

SMART FARMER – IoT ENABLED SMART FARMER APPLICATION

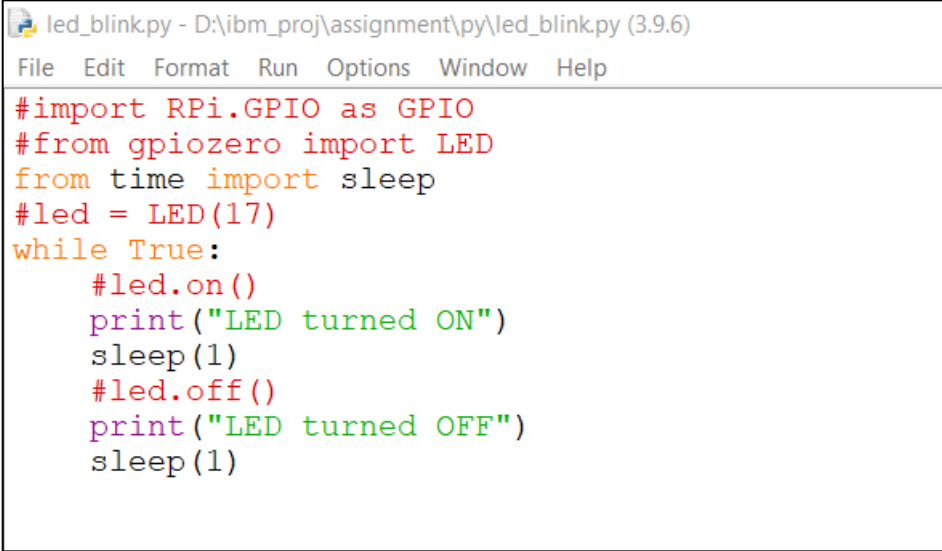
ASSIGNMENT - 3

Write a python code for blinking LED and Traffic Lights for Raspberry Pi.

(i) Python Code for Blinking LED:

```
#import RPi.GPIO as GPIO
#from gpiozero import LED
from time import sleep
#led = LED(17)
while True:
    #led.on()
    print("LED turned ON")
    sleep(1)
    #led.off()
    print("LED turned OFF")
    sleep(1)
```

Editor Window:

A screenshot of a Python IDE window titled 'led_blink.py - D:\ibm_proj\assignment\py\led_blink.py (3.9.6)'. The window has a menu bar with 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. The code is displayed with syntax highlighting: comments are in red, keywords in orange, strings in green, and identifiers in black. The code is identical to the one provided in the previous block.

```
led_blink.py - D:\ibm_proj\assignment\py\led_blink.py (3.9.6)
File Edit Format Run Options Window Help
#import RPi.GPIO as GPIO
#from gpiozero import LED
from time import sleep
#led = LED(17)
while True:
    #led.on()
    print("LED turned ON")
    sleep(1)
    #led.off()
    print("LED turned OFF")
    sleep(1)
```

Output Window:

A screenshot of an IDLE Shell 3.9.6 window. The window has a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area displays a list of 18 lines of text, each starting with 'LED turned' followed by either 'ON' or 'OFF'. The text is color-coded: 'ON' is blue and 'OFF' is red. The sequence of states is: ON, OFF, ON, OFF, ON, OFF, ON, OFF, ON, OFF, ON, OFF, ON, OFF, ON, OFF, ON, OFF. The window has standard Windows-style window controls (minimize, maximize, close) in the top right corner.

```
*IDLE Shell 3.9.6*
File Edit Shell Debug Options Window Help
LED turned ON
LED turned OFF
LED turned ON
LED turned OFF
LED turned ON
LED turned OFF
LED turned ON
LED turned OFF
LED turned ON
LED turned OFF
LED turned ON
LED turned OFF
LED turned ON
LED turned OFF
LED turned ON
LED turned OFF
LED turned ON
LED turned OFF
```

(ii) Python Code for Traffic Lights:

```
import RPi.GPIO as GPIO
import time
import signal
import sys

#setup
GPIO.setmode(GPIO.BCM)
GPIO.setup(9, GPIO.OUT)
GPIO.setup(10, GPIO.OUT)
GPIO.setup(11, GPIO.OUT)

#Turn off all lights
def allLightOff(signal, frame):
    GPIO.output(9,False)
    GPIO.output(10,False)
    GPIO.output(11,False)
    GPIO.cleanup()
    sys.exit(0)

signal.signal(signal.SIGINT, allLightsOff)

#Forever Loop
while True:
    #Red
```

```
GPIO.output(9, True)
time.sleep(3)
GPIO.output(10, True)
time.sleep(1)
#Green
GPIO.output(9,False)
GPIO.output(10,False)
GPIO.output(11,True)
time.sleep(5)
#Amber
GPIO.output(11,False)
GPIO.output(10,True)
time.sleep(2)
#Amber off
GPIO.output(10,False)
```

Editor Window:

```
import RPi.GPIO as GPIO
import time
import signal
import sys
#setup
GPIO.setmode(GPIO.BCM)
GPIO.setup(9, GPIO.OUT)
GPIO.setup(10, GPIO.OUT)
GPIO.setup(11, GPIO.OUT)
#Turn off all lights
def allLightOff(signal, frame):
    GPIO.output(9,False)
    GPIO.output(10,False)
    GPIO.output(11,False)
    GPIO.cleanup()
    sys.exit(0)
signal.signal(signal.SIGINT, allLightsOff)
#Forever Loop
while True:
    #Red
    GPIO.output(9, True)
    time.sleep(3)
    GPIO.output(10, True)
    time.sleep(1)
    #Green
    GPIO.output(9,False)
    GPIO.output(10,False)
    GPIO.output(11,True)
    time.sleep(5)
    #Amber
    GPIO.output(11,False)
    GPIO.output(10,True)
    time.sleep(2)
    #Amber off
    GPIO.output(10,False)
```