Teach-In 2014 with Raspberry Pi : Part 10

These are text files of the source code listings printed in EPE.

They appear in the same order as in the articles.

Separate listings are split by four empty lines.

from tkinter import \*

root = Tk()

w = Label(root, text="Teach-In with Raspberry Pi")

w.pack()

root.mainloop()

from tkinter import \*

root = Tk()

copyright = """

Copyright (c) 2014 Everyday Electronics

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit any person to whom the Software is furnished to do so, subject to the following condition:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

"""

w = Label(root, text=copyright, anchor=W, justify=LEFT, font='Helvetica', fg='green', bg='black', wraplength=360)

w.pack()

root.mainloop()

from tkinter import \*

import time

root = Tk()

v = StringVar()

Label(root, textvariable=v).pack()

time\_now = time.strftime('%H:%M:%S')

v.set(time\_now)

root.mainloop()

from tkinter import \*

import time

root = Tk()

imgfile = PhotoImage(file="traffic\_lights.gif")

w = Label(root, image=imgfile)

w.photo = imgfile

w.pack()

from tkinter import \*

class App:

def \_\_init\_\_(self, master):

frame = Frame(master)

master.wm\_title("Select Option")

frame.pack()

self.button1 = Button(frame, text="Button 1", command=self.button1)

self.button1.pack(side=LEFT)

self.button2 = Button(frame, text="Button 2", command=self.button2)

self.button2.pack(side=LEFT)

self.button3 = Button(frame, text="Button 3", command=self.button3)

self.button3.pack(side=LEFT)

self.quit = Button(frame, text=" QUIT ", fg="red", command=self.quit)

self.quit.pack(side=LEFT)

def button1(self):

print("Button 1 clicked")

def button2(self):

print("Button 2 clicked")

def button3(self):

print("Button 3 clicked")

def quit(self):

print("QUIT button clicked and window closed ...")

root.destroy()

root = Tk()

app = App(root)

root.mainloop()

# BCD converter - 1

# Input: 8-bit BCD encoded binary string

# Output: Equivalent decimal integer

# Python 3.x

def int\_to\_bcd(x):

bcdstring = ''

while x > 0:

nibble = x % 16

bcdstring = str(nibble) + bcdstring

x >>= 4

return int(bcdstring)

# Get the BCD value from the user

print("BCD Converter")

user = input("Please enter 8-bit BCD: ")

# Convert the user's binary string to an integer

value = int((user), 2)

# Print the result

print("BCD equivalent = " + str(int\_to\_bcd(value)))

# BCD converter - 2

# Input: Decimal integer

# Output: Equivalent BCD encoded string

# Python 3.x

def bcd\_to\_int(x):

binstring = ''

while True:

q, r = divmod(x, 10)

nibble = bin(r).replace('0b', "")

while len(nibble) < 4:

nibble = '0' + nibble

binstring = nibble + binstring

if q == 0:

break

else:

x = q

return int(binstring, 2)

# Get the decimal integer from the user

print("BCD Converter")

user = input("Please enter decimal integer: ")

# Convert the user's decimal string to an integer

value = int(user)

# Print the result and strip the leading 0b

print("BCD equivalent = " + str(bin(bcd\_to\_int(value))[2:]))

sudo modprobe i2c-dev

sudo modprobe rtc-pcf8563

sudo su

root@raspberrypi: /home/pi #

echo pcf8563 0x51 > /sys/class/i2c-adapter/i2c-0/new\_device

echo pcf8563 0x51 > /sys/class/i2c-adapter/i2c-1/new\_device

ls /dev/rtc0

# Set PCF8563 clock

# For Python 2.x

from Tkinter import \*

import smbus

import time

import subprocess

bus = smbus.SMBus(0)

root = Tk()

label1 = Label( root, text="Time HH:MM:SS")

E1 = Entry(root, bd =5)

mytime = "00:00:00"

# Convert a decimal number to BCD

def dectobcd(n):

bcdstring = ''

if n < 10:

bcdstring = '0000'

for i in str(n):

bcdstring += bcddigit(i)

return int(bcdstring,2)

return hex(int(bcdstring,2))

# Convert to BCD digit

def bcddigit(q):

p = int(q)

r = bin(p)[2:]

return ('0000' + r)[-4:]

# Set the time

def setTime():

mytime = E1.get()

timelist = mytime.split(":")

hours = (timelist[0])

hournum = int(hours)

minutes = (timelist[1])

minnum = int(minutes)

seconds = (timelist[2])

secnum = int(seconds)

# Get new data to write back to the PC8563

sec\_data = str(dectobcd(secnum))

min\_data = str(dectobcd(minnum))

hrs\_data = str(dectobcd(hournum))

# Write data to the PCF8562 registers

command\_string = "i2cset -y 0 0x51 0x02 " + sec\_data

return\_code = subprocess.call([command\_string], shell=True)

command\_string = "i2cset -y 0 0x51 0x03 " + min\_data

return\_code = subprocess.call([command\_string], shell=True)

command\_string = "i2cset -y 0 0x51 0x04 " + hrs\_data

return\_code = subprocess.call([command\_string], shell=True)

submit = Button(root, text ="Update clock", command = setTime)

label1.pack()

E1.pack()

submit.pack(side =BOTTOM)

root.mainloop()

# Digital clock using PCF8563

# For Python 2.x

from Tkinter import \*

import smbus

import time

bus = smbus.SMBus(0)

root = Tk()

timenow = ''

clock = Label(root, font=('times', 20, 'bold'), bg='green')

clock.pack(fill=BOTH, expand=1)

def tick():

global timenow

# get the current local time from the RTC

bus.write\_byte(0x51, 0x80)

time.sleep(0.1)

cs1\_data = bus.read\_byte(0x51)

cs2\_data = bus.read\_byte(0x51)

sec\_data = bus.read\_byte(0x51)

min\_data = bus.read\_byte(0x51)

hrs\_data = bus.read\_byte(0x51)

sec1 = ((sec\_data >> 4) & 0x07)

sec2 = (sec\_data & 0x0f)

min1 = ((min\_data >> 4) & 0x07)

min2 = (min\_data &0x0f)

hrs1 = ((hrs\_data >> 4) & 0x03)

hrs2 = (hrs\_data &0x0f)

seconds = (str(sec1) + str(sec2))[-2:2]

minutes = (str(min1) + str(min2))[-2:2]

hours = (str(hrs1) + str(hrs2))[-2:2]

timenow = hours + ":" + minutes + ":" + seconds

clock.config(text=timenow)

# update the clock display every 200ms

clock.after(200, tick)

tick()

root.mainloop()

# Set PCF8563 clock output frequency

# For Python 2.x

from Tkinter import \*

import smbus

import subprocess

bus = smbus.SMBus(0)

class App:

def \_\_init\_\_(self, master):

frame = Frame(master)

master.wm\_title("Select Clock Output")

frame.pack()

self.button1 = Button(frame, text="32.768kHz", command=self.button1)

self.button1.pack(side=LEFT)

self.button2 = Button(frame, text=" 1024Hz ", command=self.button2)

self.button2.pack(side=LEFT)

self.button3 = Button(frame, text=" 32Hz ", command=self.button3)

self.button3.pack(side=LEFT)

self.button4 = Button(frame, text=" 1Hz ", command=self.button4)

self.button4.pack(side=LEFT)

self.quit = Button(frame, text=" QUIT ", fg="red", command=self.quit)

self.quit.pack(side=LEFT)

def button1(self):

return\_code = subprocess.call(["i2cset -y 0 0x51 0x0d 0x80"], shell=True)

def button2(self):

return\_code = subprocess.call(["i2cset -y 0 0x51 0x0d 0x81"], shell=True)

def button3(self):

return\_code = subprocess.call(["i2cset -y 0 0x51 0x0d 0x82"], shell=True)

def button4(self):

return\_code = subprocess.call(["i2cset -y 0 0x51 0x0d 0x83"], shell=True)

def quit(self):

root.destroy()

root = Tk()

app = App(root)

root.mainloop()

MMDDhhmmyyyy

sudo date 040710272014

Mon Apr 7 10:27:00 BST 2014

sudo date

Mon Apr 7 10:27:00 BST 2014

sudo modprobe i2c-dev

sudo modprobe rtc-pcf8563

sudo su

echo pcf8563 0x51 > /sys/class/i2c-adapter/i2c-0/new\_device

echo pcf8563 0x51 > /sys/class/i2c-adapter/i2c-1/new\_device

hwclock –-systohc

hwclock -r

Finally, to set the Linux/Debian system time to the value in the ‘real’ hardware clock you can use:

hwclock -r

sudo nano /etc/rc.local

echo pcf8563 0x51 > /sys/class/i2c-adapter/i2c-0/new\_device

echo pcf8563 0x51 > /sys/class/i2c-adapter/i2c-1/new\_device

sudo hwclock –s

hwclock –-set –-date=”2014-04-07 16:09:30”

sudo apt-get install chrony

sudo chronyc

chronyc> tracking

Reference ID : 193.227.197.2 (yikes.bl2.tolna.net)

Stratum : 3

Ref time (UTC) : Mon Apr 7 09:53:16 2014

System time : 0.000000235 seconds slow of NTP time

Frequency : 42.852 ppm fast

Residual freq : 7.319 ppm

Skew : 53.437 ppm

Root delay : 0.133704 seconds

Root dispersion : 0.089000 seconds