ANEXO I

PROGRAMAS EN PYTHON

JAVIER URREA RIVAS

CONTENIDO

1 Programa en Python

A continuación, se muestran los programas en Python.
Pueden contener algún error de ortografía ya que, a fin de evitar errores de compilación al usar algún carácter no anglosajón, dichos caracteres se han cambiado o suprimido.

1.1 Programa configuración cámara

```
import cv2
import numpy as np
start_point = (350, 170)
end_point = (450, 260)
color = (0, 255, 0)
thickness = 2
cap = cv2.VideoCapture(0)
while True:
  ret, frame = cap.read()
  frame90=cv2.rotate(frame, cv2.ROTATE_90_CLOCKWISE)
  frame180=cv2.rotate(frame90, cv2.ROTATE_90_CLOCKWISE)
  if ret==True:
    frameHSV = cv2.rectangle(frame180,start_point, end_point, color, thickness)
    cv2.imshow('frame',  frame180)
    if cv2.waitKey(1) & 0xFF == ord('s'):
      break
cap.release()
cv2.destroyAllWindows()
```

1.2 Programa Realización foto mediante señal del sensor ultrasonidos

```
# Importamos las librerias
import RPi.GPIO as GPIO
from gpiozero import DistanceSensor
from picamera import PiCamera
import time
TRIG = 4 #Variable que contiene el GPIO al cual conectamos TRIG del sensor
ECHO = 17 #Variable que contiene el GPIO al cual conectamos ECHO del sensor
ultrasonic = DistanceSensor(echo=17 , trigger=4, threshold_distance=0.9)
GPIO.setmode(GPIO.BCM)     #Establecemos el modo segun el cual nos refiriremos a los GPIO de nuestra RPi
GPIO.setup(TRIG, GPIO.OUT) #Configuramos el pin TRIG como una salida
GPIO.setup(ECHO, GPIO.IN)  #Configuramos el pin ECHO como una salida
camera = PiCamera()
camera.rotation = 180
camera.resolution = (2592, 1944)
camera.framerate = 30
counter = 0
#Contenemos el codigo principal en una estructura try para limpiar los GPIO al terminar o presentarse un error
try:
   #Implementamos un loop infinito
   while True:
      # Ponemos en bajo el pin TRIG y despues esperamos 0.5 seg para que el transductor se estabilice
      GPIO.output(TRIG, GPIO.LOW)
      time.sleep(0.5)
      #Ponemos en alto el pin TRIG esperamos 10 uS antes de ponerlo en bajo
      GPIO.output(TRIG, GPIO.HIGH)
      time.sleep(0.00001)
      GPIO.output(TRIG, GPIO.LOW)
          time.sleep(0.00001)
```

```
    # En este momento el sensor envia 8 pulsos ultrasonicos de 40kHz y coloca su pin ECHO en alto
    # Debemos detectar dicho evento para iniciar la medicion del tiempo

    while True:
      pulso_inicio = time.time()
      if GPIO.input(ECHO) == GPIO.HIGH:
        break

    # El pin ECHO se mantendra en HIGH hasta recibir el eco rebotado por el obstaculo.
    # En ese momento el sensor pondra el pin ECHO en bajo.
    # Prodedemos a detectar dicho evento para terminar la medicion del tiempo

    while True:
      pulso_fin = time.time()
      if GPIO.input(ECHO) == GPIO.LOW:
        break

    # Tiempo medido en segundos
    duracion = pulso_fin - pulso_inicio

    #Obtenemos la distancia considerando que la senal recorre dos veces la distancia a medir y que la velocidad del sonido es 343m/s
    distancia = (34300 * duracion) / 2

    # Imprimimos resultado
    print( "Distancia: %.2f cm" % distancia)
    if distancia < 5:

        #camera.start_preview()

        camera.exposure_mode = 'sports'
```

```
            camera.capture('/home/pi/Desktop/tfg/Fotos_para_procesar/image%s.png' %
counter)
            counter = counter + 1
            #camera.stop_preview()
            #time.sleep (0,5)
finally:
   # Reiniciamos todos los canales de GPIO.
   GPIO.cleanup()
```

1.3 Programa procesado de imagen y tratamiento de datos en las bases de datos

```
#Librerias
import numpy as np
#Libreria OpenCv para inteligencia artificial (procesado de imagen)
import cv2
#Libreria Descodificacion Datamatrix
from pylibdmtx import pylibdmtx
#Libreria Puerto serie Raspberry
import RPi.GPIO as GPIO
from time import sleep
import os.path

#Libreria Base de datos y monitorizacion Adafruit io
from Adafruit_IO import *

#Libreria Documento tiempo real google docs
import gspread
from oauth2client.service_account import ServiceAccountCredentials
from pprint import pprint
```

```
#Libreria para fecha y hora de la lectura
from datetime import date
from datetime import datetime

#Libreria buscar datos del producto en base de datos
import pandas as pd

#Usuario y key para enviar datos a la base de datos de Adafruit io
aio= Client('Prototipo','aio_Zyzc46j46MA4sRy1BO8KWfCUsDGI')

#Sincronizacion cuenta y documento google docs
scope =
["https://spreadsheets.google.com/feeds",'https://www.googleapis.com/auth/spreadsheets',"
https://www.googleapis.com/auth/drive.file","https://www.googleapis.com/auth/drive"]

creds = ServiceAccountCredentials.from_json_keyfile_name("creds.json", scope)
client = gspread.authorize(creds)

#Contadores internos del programa
cont_doc=4
cont_unidades=0
cont_unidades_totales=0

#Deshabilito los warnings
GPIO.setwarnings(False)

#Selecciono el GPIO mode para el zumbador
GPIO.setmode(GPIO.BCM)

#Pin output del zumbador es el pin 23
buzzer=23
```

```
#Variables para edicion de imagen
ancho_izquierda = 1400
ancho_derecha = 1800
alto_arriba = 700
alto_abajo = 1000
counter=0

#Variables programa
lote_actual=0

#Tipos de datamatrix

datamatrix_tipo1=102 #Datamatrix sin control T&T
datamatrix_tipo2=123 #Datamatrix con control T&T

def Procesado_imagen():

    #Recorto la imagen
    image2 = image1[alto_arriba:alto_abajo,ancho_izquierda:ancho_derecha]

    #Modifico la resolucion de la imagen
    image = cv2.resize(image2, (0,0), fx=0.7, fy=0.7)

    #Modifico el contraste
    contrast_img = cv2.addWeighted(image, 1.3, np.zeros(image.shape, image.dtype), 0, 0)

    #La paso a escala de grises para un mejor procesado
    gray_img = cv2.cvtColor(contrast_img, cv2.COLOR_BGR2GRAY)

    msg = pylibdmtx.decode(gray_img)
    return (msg)
```

```
def Buzzer():
    GPIO.setup(buzzer,GPIO.OUT)
    GPIO.output(buzzer,GPIO.HIGH)

    sleep(0.5) # Retardo de 5 segundos
    GPIO.output(buzzer,GPIO.LOW)
    sleep(0.5) # Retardo de 5 segundos
    return

def buzzer_error():

    GPIO.setup(buzzer,GPIO.OUT)
    contador_beep=0

    while (contador_beep < 10):
        GPIO.output(buzzer,GPIO.HIGH)
        sleep(0.5) # Retardo de 5 segundos
        GPIO.output(buzzer,GPIO.LOW)
        sleep(0.5) # Retardo de 5 segundos
        contador_beep = contador_beep + 1
    contador_beep = 0
    return
```

```
if __name__ == '__main__':
  while True:
    while not os.path.isfile('/home/pi/Desktop/tfg/Fotos_para_procesar/image%s.png' %
counter):
        pass

    #Cargo la imagen
    image1 = cv2.imread('/home/pi/Desktop/tfg/Fotos_para_procesar/image%s.png' % counter,
cv2.IMREAD_UNCHANGED)
    counter = counter + 1
    datamatrix_sin_procesar = Procesado_imagen()
    procesado1 =str(datamatrix_sin_procesar)

    #Cuenta los digitos para depurarlos a posteriori
    contador = len(procesado1)
    print("El numero tiene %s digitos" % (contador))

    if datamatrix_tipo1==contador:
        datamatrix =  procesado1[15:49]
        producto = datamatrix[7:15]
        lote = datamatrix[26:34]
        caducidad_ano = datamatrix[18:20]
        caducidad_mes = datamatrix[20:22]
        caducidad_dia = datamatrix[22:24]
        caducidad = caducidad_dia + str('.') + caducidad_mes + str('.') + str(20)+ caducidad_ano

        gtin = datamatrix[3:16]
        numero_serie= str("Producto NO T&T")

    elif datamatrix_tipo2==contador:
        datamatrix =  procesado1[15:71]
        producto = datamatrix[7:15]
```

```
    lote = datamatrix[48:56]
    caducidad_ano = datamatrix[40:42]
    caducidad_mes = datamatrix[42:44]
    caducidad_dia = datamatrix[44:46]
    caducidad = caducidad_dia + str('.') + caducidad_mes + str('.') + str(20)+ caducidad_ano
    gtin = datamatrix[2:16]
    numero_serie = datamatrix[18:38]

#Depuro el datamatrix

print("Datamatrix: %s" % (datamatrix))
print("Producto: %s" % (producto))
print("Lote: %s" % (lote))
print("Caducidad: %s" % (caducidad))
print("Gtin: %s" % (gtin))
print("SN_T&T: %s" % (numero_serie))

#Fecha y hora actual
hora = str (datetime.now())

#Envio datos al documento google docs
sheet = client.open("Prototipo industria 4.0").sheet1

datamatrix_entero = datamatrix[0:43]
sheet.update_cell(cont_doc,1, hora)
sheet.update_cell(cont_doc,2, datamatrix_entero)
sheet.update_cell(cont_doc,3, producto)
sheet.update_cell(cont_doc,4, lote)
sheet.update_cell(cont_doc,5, caducidad)
sheet.update_cell(cont_doc,6, gtin)
sheet.update_cell(cont_doc,7, numero_serie)
```

```
    #Lectura datos de los productos del documento google docs


    if lote != lote_actual:


        sheet2 = client.open("Prototipo industria 4.0").worksheet('Sheet2')


        data = sheet2.get_all_values()
        headers = data.pop(0)
        df = pd.DataFrame(data, columns=headers)
        formato=df[df['Producto']==producto]['Formato']
        formato_str = str(formato)
        formato_procesado = formato_str [5:14]
        print(formato_procesado)
        sheet.update_cell(cont_doc,8, formato_procesado)
        solucion=df[df['Producto']==producto]['Solucion']
        solucion_str= str (solucion)
        solucion_procesado = solucion_str[5:51]
        print(solucion_procesado)
        sheet.update_cell(cont_doc,9, solucion_procesado)


        print(data)
        cont_doc = cont_doc+1
        cont_unidades = int (formato_procesado[0:2])
        cont_unidades_totales = cont_unidades_totales + cont_unidades
        print(cont_unidades_totales)
        lote_cont= "Lote: " + lote + "  Producto: " + producto + "  Formato: " + formato_procesado
+ "  Solucion: " + solucion_procesado


        lote_actual = lote
    else:
        print(formato_procesado)
        print(solucion_procesado)
```

```
        cont_unidades_totales = cont_unidades_totales + cont_unidades
        print(cont_unidades_totales)
        sheet = client.open("Prototipo industria 4.0").sheet1
        sheet.update_cell(cont_doc,8, formato_procesado)
        sheet.update_cell(cont_doc,9, solucion_procesado)
        cont_doc = cont_doc+1


    #Enviar datos a nase de datos Adafruit IO_
    base_datos = lote + str('      ') + producto + str('      ') + lote + str('      ') + caducidad + str('      ') + gtin + str('      ') + formato_procesado + str('      ') + numero_serie +  str('      ') + solucion_procesado
    aio.send("lectura",base_datos)


    cambio_type= str(df)
    aio.send("productos-bbdd",cambio_type)
    aio.send("contador",cont_unidades_totales)
    aio.send("lote-cont",lote_cont)
    #Activo el zumbador en caso de lectura correcta
    if contador>50:
        Buzzer()
     #Activo el zumbador en caso de lectura cincorrecta durante 10 segundos y vuelvo a realizar la lectura
    If contador =2:
         buzzer_error()
```

1.3 Programa para ejecutar paralelamente el programa 1.2 y 1.3

```
import subprocess

# rutas de los programas
scripts_paths = ("/home/pi/Desktop/tfg /ultrasonidos_foto.py", "/home/pi/Desktop/tfg
/proc_imag_y_bbdd.py")

# Creamos cada proceso
procesos = [subprocess.Popen(["python", script]) for script in scripts_paths]

for proceso in procesos:
    proceso.wait()
```