Class – TE B

PL 3

Title – Elevator

Input –

Import standard python libraries

import sys

import time

import select

LED\_1 = (0 \* 32) + 3

LED\_2 = (0 \* 32) + 23

LED\_3 = (0 \* 32) + 2

LED\_4 = (0 \* 32) + 26

LED\_5 = (1 \* 32) + 17

LED\_6 = (1 \* 32) + 15

LED\_7 = (0 \* 32) + 15

LED\_8 = (1 \* 32) + 14

LED\_9 = (0 \* 32) + 30

LED\_10 = (2 \* 32) + 2

LED\_11 = (1 \* 32) + 28

LED\_12 = (2 \* 32) + 3

LED\_13 = (0 \* 32) + 31

LED\_14 = (2 \* 32) + 5

LED\_15 = (1 \* 32) + 18

SW\_1 = (0 \* 32) + 14

SW\_2 = (0 \* 32) + 27

SW\_3 = (0 \* 32) + 22

SW\_4 = (2 \* 32) + 1

# DIRECTIN LEDS: to represent lift direction (up or down)

LIFT\_DIR\_1 = LED\_9

LIFT\_DIR\_2 = LED\_10

LIFT\_DIR\_3 = LED\_11

LIFT\_DIR\_4 = LED\_12

LIFT\_DIR\_5 = LED\_13

LIFT\_DIR\_6 = LED\_14

LIFT\_DIR\_7 = LED\_15

# POSITION LEDS: LEDs to indicate the current position of Lift

LIFT\_POS\_0 = LED\_5

LIFT\_POS\_1 = LED\_6

LIFT\_POS\_2 = LED\_7

LIFT\_POS\_3 = LED\_8

# LIFT BUTTONS: corresponding to each floor of the Lift

LIFT\_BUTTON\_0 = SW\_1

LIFT\_BUTTON\_1 = SW\_2

LIFT\_BUTTON\_2 = SW\_3

LIFT\_BUTTON\_3 = SW\_4

# LIFT LEDS: indication for BUTTON Press on each floor

LIFT\_LED\_0 = LED\_1

LIFT\_LED\_1 = LED\_2

LIFT\_LED\_2 = LED\_3

LIFT\_LED\_3 = LED\_4

# An array of DIRECTIN LEDS

dir\_leds = [ LIFT\_DIR\_1,

LIFT\_DIR\_2,

LIFT\_DIR\_3,

LIFT\_DIR\_4,

LIFT\_DIR\_5,

LIFT\_DIR\_6,

LIFT\_DIR\_7

]

# An array of POSITION LEDS

pos\_leds = [

LIFT\_POS\_0,

LIFT\_POS\_1,

LIFT\_POS\_2,

LIFT\_POS\_3

]

# An array of BUTTON PRESS LEDS

lift\_leds = [

LIFT\_LED\_0,

LIFT\_LED\_1,

LIFT\_LED\_2,

LIFT\_LED\_3

]

# An array of lift BUTTONs

lift\_buttons = [

LIFT\_BUTTON\_0,

LIFT\_BUTTON\_1,

LIFT\_BUTTON\_2,

LIFT\_BUTTON\_3

]

NO\_OF\_FLOORS = 4 # No of floors for Lift Simulation Operation

NO\_OF\_DIR\_LEDS = 7 # No of LEDs used for the lift direction (on Board)

DEFAULT\_LIFT\_POS = 0 # The floor no where lift is positioned when program is executed

# Array associated with each floor having 4 elements; each again having 3 elements fd, button and led

floor\_set = [

{"fd":-1, "button":LIFT\_BUTTON\_0, "led":LIFT\_LED\_0}, # fd, Button and LED for 0th (Ground) Floor

{"fd":-1, "button":LIFT\_BUTTON\_1, "led":LIFT\_LED\_1}, # fd, Button and LED for 1st Floor

{"fd":-1, "button":LIFT\_BUTTON\_2, "led":LIFT\_LED\_2}, # fd, Button and LED for 2nd Floor

{"fd":-1, "button":LIFT\_BUTTON\_3, "led":LIFT\_LED\_3} # fd, Button and LED for 3rd Floor

]

# PATH of a GPIO specific sysfsinterfce directory on Linux system

SYSFS\_GPIO\_DIR = "/sys/class/gpio"

defgpioExport (gpio):

try:

fo = open(SYSFS\_GPIO\_DIR + "/export","w")

fo.write(gpio)

fo.close()

return

except IOError:

return

defgpioUnexport (gpio):

try:

fo = open(SYSFS\_GPIO\_DIR + "/unexport","w")

fo.write(gpio)

fo.close()

return

except IOError:

return

defgpioSetDir (gpio, flag):

try:

fo = open(SYSFS\_GPIO\_DIR + "/gpio" + gpio + "/direction" ,"w")

fo.write(flag)

fo.close()

return

except IOError:

return

defgpioSetVal (gpio, val):

try:

fo = open(SYSFS\_GPIO\_DIR + "/gpio" + gpio + "/value" ,"w")

fo.write(val)

fo.close()

return

except IOError:

return

defgpioSetEdge (gpio, flag):

try:

fo = open(SYSFS\_GPIO\_DIR + "/gpio" + gpio + "/edge" ,"w")

fo.write(flag)

fo.close()

return

except IOError:

return

defliftLEDExit (gpio):

gpioSetVal(gpio, val="0")

gpioUnexport(gpio)

return

defliftLEDInit (gpio):

gpioExport(gpio)

gpioSetDir(gpio, flag="out")

gpioSetVal(gpio, val="0")

return

defliftLEDOn (gpio):

gpioSetVal(gpio, val="1")

return

defliftLEDOff (gpio):

gpioSetVal(gpio, val="0")

return

defliftButtonExit (gpio):

gpioUnexport(gpio)

return

defliftButtonInit (gpio):

gpioExport(gpio)

gpioSetDir(gpio, flag="in")

gpioSetEdge(gpio, flag="falling")

return

defliftInitAll():

for i in range(0, NO\_OF\_DIR\_LEDS):

liftLEDInit(str(dir\_leds[i]))

for i in range(0, NO\_OF\_FLOORS):

liftLEDInit(str(pos\_leds[i]))

liftLEDInit(str(lift\_leds[i]))

liftButtonInit(str(lift\_buttons[i]))

return

defliftExitAll():

for i in range(0, NO\_OF\_DIR\_LEDS):

liftLEDExit(str(dir\_leds[i]))

for i in range(0, NO\_OF\_FLOORS):

liftLEDExit(str(pos\_leds[i]))

liftLEDExit(str(lift\_leds[i]))

liftButtonExit(str(lift\_buttons[i]))

print "\n=== Demonstration END ===\n"

return

defliftDefaultPos():

liftLEDOn(str(pos\_leds[DEFAULT\_LIFT\_POS]))

return

defliftDirUp():

for i in range(0, NO\_OF\_DIR\_LEDS):

liftLEDOn(str(dir\_leds[i]))

time.sleep(0.5)

for i in range(0, NO\_OF\_DIR\_LEDS):

liftLEDOff(str(dir\_leds[i]))

return

defliftDirDown():

for i in range(NO\_OF\_DIR\_LEDS, 0, -1):

liftLEDOn(str(dir\_leds[i-1]))

time.sleep(0.5)

for i in range(0, NO\_OF\_DIR\_LEDS):

liftLEDOff(str(dir\_leds[i]))

return

defGetButtonVal():

try:

fo0 = open(SYSFS\_GPIO\_DIR + "/gpio" + str(LIFT\_BUTTON\_0) + "/value" ,"r") # Open and get file descriptor of button file of 0th floor

fo0.read(2) # Make dummy read() call on it

floor\_set[0]["fd"] = fo0 # store fd in 0th element of floor\_set array

fo1 = open(SYSFS\_GPIO\_DIR + "/gpio" + str(LIFT\_BUTTON\_1) + "/value" ,"r") # Open and get file descriptor of button file of 1st floor

fo1.read(2) # Make dummy read() call on it

floor\_set[1]["fd"] = fo1 # store fd in 1st element of floor\_set array

fo2 = open(SYSFS\_GPIO\_DIR + "/gpio" + str(LIFT\_BUTTON\_2) + "/value" ,"r") # Open and get file descriptor of button file of 2nd floor

fo2.read(2) # Make dummy read() call on it

floor\_set[2]["fd"] = fo2 # store fd in 2nd element of floor\_set array

fo3 = open(SYSFS\_GPIO\_DIR + "/gpio" + str(LIFT\_BUTTON\_3) + "/value" ,"r") # Open and get file descriptor of button file of 3rd floor

fo3.read(2) # Make dummy read() call on it

floor\_set[3]["fd"] = fo3 # store fd in 3rd element of floor\_set array

print "\nWaiting for button press ..."

# Call to select(). Program will block for input and it will return once button is pressed.

# We are passing three lists (read fds, write fds and exception fds) to select() function

# It will retrun three lists as r, w and ex respectively

# In this case first two lists are NULL, In third list fds associated with lift buttons are passed

# select() will retrun NULL in r & w and list of fds on which data is available in ex

r, w, ex = select.select([], [], [fo0, fo1, fo2, fo3])

# Detect on which floor button is pressed

for i in range(len(floor\_set)): # Run a loop for all (4) floors

if floor\_set[i]["fd"] in ex: # Check current fd is present in ex (ex=list of fds on which data is available)

print "LIFT button is pressed for floor #%d" % i # Print the floor no indicated by i

liftLEDOn(str(floor\_set[i] ["led"])) # Glow the corresponding LED for floor indicated by i to show button press event

time.sleep(1) # Wait for 1 second

but = i # Store value of i in but variable

fo = floor\_set[i]["fd"] # Get the current fd from array

fo.seek(0, 0); # Call seek() so that fd will point at begining of the file

str1 = fo.read(1) # Make dummy read() call on it

fo0.close() # Close fd for 0th floor

fo1.close() # Close fd for 1st floor

fo2.close() # Close fd for 2nd floor

fo3.close() # Close fd for 3rd floor

return but

except IOError:

return

try:

print "\nLift Operation Simulation using Python\n"

print "-----------------------------------------------\n"

liftInitAll() # Initialize all lift Buttons and LEDs

liftDefaultPos() # Set dafault position of the lift (0th floor)

cur\_flr = DEFAULT\_LIFT\_POS # Variable for current lift floor (initially 0)

while True:

new\_flr = GetButtonVal() # Get a new floor value by detecting a floor no to which button user calls the lift

if new\_flr>cur\_flr: # if (new floor > current floor) means lift is called to upper floor

tmp = cur\_flr # store current floor no into tmp variable

print "LIFT going UP to floor #%d" %new\_flr # print destination floor

while (tmp != new\_flr): # Use tmp value (incremental); till it becomes destination

liftDirUp() # Glow direction LEDs in upward direction

time.sleep(0.01) # sleep for 10 ms

liftLEDOff(str(pos\_leds[tmp])) # Turn off position LED at the floor pointed by tmp

tmp += 1 # Increment tmp value by 1

liftLEDOn(str(pos\_leds[tmp])) # Turn ON position LED at the floor pointed by tmp. Lift is one floor UP

time.sleep(0.5) # Sleep for 0.5 second (500 ms)

elifnew\_flr<cur\_flr: # if (new floor < current floor) means lift is called to lower floor

tmp = cur\_flr # store current floor no into tmp variable

print "LIFT going DOWN to floor #%d" %new\_flr # print destination floor

while (tmp != new\_flr): # Use tmp value (decremental); till it becomes destination

liftDirDown() # Glow direction LEDs in downward direction

time.sleep(0.01) # Sleep for 10 ms

liftLEDOff(str(pos\_leds[tmp])) # Turn off position LED at the floor pointed by tmp

tmp -= 1 # Decrement tmp value by 1

liftLEDOn(str(pos\_leds[tmp])) # Turn ON position LED at the floor pointed by tmp. Lift is one floor DOWN

time.sleep(0.5) # sleep for 0.5 second (500 ms)

cur\_flr = new\_flr # Once lift reaches the destination; current floor points to destination floor no

liftLEDOff(str(lift\_leds[new\_flr])) # Turn OFF button press indication LED of the destinaton floor

time.sleep(1) # Sleep for 1 second

liftExitAll() # Clean up all GPIOs

exit() # Exit from Program

except KeyboardInterrupt: # CTRL-C Exception Handler to cleanup and exit safely from program

liftExitAll()

print "Program Exit due to CTRL-C"

exit()

sys.exit(0)