Sinhgad College of Engineering

Department of Computer Engineering

Name : Prashant Kumar

Roll No : 305139 **PRN Number**: 71813716H

Class: TE **Div** : 2 **Batch** : B

Name of Laboratory: Embedded Systems and Internet Of Things Laboratory

List of Assignments

Sr.No.	Title of Assignment	Remark	Signature
1	Study of Raspberry-Pi, Beagle board, Arduino and other micro controller (History & Elevation)		
2	Study of different operating systems for Raspberry-Pi /Beagle board. Understanding the process of OS installation on Raspberry-Pi /Beagle board		
3	Study of Connectivity and configuration of Raspberry-Pi /Beagle board circuit with basic Peripherals, LEDS. Understanding GPIO and its use in program.		
4	Understanding connectivity of Beagle bone board circuit with IR sensor. Write an application to detect obstacle and notify user using LEDs.		
5	Understanding and Connectivity of Raspberry-Pi/Beagle board with camera. Write an application to capture and store the image.		
6	To write an application to and demonstrate the change in Beagle Board/ ARM Cortex A5 Microprocessor /CPU frequency or square wave of programmable frequency.		
7	Write an application using Raspberry-Pi /Beagle board to control the operation of stepper motor.		
8	Write an application using Raspberry-Pi /Beagle board to control the operation of a hardware simulated lift elevator.		
9	Develop a network based application by setting IP address on BeagleBoard/ARM Cortex A5.		
10	Create a simple web interface for Raspberry-pi/Beagle Board to connect the connected LEDs remotely through the interface.		

Assignment No - A1

Title of Program - Study of Raspberry-Pi, Beagle board, Arduino and other micro controller (History & Elevation)

Objective - To Study of Raspberry-Pi, Beagle board, Arduino and other micro controller (History & Elevation)

Theory-

Introduction to Embedded System

An **embedded system** is a computer system with a dedicated function within a larger mechanical or electrical system, often with real-time computing constraints. It is *embedded* as part of a complete device often including hardware and mechanical parts. Embedded systems control many devices in common use today. Ninety-eight percent of all microprocessors are manufactured as components of embedded systems.

Examples of properties of typical embedded computers when compared with general-purpose counterparts are low power consumption, small size, rugged operating ranges, and low per-unit cost. This comes at the price of limited processing resources, which make them significantly more difficult to program and to interact with. However, by building intelligence mechanisms on top of the hardware, taking advantage of possible existing sensors and the existence of a network of embedded units, one can both optimally manage available resources at the unit and network levels as well as provide augmented functions, well beyond those available. For example, intelligent techniques can be designed to manage power consumption of embedded systems.

Modern embedded systems are often based on microcontrollers (i.e. CPU's with integrated memory or peripheral interfaces),but ordinary microprocessors (using external chips for memory and peripheral interface circuits) are also common, especially in more-complex systems. In either case, the processor(s) used may be types ranging from general purpose to those specialized in certain class of computations, or even custom designed for the application at hand. A common standard class of dedicated processors is the digital signal processor (DSP).

Why we use embedded system?

The **uses** of **embedded systems** are virtually limitless, because every day new products are introduced to the market that utilize **embedded** computers in novel ways. In recent years, hardware such as microprocessors, microcontrollers, and FPGA chips have become much cheaper.

An **embedded system** is a computer **system** with a dedicated **function** within a larger mechanical or electrical **system**, often with real-time computing constraints. It is **embedded** as part of a complete device often including hardware and mechanical parts.

Raspberry Pi

The **Raspberry Pi** is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and in developing countries.The original model became far more popular than anticipated, selling outside its target market for uses such as robotics. It does not include peripherals (such as keyboards, mice and cases). However, some accessories have been included

in several official and unofficial bundles.

According to the Raspberry Pi Foundation, over 5 million Raspberry Pis were sold by February 2015, making it the best-selling British computer. By November 2016 they had sold 11 million units, and 12.5m by March 2017, making it the third best-selling "general purpose computer". In July 2017, sales reached nearly 15 million.

Evolution of Raspberry Pi :

The first generation (Raspberry Pi 1 Model B) was released in February 2012, followed by the simpler and cheaper Model A. In 2014, the Foundation released a board with an improved design, Raspberry Pi 1 Model B+. These boards are approximately credit-card sized and represent the standard mainline form-factor. Improved A+ and B+ models were released a year later. A “Compute Model” was released in April 2014 for embedded applications. The Raspberry Pi 2 which added more RAM was released in February 2015.

Raspberry Pi Technology

The raspberry pi comes in two models, they are model A and model B. The main difference between model A and model B is USB port. Model A board will consume less power and that does not include an Ethernet port. But, the model B board includes an Ethernet port and designed in china. The raspberry pi comes with a set of open source technologies, i.e. communication and multimedia web technologies.In the year 2014, the foundation of the raspberry pi board launched the computer module, that packages a model B raspberry pi board into module for use as a part of embedded systems, to encourage their use.

Raspberry Pi Hardware Specifications

The raspberry pi board comprises a program memory (RAM), processor and graphics chip, CPU, GPU, Ethernet port, GPIO pins, Xbee socket, UART, power source connector. And various interfaces for other external devices. It also requires mass storage, for that we use an SD flash memory card. So that raspberry pi board will boot from this SD card similarly as a PC boots up into windows from its hard disk.

Essential hardware specifications of raspberry pi board mainly include SD card containing Linux OS, US keyboard, monitor, power supply and video cable. Optional hardware specifications include USB mouse, powered USB hub, case, internet connection, the Model A or B: USB WiFi adaptor is used and internet connection to Model B is LAN cable.

Applications of Raspberry Pi

The raspberry pi boards are used in many applications like Media streamer, Arcade machine, Tablet computer, Home automation, Carputer, Internet radio, Controlling robots, Cosmic Computer, Hunting for meteorites, Coffee and also in raspberry pi based projects.

Evolution of Beagle Board :

1. **BeagleBone :**

   Announced in the end of October 2011, the BeagleBone is a barebone development board with a Sitara ARM Cortex-A8 processor running at 720 MHz, 256 MB of RAM, two 46-pin expansion connectors, on-chip Ethernet, a microSD slot, and a USB host port and multipurpose device port which includes low-level serial control and JTAG hardware debug connections, so no JTAG emulator is required. The BeagleBone was initially priced at $89 USD.

2. **BeagleBone Black**

Launched in April 23, 2013 at a price of $45. Among other differences, it increases RAM to 512 MB, the processor clock to 1 GHz, and it adds HDMI and 2 GB of eMMC flash memory. The BeagleBone Black also ships with Linux kernal 3.8, upgraded from the original BeagleBone's Linux kernel 3.2, allowing the BeagleBone Black to take advantage of Direct Rendering Manager (DRM). BeagleBone Black Revision C (released in 2014) increased the size of the flash memory to 4 GB. This enables it to ship with Debian GNU/Linux installed. Previous revisions shipped with Ångström Linux.

3. **BeagleBoard-X15**

The BeagleBoard-X15 was planned for November 2015. It is based on the TI Sitara AM5728 processor with two ARM Cortex-A15 cores running at 1.5 GHz, two ARM Cortex-M4 cores running at 212 MHz and two TI C66x DSP cores running at 700 MHz. The used processor provides USB 3.0 support and has a PowerVR Dual Core SGX544 GPU running at 532 MHz.

BeagleBoard Black

The **BeagleBoard Black** is a low-power open-source single-board computer produced by Texas Instruments in association with Digi-Key and Newark element14. The BeagleBoard was also designed with open source software development in mind, and as a way of demonstrating the Texas Instrument's OMAP3530 system-on-a-chip.[8] The board was developed by a small team of engineers as an educational board that could be used in colleges around the world to teach open source hardware and software capabilities. It is also sold to the public under the Creative Commons share-alike license. The board was designed using Cadence OrCAD for schematics and Cadence Allegro for PCB manufacturing; no simulation software was used.

Features

A modified version of the BeagleBoard called the BeagleBoard-xM started shipping on August 27, 2010. The BeagleBoard-xM measures in at 82.55 by 82.55 mm and has a faster CPU core (clocked at 1 GHz compared to the 720 MHz of the BeagleBoard), more RAM (512 MB compared to 256 MB), onboard Ethernet jack, and 4 port USB hub. The BeagleBoard-xM lacks the onboard NAND and therefore requires the OS and other data to be stored on a microSD card. The addition of the Camera port to the -xM provides a simple way of importing video via Leopard Board cameras.

Fig. BeagleBoard Black

Arduino

Arduino is a collection of three things. There are Hardware prototype platform, Arduino language and IDE & libraries. The Arduino boards are micro-controllers, not a full-fledged computer. They don't run a full operating system, but simply write the code and execute as their firmware interprets it.

Evolution of Arduino :

Arduino was created in Ivrea Italy as a Masters thesis project. The goal of Arduino was to allow nontechnical individuals to create technical projects of their own. The Arduino was intended to be affordable. Over 700,000 Arduino boards have been commercially produced since its founding.

1. **Serial Arduino**
    - Release Year - 2005
    - Processor - ATmega8
    - Frequency - 16MHz
    - Host Interface - DE-9 Serial Connection
    - Uses RS232 as an interface for programming or communication with a computer.
    - Specifically designed to be easily assembled with the most simple components.

2. **Arduino Nano**
    - Release Date - May 15, 2008
    - Processor - ATmega328 (ATmega168 before v3.0)
    - Frequency - 16 MHz
    - Host Interface - USB
    - Uses a surface - mounted processor
    - Lacks only a DC power jack and uses a Mini-B USB instead of standard on

3. **Arduino UNO**
    - Release Date - September 24, 2010, • Processor - ATmega328P
    - Frequency - 16 MHz, • Host Interface – USB, • Uses FTDI chip for USB

Advantages

Following are some of the main advantages of Arduino.

- Very easy to get started.
- Can be used for real-time applications for both hardware, software and IDE is open source.
- Not much programming knowledge needed to do basic stuff.
- It is very easy to extend and has tons of user contributed shields and libraries. Shields are available to do attractive much anything.

Difference between Arduino, Raspberry Pi and BeagleBone Black :

The Arduino, Raspberry Pi, BeagleBone and PCDuino may look quite similar for you, but they are in fact very different devices. The Arduino is a microcontroller. A microcontroller is just one tiny part of a computer. The arduino can be programmed in C, but can't run an operating system.

Comparison between Arduino, Raspberry Pi and BeagleBone Black :

Name	Arduino	RaspberryPi (version 1)	Raspberry Pi (version 3)	BeagleBone Black
Price	Less	Costly	Costly	Costlier
Processor	ATMega 328	ARM11	4*ARM Cortex-A53	ARM CORTEX-A8
Clock Speed	16 MHz	900 MHz	1.2 GHz	700 MHz
RAM	2 KB	256 MB	1GB DDR2	256 MB
Flash	32 KB	SD card	32 GB (micro SD)	4 GB (micro SD)
EEPROM	1 KB	-	-	-
I/P voltage	7-12 V	5 V	5 V	5 V
Min Power	42 mA (.3 W)	700 mA (3.5 W)	1.34 A	170 mA (.85 W)
GPIO	14	40	40	65
Analog I/P	6-10 bit	N/A	N/A	7-12 bit
Ethernet	N/A	10/100	10/100	10/100
USB Master	N/A	2 USB 2.0	4 USB 2.0	1 USB 2.0
Video Output	N/A	HDMI, Composite	HDMI	N/A
Audio Output	N/A	HDMI, Analog	HDMI, Audio	Analog
Dev IDE	Arduino tool	IDLE, Scratch, Squeak/Linux	BlueJ, Geany, AdaFruit Web IDE	Python, Scratch, Squeak, Cloud9/Linux
Serial Peripheral Interface (SPI)	1	1	2	1
Multitasking	No	Yes	Yes	Yes
Operating System	None	Linux Distributors	Arch Linux ARM, Raspbian OS, Windows 10 IOT core, Pidora	Linux, Android, Cloud9 ID

Conclusion:

Thus we have understood the concept of Raspberry-Pi, BeagleBoard Black, Arduino and Microcontroller.

Assignment No - A2

Title of Program - Study of different operating systems for Raspberry-Pi /Beagle board. Understanding the process of OS installation on Raspberry-Pi /Beagle board

Objective - To Study different operating systems for Raspberry-Pi /Beagle board and to understand the process of OS installation on Raspberry-Pi /Beagle board

Theory-

The BeagleBone Black includes a 2GB or 4GB on-board eMMC flash memory chip. It comes with the Debian distribution factory pre-installed. You can flash new operating systems including Angstrom, Ubuntu, Android, and others. The following pages will illustrate the steps to getting the latest of each type of supported distribution onto the on-board eMMC. In addition to the eMMC, you can also boot directly from a microSD card similarly to the original BeagleBone.

Operating Systems

The Raspberry Pi Foundation recommends the use of Raspbian, a Debian-based Linux operating system. Other third-party operating systems available via the official website include Ubuntu MATE, Windows 10 IoT Core, RISC OS and specialized distributions for the Kodi media centre and classroom management. Many other operating systems can also run on the Raspberry Pi.

Other operating systems (not Linux-based)

- RISC OS Pi (a special cut down version RISC OS Pico, for 16 MB cards and larger for all models of Pi 1 & 2, has also been made available.)
- FreeBSD
- NetBSD
- Plan 9 from Bell Labs and Inferno (in beta)
- Windows 10 IoT Core – a no-cost edition of Windows 10 offered by Microsoft that runs natively on the Raspberry Pi 2.
- xv6– is a modern reimplementation of Sixth Edition Unix OS for teaching purposes; it is ported to Raspberry Pi from MIT xv6; this xv6 port can boot from NOOBS.
- Haiku – is an opensource BeOS clone that has been compiled for the Raspberry Pi and several other ARM boards. Work on Pi 1 began in 2011, but only the Pi 2 will be supported.
- HelenOS – a portable microkernel-based multiserver operating system; has basic Raspberry Pi support since version 0.6.0

Other operating systems (Linux-based)

- Android Things – an embedded version of the Android operating system designed for IoT device development.
- Arch Linux ARM – a port of Arch Linux for ARM processors.
- openSUSE

- SUSE Linux Enterprise Server 12 SP2
- Raspberry Pi Fedora Remix
- Gentoo Linux
- CentOS for Raspberry Pi 2 and later
- Devuan - a version of Debian with sysvinit instead of systemd
- RedSleeve (a RHEL port) for Raspberry Pi 1
- Slackware ARM – version 13.37 and later runs on the Raspberry Pi without modification. The 128–496 MB of available memory on the Raspberry Pi is at least twice the minimum requirement of 64 MB needed to run Slackware Linux on an ARM or i386 system.
- (Whereas the majority of Linux systems boot into a graphical user interface, Slackware's default user environment is the textual shell / command line interface.) The Fluxbox window manager running under the X Window System requires an additional 48 MB of RAM.
- OpenWrt – is primarily used on embedded devices to route network traffic.
- Kali Linux – is a Debian-derived distro designed for digital forensics and penetration testing.
- SolydXK – is a light Debian-derived distro with Xfce.
- Ark OS – is designed for website and email self-hosting.
- Sailfish OS with Raspberry Pi 2 (due to use ARM Cortex-A7 CPU; Raspberry Pi 1 uses different ARMv6 architecture and Sailfish requires ARMv7.)
- Tiny Core Linux – a minimal Linux operating system focused on providing a base system using BusyBox and FLTK. Designed to run primarily in RAM.
- Alpine Linux – is a Linux distribution based on musl and BusyBox, primarily designed for "power users who appreciate security, simplicity and resource efficiency".
- Void Linux – a rolling release Linux distribution which was designed and implemented from scratch, provides images based on musl or glibc.
- Fedora 25 – supports Pi 2 and later (Pi 1 is supported by some unofficial derivatives, e.g. listed here.).
- Media centre operating systems
- Daylight Linux – An ultra-lightweight operating system with the Fluxbox interface
- Raspberry Digital Signage – An operating system designed for digital signage deployments.

OS Installation :

Step 1: Download the latest software image

Download the lastest Debian image from beagleboard.org/latest-images. The "IoT" images provide more free disk space if you don't need to use a graphical user interface (GUI). The Debian distribution is provied for the boards. The file you download will have an .img.xz extension. This is a compressed sector-by-sector image of the SD card.

Step 2: Install SD card programming utility Download and install Etcher.

Some general help on programming SD cards can be found on the Ubuntu Image Writer page.

Step 3: Connect SD card to your computer

Use your computer's SD slot or a USB adapter to connect the SD card to your computer.

Step 4: Write the image to your SD card

Use Etcher to write the image to your SD card. Etcher will transparently decompress the image on-the-fly before

writing it to the SD card.

Step 5: Eject the SD card

Eject the newly programmed SD card.

Step 6: Boot your board off of the SD card

Insert SD card into your (powered- down) board, hold down the USER/BOOT button (if using Black) and apply power, either by the USB cable or 5V adapter.

Assignment No – A3

Title of Program - Study of Connectivity and configuration of Raspberry-Pi /Beagle board circuit with basic Peripherals, LEDS. Understanding GPIO and its use in program.

Objective - To learn Connectivity and configuration of Raspberry-Pi /Beagle board circuit with basic peripherals, LEDS, GPIO and its use in program.

Code of Program –

```
1. import Adafruit_BBIO.GPIO as GPIO
2. import time
3.  GPIO.setup("P8_10", GPIO.OUT)
4. while True:
5.     GPIO.output("P8_10", GPIO.HIGH)
6.     time.sleep(0.5)
7.     GPIO.output("P8_10", GPIO.LOW)
8.     time.sleep(0.5)
```

python blink.py

We have studied Connectivity and configuration of Raspberry-Pi /Beagle board circuit with basic peripherals, LEDS and GPIO and its use in program

Assignment No – B1

Title of Program - Understanding connectivity of Beagle bone board circuit with IR sensor. Write an application to detect obstacle and notify user using LEDs.

Objective - To understand connectivity of Beagle bone board circuit with IR sensor.

Code of Program –

```
1. import RPi.GPIO as GPIO
2. import time
3. GPIO.setmode(GPIO.BCM)
4. GPIO.setwarning(False)
5. GPIO.setup(18,GPIO.OUT)
6. while True:
7.         GPIO.output(18,GPIO.HIGH)
8.         time.sleep(2)
9.         GPIO.output(18,GPIO.LOW)
10.        time.sleep(2)
```

Input:

Any obstacle that comes in the range of the sensor is detected. IR sensors actually measure the heat being emitted from the object. So the heat is the actual input for the sensors.

The IR sensor gets its input from pin number P9_12.

Output:

The output is shown by the LED and the buzzer. When an obstacle is detected, the LED glows and buzzer is turned on, i.e. whenever heat is sensed by the sensor, output is shown.

The output is shown by making the pins P8_7 (led) and P8_8 high (buzzer).

Assignment No – B2

Title of Program - Understanding and Connectivity of Raspberry-Pi/Beagle board with camera. Write an application to capture and store the image.

Objective - To understand connectivity of Beagle bone board circuit with camera and to write application to store and capture the image

Code of Program –

```
import RPi.GPIO as GPIO


GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(36, GPIO.IN)
while True :
  prox_val = GPIO.input(36)
  if prox_val :
    print("obstacle detected\n")
  else :
    print("No obstacle detected\n")
```

Compile/ Exe/ Input /Output (Clear screen, Compile, Run the program and Paste output)

Input:

```
$ sudo apt-get update
$ sudo apt-get install python-picamera python3-picamera
$ sudo apt-get upgrade
```

<u>To Capture images with PiCamera</u>

```
import picamera
camera = picamera.PiCamera()
camera.capture('image.jpg')
```

Output:

The LED glows when the preview is started. The Camera captures and stores the image at the path mentioned in the capture function. Once the image is captured LED is turned off.

Assignment No – B3

Title of Program - To write an application to and demonstrate the change in Beagle Board/ ARM Cortex A5 Microprocessor /CPU frequency or square wave of programmable frequency.

Objective - To understand the basic working principle of DC motors.
To understand how to write programs for Beagle bone Black in Python.

Code of Program –

```
import RPi.GPIO as gpio

from time import sleep

from picamera import PiCamera


gpio.setmode(gpio.BCM)

gpio.setup(17,gpio.IN)

gpio.setup(27,gpio.OUT)


if(gpio.input(17)==0):
        gpio.output(27,1)
        camera = PiCamera()
        camera.start_preview()
        camera.vflip = True
        sleep(2)
        camera.capture('foo.jpg', use_video_port=True)
        camera.stop_preview()
        camera.close()
        gpio.output(27, 0)
```

Compile/ Exe/ Input /Output (Clear screen, Compile, Run the program and Paste output)

Input and Output:

Duty Cycle		Frequency	Result
0	2	No Rotation	
1	2	Slow	
20	2	Speed Increases	
60	2	Speed Increases	
100	2	Top Speed	
60	1	Maximum Fluctuation	
60	20	Low Fluctuation	
60	100	Very Low Fluctuation	

Assignment No – C1

Title of Program - Write an application using Raspberry-Pi /Beagle board to control the operation of stepper motor.

Objective - To study working principle of stepper motor.
To understand how to write stepper motor program for Beagle bone Black in Python.

Code of Program –

```
import RPi.GPIO as GPIO
import timeGPIO.setmode(GPIO.BOARD)control_pins = [7,11,13,15]for pin in control_pins:
  GPIO.setup(pin, GPIO.OUT)
  GPIO.output(pin, 0)halfstep_seq = [
  [1,0,0,0],
  [1,1,0,0],
  [0,1,0,0],
  [0,1,1,0],
  [0,0,1,0],
  [0,0,1,1],
  [0,0,0,1],
  [1,0,0,1]
]for i in range(512):
  for halfstep in range(8):
    for pin in range(4):
      GPIO.output(control_pins[pin], halfstep_seq[halfstep][pin])
    time.sleep(0.001)GPIO.cleanup()
```

Compile/ Exe/ Input /Output (Clear screen, Compile, Run the program and Paste output)

Input and Output:

Input (Steps)	Output (Moment)
Full Step	13.84

Assignment No – C2

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

Objective - To understand the architecture of Beagle bone Black . To understand how to write programs for Beagle bone Black in Python.

Program –

```
import time
import RPi.GPIO as GPIO
red_led1 = 36
yellow_led1 = 38
green_led1 = 40
red_led2 = 8
yellow_led2 = 10
green_led2 = 12
red_led3 = 11
yellow_led3 = 13
green_led3 = 15
red_led4 = 19
yellow_led4 = 21
green_led4 = 23
lane1= int(input("lane1:"))
print(lane1)
lane2= int(input("lane2:"))
print(lane2)
lane3= int(input("lane3:"))
print(lane3)
lane4= int(input("lane4:"))
print(lane4)
RUNNING = True
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False);
GPIO.setup(red_led1, GPIO.OUT)
GPIO.setup(yellow_led1, GPIO.OUT)
GPIO.setup(green_led1, GPIO.OUT)
GPIO.setup(red_led2, GPIO.OUT)
GPIO.setup(yellow_led2, GPIO.OUT)
GPIO.setup(green_led2, GPIO.OUT)
GPIO.setup(red_led3, GPIO.OUT)
GPIO.setup(yellow_led3, GPIO.OUT)
GPIO.setup(green_led3, GPIO.OUT)
GPIO.setup(red_led4, GPIO.OUT)
GPIO.setup(yellow_led4, GPIO.OUT)
GPIO.setup(green_led4, GPIO.OUT)
def trafficState1(red1, yellow1, green1):
```

```
GPIO.output(red_led1, red1)
GPIO.output(yellow_led1, yellow1)
GPIO.output(green_led1, green1)
def trafficState2(red2, yellow2, green2):
GPIO.output(red_led2, red2)
GPIO.output(yellow_led2, yellow2)
GPIO.output(green_led2, green2)
def trafficState3(red3, yellow3, green3):
GPIO.output(red_led3, red3)
GPIO.output(yellow_led3, yellow3)
GPIO.output(green_led3, green3)
def trafficState4(red4, yellow4, green4):
GPIO.output(red_led4, red4)
GPIO.output(yellow_led4, yellow4)
GPIO.output(green_led4, green4)
print "Traffic Light Simulation. Press CTRL + C to quit"
loop
try:
   while RUNNING:
print("Green Light ON for Lane 3 ")
trafficState1(0,0,1)
trafficState2(1,0,0)
trafficState3(1,0,0)
trafficState4(1,0,0)
time.sleep(lane3)
trafficState1(0,1,0)
trafficState2(1,0,0)
trafficState3(1,0,0)
trafficState4(1,0,0)
time.sleep(5)
print("Green Light ON for Lane 4")
trafficState1(1,0,0)
trafficState2(0,0,1)
trafficState3(1,0,0)
trafficState4(1,0,0)
time.sleep(lane4)
trafficState1(1,0,0)
trafficState2(0,1,0)
trafficState3(1,0,0)
trafficState4(1,0,0)
time.sleep(5)
print("Green Light ON for Lane 1")
trafficState1(1,0,0)
trafficState2(1,0,0)
trafficState3(0,0,1)
trafficState4(1,0,0)
time.sleep(lane1)
trafficState1(1,0,0)
trafficState2(1,0,0)
trafficState3(0,1,0)
trafficState4(1,0,0)
```

```
time.sleep(5)
print("Green Light ON for Lane 2")
trafficState1(1,0,0)
trafficState2(1,0,0)
trafficState3(1,0,0)
trafficState4(0,0,1)
time.sleep(lane2)
trafficState1(1,0,0)
trafficState2(1,0,0)
trafficState3(1,0,0)
trafficState4(0,1,0)
time.sleep(5)
except KeyboardInterrupt:
RUNNING = False
print "\Quitting"
finally:
GPIO.cleanup()
```

Output: Glowing Traffic Lights

Assignment No – C3

Title of Program - Write an application using Raspberry-Pi /Beagle board to control the operation of a hardware simulated lift elevator.

Objective - To understand the operations of LIFT.
To understand how to write programs for Beagle bone Black in Python.

Code of Program –

```
import Adafruit_BBIO.GPIO as GPIO
import time

GPIO.setup("P8_7",GPIO.IN)
GPIO.setup("P8_8",GPIO.IN)
GPIO.setup("P8_9",GPIO.IN)
GPIO.setup("P8_10",GPIO.IN)
GPIO.setup("P8_11",GPIO.OUT)
GPIO.setup("P8_12",GPIO.OUT)
GPIO.setup("P8_13",GPIO.OUT)
GPIO.setup("P8_14",GPIO.OUT)
GPIO.setup("P8_15",GPIO.OUT)
GPIO.setup("P8_16",GPIO.OUT)
GPIO.setup("P8_17",GPIO.OUT)
GPIO.setup("P8_18",GPIO.OUT)
GPIO.setup("P9_11",GPIO.OUT)
GPIO.setup("P9_12",GPIO.OUT)
GPIO.setup("P9_13",GPIO.OUT)
GPIO.setup("P9_14",GPIO.OUT)
GPIO.setup("P9_15",GPIO.OUT)
GPIO.setup("P9_16",GPIO.OUT)
GPIO.setup("P9_23",GPIO.OUT)
GPIO.setup("P9_24",GPIO.OUT)

num=0
var1=0
var2=0
var3=0
var4=0
var5=0
var6=var7=var8=var9=var10=0
p=1

def led(m,ch):
        if((m>=0) & (m<=3)):
                if ch == 0:
                        GPIO.output("P9_23",GPIO.HIGH)
                        GPIO.output("P9_24",GPIO.HIGH)
                elif ch == 1:
                        GPIO.output("P9_23",GPIO.LOW)
                        GPIO.output("P9_24",GPIO.LOW)
```

```
                GPIO.output("P9_16",GPIO.LOW)
                GPIO.output("P9_15",GPIO.HIGH)
        elif ch == 2:
                GPIO.output("P9_15",GPIO.LOW)
                GPIO.output("P9_16",GPIO.LOW)
                GPIO.output("P9_14",GPIO.LOW)
                GPIO.output("P9_13",GPIO.HIGH)
        elif ch == 3:
                GPIO.output("P9_13",GPIO.LOW)
                GPIO.output("P9_14",GPIO.LOW)
                GPIO.output("P9_12",GPIO.LOW)
                GPIO.output("P9_11",GPIO.HIGH)
elif m>3:
        if ch == 3:
                GPIO.output("P9_12",GPIO.LOW)
                GPIO.output("P9_11",GPIO.HIGH)
        elif ch == 2:
                GPIO.output("P9_11",GPIO.LOW)
                GPIO.output("P9_12",GPIO.LOW)
                GPIO.output("P9_13",GPIO.LOW)
                GPIO.output("P9_14",GPIO.HIGH)
        elif ch == 1:
                GPIO.output("P9_13",GPIO.LOW)
                GPIO.output("P9_14",GPIO.LOW)
                GPIO.output("P9_15",GPIO.LOW)
                GPIO.output("P9_16",GPIO.HIGH)
        elif ch == 0:
                GPIO.output("P9_15",GPIO.LOW)
                GPIO.output("P9_16",GPIO.LOW)
                GPIO.output("P9_23",GPIO.HIGH)
                GPIO.output("P9_24",GPIO.HIGH)


while True:
        #print(GPIO.input("P8_7"),GPIO.input("P8_8"),GPIO.input("P8_9"),GPIO.input("P8_10"))
        if (GPIO.input("P8_10")==0 and var1==0 ):#0
                GPIO.output("P8_11",GPIO.LOW)
                GPIO.output("P8_12",GPIO.LOW)
                GPIO.output("P8_13",GPIO.LOW)
                GPIO.output("P8_14",GPIO.LOW)
                GPIO.output("P8_15",GPIO.LOW)
                GPIO.output("P8_16",GPIO.LOW)
                GPIO.output("P8_17",GPIO.HIGH)
                GPIO.output("P8_18",GPIO.HIGH)
                if var6==1:
                        led(4,0)
                else:
                        led(0,0)
                time.sleep(1)
                var2=var6=0
                var1=1
```

```
                var4=var5=var3=p=1
                var7=var8=var9=1
elif (GPIO.input("P8_8")==0 and var6==0):#1
                GPIO.output("P8_11",GPIO.HIGH)
                GPIO.output("P8_12",GPIO.LOW)
                GPIO.output("P8_13",GPIO.LOW)
                GPIO.output("P8_14",GPIO.HIGH)
                GPIO.output("P8_15",GPIO.HIGH)
                GPIO.output("P8_16",GPIO.HIGH)
                GPIO.output("P8_17",GPIO.HIGH)
                GPIO.output("P8_18",GPIO.HIGH)
                if (var3==1 and p==0):
                        led(4,1)
                else:
                        led(0,1)
                time.sleep(1)
                num=0
                var1=var3=var7=p=0
                var6=var5=1
elif (GPIO.input("P8_9")==0 and var3==0):#2
                GPIO.output("P8_11",GPIO.LOW)
                GPIO.output("P8_12",GPIO.LOW)
                GPIO.output("P8_13",GPIO.HIGH)
                GPIO.output("P8_14",GPIO.LOW)
                GPIO.output("P8_15",GPIO.LOW)
                GPIO.output("P8_16",GPIO.HIGH)
                GPIO.output("P8_17",GPIO.LOW)
                GPIO.output("P8_18",GPIO.HIGH)
                if (var5 == 1 and p==1):
                        led(4,2)
                else:
                        led(0,2)
                time.sleep(1)
                var6=var5=var4=p=0
                var3=var1=1
elif (GPIO.input("P8_7")==0 and var5==0):#3
                GPIO.output("P8_11",GPIO.LOW)
                GPIO.output("P8_12",GPIO.LOW)
                GPIO.output("P8_13",GPIO.LOW)
                GPIO.output("P8_14",GPIO.LOW)
                GPIO.output("P8_15",GPIO.HIGH)
                GPIO.output("P8_16",GPIO.HIGH)
                GPIO.output("P8_17",GPIO.LOW)
                GPIO.output("P8_18",GPIO.HIGH)
                led(0,3)
                time.sleep(1)
                var1=var6=var5=var4=p=1
                var3=var9=var8=0
elif (GPIO.input("P8_8")==0 and var8==0):#3-1
                GPIO.output("P8_11",GPIO.LOW)
                GPIO.output("P8_12",GPIO.LOW)
```

```
                    GPIO.output("P8_13",GPIO.HIGH)
                    GPIO.output("P8_14",GPIO.LOW)
                    GPIO.output("P8_15",GPIO.LOW)
                    GPIO.output("P8_16",GPIO.HIGH)
                    GPIO.output("P8_17",GPIO.LOW)
                    GPIO.output("P8_18",GPIO.HIGH)
                    led(4,2)
                    time.sleep(1)
                    GPIO.output("P8_11",GPIO.HIGH)
                    GPIO.output("P8_12",GPIO.LOW)
                    GPIO.output("P8_13",GPIO.LOW)
                    GPIO.output("P8_14",GPIO.HIGH)
                    GPIO.output("P8_15",GPIO.HIGH)
                    GPIO.output("P8_16",GPIO.HIGH)
                    GPIO.output("P8_17",GPIO.HIGH)
                    GPIO.output("P8_18",GPIO.HIGH)
                    led(4,1)
                    time.sleep(1)
                    var1=var3=var7=0
                    var8=var5=1
elif (GPIO.input("P8_7")==0 and var7==0):#1-3
                    GPIO.output("P8_11",GPIO.LOW)
                    GPIO.output("P8_12",GPIO.LOW)
                    GPIO.output("P8_13",GPIO.HIGH)
                    GPIO.output("P8_14",GPIO.LOW)
                    GPIO.output("P8_15",GPIO.LOW)
                    GPIO.output("P8_16",GPIO.HIGH)
                    GPIO.output("P8_17",GPIO.LOW)
                    GPIO.output("P8_18",GPIO.HIGH)
                    led(0,2)
                    time.sleep(1)
                    GPIO.output("P8_11",GPIO.LOW)
                    GPIO.output("P8_12",GPIO.LOW)
                    GPIO.output("P8_13",GPIO.LOW)
                    GPIO.output("P8_14",GPIO.LOW)
                    GPIO.output("P8_15",GPIO.HIGH)
                    GPIO.output("P8_16",GPIO.HIGH)
                    GPIO.output("P8_17",GPIO.LOW)
                    GPIO.output("P8_18",GPIO.HIGH)
                    led(0,3)
                    time.sleep(1)
                    var3=var8=var9=0
                    var1=var5=var7=1
elif (GPIO.input("P8_10")==0 and var9==0):#3-0
                    GPIO.output("P8_11",GPIO.LOW)
                    GPIO.output("P8_12",GPIO.LOW)
                    GPIO.output("P8_13",GPIO.HIGH)
                    GPIO.output("P8_14",GPIO.LOW)
                    GPIO.output("P8_15",GPIO.LOW)
                    GPIO.output("P8_16",GPIO.HIGH)
                    GPIO.output("P8_17",GPIO.LOW)
```

```
                        GPIO.output("P8_18",GPIO.HIGH)
                        led(4,2)
                        time.sleep(1)
                        GPIO.output("P8_11",GPIO.HIGH)
                        GPIO.output("P8_12",GPIO.LOW)
                        GPIO.output("P8_13",GPIO.LOW)
                        GPIO.output("P8_14",GPIO.HIGH)
                        GPIO.output("P8_15",GPIO.HIGH)
                        GPIO.output("P8_16",GPIO.HIGH)
                        GPIO.output("P8_17",GPIO.HIGH)
                        GPIO.output("P8_18",GPIO.HIGH)
                        led(4,1)
                        time.sleep(1)
                        GPIO.output("P8_11",GPIO.LOW)
                        GPIO.output("P8_12",GPIO.LOW)
                        GPIO.output("P8_13",GPIO.LOW)
                        GPIO.output("P8_14",GPIO.LOW)
                        GPIO.output("P8_15",GPIO.LOW)
                        GPIO.output("P8_16",GPIO.LOW)
                        GPIO.output("P8_17",GPIO.HIGH)
                        GPIO.output("P8_18",GPIO.HIGH)
                        led(4,0)
                        time.sleep(1)
                        var1=var3=var5=var9=p=1
                        var2=var6=0
elif (GPIO.input("P8_7")==0 and var2==0 ):#0-3
                        GPIO.output("P8_11",GPIO.HIGH)
                        GPIO.output("P8_12",GPIO.LOW)
                        GPIO.output("P8_13",GPIO.LOW)
                        GPIO.output("P8_14",GPIO.HIGH)
                        GPIO.output("P8_15",GPIO.HIGH)
                        GPIO.output("P8_16",GPIO.HIGH)
                        GPIO.output("P8_17",GPIO.HIGH)
                        GPIO.output("P8_18",GPIO.HIGH)
                        led(0,1)
                        time.sleep(1)
                        GPIO.output("P8_11",GPIO.LOW)
                        GPIO.output("P8_12",GPIO.LOW)
                        GPIO.output("P8_13",GPIO.HIGH)
                        GPIO.output("P8_14",GPIO.LOW)
                        GPIO.output("P8_15",GPIO.LOW)
                        GPIO.output("P8_16",GPIO.HIGH)
                        GPIO.output("P8_17",GPIO.LOW)
                        GPIO.output("P8_18",GPIO.HIGH)
                        led(0,2)
                        time.sleep(1)
                        GPIO.output("P8_11",GPIO.LOW)
                        GPIO.output("P8_12",GPIO.LOW)
                        GPIO.output("P8_13",GPIO.LOW)
                        GPIO.output("P8_14",GPIO.LOW)
                        GPIO.output("P8_15",GPIO.HIGH)
```

```
                    GPIO.output("P8_16",GPIO.HIGH)
                    GPIO.output("P8_17",GPIO.LOW)
                    GPIO.output("P8_18",GPIO.HIGH)
                    led(0,3)
                    time.sleep(1)
                    var1=var2=var4=var5=var6=1
                    var3=var8=var9=0
elif (GPIO.input("P8_9")==0 and var2==0):#0-2
                    GPIO.output("P8_11",GPIO.HIGH)
                    GPIO.output("P8_12",GPIO.LOW)
                    GPIO.output("P8_13",GPIO.LOW)
                    GPIO.output("P8_14",GPIO.HIGH)
                    GPIO.output("P8_15",GPIO.HIGH)
                    GPIO.output("P8_16",GPIO.HIGH)
                    GPIO.output("P8_17",GPIO.HIGH)
                    GPIO.output("P8_18",GPIO.HIGH)
                    led(0,1)
                    time.sleep(1)
                    GPIO.output("P8_11",GPIO.LOW)
                    GPIO.output("P8_12",GPIO.LOW)
                    GPIO.output("P8_13",GPIO.HIGH)
                    GPIO.output("P8_14",GPIO.LOW)
                    GPIO.output("P8_15",GPIO.LOW)
                    GPIO.output("P8_16",GPIO.HIGH)
                    GPIO.output("P8_17",GPIO.LOW)
                    GPIO.output("P8_18",GPIO.HIGH)
                    led(0,2)
                    time.sleep(1)
                    var4=var5=var6=p=0
                    var2=var1=var3=1
elif (GPIO.input("P8_10")==0 and var4==0):#2-0
                    GPIO.output("P8_11",GPIO.HIGH)
                    GPIO.output("P8_12",GPIO.LOW)
                    GPIO.output("P8_13",GPIO.LOW)
                    GPIO.output("P8_14",GPIO.HIGH)
                    GPIO.output("P8_15",GPIO.HIGH)
                    GPIO.output("P8_16",GPIO.HIGH)
                    GPIO.output("P8_17",GPIO.HIGH)
                    GPIO.output("P8_18",GPIO.HIGH)
                    led(4,1)
                    time.sleep(1)
                    GPIO.output("P8_11",GPIO.LOW)
                    GPIO.output("P8_12",GPIO.LOW)
                    GPIO.output("P8_13",GPIO.LOW)
                    GPIO.output("P8_14",GPIO.LOW)
                    GPIO.output("P8_15",GPIO.LOW)
                    GPIO.output("P8_16",GPIO.LOW)
                    GPIO.output("P8_17",GPIO.HIGH)
                    GPIO.output("P8_18",GPIO.HIGH)
                    led(4,0)
                    time.sleep(1)
```

var1=var3=var5=var7=p=1
var8=var9=1
var2=var6=0

Input and Output:

Sr. No.	Input	Output
1	P8_10	Ground Floor
2	P8_8	First Floor
3	P8_9	Second Floor
4	P8_7	Top Floor

Assignment No – D1

Title of Program – Develop a network based application by setting IP address on BeagleBoard/ARM Cortex A5.

Objective - To understand the concept of socket programming.
To understand how to write programs for Beagle bone Black in Python.

Code of Program –

```
from picamera import PiCamera
from time import sleep

import smtplib
from email.MIMEMultipart import MIMEMultipart
from email.MIMEText import MIMEText
from email.MIMEBase import MIMEBase
from email import encoders

camera = PiCamera()

camera.start_preview()
sleep(5)
camera.capture('/home/pi/Desktop/image.jpg')
camera.stop_preview()


fromaddr = "manisha.intelidemics@gmail.com"
toaddr = "anumahale44@gmail.com"

msg = MIMEMultipart()

msg['From'] = fromaddr
msg['To'] = toaddr
msg['Subject'] = "capture Images Send using Raspberry pi"

body = "camera capture image send successfully"

msg.attach(MIMEText(body, 'plain'))

filename = "image.jpg"
attachment = open("/home/pi/Desktop/image.jpg", "rb")

part = MIMEBase('application', 'octet-stream')
part.set_payload((attachment).read())
encoders.encode_base64(part)
part.add_header('Content-Disposition', "attachment; filename= %s" % filename)

msg.attach(part)
```

```
server = smtplib.SMTP('smtp.gmail.com', 587)
server.starttls()
server.login(fromaddr, "manisha@123")
text = msg.as_string()
server.sendmail(fromaddr, toaddr, text)
server.quit()
```

Input and Output:

	Input	Output
Server	Client Message	Server Message
Client	Server Message	Client Message

Assignment No – D2

Title of Program - Create a simple web interface for Raspberry-pi/Beagle Board to connect the connected LEDs remotely through the interface.

Objective - To create a simple web interface to control the connected LEDs remotely.

Code of Program –

```
<html>
	<head>
	</head>
	<body>
		<form method="get" action="index.php">
			<input="submit" value="OFF" name="off">
			<input="submit" value="ON" name="on">
			<input="submit" value="BLINK" name="blink">
		</form>

		<?php
		shell_exec("/usr/local/bin/gpio -g mode 17 out");
		if(isset($_GET['off']))
		{
			echo "LED is off";
			shell_exec("/usr/local/bin/gpio -g write 17 0");
		}
		else if(isset($_GET['on']))
		{
			echo "LED is on";
			shell_exec("/usr/local/bin/gpio -g write 17 1");
		}
		else if(isset($_GET['blink']))
		{
			echo "LED is blinking";
			for($x = 0;$x<=4;$x++)
			{
				shell_exec("/usr/local/bin/gpio -g write 17 1");
				sleep(1);
				shell_exec("/usr/local/bin/gpio -g write 17 0");
				sleep(1);
			}
		}
		?>

	</body>
</html>
```

Conclusion:

We have successfully implemented the web interface for Raspberry-pi to control the connected LEDs remotely through the interface.