**Experiment No. 10**

**Title:** Write an application using Raspberry-Pi /Beagle board to control the operation of a hardware simulated traffic signal.

**Aim:** To understand the interface between traffic signal simulation device and Raspberry Pi Board.

Demonstrate a smart traffic system using Raspberry Pi and Python.

**Hardware Requirement**:

Raspberry Pi board with an SD card and the usual peripherals, we will also  need:

* 1x Solderless breadboard
* All king of jumper leads
* 1x Tactile button
* 3x LEDs (Red,Greean and Yellow)
* 3x 330 ohm Register
* Button

or a **Simulated traffic signal board.**

**Software Requirement:** Raspbian O.S, Python.

**Theory:**

**GPIO pins**

One powerfull feature of the Raspberry Pi is the row of GPIO pins along the top edge of the boadr. GPIO stands for General-Purpose Input/Output. These pins are a physical interface between the Raspberry Pi and the outside world.

**3.3 volts**

Anything connected to these pins will always get 3.3 v of power.

**5 volts**

Anything connected to these pins will always get 5 v of power.

**GND**

Zero volts, used to complete a circuit

**GPIO**

These pins are for general-purpose use and can be configures as input or output pins

**ID\_SC/ID\_SD/DNC**

Special purpose pins.

**Lighting an LED**

LEDs are delicate little things. if you put too much current through them they will pop. To limit the current going through the LED, you should always use a register in series with it.

Long leg an LED to the Pi 3.3v and the short leg to a GND pin. Automatically LED will turn ON

**Making Traffic Light Signal**

|  |  |  |
| --- | --- | --- |
| LEDs | Raspberry Pi board Pins |  |
| NRED  NAMBER  NGREEN | 17  10  6 | LED Pin nos to make North traffic signal ON(Green signal). Function called is NState(0,0,1) |
| SRED  SAMBER  SGREEN | 18  12  7 | LED Pin nos to make South traffic signal ON(Green signal). Function called is SState(0,0,1) |
| ERED  EAMBER  EGREEN | 19  13  8 | LED Pin nos to make East traffic signal ON(Green signal). Function called is EState(0,0,1) |
| WRED  WAMBER  WGREEN | 20  16  9 | LED Pin nos to make West traffic signal ON(Green signal). Function called is WState(0,0,1) |

**Implementation:**

To make traffic signal working, we need to initialize the LEDs of all four direction signals.

Once we initialize the LED we need to decide the Rotation based on which we need our traffic signal working such as clock wise or anti clockwise.

To start with one signal, green LED should be set HIGH for that side signal. Rest all LEDs should be set LOW at that time. After specified delay, Green LED should be set LOW and Amber LED should set high and then RED LED of the same signal should set HIGH.

As soon as we set the RED LED HIGH, Green LED of next signal should be HIGH keeping all other LEDs LOW. Likewise the rotations goes on and traffic signal starts working.

We can set delay to keep Green/Red LED ON according to our requirement for each signal.

That way we can implement the traffic signal by just changing the LED pattern ON/OFF.

**Conclusion:** We have successfully created smart traffic signal system using Raspberry Pi.

**Code:**

import time

import RPi.GPIO as GPIO

TRUE = 1

#north led definition

NRED = 17

NAMBER = 10

NGREEN = 6

#east led definition

SRED = 18

SAMBER = 12

SGREEN = 7

#south led definition

ERED = 19

EAMBER = 13

EGREEN = 8

#west led definition

WRED = 20

WAMBER = 16

WGREEN = 9

GPIO.setmode(GPIO.BCM)

GPIO.setup(NRED,GPIO.OUT)

GPIO.setup(NAMBER,GPIO.OUT)

GPIO.setup(NGREEN,GPIO.OUT)

GPIO.setup(SRED,GPIO.OUT)

GPIO.setup(SAMBER,GPIO.OUT)

GPIO.setup(SGREEN,GPIO.OUT)

GPIO.setup(WRED,GPIO.OUT)

GPIO.setup(WAMBER,GPIO.OUT)

GPIO.setup(WGREEN,GPIO.OUT)

GPIO.setup(ERED,GPIO.OUT)

GPIO.setup(EAMBER,GPIO.OUT)

GPIO.setup(EGREEN,GPIO.OUT)

def NState(r,a,g):

GPIO.output(NRED,r)

GPIO.output(NAMBER,a)

GPIO.output(NGREEN,g)

def SState(r,a,g):

GPIO.output(SRED,r)

GPIO.output(SAMBER,a)

GPIO.output(SGREEN,g)

def EState(r,a,g):

GPIO.output(ERED,r)

GPIO.output(EAMBER,a)

GPIO.output(EGREEN,g)

return

def WState(r,a,g):

GPIO.output(WRED,r)

GPIO.output(WAMBER,a)

GPIO.output(WGREEN,g)

return

try:

while TRUE:

NState(1,1,1)

SState(1,1,1)

EState(1,1,1)

WState(1,1,1)

#S on

NState(1,0,0)

SState(0,0,1)

EState(1,0,0)

WState(1,0,0)

time.sleep(4)

SState(0,1,0)

time.sleep(0.5)

#E on

NState(1,0,0)

SState(1,0,0)

EState(0,0,1)

WState(1,0,0)

time.sleep(4)

EState(0,1,0)

time.sleep(0.5)

#W on

NState(1,0,0)

SState(1,0,0)

EState(1,0,0)

WState(0,0,1)

time.sleep(4)

WState(0,1,0)

time.sleep(0.5)

#N on

NState(0,0,1)

SState(1,0,0)

EState(1,0,0)

WState(1,0,0)

time.sleep(4)

NState(0,1,0)

time.sleep(0.5)

# If CTRL+C is pressed the main loop is broken

except KeyboardInterrupt:

RUNNING = False

print "\Quitting"

# Actions under 'finally' will always be called

finally:

# Stop and finish cleanly so the pins

# are available to be used again

GPIO.cleanup()