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

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

Conclusion:

We have studied different operating systems for Raspberry-Pi /Beagle board and the process of OS installation on Beagle board

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 –

```python
import Adafruit_BBIO.GPIO as GPIO
import time
 GPIO.setup("P8_10", GPIO.OUT)
while True:
   GPIO.output("P8_10", GPIO.HIGH)
   time.sleep(0.5)
   GPIO.output("P8_10", GPIO.LOW)
   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 –

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setwarning(False)
GPIO.setup(18,GPIO.OUT)
while True:
        GPIO.output(18,GPIO.HIGH)
        time.sleep(2)
        GPIO.output(18,GPIO.LOW)
        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
```

To Capture images with PiCamera

```
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):
```

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)
```

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.