

Sensor de Huella dactilar controlado por una Raspberry Pi 3+

Federico Lukács Álvarez

25 de junio de 2019

Índice general

Resumen

Prefacio

Tipos de Autentificación biométrica

Autentificación biométrica mediante huella dactilar

Objetivo del proyecto

Metodología

- Modelo conceptual

- Hardware necesario

- Software del proyecto, librerías y programación.

Conclusiones

Bibliografía

Resumen

Este proyecto consiste en usar una Raspberry Pi 3, programarla para que pueda interactuar con sensores, obtener datos y estos enviarlos para su posterior análisis.

En este caso se creará un sistema de control de entrada y salida de trabajadores conectando un sensor de huella a una Raspberry y se enviará una plantilla al correo electrónico del encargado de Recursos Humanos.

Prefacio

Este proyecto consiste en programar los sensores con Python 3 y analizar los datos. Para esto utilizaremos la Raspberry conectada a un fingerprint (sensor de huellas dactilares), un teclado numérico de 3x4 y una pantalla LCD.

El supuesto es una empresa con 20 empleados aproximadamente que fichan al entrar a las nueve de la mañana y fichan a la salida a las seis de la tarde, en un solo turno. El programa automáticamente envía doce del día el fichaje de la entrada al responsable de recursos humanos y a las ocho de la noche envía el resumen de entradas y salidas.

Otro modulo importante es el de la creación de una base de datos con la tupla número de empleado y huella dactilar esta se hará con la Raspberry conectada al fingerprint, se agregarán todos los empleados cuando se instale el dispositivo y posteriormente se ingresarán a la base de datos los empleados que se contraten y se borrarán los que se van de la empresa.

Se programarán tres periféricos con la Raspberry un LCD que muestra la información de entrada, salida y administración, un teclado para que el sistema sepa la acción de entrada o salida y el sensor de huellas.

El sistema se simplificará para permitir llevarlo a cabo, se supondrá que los empleados ficharan correctamente la entrada y la salida, hay un turno y el trabajo es de lunes a viernes, sin embargo, a futuro se pueden hacer controles mas precisos como doble fichaje, replicar las bases de datos de empleados para que fichen en distritos puntos de trabajo, etc.

Tipos de Autenticación biométrica

Los tipos de verificación biométrica son:

1.- Sistemas de Verificación de Voz

En los sistemas de reconocimiento de voz se intenta identificar una serie de sonidos y sus características para decidir si el usuario es quien dice ser.

2.- Sistemas de Verificación de Escritura

La escritura -generalmente la firma- no es una característica estrictamente biométrica, pero suele agrupar dentro de esta categoría, el objetivo aquí no es interpretar o entender lo que el usuario escribe, sino autenticarlo basándose en ciertos rasgos característicos de su escritura.

3.- Sistemas de Verificación de Huellas Dactilares

La huella dactilar de un individuo ha sido un patrón bastante usado desde antes de la existencia de sensores biométricos, para la identificación individual de las personas, es un sistema muy fiable y difícil de engañar.

4.- Sistemas de Verificación de Patrones Oculares

Los modelos de autenticación biométrica basados en patrones oculares se dividen en dos tipos diferentes: los patrones de retinas, y los del iris. Estos modelos se suelen considerar los más seguros por que tienen una fiabilidad muy alta.

5.- Sistemas de Verificación de Geometría de la Mano

Los sistemas de autenticación basados en el análisis de la geometría de la mano son sin duda los más rápidos dentro de los biométricos: con una probabilidad de error aceptable, en aproximadamente un segundo son capaces de determinar si una persona es quien dice ser.

	Voz	Escritura Firma	Huellas Dactilares	Ojo Retina	Ojo Iris	Geometría de Mano
Fiabilidad	alta	alta	alta	muy alta	muy alta	alta
Facilidad de Uso	alta	alta	alta	baja	media	alta
Prevención de Ataques	media	media	alta	muy alta	muy alta	alta
Aceptación	alta	muy alta	media	media	media	alta
Estabilidad	media	media	alta	alta	alta	media
Autenticación	si	si	si	si	si	si
Interferencias	ruidos, resfriados	firmas fáciles	suciedad, heridas	irritación	gafas	artritis, reumatismo

Tabla 1: resumen de tipos de autenticación.

Autenticación biométrica mediante huella dactilar

La identificación basada en huella dactilar es la más antigua de las técnicas biométricas debido a que se considera que las huellas dactilares son únicas e inalterables.

Es el rasgo biométrico más utilizado para autenticación. Se han desarrollado una amplia gama de tecnologías de captura y tiene como ventajas su alta precisión y facilidad de uso.

Existen dos tipos de técnicas para medir las huellas dactilares:

1.- Basadas en minucias:

Esta técnica basa su mecanismo de autenticación en las minucias, es decir, en determinadas formas fácilmente identificables existentes en la huella dactilar. Así, se registra el tipo de minucia y su posición dentro de la huella, estableciendo una serie de mediciones. De esta forma, el modelo o plantilla correspondiente a cada usuario es un esquema en el que se indican las minucias que se han de detectar, su posición y las distancias que separan unas de otras.

2.- Basadas en correlación:

Mediante la utilización de esta técnica se analiza el patrón global seguido por la huella dactilar, es decir, el esquema general del conjunto de la huella en lugar de las minucias. Esta técnica requiere un registro preciso, pero su principal inconveniente es que se ve afectada por la traslación y la rotación de la imagen.



Figura 1: Tipos de técnicas para medir huellas

Objetivo del proyecto

El objetivo del proyecto es utilizar una Raspberry Pi conectada a sensores que envíen información de forma remota ya sea a una nube o correo electrónico, en el caso específico de este proyecto la recolección de datos a través de un sensor de huella dactilar para el control de entrada y salida del lugar de trabajo.

Metodología

Modelo conceptual

El modelo de fichaje se simplificará por el tiempo de proyecto, el empleado al fichar se apunta con la huella, si la huella no la reconoce la pide de nuevo y no ficha, si la reconoce se genera un campo en una plantilla con los siguientes campos número de huella sensor, número empleado, fecha y hora y se guardará en un archivo cvs que se actualizará con cada entrada, el proceso se ve en el diagrama de flujo de la figura 2.

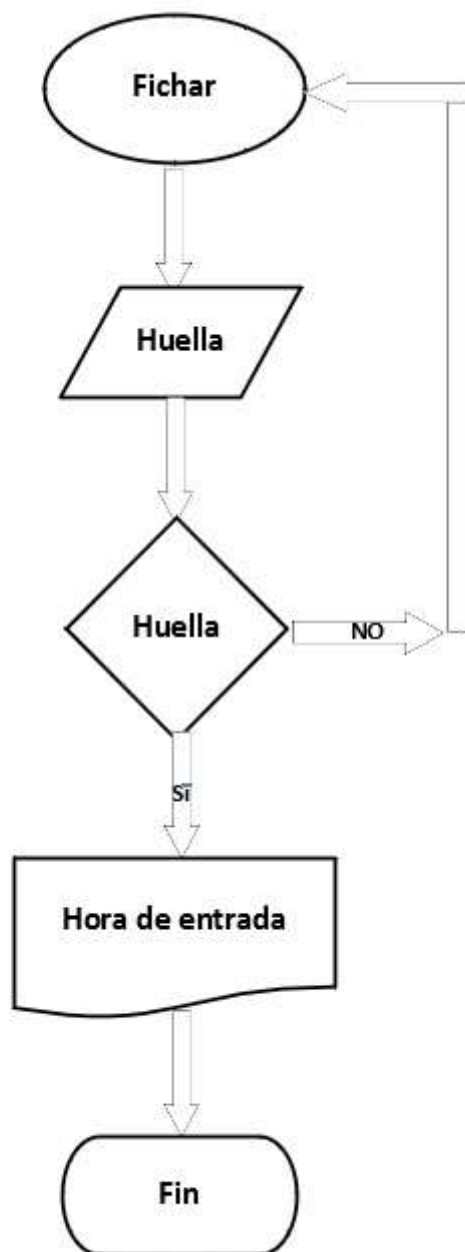


Figura 2: Diagrama de flujo del proceso de fichar

Hardware necesario

El hardware necesario es el siguiente, para realizar este proyecto es el siguiente:

- Raspberry Pi 3 plus
- Pantalla 2C 1602 LCD
- Teclado numérico de 3x4
- Adaptador UART FTDI232 a mini USB

- FingerPrint GT521F52
- Breadboard MB-102
- GPIO T Tipo de Placa de Expansión+

En la siguiente imagen se puede ver todos los dispositivos conectados:

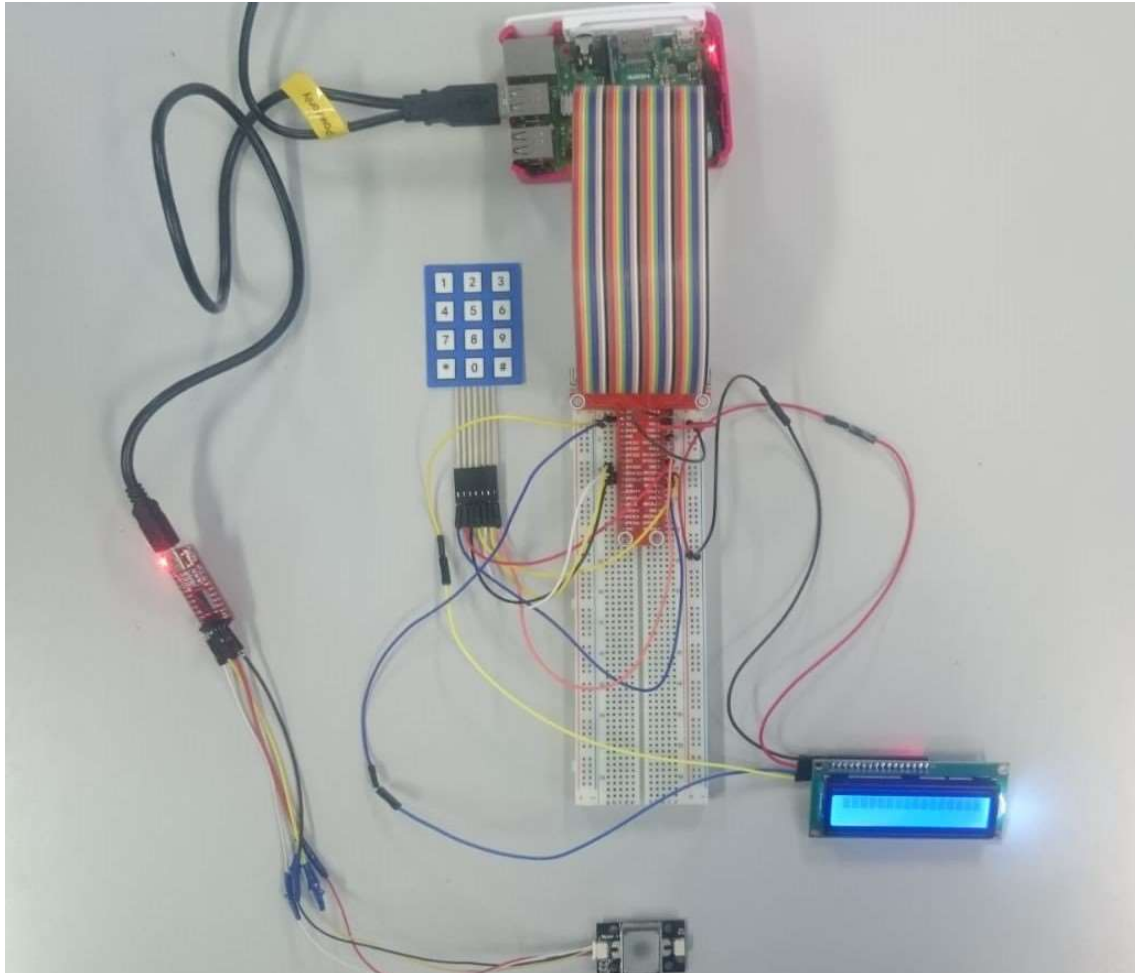


Foto 2: Raspberry Pi con fingerprint.

Software del proyecto, librerías y programación.

Para hacer funcionar el fingerPrint se utilizó una librería que esta alojada en un proyecto GIT, el fabricante del sensor tiene los códigos en C++, y en esta librería están en Python. Para la configuración del keypad se utilizó una librería ya programada de la página Raspberry Pi Tutorials, y el LCD se configuró con la librería de fabricante Osoyoo que venía con Raspberry Pi Starter Kit V1.

El código de funcionamiento del sensor de huellas se detalla a continuación, aún está en proceso de programación porque el sensor no funcionaba.

```
*huella.py - C:\Users\Alumne\Desktop\sensor huellas\Proyecto final\huella.py (3.7.3)*
File Edit Format Run Options Window Help

import time
import GT_521F52
import datetime
import smbus
import pandas as pd
import RPi.GPIO as GPIO
from random import choice
from time import sleep
from keypad import keypad
import os

comando='python3 i2cl602_lcd.py'
os.system(comando)
GPIO.setwarnings(False)

# Define some device parameters
I2C_ADDR = 0x27 # I2C device address, if any error, change this address to 0x3F
LCD_WIDTH = 16 # Maximum characters per line

# Define some device constants
LCD_CHR = 1 # Mode - Sending data
LCD_CMD = 0 # Mode - Sending command

LCD_LINE_1 = 0x80 # LCD RAM address for the 1st line
LCD_LINE_2 = 0xC0 # LCD RAM address for the 2nd line
LCD_LINE_3 = 0x94 # LCD RAM address for the 3rd line
LCD_LINE_4 = 0xD4 # LCD RAM address for the 4th line

LCD_BACKLIGHT = 0x08 # On
#LCD_BACKLIGHT = 0x00 # Off

ENABLE = 0b000000100 # Enable bit

# Timing constants
E_PULSE = 0.0005
E_DELAY = 0.0005

#Open I2C interface
#bus = smbus.SMBus(0) # Rev 1 Pi uses 0
bus = smbus.SMBus(1) # Rev 2 Pi uses 1

def lcd_init():
    # Initialise display
    lcd_byte(0x33,LCD_CMD) # 110011 Initialise
    lcd_byte(0x32,LCD_CMD) # 110010 Initialise
    lcd_byte(0x06,LCD_CMD) # 000110 Cursor move direction
    lcd_byte(0x0C,LCD_CMD) # 001100 Display On,Cursor Off, Blink Off
    lcd_byte(0x28,LCD_CMD) # 101000 Data length, number of lines, font size
    lcd_byte(0x01,LCD_CMD) # 000001 Clear display
    time.sleep(E_DELAY)

def lcd_byte(bits, mode):
    # Send byte to data pins
    # bits = the data
    # mode = 1 for data
    #       0 for command

    bits_high = mode | (bits & 0xF0) | LCD_BACKLIGHT
    bits_low = mode | ((bits<<4) & 0xF0) | LCD_BACKLIGHT
```

Ln: 111 Col: 0


```

# huella.py - C:\Users\Alumne\Desktop\sensor huellas\Proyecto final\huella.py (3.7.3)*
File Edit Format Run Options Window Help

# High bits
bus.write_byte(I2C_ADDR, bits_high)
lcd_toggle_enable(bits_high)

# Low bits
bus.write_byte(I2C_ADDR, bits_low)
lcd_toggle_enable(bits_low)

def lcd_toggle_enable(bits):
    # Toggle enable
    time.sleep(E_DELAY)
    bus.write_byte(I2C_ADDR, (bits | ENABLE))
    time.sleep(E_PULSE)
    bus.write_byte(I2C_ADDR, (bits & ~ENABLE))
    time.sleep(E_DELAY)

def lcd_string(message, line):
    # Send string to display

    message = message.ljust(LCD_WIDTH, " ")

    lcd_byte(line, LCD_CMD)

    for i in range(LCD_WIDTH):
        lcd_byte(ord(message[i]), LCD_CHR)

def main():
    # Main program block

    # Initialise display
    lcd_init()

def huella():
    #lectura de huella
    lectura = choice(empleados)
    return lectura

#código principal

empleados = []
lectura = []
registro = []
#control = []
#Opcion = str
#digit = None
seq = []
lcd_string("", LCD_LINE_1)
lcd_string("", LCD_LINE_2)

while True:

    # Inicializar keypad
    kp = keypad(columnCount = 3)

    # waiting for a keypress
    digit = None
    while digit == None or (digit != 1 and digit != 2 and digit != 3):
        df = pd.DataFrame(registro, columns = ['Estado', 'Empleado', 'hora', 'Fecha'])
        df.to_csv('Fichaje '+str(time.strftime("%d-%m-%y")) + '.csv')
        digit = kp.getKey()

```

```
*huella.py - C:\Users\Alumne\Desktop\sensor huellas\Proyecto final\huella.py (3.7.3)*
File Edit Format Run Options Window Help

digit = None
while digit == None or (digit != 1 and digit != 2 and digit != 3) :
    df = pd.DataFrame(registro, columns = ['Estado', 'Empleado', 'hora', 'Fecha'])
    df.to_csv('Fichaje '+str(time.strftime("%d-%m-%y")) + '.csv')
    digit = kp.getKey()
    lcd_string("Entrada: Press 1",LCD_LINE_1)
    lcd_string("Salida : Press 2",LCD_LINE_2)

    #Opcion = str(input('1: Entrada, 2: Salida: '))

    if digit == 1:
        print(digit)
        numero_empleado = huella()
        #control.append(numero_empleado)
        #print(control)
        #if numero_empleado not in control:
        lectura = ('Entrada',numero_empleado,time.strftime("%H:%M:%S"),time.strftime("%d/%m/%y")
        registro.append(lectura)
        print(registro)
        print(lectura)

        #else:
        # lcd_string("'Usted ya ha fichado'",LCD_LINE_1)
        # lcd_string("",LCD_LINE_2)

    if digit == 2:
        numero_empleado = huella()
        lectura = ('Salida',numero_empleado,time.strftime("%H:%M:%S"),time.strftime("%d/%m/%y")
        registro.append(lectura)
        print(registro)
        print (lectura)

    if digit == 3:

        try:
            p = GT_521F52.PyFingerprint_GT_521F52('/dev/ttyUSB0')

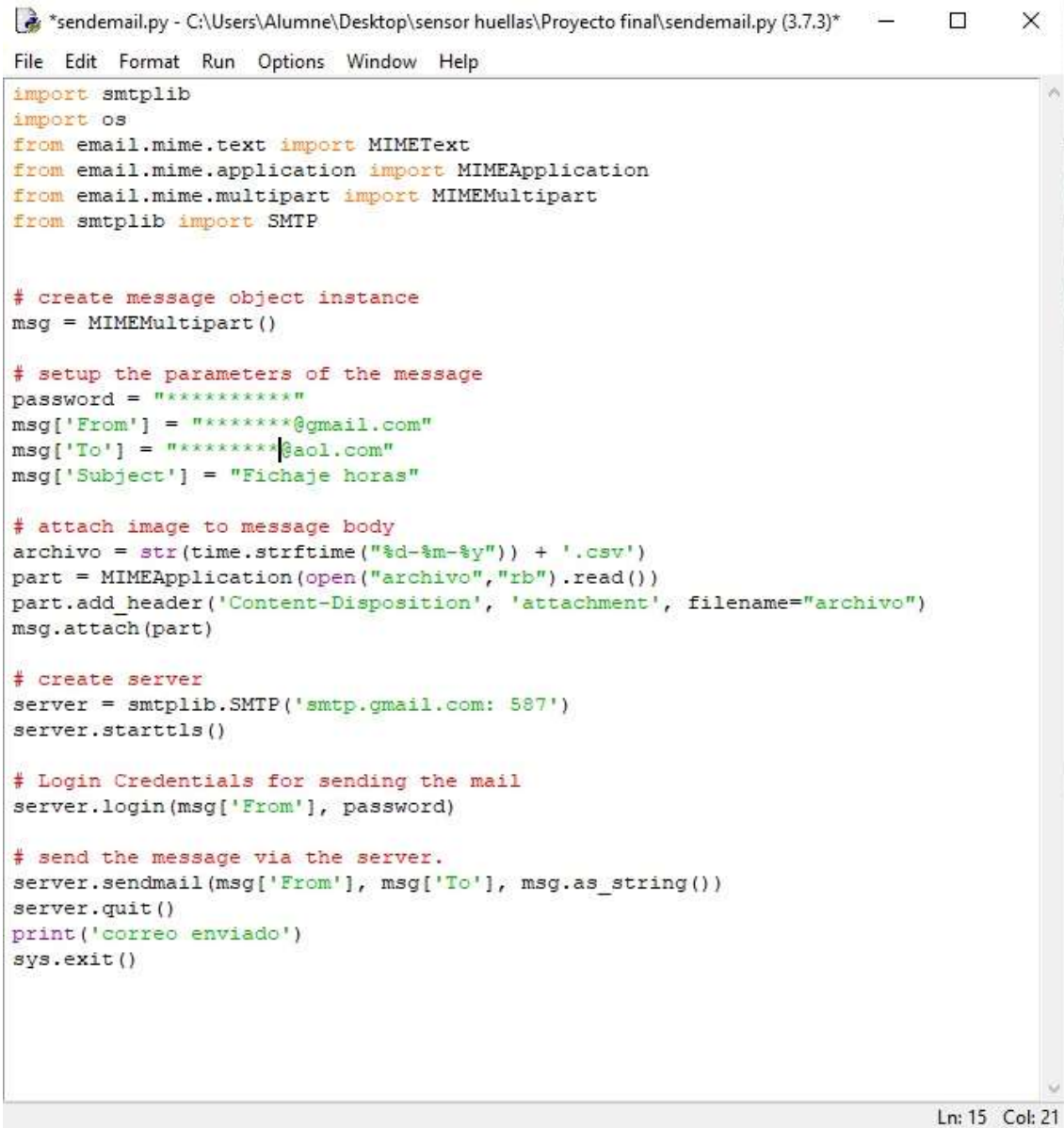
        except Exception as e:
            print("Something went wrong")
            print('Exception message: ' + str(e))

        try:
            lcd_string("Ponga dedo",LCD_LINE_1)
            lcd_string("tres veces",LCD_LINE_2)
            time.sleep(2)
            p.enrollUser()
            #print("Now lets identify you finger")
            time.sleep(2)
            pos = p.IdentifyUser()
            if(pos):
                lcd_string("Tu dedo es el ",LCD_LINE_1)
                lcd_string(("Numero" + str(pos)),LCD_LINE_2)
                empleado.adde

        # For other functions simply check the datasheet and the api
        except Exception as e:
            lcd_string(('Exception message: ' + str(e)),LCD_LINE_1)
            lcd_string(("Numero" + str(pos)),LCD_LINE_2)

Ln: 149 Col: 0
```

El siguiente código es para enviar la plantilla creada diariamente, se ejecutará con los comandos cron y crontab de linux para llamar el programa diariamente y enviarlo a la dirección de correo electrónico del responsable de recursos humanos.



```
import smtplib
import os
from email.mime.text import MIMEText
from email.mime.application import MIMEApplication
from email.mime.multipart import MIMEMultipart
from smtplib import SMTP

# create message object instance
msg = MIMEMultipart()

# setup the parameters of the message
password = "*****"
msg['From'] = "*****@gmail.com"
msg['To'] = "*****@aol.com"
msg['Subject'] = "Fichaje horas"

# attach image to message body
archivo = str(time.strftime("%d-%m-%y")) + '.csv'
part = MIMEApplication(open("archivo", "rb").read())
part.add_header('Content-Disposition', 'attachment', filename="archivo")
msg.attach(part)

# create server
server = smtplib.SMTP('smtp.gmail.com: 587')
server.starttls()

# Login Credentials for sending the mail
server.login(msg['From'], password)

# send the message via the server.
server.sendmail(msg['From'], msg['To'], msg.as_string())
server.quit()
print('correo enviado')
sys.exit()
```

Ln: 15 Col: 21

Conclusiones

El sensor de huellas es un método que es fiable y de bajo coste para la identificación de las personas, este proyecto consiste en identificar a los empleados de una pequeña empresa para el control de entrada y salida.

Por el tiempo para realizar el proyecto se centró en un proceso de control de entrada y salida sencillo, sin embargo, el programa se puede ir actualizando e ir agregándole módulos como por ejemplos envío de informes de empleados que llegan fuera de horario módulo de descansos, etc. Esto se puede programar y actualizar por Docker.

Se optó por el envío de informes por medio de correo electrónico en vez de usar una nube corporativa por un tema de seguridad ya que hay información que atañe sólo al personal de RRHH y este método es mas seguro y directo, cuando se programó el envío de correo la API que los envía en muchos lados estaba programada para enviar fotos, lo que había que hacer era programar el archivo adjunto como un binario y enviarlo en vez de hacerlo como foto. El código es el siguiente:

```
“# Adjuntar archivo al cuerpo del mensaje

archivo = str(time.strftime("%d-%m-%y")) + '.csv'

part = MIMEApplication(open("archivo", "rb").read())

part.add_header('Content-Disposition', 'attachment', filename="archivo")

msg.attach(part)”
```

Una dificultad o desafío que hubo fue elegir un fingerprint que no ha sido tan masivo para que fuera compatible con la Raspberry, había menos información en la web sobre el con solo una librería programada en Python, ya que el fabricante tiene sus librerías programadas en C y un ejecutable para Windows. El fingerprint no tenía el cable Mini Micro JST de cuatro pines por lo que se soldó directamente los cables al fingerprint que el fabricante daba la opción al conectarlo por la GPIO no funcionó y aquí hubo otro problema ya que la librería en Python estaba programada para conexión por USB.

La librería al importarla por pip no llamaba a todos los comandos por lo que no funcionaba correctamente, la solución para este problema fue descargar directamente la librería en la carpeta de trabajo del programa principal, la librería descargada de GIT es GT_521F52 y se llamó directamente como se ve aquí:

```
“import GT_521F52 as FP

from GT_521F52 import PyFingerprint_GT_521F52 as lectorFP

import time

try:

    p = FP.lectorFP('/dev/ttyUSB0’)”
```

Se tomó la decisión de comprar el cable JST de cuatro pines y un adaptador UART para conectarlo con USB, esto produjo a más retraso ya que no se podía terminar la API ya que el fingerprint no se podía conectar, al llegar los cables y conectores por USB el dispositivo no funcionaba se soldaron los cables al UART de diferentes formas y en un día no se pudo hacer que el dispositivo leyera una huella.

Volviendo pasos atrás y después de haber probado todas las opciones se retocaron las soldaduras que se hicieron en el fingerprint que a primera vista no hacían ningún puente entre los conectores, se limpió la soldadura y después de se probó el fingerprint y funcionó perfectamente.

Se resolvió este gran inconveniente y ahora ya se puede usar y programar el sensor de huellas, para resolverlo se volvió a los pasos anteriores y un pequeño proceso que se verificó y aparentemente estaba bien produjo este gran incidente.

Bibliografía

Estrategia Magazine:

<https://www.estrategiamagazine.com/tecnologia/los-controles-biometricos-verificacion-de-voz-escritura-huellas-patron-oculares-retina-iris-geometria-de-la-mano/>

Instituto Nacional de Ciber seguridad:

https://www.incibe.es/sites/default/files/contenidos/guias/doc/guia_tecnologias_biometricas_aplicadas_ciberseguridad_metad.pdf

Librería Sensor fingerprint GT-521F52

<https://github.com/CESARBR/GT521F52>

Configuración del keypad

<https://tutorials-raspberrypi.com/connezc-raspberry-pi-kecpad-code-lock/>