```
# Simple GUI to receive bell signal as first trial for the door annunciator system
# 26.2.17
# by Julian Rogers
# bell_receive1
# version has reduced window size to fit on Pi 7" screen
# This puts up some labels and a button on the screen
# The label indicates when a bell signal has been received
# The button goes red when this happens until it is pressed to reset
# When a notification is received, the message is bounced back to the sender as confirmation.
# This runs on a Pi. The “bell” itself is running on a Pi Zero.


#remote ip address - enter your IP addresses
IP_ZERO = "xxx.xxx.xxx.xxx"
IP_THIS_PI ="xxx.xxx.xxx.xxx"


THIS_PORT = xxxx                #enter your port for the computer running this program
BACKGROUND = "gray"

from tkinter import *   #GUI

import datetime
import time
import socket           #UDP
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

# create the root window
root = Tk()

# modify the window
root.title("BELL RECEIVE")
#root.geometry("700x430")
root.configure(bg = BACKGROUND)
#root.attributes('-fullscreen', True)   #eliminates the title bar

w, h = root.winfo_screenwidth(), root.winfo_screenheight()
root.geometry("%dx%d+0+0" % (w, h))

# create a frame
app = Frame(root)
app.configure(bg = BACKGROUND)
app.grid()

title_lab = Label(app, text = "Bell Receiver Test", font = ("Arial Bold", 20), fg = "maroon", bg = "gray")
title_lab.grid(row = 1, column = 1, columnspan = 8)

blank_lab = Label(app, bg = "gray")
blank_lab.grid(row = 2, column = 1)

reset_but = Button(app, text = "Reset", font = ("Arial", 16), fg = "maroon", bg = "light blue")
reset_but.grid(row = 3, column = 1)

def reset_but_reset():
    reset_but.config(bg = "light blue")

reset_but.config(command = reset_but_reset)

#use "Button" rather than "Label" to match look of "reset_but"
received_label = Button(app, text = "Waiting...", font = ("Arial", 16), fg = "maroon", bg = "light blue")
received_label.grid(row = 3, column = 2)


def do_stuff():
    received = "Waiting..."

    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.bind((IP_THIS_PI, THIS_PORT))
    try:
        sock.settimeout(1)
        while True:
            received, addr = sock.recvfrom(100)
            if received:
                sent = sock.sendto(received, addr)
                reset_but.config(bg = "red" )


    except socket.error:
        received_label.config(text = "error...")
    received_label.config(text = received)


# calls a function after given time
def after(self, ms, func = None, *args):
    """call function after a given time"""

# updates screen every 0.1 seconds
def task():
    do_stuff()
    root.after(100, task)

# calls the screen update every 0.1 seconds
root.after(100, task)

#------------------------------------------------------------------------------------------
# kick off the window's event-loop
root.mainloop()
```