

TRAFFIC MANAGEMENT

PHASE-5

PROJECT DOCUMENTATION & SUBMISSION

OBJECTIVE :

In 2014, 54% of the total global population was urban residents .The prediction was a growth of nearly 2% each year until 2023 leading to more pressure on the transportation system of cities .Cities should be making their streets run smarter instead of just making them bigger or building more roads .This leads to the proposed system which will use a Raspberry pi and camera for tracking the number of vehicles leading to time-based monitoring of the system.

SOURCE CODE :

```
import pyfirmata

import time  comp='COM8'

board =
pyfirmata.Arduino(comp)

led_1=board.get_pin('d:2:o')
led_2=board.get_pin('d:3:o')
led_3=board.get_pin('d:4:o')
```

```
led_4=board.get_pin('d:5:o')
```

```
led_5=board.get_pin('d:6:o')
```

```
led_6=board.get_pin('d:7:o')
```

```
l1=[0,0]
```

```
def
```

```
normaltime():
```

```
return 2 def
```

```
maxtime():
```

```
return 10 def
```

```
val1():
```

```
return l1[0]
```

```
def val2():
```

```
return l1[1]
```

```
def
```

```
signalamb1():
```

```
l1[0]=1
```

```
l1[1]=0
```

```
led_2.write(0)
```

```
led_5.write(0)
```

```
led_3.write(1)
```

```
led_4.write(1)
```

```
def
```

```
signalamb2():
```

```
l1[0]=0
```

```
l1[1]=1
```

```
led_2.write(0)
```

```
led_6.write(1)
```

```
led_1.write(1)
```

```
led_5.write(0)
```

```
def signal1():
```

```
l1[0]=1
```

```
    l1[1]=0
```

```
led_2.write(0)
```

```
led_5.write(0)
```

```
led_3.write(1)
```

```
led_4.write(1)
```

```
def wait():
```

```
l1[0]=0
```

```
l1[1]=0
```

```
led_2.write(1)
```

```
led_3.write(0)
```

```
led_4.write(0)
```

```
led_5.write(1)
```

```
led_1.write(0)
```

```
led_6.write(0)
def signal2():
    l1[0]=0
    l1[1]=1
    led_2.write(0)
    led_6.write(1)
    led_1.write(1)
    led_5.write(0)
    def
    led_norm():
        l1[0]=1
        l1[1]=0
        led_2.write(0)
        led_5.write(0)
        led_3.write(1)
        led_4.write(1)

        time.sleep(5)
    led_2.write(1)
    led_3.write(0)
    led_4.write(0)
    led_5.write(1)
    time.sleep(1)
```

```
l1[0]=1
l1[1]=0
led_2.write(0)
led_6.write(1)
led_1.write(1)
led_5.write(0)
time.sleep(5)
led_2.write(1)
led_1.write(0)
led_6.write(0)
led_5.write(1)
time.sleep(1)
def
signalmax_1():
    l1[0]=1
    l1[1]=0
    led_2.write(0)
    led_5.write(0)
    led_3.write(1)
    led_4.write(1)
    time.sleep(15)
    led_2.write(1)
```

```
led_3.write(0)
led_4.write(0)
led_5.write(1)
time.sleep(1)
l1[0]=1
l1[1]=0
led_2.write(0)
led_6.write(1)
led_1.write(1)
led_5.write(0)
time.sleep(15)
led_2.write(1)
led_1.write(0)
led_6.write(0)
led_5.write(1)
time.sleep(1)
def
signalmax_2():
l1[0]=1 l1[1]=0
led_2.write(0)
led_5.write(0)
led_3.write(1)
```

```
led_4.write(1)
time.sleep(15)
led_2.write(1)
led_3.write(0)
led_4.write(0)
led_5.write(1)
time.sleep(1)
l1[0]=1
l1[1]=0
led_2.write(0)
led_6.write(1)
led_1.write(1)
led_5.write(0)
time.sleep(15)
led_2.write(1)
led_1.write(0)
led_6.write(0)
led_5.write(1)
time.sleep(1)

{

import cv2
from signals import *
```

```
import time
cascade_src = 'cars.xml'
video_src = 'dataset/test2.mp4'
video_src1 = 'dataset/test3.mp4'
l=[0,0]

cap =
cv2.VideoCapture("dataset/test2.mp
4")

cap1 =
cv2.VideoCapture("dataset/test3.mp
4")

car_cascade =
cv2.CascadeClassifier(cascade_src)

ret, img = cap.read()
ret1,img1 = cap1.read()

num=0

def returnval():
    return l[:]

def fun1(time1):
    now=time.time()
    timer = 0
    while timer<=time1:
        ret, img = cap.read()
        ret1,img1 = cap1.read()
```



```

    if (type(img) == type(None)):
        break

    if(type(img1)==type(None)):
        break

    gray = cv2.cvtColor(img,
cv2.COLOR_BGR2GRAY)

    gray1 = cv2.cvtColor(img1,
cv2.COLOR_BGR2GRAY)

    #cv2.imshow('video', gray)

    cars =
car_cascade.detectMultiScale(gray,
1.1, 1)

    cars1 =
car_cascade.detectMultiScale(gray1,
1.1, 1)

    for (x,y,w,h) in cars:

cv2.rectangle(img,(x,y),(x+w,y+h),(
0,0,255),2)

    for (x,y,w,h) in cars1:

cv2.rectangle(img1,(x,y),(x+w,y+h),
(0,0,255),2)

    cv2.imshow('video', img)

    cv2.imshow('video1', img1)

```

```

    if(val1()==0):
        l[0]=len(cars)
    if(val2()==0):
        l[1]=len(cars1)
    if cv2.waitKey(33) == 27:
        break

    end = time.time()
    timer = round(end-now)
    print(timer)
    cv2.destroyAllWindows()

    #return num
}

```

OUTPUT :

1. Initially, all the lights are off because of the setup function.
2. The "Loop" function starts executing.
3. It turns on the red light and turns off the yellow and green lights. The red light remains on for 5 seconds.
4. After 5 seconds, it turns off the red light and turns on the green light. The green light remains on for 5 seconds.
5. This cycle repeats indefinitely, with the traffic lights changing between red and green every 5 seconds.

So, the expected output of this code is a simulated traffic light sequence where the red light is on for 5 seconds, followed by the green light for 5 seconds, and this cycle continues in a loop. You would observe the physical LEDs connected to the GPIO pins behaving accordingly if you had the necessary hardware set up.

