ASSIGNMENT 3

DOMAIN: IOT

TEAM ID: PNT2022TMID29654

TEAM MEMBERS:

- DHOSHINIE S - 513119106016
- JAPAKUMAR M - 513119106031
- JAYAKUMAR B - 513119106032
- GAUTAM VINAY S – 513119106302

QUESTION:

Write python code for blinking LED and Traffic lights for Raspberry pi.

CODE FOR BLINKING LED:

```
import RPi.GPIO as GPIO
import time

ledPin = 16
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
GPIO.setup(ledPin,GPIO.OUT)

while True:
    GPIO.output(ledPin, GPIO.HIGH)
    print('LED ON')
    time.sleep(1)
    GPIO.output(ledPin, GPIO.LOW)
```

```
    print('LED OFF')
    time.sleep(1)
```

CODE FOR TRAFFIC LIGHTS:

ss

```
using System;
using System.Device.Gpio;
using Almostengr.TrafficPi.LampControl.CmdLine;
using Almostengr.TrafficPi.LampControl.Services;
using Almostengr.TrafficPi.LampControl.Workers;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

namespace Almostengr.TrafficPi.LampControl
{
    public class Program
    {
        public static void Main(string[] args)
        {
            if (args.Length == 0 || args.Length >= 2)
            {
                ShowHelp();
            }
            else if (args[0] == "--manual")
            {
                ManualConsole.RunProgram();
            }
```

```
            else
            {
                CreateHostBuilder(args).Build().Run();
            }
        }

        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureServices((hostContext, services) =>
                {
                    services.AddSingleton<GpioController>();

                    services.AddSingleton<ISignalIndicationService,
SignalIndicationService>();
                    services.AddSingleton<ISensorService,
CarWaitingSensorService>();

#if RELEASE
                    services.AddSingleton<IGpioService, GpioService>();
#else
                    services.AddSingleton<IGpioService, MockGpioService>();
#endif

                    switch (args[0])
                    {
                        case "--us":
                            services.AddHostedService<UsTrafficWorker>();
                            break;
```

```
                case "--ussensor":
services.AddHostedService<UsTrafficWithSensorWorker>();
                    break;

                case "--usflasher":
services.AddHostedService<UsTrafficWithFlasherWorker>();
                    break;

                case "--rglight":
services.AddHostedService<RedLightGreenLightWorker>();
                    break;

                case "--rglightyellow":
services.AddHostedService<RedLightGreenLightWithYellowWorker>();
                    break;

                case "--flashred":
                    services.AddHostedService<FlashRedWorker>();
                    break;

                case "--flashyellow":
                    services.AddHostedService<FlashYellowWorker>();
                    break;
```

```
case "--flashgreen":
    services.AddHostedService<FlashGreenWorker>();
    break;

case "--solidred":
    services.AddHostedService<RedLightWorker>();
    break;

case "--solidyellow":
    services.AddHostedService<YellowLightWorker>();
    break;

case "--solidgreen":
    services.AddHostedService<GreenLightWorker>();
    break;

case "--alllights":
    services.AddHostedService<AllLightsWorker>();
    break;

case "--partymode":
    services.AddHostedService<PartyModeWorker>();
    break;

case "--tm1to2":
    services.AddHostedService<Tm1To2Worker>();
    break;
```

```
                case "--tm2to3":
                    services.AddHostedService<Tm2To3Worker>();
                    break;

                case "--tm4to6":
                    services.AddHostedService<Tm4To6Worker>();
                    break;

                case "--tm5to7":
                    services.AddHostedService<Tm5To7Worker>();
                    break;

                default:
                    Console.WriteLine(args[0]);
                    ShowHelp();
                    break;
            }
        });

    public static void ShowHelp()
    {
        Console.WriteLine("===== PROGRAM HELP =====");
        Console.WriteLine();
        Console.WriteLine("--us - Run the signal using the US signal
pattern");
        Console.WriteLine("--ussensor - Run the signal using the US
signal pattern with a sensor");
```

```
            Console.WriteLine("--usflasher - Run the signal using the US
signal pattern with a flasher");
            Console.WriteLine("--manual - Manually control each light");
            Console.WriteLine("--rglight - Run red light, green light");
            Console.WriteLine("--rglightyellow - Run red light, green light with
yellow");
            Console.WriteLine("--partymode - Randomly flash a signal
color(s)");
            Console.WriteLine("--flashred - Flash red signal");
            Console.WriteLine("--flashyellow - Flash yellow signal");
            Console.WriteLine("--flashgreen - Flash green signal");
            Console.WriteLine("--tm1to2 - Toastmasters 1 to 2 minute
speech");
            Console.WriteLine("--tm2to3 - Toastmasters 2 to 3 minute
speech");
            Console.WriteLine("--tm4to6 - Toastmasters 4 to 6 minute
speech");
            Console.WriteLine("--tm5to7 - Toastmasters 5 to 7 minute
speech");
            Console.WriteLine("--solidred - Solid red signal");
            Console.WriteLine("--solidyellow - Solid yellow signal");
            Console.WriteLine("--solidgreen - Solid green signal");
            Console.WriteLine("--alllights - All lights on solid");
        }


    }
}
```