Course Title: IOT/ Embedded Systems Laboratory	Course code: CSL67
Lab Session: 1	Student Name: Swapnonil Bandyopadhyay
Title:	USN : 1MS17CS124
Credit: 0:0:1:0	Date : 28/01/2020
Faculty Signature:	Marks(out of 10):

Answer the following questions:

1) Analyse which control registers are used in this program and what does the value set mean. Set the port C ,pin 12-14 as output and give the value stored in PMD and DOUT

```
main
    ldr r1, =0x50004080
        ldr r2, =0x08
        ldr r3, =0x01
        lsls r4, r3, #30

        str r4, [r1]
        ldr r4, =0x00008000
        adds r1, r1,r2
        str r4, [r1]
stop b  stop
    end
```

Inference: The base address of the registers starts from 0x60004000 and as port C is used, the offset added is 80, to the address 0x50004080 is stored in the register r1 to mark the address of port C. The value 0x01 is loaded into register r3 to later use to set output mode of pin number 15. which is output mode. This shifted value is stored in R4 which is stored into register R1 to that is now set to updated. An integer value 0x00008000 , the address is R1 and R2 = 0x08 is added to R1. The offset of 0x08 added to R1 states the register GP10 - and what R4 is stored in R1, the output value of 0x0000 stored as of value of 0x00008000 is 8000 is DOUT register. Thus the pin 15 is in o/p mode with value 0x000080.

2)Give a brief over view on the following control registers

PMD
OFFD
DOUT
DMASK
DBEN
IMD
IEN
ISRC

(1) The PMD control register is used to set mode of a pin.
→ GPIOn-PMD when set to 000 puts the GPIOx pins in ip mode.
→ When set to 016 puts the GPIOn port pin in o/p mode.
→ GPIOn-PMD set to 106 gives open-drain mode.
→ GPIOn-PMD set to 116 gives quasi-bidirectional mode.

(2) OFFD

It is the digital i/p path disable control register. the bits are used to control if digital i/p path of corr. GPIO pin is disabled to avoid trespage in case of analog i/p signal.

Inferences: 1 = Disable I/O digital i/p path input tied to low
0 = Enable I/O digital i/p path.

(3) DOUT - o/p value

Control status of a GPIO pin when the the GPIO pin is configured as o/p / open-drain & quasi-mode -
1 = Pins will drive high
2 = Pins will drive low.

(4) DMask - Data o/p write mask.

(5) Bits used to protect the corr. register of GPIOn-DOUT bit[n]:
1 = GPIOn-Dout[n] bit is protected, 0 = updated.

(6) DBEN - Input signal de-bound enable
Used to enable debounce fn for each corr. bit. The debounce fn is valid for edge-triggered interrupt, if the ip is level triggered, it ignored.
1 = bit[n] de-bounce enabled

Course Title:**IOT/ Embedded Systems Laboratory**	Course code: **CSL67**
Lab Session: **2**	Student Name: Swapnanil Bandyopadhyay
Title: IOT - laboratory - 2	USN : 1MS17CS124
Credit: 0:0:1:0	Date :
Faculty Signature:	Marks(out of 10):

Answer the following questions:

1) Write a program to write into the port. use setportbits function to set the port A pins A12-A14 and C12 to C15

```
#include<stdio.h>
#include "NUC1xx.h"
#include "Driver\DrvGPIO.h"
#include "Driver\DrvSYS.h"
int main(void)
{UNLOCKREG();
 DrvSYS_Open(48000000);
  LOCKREG();
  while(1)
  {
    DrvGPIO_setPortBits(E_GPC,0x0000);
  }
  return 0;
  }
```

2)Write a program to identify the A port Pins.A0-A15 by making use of GetPortBits function to

```
#include <stdio.h>
#include "NUCxx.h"
#include "Driver/DrvSys.h"
#include "Driver/DrvGPIO.h"
#include "NUC1xx-LB-002\LCD_Driver.h"
int main(void)
```

```
{ int32_t number;
  char TEXT0[16] = "smpltaypad";
  char TEXT1[16];
  UNLOCKREG();
  Initial_panel();
  clr_all_panel();
  print_lcd(0,TEXT0);
while(1)
{ number = DrvGPIO_GetPortBits(E_GPA);
  sprintf(TEXT1, "%d", number);
  print_lcd(1,TEXT1);
```

Inference:

```
if (number == 0xfffc)
    print_lcd(2,"A0");
else if (number == 0xfffd)
    print_lcd(2,"A1");
else if (number == 0xfffb)
    print_lcd(2,"A2");
else if (number == 0xffff)
    print_lcd(2,"A3");
else if (number == 0xfff7)
    print_lcd(2,"A4");
else if (number == 0xfffe)
    print_lcd(2,"A5");
else if (number == 0xfff6)
    print_lcd(2,"A6");
else if (number == 0xfff4)
    print_lcd(2,"A7");
else if (number == 0xfff9)
    print_lcd(2,"A8");
else if (number == 0xfff8)
    print_lcd(2,"A9");
else if (number == 0xfffa)
    print_lcd(2,"A10");
else if (number == 0xfff5)
    print_lcd(2,"A11");
else if (number == 0xfff1)
    print_lcd(2,"A12");
else if (number == 0xfff0)
    print_lcd(2,"A13");
else if (number == 0xfff2)
    print_lcd(2,"A14");
else
    print_lcd(2,"A15");
```

Course Title:**IOT/ Embedded Systems Laboratory**	Course code: **CSL67**
Lab Session: 3	Student Name: Swaprovil. B
Title: IOT - 3	USN : 1MS17CS124
Credit: 0:0:1:0	Date :
Faculty Signature:	Marks(out of 10):

Answer the following questions:

1)Write a program to convert analog /digital data using potentiometer provided on the board/external also.make use of ADC driver file

```
#include <stdio.h>
#include "NUC1xx.h"
#include "Driver\DrvSYS.h"
#include "Driver\DrvUART.h"
#include "Driver\DrvGPIO.h"
#include "Driver\DrvADC.h"
#include "LCD_Driver.h"

int32_t main()
{
  int32 char TEXT[16];
  int32_t value;
  float one;
  UNLOCKREG();
  SYSCLK->PWRCON.XTL12M_EN = 1;
  SYSCLK->CLKSEL0.HCLK_S = 0;
  LOCKREG();
  Initial_panel();
  clr_all_panel();
  DrvADC_Open(ADC_SINGLE_END, ADC_SINGLE_OP, 0x40,
  INTERNAL_HCK, 1);
  while(1)
{ DrvADC_StartConvert();
  while(DrvADC_IsConversionDone() == FALSE)
      value = ADC->ADDR[6].RSLT & 0xFFF;
```

```
        ans = (value * 3.3) /4095;
        sprintf (TEXT, "The value = %.f", ans);
        print_lcd(0,TEXT);
    }
}
```

2)Write a program to convert analog /digital data using potentiometer provided on the board/external .make use of ADC Init function

```
#include <stdio.h>
#include "NUC100.h"
#include "Driver\DrvSYS.h"
#include "NUC100 - LB - 002\LCD_Driver.h"
void InitADC(void)
{ GPIOA -> OFFD1 = 0x00800000;
  SYSCLK -> CLKSEL1.ADC_S = 2;
  SYSCLK -> APBCLK.ADC_EN = 1;
  ADC -> ADCR.ADEN = 1;
  ADC -> ADCR.DIFFEN = 0;
  ADC -> ADCR.ADMD = 0;
  ADC -> ADCHER.CHEN = 0x80;
  ADC -> ADSR.ADF = 1;
  ADC -> ADCR.ADIE = 1;
  NVIC_EnableIRQ(ADC_IRQn);
  ADC -> ADCR.ADST = 1;
}
int main(void)
{ int32_t adc; UNLOCKREG();
  SYSCLK -> PWRCON.XTL12M_EN = 1;
  SYSCLK -> CLKSEL0.HCLK_S = 0;
```

Inference: Examine ADC control registers for storing data and selecting channel

```
  LOCKREG();
  InitADC();
  while(1){
  while(ADC->ADSR.ADF == 0)
        ADC -> ADSR.ADF = 1;
    adc = ADC -> ADDR[7].RSLT;
    print_lcd(1, adc);
    ADC -> ADCR.ADST = 1;
}}
```

Course Title: IOT/ Embedded Systems Laboratory	Course code: CSL67
Lab Session: 4	Student Name: Swaphonil. B
Title: IOT-4	USN : 1MS17CS124
Credit: 0:0:1:0	Date :
Faculty Signature:	Marks(out of 10):

Answer the following questions:

1)Features of Nodemcu and various functions that can be handled

Nodemcu is a development board which runs on the ESP8266 with the Espressif NON-OS-SDK, and hardware based on the ESP-12 module. The device features 4mb of flash memory, 80MHz of system clock, around 50K of usable ram and on-chip wifi transceiver.

Device Summary:

Microcontroller : Tensilica 32-bit RISC CPU Xtensa LX106

Operating voltage : 3.3V

Input voltage : 7-12V

Digital I/O pins : 16

ADC pins : 1

UARTS : 1

SPIS : 1

I2Cs : 1

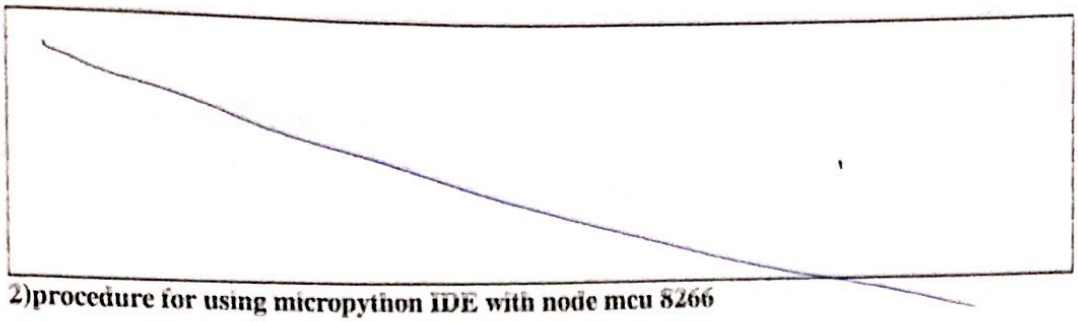
Flash memory : 4MB

SRAM : 64KB

clock speed : 80M Hz

Wifi : IEEE 802.11 b/g/n

Nodemcu can be used to connect to wifi, cloud applications, IOT projects using the board Analog to Digital conversion, pulse width modulation, controlling I-WW-devices, Neopixel, leds, various sensors and relays.



2)procedure for using micropython IDE with node mcu 8266

- Getting started with Micropython on the ESP8266
- Install uPyCraft IDE
- Flashing Micropython Firmware
- Install CP212 driver
- Open IDE
- Create main.py ; contains your code, it is executed immediately after the boot.py .
- File > Refresh Directly
- Tools > Serial > Choose COM
- Tools > Burn Firmware
- Save file
- Download and run

3)write programs in micropyton to use GPIO,ADC,PWM

~~import~~ ADC

```
import machine
adc = machine.ADC(0)
adc.read(1)
```

PWM

```
import machine
p12 = machine.Pin(2)
pw12 = machine.PWM(p12)
pw12.freq(12)
pw12.duty(10)
pw12
```

GPIO

```
pin = machine.Pin(2)
pin = machine.Pin(2, machine.Pin.OUT)
pin.value(0)
```

Course Title:IOT/ Embedded Systems Laboratory	Course code: CSL67
Lab Session: 5	Student Name: Swaptanil .B
Title: IOT - 5	USN :1MS17CS124
Credit: 0:0:1:0	Date :
Faculty Signature:	Marks(out of 10):

Answer the following questions:

1) Nodemcu and various network functions

You can establish a Wifi connection and define i/o ~~through~~ acc to your nodes exactly like an arduino, turning your ESP8266 into a web server and lot more

Network functions :

- network module is used to configure the wifi connection.
- 2 wifi interfaces : one for station network. wlan (network.STA_IF)
  one for AP network.wlan (network.AP_IF)
- Check if interfaces are active : sta_if.active()
  ap_if.active()
- check network settings : ap_if.config()
- connect to your wifi network : sta_if.connect("<SSID>", "<pass>");
- check if connection is established : sta_if.isconnected()
- disable access point interface : ap_if.active(False)
  using sockets for network tasks : important sockets
- Get IP address of server : addr = socket.getaddrinfo(host, port)[0][-1]
- make socket and connect : s = socket.socket()
  s.connect(addr)

2)write the different functions to be activated in boot.py and main.py for setting a webpage

```
#in boot.py
import socket
import network
ssid = "REPLACE WITH YOURSSID"
password = "REPLACE WITH YOUR PASSWORD"
station = network.WLAN(network.STA_IF)
station.active(True)
station.connect(ssid, password)
while not station.is_connected():
    pass
print("connection successful")
print(station.ifconfig())
```

```
#main.py
def web_page():
    html = "<html><head><title>My webpage</title>
        </head><body>Hi there! Have a nice day :)</
        body></html>"
    return html
```

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(("", 8080))
s.listen(5)
while True:
    conn, addr = s.accept()
    print("Got a connection from %s" % str(addr))
    response = web_page()
    conn.send("HTTP/1.1 200 OK\n")
    conn.send("Content-type: text/html\n")
    conn.send("Connection: close\n\n")
    conn.sendall(response)
    conn.close()
```

Course Title:IOT/ Embedded Systems Laboratory	Course code: CSL67
Lab Session: 6	Student Name:
Title: IOT - 6	USN :
Credit: 0:0:1:0	Date :
Faculty Signature:	Marks(out of 10):

Answer the following questions:

1) **Write a micro python program to light and LED with a switch**
2) **Write a micro python program to a toggle led with a switch**

```
import time
from machine import Pin
led = Pin(2, PIN.OUT)
switch = Pin(14, PIN.IN, PIN.PULL_UP)
while True:
    if not switch.value():
    time.sleep_ms(300);
    while not switches.value():
        pass
```

3) Write a micro python program to interface DHT 22 with a nodemcu and read humidity and temperature

```
from machine import Pin
from time import sleep
import dht
sensor = dht.DHT22(Pin(14))
while True:
    try:
        sleep(2)
        sensor.measure()
        temp = sensor.temperature()
        hum = sensor.humidity()
        temp_f = temp * (9/5) + 32.0
        print('temperature: %3.1f C' % temp)
        print('temperature: %3.1f F' % temp_f)
        print('humidity: %3.1f %%' % hum)
    except OSError as e:
        print('Failure', str(e))
```

3)Write a micro python program to configure nodemcu as an access point and also as a web sever

```
import socket
import network
import esp
```

```
esp.os_debug(None)
import gc
gc.collect()
ssid = 'Micropython-AP'
password = '12345678'
ap = network.WLAN(network.AP_IF)
ap.active(True)
ap.config(essid = ssid, password = password)
while(ap.active() == False)
    pass
print('connection successful')
print(ap.ifconfig())
def web_page():
    html = "<html><head></head><body>Hello there!</body></html>"
    print(ap.ifconfig())
    return html
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('',80))
s.listen(5)
while True:
    conn, addr = s.accept()
    print('got a connection from %s' % str(addr))
    request = conn.recv(1024)
    print('Content = %s' % str(request))
    response = web_page()
    conn.send(response)
    conn.close()
```

Course Title:IOT/ Embedded Systems Laboratory	Course code: CSL67
Lab Session: 7	Student Name: Swaprankl. B
Title: IOT - 7	USN : 1MJ17CS124
Credit: 0:0:1:0	Date : 24/2/2020
Faculty Signature:	Marks(out of 10):

Answer the following questions:

1) **Write a program to use RTC with NUVTon 140 board to set date and time and set alarm**

```
#include <stdio.h>
#include <string.h>
#include "NUC1xx.h"
#include "LCD_Driver.h"
#include "DrvRTC.h"
static uint8_t Alarm_E=1;
void InitRTC(void)
{ UNLOCKREG();
  SYSCLK->PWRCON.XTL32K_EN=1;
  SYSCLK->APBCLK.RTC_EN=1;
  START_RTC();
  RTC->TSSR.HR24_HR12=1;
  set_CLR(2,3,0,3,1,8);
  set_TLR(1,0,2,8,2,0);
  set_CAR(2,3,0,3,1,8);
  set_TAR(1,0,2,8,3,0);
  RTC->RIER.AIER=1;
  RTC->RIER.TIER=1;
  NVIC_EnableIRQ(RTC_IRQn);
}
void RTC_IRQHandler(void)
{ uint32_t clock, calendar;
  char TEXT_RTC[16]="RTC";
  char TEXT_RTC[16]="RTC:";
  char cal[16];
  if(inpw(&RTC->RIIR)&0x2)
{ clock=inpw(&RTC->TLR)&0xFFFFFF;
  calendar=inpw(&RTC->CLR)&0xFFFF;
  sprintf(TEXT_RTC1,"%.02x",(clock>>16)&0xFF)
  print_Ln(12,TEXT_RTC);
```

```
if (input (&RTC->RIIR) & 0x1)
{ printf lcd(2, "RIIR");
  GPIOC -> DOUT = 0xFF;
  Alarm_E = 0; }
int 32_t main()
UNLOCKREG();
SYSCLK -> PWRCON.XTLK22K_EN = 1;
```

```
LOCKREG();
Initial_panel();
clr_all_panel();
print_lcd(0, "Sample RTC");
Init RTC();
while(Alarm_E)
{ WDT -> WCTR
  WTR = 0; --NOPC(); }
}
```

2) Discuss the control registers used to set the date and time and the alarm date and time

The RTC controller provides the time message in Time loading Register (TLR) as well as calendar loading Register (CLR) also alarm function can preset the alarm time in time Alarm Register (TAR) and alarm calendar in calendar alarm register (CAR). They are all BCD counter.

INIR → RTC initialization register
AER → RTC Access enable register, Reset 0
FCR → RTC Frequency compensation register
TLR → Calendar loading register, Reset : 00:00:00
CLR → Calendar loading register, Reset : 05/11/1
TSSR → Time scale register, Reset 1
DWR → Day of the week register
TAR → Time alarm register, reset = 00:00:00
CAR → Calendar Alarm register, reset = 00:00:00
LIR → RTC leap year Indication Register

3)Write about the interrupt handlers used by the RTC

The RTC supports periodic Time Tack and Alarm match interrupts. The periodic interrupt has 8 period options 1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2 and 1 second which one selected by TTR[2:0]. When RTC counter in

TLR and CLR are equal to alarm setting time. registers TAR and CAR, the alarm interrupt flag [RIIR.AIF] is set and the alarm interrupted is requested if the alarm interrupt is enabled.
[Alarm match Interrupt]

RIER → RTC interrupt enable register
RIIR → RTC interrupt indication register
TTR → RTC Time tick register

Course Title:IOT/ Embedded Systems Laboratory	Course code: CSL67
Lab Session:8	Student Name: Swaphonil. B
Title: IOT - 8	USN : 1MS17CS124
Credit: 0:0:1:0	Date : 31/3/2020
Faculty Signature:	Marks(out of 10):

Answer the following questions:

1) **Feature of Raspberry Pi and its pin out**

The raspberry Pi is a series of small single-board computers developed in the UK by the Raspberry Pi foundation to promote teaching of basic computer science in schools and in developing countries.

Features of Raspberry Pi: (model A)

256 MB SDRAM memory

Single 2.0 USB connector

Dual core video core IV multimedia coprocessor

HDMI (rev 1.3 & 1.4) composite RCA (PAL & NTSC) video out

3.5 mm Jack, HDMI, Audio out

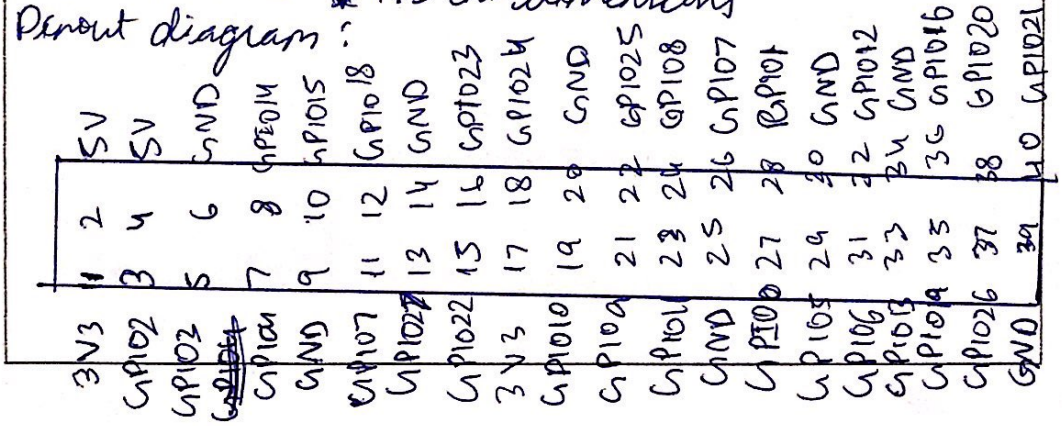
SD, MMC, SDIO card slot on board storage

Linux OS

Broadcom BCM 2835 SOC full HD multimedia processor

8.6 cm * 5.4 cm * 1.5 cm dimensions

Pinout diagram:



2) Write a blink program to run on raspberry pi / write the steps to enable SSH and run the blink program from remote terminal

```
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(11,GPIO.OUT)
while True:
    GPIO.output(11,True)
    time.sleep(1)
    GPIO.output(11,False)
    time.sleep(1)
```

SSH :- (secure shell) You can access the command line of a raspberry Pi remotely from another computer or device on the same network using SSH.

1.) Set up your local network and wireless connectivity.
2.) Enable SSH :
- Launch Raspberry Pi configuration from the preferences menu.
- Navigate to the interfaces tab
- Select "enabled" next to SSH
- click "OK".

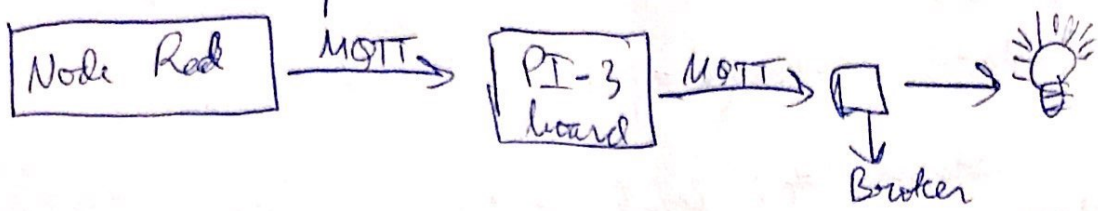
MQTT and Node-Red (Lab 9 and 10)

IMS17CS124
Roopanand.B

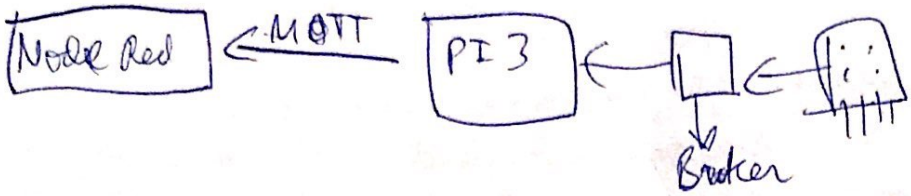
It is a simple messaging protocol, used for ~~user~~ constrained devices and low bandwidth, hence it is perfect for IOT. MQTT stands for Message Queing Telemetry transport.

1) High-level overview

1.) Control an output



2) Read and publish data from sensors or devices.



Concepts in MQTT

MQTT works on the publish-subscribe (pub-sub for short) basis

- Device 1 publishes the data
- Device 2 receives the message (command/data) by subscribing to the data being published by device 1.

2.) Concepts

- The MQTT uses the publish subscribe (pub-sub) system. A device publishes data on a topic, and the device that has subscribed to the topic, receives the published data.
- Topics are the way you register interest for the upcoming and incoming messages or to specify where you want the publish the message. They are represented with a string followed by a forward slash.
- The broker handles the job of MQTT to send, receive messages register interests for topics, etc. Mosquitto is a lightweight broker for MQTT

- Node red - It is a programming tool for wiring together hardware devices, API's and online services. Primarily, it is a visual tool designed for IOT, but it can be used for other applications to very quickly assemble flows of various services.

3) Installation & implementation

Windows (node-red)

→ npm install -g --unsafe-perm node-red
→ node-red
→ open on browser http://localhost:1880/
→ npm install node-red-dashboard

Mosquitto

- Download Mosquitto broker for windows
- install by following the steps of wizard.
- Open windows cmd/powershell and navigate to installation folder
- Ensure node js is installed.
- Copy files from installation folder to MQTT folder.
- Go to services and start server.
- Use "mosquitto-sub" to subscribe to a topic on terminal 1.

mosquitto_sub -t 'test/topic' -v

- Publish "Hello" message to topic test:

Open a new terminal and enter the following :-

$ mosquitto -sub -d -t test

On second window publish the message.

$ mosquitto -pub -d -t test -m "Hello"

We can see that both of terminals received the published message

Node-red

→ ① - Establish SSH connection with Raspberry PI, and enter the following :

$ bash <(curl -sL https://raw.githubusercontent.com/node-red/raspbian-deb-package/master/resources/update-nodejs-and-nodered

~~Test the installation~~

- Create as simple flow by using the inject and debug node. Connect them together. Add string to inject node. and then deploy.
- Open debug and click inject node. We can see the string generated.

~~Utility~~

3.) ~~Utility~~

~~Node-Red~~ MQTT :

It is used to collect data from other devices and transport it to IT infrastructure. The benefits are as of follows.

→ distribute information more efficiently.
→ reduce network bandwidth consumption.
→ reduce update rates
→ good for remote sensing.
→ saves development time.

Node-Red :

→ Access RPI GPIOs
→ Establish MQTT connection with other boards.
→ Create a responsive GUI.
→ Retrieve web data.
→ Create triggered events.
→ Store and retrieve data from database.

Difference bew MQTT and HTTP protocols

MQTT	HTTP
• Data centric	• Document centric
• Publish / Subscribe	• Request / Response
• simple	• complex
• small, with a compact binary header.	• Larger, partly because status details are text based.
• Three quality of service settings	• All messages get same level of service.
• Libraries for C and Java.	• Depends on application but typically not small.
• Support 1-1, 1-0, 1-n.	• 1-1 only.

1MS17CS124
Swaprondl. B

Lab 11:

Theory: Pub/sub is an asynchronous messaging service that decouples services that produce events from services that process events. It offers durable message storage and real time message delivery with high availability and consistent performance at scale. The core concepts involved are:

→ topic - named resource to which messages sent by publishers
→ subscription - stream of messages from a single topic, to be delivered to the subscribing application.
→ message - data and attributes that are sent by a publisher.
→ message attribute - key-value pair that a publisher can define.

Implementation + Utilities:

1.) Go to Qwiklab - Google Cloud Pub/Sub.
2) Start Lab: login using generated credentials.
3.) open google cloud command line tool: gcloud.
4.) Create pub/sub topic: gcloud pubsub topics create mytopic.
5.) list topics: gcloud pubsub topics list.
6.) delete topics: gcloud pubsub topics delete mytopic.
7.) Create a subscription: gcloud
8.) List subscriptions: gcloud pubsub topics list-subscriptions mytopic
9.) delete subscription: gcloud pubsub subscription delete test1
10.) publish message: gcloud pubsub topics publish mytopic --message "Hello".
11.) Pull subscription: gcloud pubsub subscriptions pull mysubscription --auto-ack.
12.) Limit: gcloud pubsub subscriptions pull mysubscriptions --auto-ack --limit = 3

Lab-12

IOT / Embedded systems : CSL67
Name : Swaphonil Bandyopadhyay
USN : 1MS17CS124
Date : 27/5/2020

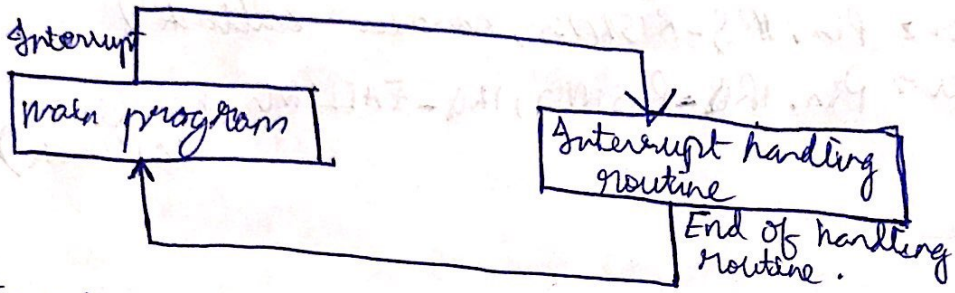
Interrupts in 8226 Microchip

Details of the interrupt pins:

In ESP8226, you can use all GPIO pins except GPIO16 as interrupt pins.

In system programming, an interrupt is a signal to the the processor emitted by hardware or software indicating an event that needs immediate attention. Pin interrupts can be used to invoke callbacks.

Signal to be used as interrupts



To raise an interrupt signal in micropython, the irq() method is used.

- You ned to set the interrupt pin as input.
- Attach an interrupt using irq():

```
pin.irq (trigger=Pin.IRQ_RISING, handler=handle_interrupt)
```

Note: IRQ_FALLING can also be used to trigger on falling edge.

The irq() method accepts the following arguments:

trigger: Pin.IRQ_RISING : Trigger on HIGH to LOW
Pin.IRQ_FALLING: Trigger on LOW to HIGH
3 : Both edges.

handler : function called when an interrupt is detected.

Interrupt handler

We define a callback function, which takes a single argument, being the pin that triggered the function.

Sample code:

```
def callback(p):
    print('Pin change', p)
```

This function is called everytime pin 'p' invokes an interrupt signal. It is recommended to keep the callback function as simple as possible.

Sample code example of interrupt 8226

```
def callback(p):
    print('pin change', p)
from machine import Pin
p0 = pin(0, Pin.IN)
p2 = Pin(2, Pin.IN)
p0.irq(trigger = Pin.IRQ_RISING, handler = callback)
p2.irq(trigger = Pin.IRQ_RISING, IRQ_FALLING, handler =
                                                callback)
```

Lab - 13

IOT / Embedded Systems
Name : Swapnil . B
USN : 1MS17CS124
Date : 27/5/2020

Deep sleep and wake up in ESP8266 using timer/external signal

Using Timer :

The ESP8266 module has the deep sleep mode which allows to put it in hibernation to save the battery.
There are slightly different ways to wake up the ESP8266 with a timer after deep sleep. One of the easiest ways is using the following functions:

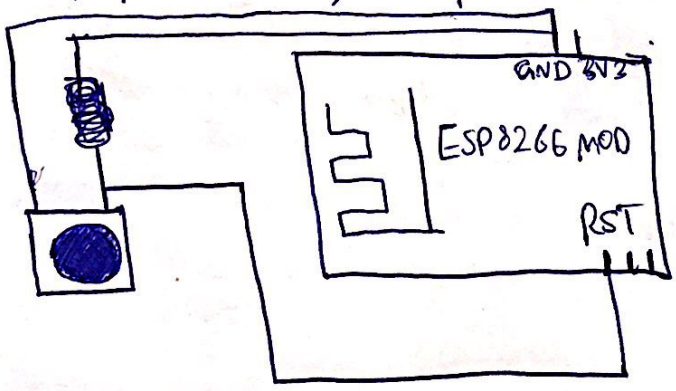
```
def deep-sleep (msecs)
    rtc = machine.RTC()
    rtc.irq(trigger = rtc.ALARM0, wake = machine.DEEPSLEEP)
    rtc.alarm(rtc.ALARM0, msecs)
    machine.deepsleep()
```

```
deepsleep(5000)
```

Using External signal :

The reset pin (RST) is also used to wake up the 8226. If we put it in deep sleep for an indefinite time, it will only be woken up when something resets the board. So, we can wire something to the RST pin and use it as an external wake up.

Example : Press of push button, reset pin LOW and wake 8266 up.




```
import machine
from machine import Pin
from time import sleep
led = Pin(2, Pin.OUT)
```

```
led.value(0)
sleep(1)
led.value(1)
sleep(1)

sleep(5)
print('I'm awake, but I'm going to sleep')
sleep(1)

# sleep for indefinite time
machine.deepsleep()
```