Teach-In 2014 with Raspberry Pi: Part 9

by Mike and Richard Tooley

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.

# HTML code to be rendered

html\_str = """

<html>

<head>

</head>

<body>

<h2>Hello Raspberry Pi!</h2>

</body>

</html>

"""

# Write the html to a file

file = open("hello.html","w")

file.write(html\_str)

file.close()

# Import the libraries

import webbrowser

import time

now = time.time()

# HTML code to be rendered

html\_str1 = """

<html>

<head>

</head>

<body>

<h2>Time now:

"""

html\_str2 = """

</h2>

</body>

</html>

"""

html\_str = html\_str1 + time.ctime(now) + html\_str2

# Write the html to a file

file = open("time.html","w")

file.write(html\_str)

file.close()

# Open the web browser and display the file

webbrowser.open("time.html")

# Import the libraries

import webbrowser

import time

now = time.time()

with open('time.html', 'w') as myFile:

myFile.write('<html><head></head><body><h2>')

myFile.write('Time now: ')

myFile.write(time.ctime(now))

myFile.write('</h2></body></html>')

# Open the web browser and display the file

webbrowser.open("time.html")

# Import the library

import webbrowser

# HTML code to be rendered

html\_str = """

<html>

<head>

</head>

<body>

<h1>MCP4725 DAC pin connections</h1>

<img src="fig1.gif"></img>

<img src="fig2.png"></img>

</body>

</html>

"""

# Write the html to a file

file = open("dac.html","w")

file.write(html\_str)

file.close()

# Open the web browser and display the file

webbrowser.open("dac.html")

# Import the GPIO library module

import RPi.GPIO as GPIO

# Set the GPIO pin numbering convention

GPIO.setmode(GPIO.BOARD)

# Configure the GPIO

GPIO.setmode(GPIO.BOARD)

GPIO.setup(11, GPIO.OUT) # White LEDs

GPIO.setup(13, GPIO.OUT) # IR LEDs

# To turn the white LEDs on

def white\_on():

GPIO.output(11, True)

# To turn the white LEDs off

def white\_off():

GPIO.output(11, False)

# To turn the IR LEDs on

def IR\_on():

GPIO.output(13, True)

# To turn the IR LEDs on

def IR\_off():

GPIO.output(13, False)

white\_off()

IR\_off()

# Import the GPIO library module

import RPi.GPIO as GPIO

import tkinter as tk

# Set the GPIO pin numbering convention

GPIO.setmode(GPIO.BOARD)

# Configure the GPIO

GPIO.setmode(GPIO.BOARD)

GPIO.setup(11, GPIO.OUT) # White LEDs

GPIO.setup(13, GPIO.OUT) # IR LEDs

# User-defined functions

def white\_on():

GPIO.output(11, True)

def white\_off():

GPIO.output(11, False)

def IR\_on():

GPIO.output(13, True)

def IR\_off():

GPIO.output(13, False)

# Initialise with both sets of LEDs off

white\_off()

IR\_off()

# Print key selection

print("Camera lighting controller")

print("==========================")

print("[W] white on, [S] white off, [R] IR on, [F] IR off, [Q] to quit")

# Respond to a key press

while True:

x = ""

x = input("Enter command letter : ")

if x == "q":

print("Program terminated")

break

elif x == "w":

print("White ON")

white\_on()

elif x == "s":

print("White OFF")

white\_off()

elif x == "r":

print("Infra red ON")

IR\_on()

elif x == "f":

print("Infra red OFF")

IR\_off()

# Import the library modules

import RPi.GPIO as GPIO

from tkinter import \*

master = Tk()

# Set the GPIO pin numbering convention

GPIO.setmode(GPIO.BOARD)

# Configure the GPIO

GPIO.setmode(GPIO.BOARD)

GPIO.setup(11, GPIO.OUT) # White LEDs

GPIO.setup(13, GPIO.OUT) # IR LEDs

# User-defined functions

def white\_on():

GPIO.output(11, True)

def white\_off():

GPIO.output(11, False)

def IR\_on():

GPIO.output(13, True)

def IR\_off():

GPIO.output(13, False)

# User-defined functions to handle checkbutton events

def go():

print("OK clicked")

def normal():

if w.get()==1:

print("White on")

white\_on()

else:

print("White off")

white\_off()

def infrared():

if x.get()==1:

print("IR on")

IR\_on()

else:

IR\_off()

print("IR off")

# Set up window size and title

master.geometry("%dx%d%+d%+d" % (180, 110, 0, 0))

master.wm\_title("CAM-1")

w = IntVar() # Normal checkbutton state

x = IntVar() # Infra-red checkbutton state

# Initialise with both sets of LEDs off

white\_off()

IR\_off()

print("White off")

print("IR off")

# Place a message and checkbuttons in the window

Message(master, text="LIGHTING", width=150).pack(padx=25, anchor=CENTER)

Checkbutton(master, text="Normal", variable=w, command=normal).pack(padx=25, anchor=NW)

Checkbutton(master, text="Infra-red", variable=x, command=infrared).pack(padx=25, anchor=NW)

master.mainloop()

# Custard Pi 2 digital output

# First we need to import the required library modules

import RPi.GPIO as GPIO

import time

# Set the GPIO pin numbering convention

GPIO.setmode(GPIO.BOARD)

# Set up the required GPIO pins as outputs

GPIO.setup(11, GPIO.OUT)

GPIO.setup(12, GPIO.OUT)

GPIO.setup(13, GPIO.OUT)

GPIO.setup(15, GPIO.OUT)

# Set all outputs low

GPIO.output(11, True)

GPIO.output(12, True)

GPIO.output(13, True)

GPIO.output(15, True)

# Wait 10 seconds

time.sleep(10)

# Set all outputs high

GPIO.output(11, False)

GPIO.output(12, False)

GPIO.output(13, False)

GPIO.output(15, False)

# Wait ten seconds

time.sleep(10)

# Set all outputs low again

GPIO.output(11, True)

GPIO.output(12, True)

GPIO.output(13, True)

GPIO.output(15, True)

# Tidy up

GPIO.cleanup()

# Custard Pi 2 analogue input

# First we need to import the required library modules

import RPi.GPIO as GPIO

import time

# Set the GPIO pin numbering convention

GPIO.setmode(GPIO.BOARD)

# Set up the required GPIO pins as outputs and inputs

GPIO.setup(24, GPIO.OUT)

GPIO.setup(23, GPIO.OUT)

GPIO.setup(19, GPIO.OUT)

GPIO.setup(21, GPIO.IN)

# Initialise the ADC chip

GPIO.output(24, True)

GPIO.output(23, False)

GPIO.output(19, True)

word1= [1, 1, 1, 1, 1]

GPIO.output(24, False)

vin = 0

for x in range (0,5):

GPIO.output(19, word1[x])

time.sleep(0.01)

GPIO.output(23, True)

time.sleep(0.01)

GPIO.output(23, False)

for x in range (0,12):

GPIO.output(23, True)

time.sleep(0.01)

bit=GPIO.input(21)

time.sleep(0.01)

GPIO.output(23, False)

value = bit \* 2 \*\* (12-x-1)

vin = vin + value

print x, bit, value, vin

GPIO.output(24, True)

volt = vin \* 3.3/4096

print volt

GPIO.cleanup()

import sys

sys.exit()