Below is the code for the SmartDoorbell. The reason I have saved it on a word doc is because it cannot be tested because it requires the hardware components to fullthe hardware components to fully function.

from gpiozero import Button

import RPi.GPIO as GPIO

from datetime import datetime

from picamera import PiCamera

from picamera import PiCameraError

import os

import time

camera = None

button\_pressed = 0

start\_record = 0

button = Button(19)

# Define GPIO to LCD mapping

LCD\_RS = 25

LCD\_E  = 24

LCD\_D4 = 23

LCD\_D5 = 17

LCD\_D6 = 18

LCD\_D7 = 22

# Define some device constants

LCD\_WIDTH = 16    # Maximum characters per line

LCD\_CHR = True

LCD\_CMD = False

LCD\_LINE\_1 = 0x80 # LCD RAM address for the 1st line

LCD\_LINE\_2 = 0xC0 # LCD RAM address for the 2nd line

# Timing constants

E\_PULSE = 0.0005

E\_DELAY = 0.0005

def main():

    # Main program block

    GPIO.setwarnings(False)

    GPIO.setmode(GPIO.BCM)       # Use BCM GPIO numbers

    GPIO.setup(LCD\_E, GPIO.OUT)  # E

    GPIO.setup(LCD\_RS, GPIO.OUT) # RS

    GPIO.setup(LCD\_D4, GPIO.OUT) # DB4

    GPIO.setup(LCD\_D5, GPIO.OUT) # DB5

    GPIO.setup(LCD\_D6, GPIO.OUT) # DB6

    GPIO.setup(LCD\_D7, GPIO.OUT) # DB7

    # Initialise display

    lcd\_init()

    lcd\_string("Welcome to",LCD\_LINE\_1)

    lcd\_string("UCI",LCD\_LINE\_2)

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 = data

    # mode = True  for character

    #        False for command

    GPIO.output(LCD\_RS, mode) # RS

    # High bits

    GPIO.output(LCD\_D4, False)

    GPIO.output(LCD\_D5, False)

    GPIO.output(LCD\_D6, False)

    GPIO.output(LCD\_D7, False)

    if bits&0x10==0x10:

        GPIO.output(LCD\_D4, True)

    if bits&0x20==0x20:

        GPIO.output(LCD\_D5, True)

    if bits&0x40==0x40:

        GPIO.output(LCD\_D6, True)

    if bits&0x80==0x80:

        GPIO.output(LCD\_D7, True)

    # Toggle 'Enable' pin

    lcd\_toggle\_enable()

    # Low bits

    GPIO.output(LCD\_D4, False)

    GPIO.output(LCD\_D5, False)

    GPIO.output(LCD\_D6, False)

    GPIO.output(LCD\_D7, False)

    if bits&0x01==0x01:

        GPIO.output(LCD\_D4, True)

    if bits&0x02==0x02:

        GPIO.output(LCD\_D5, True)

    if bits&0x04==0x04:

        GPIO.output(LCD\_D6, True)

    if bits&0x08==0x08:

        GPIO.output(LCD\_D7, True)

    # Toggle 'Enable' pin

    lcd\_toggle\_enable()

def lcd\_toggle\_enable():

    # Toggle enable

    time.sleep(E\_DELAY)

    GPIO.output(LCD\_E, True)

    time.sleep(E\_PULSE)

    GPIO.output(LCD\_E, False)

    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)

main()

while True:

    try:

        if button.is\_pressed:

            os.system('pushntf.sh "DingDongClick: doorbell has rung."')

            if camera == None or not camera.recording:

                camera = PiCamera()

                camera.start\_recording("video{}.h264".format(str(datetime.now())))

                start\_record = time.time()

            if time.time() - button\_pressed < 5:

                os.system(

                    'omxplayer "/home/pi/Downloads/Copy-of-WWE-John-Cena-New-Theme-Song-2016-\_1\_.wav"')

                button\_pressed = time.time()

            else:

                os.system('omxplayer "/home/pi/Downloads/Westminster-chimes.mp3"')

                button\_pressed = time.time()

        if camera != None and camera.recording and time.time() - start\_record >= 120:

            camera.stop\_recording()

            camera.close()

            camera = None

    finally:

        if camera != None:

            camera.close()