Python ens permet fer qualsevol programa i funciona en qu

Annex

Anex 1:



Hem modificat la pantalla i en lloc d'utilitzar un scrolledtext per mostrar les instruccions, hem utilitzat un label, utilitzant un label no tenim el problema que ens passava amb el **scrolledtext** on havíem de sortir de la funció perquè ens aparegui el text.

Ara guardem el text en una variable i es mostra per pantalla a l'instant quan s'executa.

```
my_string_var = tk.StringVar(value="                ")
label = Label(root, textvariable=my_string_var)
label.place(x=250, y=50)
label.config(fg="black",     # Foreground
             bg="white",    # Background
             font=("Verdana",10))
```

Així és com definim el label, la variable **my_string_var** és on hi posem el text que volem que es mostri.

Exemple:

Quan premem el botó **Agafar Llibre**, modifiquem la variable **my_string_var** perquè ens mostri la primera instrucció i el temps que ens queda.

```
def clicked():
    global var
    global Codi
    global Codii
    global IDSQL
    global IDU
    global IDL
    global RFID
    global my_string_var
    global x
    if var==0:
        entry.delete(0, tk.END)
        var=1
        root.after(1000,clicked)
    elif var==1:
        my_string_var.set("Introdueix Targeta, temps" + str(x) + "segons")
        entry.focus()
        Codi = entry.get()
        Codii = Codi.upper()
        print(Codi)
        if Codii=="":
            x=x-1
            print(x)
            if x==0:
                var=0
                x=30
            else:
                root.after(1000,clicked)
        else:
            var = 2
            x=30
            entry.delete(0, tk.END)
            root.after(1000,clicked)
```

Abans utilitzavem el lector **RFID-RC522** per interactuar amb la **GUI**, ara que tenim el lector de codis hem modificat el programa, però ens hem trobat amb un problema que és quan utilitzem **TKinter** no podem escriure en variables, per això hem d'utilitzar un **entry**, aquí és on escrivim i cada cop que volem guardar el text que escrivim en una variable li hem de posar “entry.get()” d'aquesta manera agafem el text de l'**entry**.

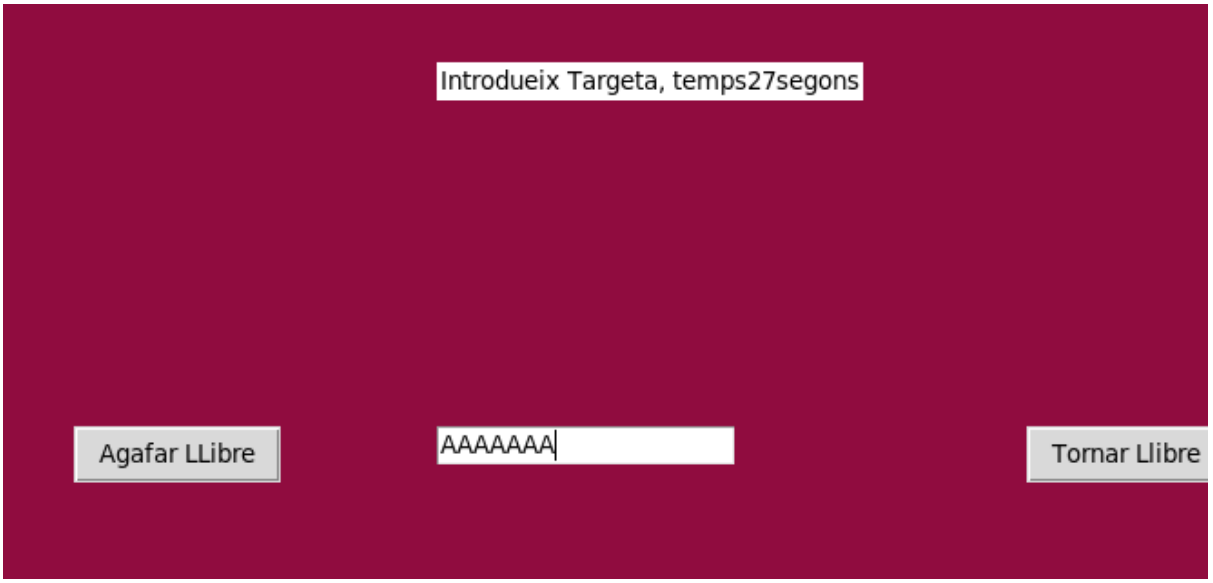
```
elif var==1:
    #Aqui esborrem el text de la pantalla i escrivim la seguent linea
    txt2.delete('0.0', END)
    txt2.insert(INSERT, "Introdueix el codi del llibre despres d'apret
    #Activem el lector RFID
    id, text = reader.read()
    #Aqui li fem la comanda d'anar a buscar el nom del propietari de l
    cursor.execute("SELECT name FROM users WHERE rfid_uid="+str(id))
    #Ara guardem el resultat en una variable, la guarda en una tupla
    #i la hem de pasar a string per poder-la utilitzar més endevant
    Persona = cursor.fetchone()
    Persona = Persona[0]
```

Definim l'entry de la següent manera:

```
entry = ttk.Entry(root)
entry.place(x=250, y=250)
```

En l'exemple anterior quan premem el botó **Agafar Llibre**, esborrem el text que pugui haver-hi a l'**entry** amb "entry.delete()" i llavors a la variable "Codi" hi guardem el text amb "entry.get()", metres la variable estigui buida s'anirà descomptant el temps i quan arribi a 0 anirà a inic, però si encara no s'ha acabat el temps i hi hem introduït text a la variable llavors passem a la següent part que és buscar el **CodiQR** a la base de dades.

El programa segueix fent el mateix que el que vam descriure a l'anterior memòria el únic que hi hem afegit aquestes dues funcions i les hem adaptat.


Introdueix Targeta, temps27segons
Agafar LLibre
AAAAAAA
Tornar Llibre

Anex2:

En el programa de seguretat utilitzem l'antena per a llegir les ID's de les etiquetes, per enviar i rebre dades de l'antena el connectem a l'USB de la **Raspberry** i utilitzem **comunicació serie**.

Per això hem importat el mòdul serie i establir la comunicació amb el port.

La informació de la comunicació la guardem a la variable "ser".

Amb aquesta variable enviem i revem dades, utilitzem "ser.write" per enviar i "ser.read" per llegir.

```
import serial
import codecs
import binascii
import mysql.connector
#outdent
import os
import pygame, sys

from pygame.locals import *
import pygame.camera

from datetime import *
from datetime import datetime

import calendar
import time


db = mysql.connector.connect(
    host="localhost",
    user="AutoBiblio",
    passwd="admin",
    database="AutoBiblio")

cursor = db.cursor()

ser = serial.Serial(port='/dev/serial/by-id/usb-Prolific_Technology_Inc._USB-Serial_Controller-if00-port0', baudrate=115200, timeout=0.1)
```

Llavors li enviem un codi que serveix per dir-li a l'antena que llegeixi les ID's de les etiquetes i ens les passi pel port.

```
while True:
    ser.write(serial.to_bytes([0xAA,0xAA,0xFF,0x08,0xC1,0x02,0x05,0x00,0xBC,0xA9,0x24]))
```

Aquest codi el trobem a la documentació de l'antena, el protocol de comunicació que segueixen l'antena i les etiquetes és la **ISO 18000-6C**.

Els codis estan en **Hexadecimal** i els hem de transformar en **binaris** per poder-los enviar.

Els nostres missatges han de seguir la següent estructura per poder establir una comunicació del **Host**(En el nostre cas la Raspberry) i el **Reader**(L'antena RFID):

- Command frame: Host → Reader

FH	FH	Address	Length	Command	Command	Parameter	Check	Check
0xAA	0xAA	RA	LEN	CMDH	CMDL	Parameter(0~32)	CRCH	CRCL

UHF Reader & Module Communication Protocol V3.0 Pag 3

Missatge:

AA AA FF 08 C1 02 05 00 BC A9 24

Els primers dos bytes, són per dir-li al lector que s'està enviant un missatge i que estigui atent.

El següent byte, és l'adreça de l'antena a qui li dirigim el missatge, aquí posem FF que equival a 255 i s'utilitza com a broadcast, és a dir que el missatge va a totes les antenes connectades, en el nostre cas només en tenim una.

El 4t byte indica la llargada del missatge contant-se a si mateix, és a dir que en aquest cas que tenim 7 bytes després del LEN, contant el LEN el seu valor és 08.

Els següents 5 bytes son la comanda, podem escriure dues comandes, el primer és el CMDH que és la comanda principal i la següent és el CMDL per si volem fer una subcomanda.

El primer byte de la comanda és per dir-li que llegeixi múltiples etiquetes.

26	Start multi-Tags inventory	0xC1	CMD_MULTI_ID	Start multiple Tags inventory by multiple antennas

UHF Reader & Module Communication Protocol V3.0 Pag 4

El següent byte li diem quina informació volem extreure dels tacs, el que demanem són les ID's de les etiquetes.

EPC: The tag's EPC ID data is stored on the EPC MemBank with the starting address of 0x02, and its length is defined by the TAG protocol control character (PC).

UHF Reader & Module Communication Protocol V3.0 Pag 22

I l'últim valor és per dir-li quantes etiquetes màximes volem emmagatzemar en una sola lectura.

Els últims dos bytes de la comanda és per indicar en quins inventaris volem guardar les dades, ho deixem com ens ve per defecte.

Start multiple Tags and multiple antennas inventory, inventory Number Range of value 0-65535.

Main	Sub	Parameter		
CMDH	CMDL	Parameter		Functional Specifications
0xC1	0x00	QV(1B)	InvNumber (2B)	Algorithm 0
0xC1	0x01			Algorithm 1
0xC1	0x02			Algorithm 2

- QV: Q Value. The scope of the amount of inventory to be labelled is defined: 2^Q-1 tags. The proposed Q value (2^Q-1 tags) is more than 20% larger than the actual number of tags. But the greater the Q value, the longer the cycle required to inventory the tags, so the setting of the Q value should be considered.
- InvNumber: Inventory Number (times), Range of value 0-65535.
  (1) InvNumber =0, Limitless inventory tags;

UHF Reader & Module Communication Protocol V3.0 Pag 21

I els últims dos bytes són pel CRC, que és un control d'errors. Aquests dos valors els obtenim amb el càlcul **CRC-16**, pero la documentació ja ens don tot el missatge i no ho hem de calcular, quan el **Reader** reb aquest missatge fa el calcul i li ha de donar el mateix resultat, si li dona diferent vol dir que s'ha produit algun error.

Despres d'enviar la trama utilitzem "ser.read" per llegir, no revem tota la trama de cop sino que les anem revent byte per byte cada cop que utilitzem "ser.read".

Exemple de la resposta:

>Reader: AA AA FF 18 C1 00 00 BB 30 00 E2 00 41 06 22 18 00 64 19 80 47 1E 21 3D 00 BD 83

UHF Reader & Module Communication Protocol V3.0 Pag 23

La part del codi en verd és la ID de l'etiqueta i és la que ens interesa.

Hem fet un loop on continuament estem llegint lo que ens envia la antena i hu guardem en la variable "x".

Sabem que les nostres etiquetes comencen per **E2** i tenen una llargada de **12 bytes**, així que hem fet un **if** on si **x** val **E2**, el guardem amb els proxims 11 bytes en una variable.

```
        if x == b'\xe2':
            for i in range(12):
                TacID = TacID + x
                x=ser.read()
            hex_b = binascii.hexlify(TacID)
            hex_s = hex_b.decode("ascii")
            ID.append(hex_s)
            sql = "SELECT IDLlibre, Llibre, Agafat FROM Llibres WHERE CodiRFID= %s"
            adr = (hex_s, )
            cursor.execute(sql, adr)
            Resultat = cursor.fetchone()
            TacID = (b'')
```

Aquest missatge ens el torna ens el torna en format binari, hem importat una llibreria que ens permet traduir aquest missatge a **ASCII** i el guardem en la variable hex_s

Missatge en binari:

```
b'\xe2\x00\x00\x1b"\x0f\x02\x83\x11p\xa3\xdb'
```

Missatge en ASCII:

```
'e200001b220f02831170a3db',
```

El següent pas que fem és mirar les ID's que l'antena va detectant si estan a la taula de registres.

Per fer-ho el que fem és primer buscar aquesta ID a la taula de llibres per comprovar que està registrada, això ho fem perque ens hem trobat que de vegades detecta malament l'etiqueta i despres tenim una ID diferent. Per això comprovem primer si esta a la base de dades i la identifiquem, si esta a la taula de Llibres, agafem el numero ID del Llibre i mirem si està a la taula de Registres.

Si està registrada, fem una fotografia i també enviem un correu al responsable de la biblioteca.

Per fer la foto ho fem important una llibreria que ens permet capturar la imatge de la camara USB i guardar-la en la carpeta que li definim amb el nom que volem.

```
if Resultat != None:
    IDL = Resultat[0]
    NomL = str(Resultat[1])
    Ag = int(Resultat[2])
    sql = "SELECT IDRegistre FROM Registre WHERE IDLlibre= %s"
    adr = (IDL, )
    IDReg = cursor.fetchone()
    TacID = (b'')
    db.commit()
    if IDReg == None and Ag == 0:
        pygame.init()
        pygame.camera.init()
        cam = pygame.camera.Camera("/dev/video0",(640,480))
        cam.start()
        image = cam.get_image()
        cam.stop()
        now = datetime.now()
        date_time = now.strftime("%m-%d-%Y,%H:%M:%S")
        Nom = "NomLLibre:" + str(NomL) + "Data:" + date_time + '.jpg'
        pygame.image.save(image, Nom)
        Ag = 1
        sql_insert = "UPDATE Llibres SET Agafat = %s WHERE IDLlibre=%s"
        cursor.execute(sql_insert, (Ag, IDL))
        Correu(NomL, date_time)
        db.commit()
```

El correu ho enviem a traves d'una llibreria que ens permet enviar missatges per Gmail, necesitem una conta de Gmail que utilitzarem per enviar els correus, llavors escrivim el missatge i el correu del destinatari.

```
def Correu(NOM, DATA):
    email = 'Correu'
    password = 'Contrasenya'
    send_to_email = "CorreudelResponsable"
    subject = 'Control seguritat Biblioteca' # The subject line
    message = "El Llibre: " + str(NOM) + "ha sortit de la biblioteca sense ser registrat a la data: " + DATA

    msg = MIMEMultipart()
    msg['From'] = email
    msg['To'] = send_to_email
    msg['Subject'] = subject

    # Attach the message to the MIMEMultipart object
    msg.attach(MIMEText(message, 'plain'))

    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.starttls()
    server.login(email, password)
    text = msg.as_string() # You now need to convert the MIMEMultipart object to a string to send
    server.sendmail(email, send_to_email, text)
    server.quit()
```

L'antena esta sempre llegint, i fa moltes lectures per segon així que les etiquetes les pot detectar multiples vegades i ens pot pasar que faigui moltes fotos quan amb una sola en fem. Per evitar aquest problema el que hem fet és afegir una columna a la taula Llibres anomenada "Agafa" que pot tindre dos valors, 0 o 1. Totes les files

valen 0, quan s'ha detectat un Llibre que no està registrat, fa la fotografia i li posa un 1 al Llibre que li ha fet la foto.

Per tornar-lo a posar a 0 hem d'anar a la pantalla de la Raspberry, apretar el botó de Deixar Llibre i escanejar el Codi de Barres del llibre.

Anex 3:

Hem decidit que els llibres que surten del recinte no serà indefinidament sinó que s'hauran de tornar abans d'un tems determinat. El tems es pot variar, pot ser de dies, setmanes, mesos...

De moment hem fet que el llibre el podràs disposar del llibre un màxim d'un mes.

Quan algun llibre li falti una setmana per fer un mes, automàticament s'enviarà un correu a la persona que tingui el llibre avisant-li de que li falta una setmana per poder tornar el llibre.

A la taula de registres utilitzem la columna creat per saber a qui li hem d'enviar un correu.

Per fer-ho hem utilitzat una variable que te guardada la data de fa 3 setmanes, aquesta data la obtenim utilitzant la data actual restant-li 3 setmanes, lo que fem és buscar a la columna creat aquesta data, si hi ha una coincidència vol dir que han passat 3 setmanes des de que s'ha registrat el llibre.

Hem començat creant un nou codi de Python i importem les llibreries necessàries.

```
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
import mysql.connector
from datetime import *
from dateutil.relativedelta import *
import calendar
import time
```

Les 3 primeres línies son per poder enviar correus per Gmail, llavors la connexió al Mysql i finalment unes llibreries que ens serveixen per calcular les dates i un calendari per saber la data actual.

La llibreria **dateutil.relativedelta** no be inclosa a Python sinó que s'ha d'instal·lar.

Per instal·lar la, hem anat a la consola i hem introduït la següent comanda:

```
pip install python-dateutil
```

```
pi@raspberrypi:~ $ pip install python-dateutil
```

Aquesta llibreria ens permet fer càlculs exactes de dates, restar i/o sumar minuts, hores, dies, mesos, anys…

```
db = mysql.connector.connect(
  host="localhost",
  user="Admin",
  passwd="Admin",
  database="Biblioteca"
)

cursor = db.cursor()
```

Després d'importar les llibreries, hem creat la connexió a la base de dades.

```
TODAY = date.today()
Data = TODAY+relativedelta(weeks=-3)
```

Hem creat dos variables, ("**TODAY**") i ("**Data**"), ("**TODAY**") hi guardarem la data actual, ho fem amb la línia "date.today()" que el que fa és anar al calendari i extreure la data actual.

La dada ho guarda en format "string", primer l'any després el mes i finalment el dia separats per "-"

exemple: "2020-05-16"

A la segona variable és on hi guardem la data 3 setmanes anterior a l'actual.

Ho fem agafant la variable ("**TODAY**") més la funció **relativedelta()**, aquesta funció, dintre hi posem lo que volem fer amb la variable ("**TODAY**"), el que hem fet és restar-li 3 setmanes.

El resultat de la operació es guarda a la variable ("**Data**").

Ara que tenim la variable ("**Data**"), hem de buscar a la taula de registres si hi ha alguna semblança.

```
cursor.execute("SELECT IDUsuari FROM Registre WHERE DATE(creat) LIKE %s", (Data, ))
Result = cursor.fetchone()
```

Ara busquem les id dels usuaris que tinguin la data de "creat" igual que la variable ("**Data**") .

Aquí quan busquem la data, en la base de dades està en format "timestamp", és a dir que guarda la data més el temps, en el nostre cas només necessitem la data per això quan fem la ordre en lloc de posar "WHERE creat" que ens agafaria en format "timestamp", posem "WHERE DATE(creat)" així li especifiquem que nomes ens agafi la data.

En aquest cas pot trobar més d'una semblança si més d'una persona registre un llibre el mateix dia.

```
while Result != None:
    Uid.append(Result)
    Result = cursor.fetchone()
    time.sleep(1)

time.sleep(1)
```

Per guardar tots els resultats, les hem de guardar una per una, és a dir que quan en una variable com en el nostre cas ("**Result**") li guardem el valor del cursor, guarda el primer resultat, després de guardar-la en la variable, el cursor tindrà el segon resultat i així successivament.

Així que utilitzem un “**while**“ que el que fa és acumular els resultats en una variable anomenada (“**Uid**”) tipus “tuple” que hem creat prèviament, fins que la variable (“**Result**”) estigui buida.

```
for x in Uid:
    time.sleep(1)
    sql = "SELECT Correu FROM Usuaris WHERE IDUsuari= %s"
    adr = (x, )
    cursor.execute(sql, adr)
    Result = cursor.fetchone()
    Co.append(Result)

time.sleep(1)
db.commit()
```

Ara que tenim les id, anem a la taula d’usuaris i busquem els seus correus, en aquest cas utilitzem un **for** que recorre tots els resultats que hi ha a la variable (“**Uid**”), que és la que hem utilitzat per acumular les id’s dels usuaris.

El que fem en aquest tros de codi és que quan entrem al **for**, és anar guardant els valors que hi ha a (“**Uid**”) a (“**x**”), el **for** s’anirà repetint fins que les dades de (“**Uid**”) s’acabin, I el primer cicle del **for**, (“**x**”) tindrà el primer valor de (“**Uid**”), quan s’acabi tindrà el següent i així successivament.

Dintre el **for** acumulem els correus en una nova variable, en aquest cas hem de tindre en compte que a l’hora de crear la variable, ho hem de fer fora del **for**, si la creem a dintre, cada vegada que el **for** acaba i es repeteix, sobreescriu la variable i no acumularà.

```
Co = ()
Uid = ()
Uid2 = ()

db = mysql.connector.connect(
    host="localhost",
    user="AutoBiblio",
    passwd="admin",
    database="AutoBiblio")
```

Ara que tenim els correus guardats, el següent pas és enviar els missatges,

```
for x in Co:
    email = 'correu'
    password = 'contrasenya'
    send_to_email = x
    subject = 'Biblioteca' # The subject line
    message = 'Has de tornar el llibre'
```

Buscant informació per internet, ens hem trobat un codi que ens serveix per enviar correus a Gmail, les primeres dos línies ens demana la conta del correu i la contrasenya, aquesta conta la utilitzarem per enviar els correus, la tercera línia ens demana a qui li volem enviar el correu, aquí hem posat tot el codi dintre d'un for que ens anirà extraient els correus de la "tuple" ("**Co**") i posant-los a la variable ("**x**") que seran els destinataris, les dos últimes línies ens demana l'assumpte del missatge i el missatge que volem enviar.

```
    msg = MIMEMultipart()
    msg['From'] = email
    msg['To'] = send_to_email
    msg['Subject'] = subject

    # Attach the message to the MIMEMultipart object
    msg.attach(MIMEText(message, 'plain'))

    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.starttls()
    server.login(email, password)
    text = msg.as_string()
    # You now need to convert the MIMEMultipart object to a string to send
    server.sendmail(email, send_to_email, text)
    server.quit()
```

Finalment el codi envia el missatge per un canal de Gmail.

Per poder enviar missatges des d'un compte hem d'activar un paràmetre perquè puguis accedir a la conta des de programes.

Primer hem accedit a questa adreça:

https://myaccount.google.com/lesssecureapps

I aquí hem activat aquesta poció.

Per acabar hem d'enviar un correu a l'encarregat de la biblioteca per si algun usuari no ha retornat el llibre al cap d'un mes.

```
Data2 = TODAY+relativedelta(month=-1)
```

Repetim la primera part de buscar la data que calculem que en aquest cas és restant-li un mes en lloc de 3 setmanas.

```
cursor.execute("SELECT IDUsuari FROM Registre WHERE DATE(creat) LIKE %s", (Data2, ))
Result = cursor.fetchone()

while Result != None:
    Uid2.append(Result)
    Result = cursor.fetchone()
    time.sleep(1)
```

I busquem a la taula si hi ha algun usuari que hagi superat aquest temps. I guardem les seves ID's a la variable ("**Uid2**").

Finalment enviem el correu si troba al menys una persona.

```
if Uid2 != None:
    email = 'Correu'
    password = 'Contrasenya'
    send_to_email = "CorreuEncarregat"
    subject = 'Usuaris que no han retornat els llibre' # The subject line
    message = 'Els seguent usuaris no han retornat els llibres: ' + str(Uid2)

    msg = MIMEMultipart()
    msg['From'] = email
    msg['To'] = send_to_email
    msg['Subject'] = subject

    # Attach the message to the MIMEMultipart object
    msg.attach(MIMEText(message, 'plain'))

    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.starttls()
    server.login(email, password)
    text = msg.as_string() # You now need to convert the MIMEMultipart object to a string to send
    server.sendmail(email, send_to_email, text)
    server.quit()
```

/*

Conclusions:

[(3)]Python ens permet crear programes molt senzills d'una forma ràpida i precisa. Però en canvi a l'hora de crear una interfície gràfica ens ha dificultat la feina, si en comptes de [(3)]Python haguéssim pogut utilitzar el Visual Studio juntament amb el windows forms el resultat hauria pogut ser molt més satisfactori.

La GUI ens ha donat molts problemes, ja que quan volem incorporar depèn de quin algorisme es bloqueja i el punter del programa no avança. Quan volem modificar un text que es mostra per pantalla, aquest no s'actualitza fins que s'ha acabat l'algorisme, per tant no ens mostra els textos d'ajuda per pantalla.

A part de la falta de coneixements de programació, La interfície gràfica Tkinter no ens ha acabat de solucionar tots el problemes ja que normalment s'utilitza per aplicacions molt senzilles. Com és molt complicada de dominar no hem pogut extreure el màxim de rendiment d'aquest element.

L'inconvenient més gran és que no hi ha documentació de Pyhon o Tkinter, la informació d'internet és pràcticament nul·la, per a cada inconvenient que ens trobem hem de fer mans i mànigues per a trobar alguna resposta que s'acosti a la situació que tenim. De vegades quan trobem codis que ens ajuden seguir, però hem de modificar gran part del nostre programa per a poder incorporar la modificació. Així que quan trobàvem una solució ens creava un problema més que havíem de solucionar per falta de coneixements base.

Ens queda pendent portar a terme la instal·lació, el disseny de l'estructura que suporta la nostre pantalla i saber exactament on posar-la.

Polir els codis i fer que s'executin els 3 codis cada cop que s'engega la raspberry per si s'apaga i es torna a engegar, s'executin els programes. Hem trobat diferents maneres de fer-ho i ens queda triar la més adequada per al nostre cas.

Al principi no disposavem dels coneixements ni de la documentació de l'antena RFID, en aquest punt hi hem invertit molt de temps. Un cop obtinguda la documentació (i l'ajuda d'un tècnic), podem avançar. Fins aquest punt el projecte

estava congelat, perqué depeniem de l'antena RFID per avançar. Els paràmetres que ens han dut més problemes són els bauds i el llenguatge de comunicació.

hem arribat fins on els nostres coneixements i capacitats físiques ens han dut

Fins a un cert límit un aficionat pot aprendre a programar però la base i molta experiència no és pot aconseguir nomes buscant per internet, això es lo que hem après.

Python ens permet crear programes molt senzills d'una forma ràpida i precisa Però en canvi a l'hora de crear una interfície gràfica ens ha dificultat la feina, si en comptes de Python haguéssim pogut utilitzar el Visual Studio juntament amb el windows forms el resultat hauria pogut ser molt més satisfactori.

La GUI ens ha donat molts problemes, ja que quan volem incorporar depèn de quin algorisme es bloqueja i el punter del programa no avança. Quan volem modificar un text que es mostra per pantalla, aquest no s'actualitza fins que s'ha acabat l'algorisme, per tant no ens mostra els textos d'ajuda per pantalla.

A part de la falta de coneixements de programació, la interfície gràfica Tkinter no ens ha acabat de solucionar tots el problemes ja que normalment s'utilitza per aplicacions molt senzilles. Com és molt complicada de dominar no hem pogut extreure el màxim de rendiment d'aquest element.

L'inconvenient més gran és que no hi ha documentació de Pyhon o Tkinter, la informació d'internet és pràcticament nul·la, per a cada inconvenient que ens trobem hem de fer mans i mànigues per a trobar alguna resposta que s'acosti a la situació que tenim. De vegades quan trobem codis que ens ajuden seguir hem de modificar gran part del nostre programa per a poder incorporar la solució. I la majoria de modificacions que apliquem ens crea un problema més que hem de solucionar, la majoria de vegades per falta de coneixements basics.

Al principi no disposàvem dels coneixements ni de la documentació de l'antena RFID, en aquest punt hi hem invertit molt de temps. Un cop obtinguda la documentació (i l'ajuda d'un tècnic), podem avançar. Fins aquest punt el projecte estava congelat, perquè depeníem de l'antena RFID per avançar. Els paràmetres que ens han dut més problemes són els bauds i el llenguatge de comunicació.

En el projecte encara queda molta feina a fer, polir els codis i fer que s'executin cada cop que s'engega la Raspberry (en cas de que es reinicií). Hem trobat diferents maneres de fer-ho, les hem aplicat però el resultat no és l'optim.

Ens queda pendent portar a terme la instal·lació, el disseny del suport de la pantalla i saber exactament on posar els diferents elements tenint en compte la distribució i les distàncies.

Com a resum del projecte hem arribat fins on els nostres coneixements i capacitats ens han dut, hem après que els novells poden arribar a un cert límit, el nostre projecte no és pot aconseguir nomes buscant per internet i conceptes bàsics, a partir d'aquest punt ja és requereixen nocions més elevades d'informatica i més experiència.