**Teach-In 2014 with Raspberry Pi: Part 5**

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.

import pickle

stockfile = 'stock.dat'

stocklist = ['2N5401', 'MPSA42', 'BC548', 'BC557', 'J112']

file = open(stockfile, 'wb')

pickle.dump(stocklist, file)

file.close()

file = open(stockfile, 'rb')

stocklist = pickle.load(file)

# pickletest1.py

# import the pickle module

import pickle

# item to be pickled

loc = 'Locator IO91sa'

# open the file for writing in binary mode

file = open('locdata.dat', 'wb')

# pickle the item and dump it in the file

pickle.dump(loc, file)

# and then close the file

file.close()

# unpickletest.py

# import the pickle module

import pickle

# open the file for reading in binary mode

file = open('locdata.dat', 'rb')

# unpickle the item into a new object

loc = pickle.load(file)

# and print the item that we've just read

print(loc)

# then close the file

file.close()

# pickle\_example.py

# import the pickle module

import pickle

import time

import datetime

# datestamp = datetime.datetime.today()

new\_data = [str(datetime.datetime.now()), 'Temp', 12, 'Wind', 15, 'NE']

# open the file for writing in binary mode

file = open('wxdata.dat', 'wb')

# pickle the weather data and dump it in the file

pickle.dump(new\_data, file)

# and then close the file

file.close()

# unpickle\_example.py

# open the file for reading in binary mode

file = open('wxdata.dat', 'rb')

# unpickle the weather data into a new object

old\_data = pickle.load(file)

# to print recovered weather data

print(old\_data)

# before closing the file

file.close()

import time

import RPi.GPIO as GPIO

# Configure GPIO

GPIO.setmode(GPIO.BOARD)

GPIO.setup(13, GPIO.IN) # Input

# Tell user to press ENTER to start

print("Press ENTER to start counting ....")

#Initialise the count

count = 0

while True:

print("Count = ", count)

# Wait for the input to go high

input\_state = False

while input\_state = False:

# Read input status

input\_state = GPIO.input(13)

# Input has gone high so increment the count

count = count + 1

# Wait for the input to go low

input\_state = True

while input\_state = True:

# Read input status

input\_state = GPIO.input(13)

# Input has gone low so go back to the start of the loop

import time

import RPi.GPIO as GPIO

# Configure GPIO

GPIO.setmode(GPIO.BOARD)

GPIO.setup(11, GPIO.OUT) # STOP switch

GPIO.setup(13, GPIO.IN) # START switch

# Tell user to press the start switch

print("Press START to start timing ....")

# Wait for switch press to start

switch\_state = False

while switch\_state = False:

# Read switch status

switch\_state = GPIO.input(13)

# START switch has been pressed so start timing

start = time.clock()

# Tell user to press the stop switch

print("Press STOP to start timing ....")

switch\_state = False

while switch\_state = False:

# Read switch status

switch\_state = GPIO.input(11)

# STOP switch has been pressed so calculate the elapsed time

elapsed = (time.clock() - start)

# Finally print the elapsed time

print("Elapsed time = ", elapsed)

#!/usr/bin/env python

import time

import os

import RPi.GPIO as GPIO

from gpiospiadc import \*

GPIO.setmode(GPIO.BOARD)

DEBUG = 1

# Define pins on the GPIO connector

SPICLK = 23

SPIMISO = 21

SPIMOSI = 19

SPICS = 24

# Set up the SPI

GPIO.setup(SPIMOSI, GPIO.OUT)

GPIO.setup(SPIMISO, GPIO.IN)

GPIO.setup(SPICLK, GPIO.OUT)

GPIO.setup(SPICS, GPIO.OUT)

sensor\_adc = 0

# Function to determine current light condition

def light(sensor\_value):

if sensor\_value >= 896:

condition = 'Dark'

elif sensor\_value >= 512:

condition = 'Twilight'

elif sensor\_value >= 134:

condition = 'Overcast'

elif sensor\_value >= 30:

condition = 'Dull'

else:

condition = 'Bright'

return condition

# Main loop starts here

while True:

sensor\_value = readadc(sensor\_adc, SPICLK, SPIMOSI, SPIMISO, SPICS)

print(light(sensor\_value))

time.sleep(1)

#!/usr/bin/env python

from tkinter import \*

import time

import os

import RPi.GPIO as GPIO

from gpiospiadc import \*

root = Tk()

root.title("kOHM")

resistance1 = ''

meter = Label(root, font=('arial', 36, 'bold'), bg='yellow')

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

GPIO.setmode(GPIO.BOARD)

DEBUG = 1

# Define pins on the GPIO connector

SPICLK = 23

SPIMISO = 21

SPIMOSI = 19

SPICS = 24

# Set up the SPI

GPIO.setup(SPIMOSI, GPIO.OUT)

GPIO.setup(SPIMISO, GPIO.IN)

GPIO.setup(SPICLK, GPIO.OUT)

GPIO.setup(SPICS, GPIO.OUT)

sensor\_adc = 7

def measure():

global resistance1

# get the current resistance from the SPI device

sensor\_value = readadc(sensor\_adc, SPICLK, SPIMOSI, SPIMISO, SPICS)

sensor\_resistance = (sensor\_value \* 3.3)/1024

resistance2 = sensor\_resistance

sv = '%.3f' % sensor\_resistance

# if the resistance has changed, update it

if resistance2 != resistance1:

resistance1 = resistance2

meter.config(text=sv)

# update the display every 200ms

meter.after(200, measure)

measure()

root.mainloop()

#!/usr/bin/env python

import time

import os

import RPi.GPIO as GPIO

from gpiospiadc import \*

GPIO.setmode(GPIO.BOARD)

DEBUG = 1

# Define pins on the GPIO connector

SPICLK = 23

SPIMISO = 21

SPIMOSI = 19

SPICS = 24

# Set up the SPI

GPIO.setup(SPIMOSI, GPIO.OUT)

GPIO.setup(SPIMISO, GPIO.IN)

GPIO.setup(SPICLK, GPIO.OUT)

GPIO.setup(SPICS, GPIO.OUT)

sensor\_adc = 0

while True:

sensor\_value = readadc(sensor\_adc, SPICLK, SPIMOSI, SPIMISO, SPICS)

sensor\_voltage = (sensor\_value \* 3.3)/1024

sensor\_temperature = (sensor\_voltage - 0.5) \* 100

print("%4d/1023 => %5.3f V => %4.1f deg.C" % (sensor\_value, sensor\_voltage, sensor\_temperature))

time.sleep(0.5)

#!/usr/bin/env python

import time

import os

import csv

import RPi.GPIO as GPIO

from gpiospiadc import \*

GPIO.setmode(GPIO.BOARD)

DEBUG = 1

# Define pins on the GPIO connector

SPICLK = 23

SPIMISO = 21

SPIMOSI = 19

SPICS = 24

# Set up the SPI

GPIO.setup(SPIMOSI, GPIO.OUT)

GPIO.setup(SPIMISO, GPIO.IN)

GPIO.setup(SPICLK, GPIO.OUT)

GPIO.setup(SPICS, GPIO.OUT)

sensor\_adc = 0

filename = input('Filename to save to: ')

filename = '/home/pi/Desktop/' + filename + '.csv'

samples = input('Number of samples: ')

interval = input('Interval between samples (seconds): ')

void = input('Press enter to start collecting data ...')

print('Thanks - now collecting data ....')

count = 1

while (count < (int(samples) + 1)):

print('Sample No. ' + str(count))

sensor\_value = 0

sensor\_value1 = 1

while sensor\_value1 != sensor\_value:

sensor\_value = readadc(sensor\_adc, SPICLK, SPIMOSI, SPIMISO, SPICS)

sensor\_value1 = readadc(sensor\_adc, SPICLK, SPIMOSI, SPIMISO, SPICS)

sensor\_voltage = (sensor\_value \* 3.3)/1024

sensor\_temperature = (sensor\_voltage - 0.5) \* 100

sensor\_data = round(sensor\_temperature, 1)

with open(filename, 'a', newline='') as f:

writer = csv.writer(f)

writer.writerow([count] + [sensor\_data])

time.sleep(int(interval))

count = count + 1

print('Finished - ' + str(count - 1) + ' samples collected')

sudo python3 /home/pi/Desktop/Work\_files/temperature.py

# Python 2.7

print "Fetching data – please wait"

# Python 3.3

print("Fetching data – please wait")

# Python 2.7

print "Fetching data – please wait",

# Python 3.3

print("Fetching data – please wait", end = "")

# Python 2.7

password = raw\_input()

# Python 3.3

password = input()

# Python 2.7

password = raw\_input("Enter your password: ")

# Python 3.3

password = input("Enter your password: ")

# Python 2.7 and 3.3

message = "User name: %s Password: %s" % ("EPE", "everyday13")

User name: EPE Password: everyday13

# Python 3.3

message = "User name: {0} Password: {1}".format("EPE", "everyday13")

# Python 3.3

message = "User name: {1} Password: {0}".format("EPE", "everyday13")

User name: everyday13 Password: EPE

# Python 3.3

message = "Direction: {0} {1} {2} {3}".format("N", "E", "S", "W")

Direction: N E S W

# Python 3.3

message = "Direction: {0} {3} {2} {1}".format("N", "E", "S", "W")

Direction: N W S E