EXPERIMENT 1

AIM: Write a program for blinking of Led using 8051 microcontroller.

OUTCOME: Students will learn how to use Keil compiler for writing programs and Flash Magic to build the program in PC.

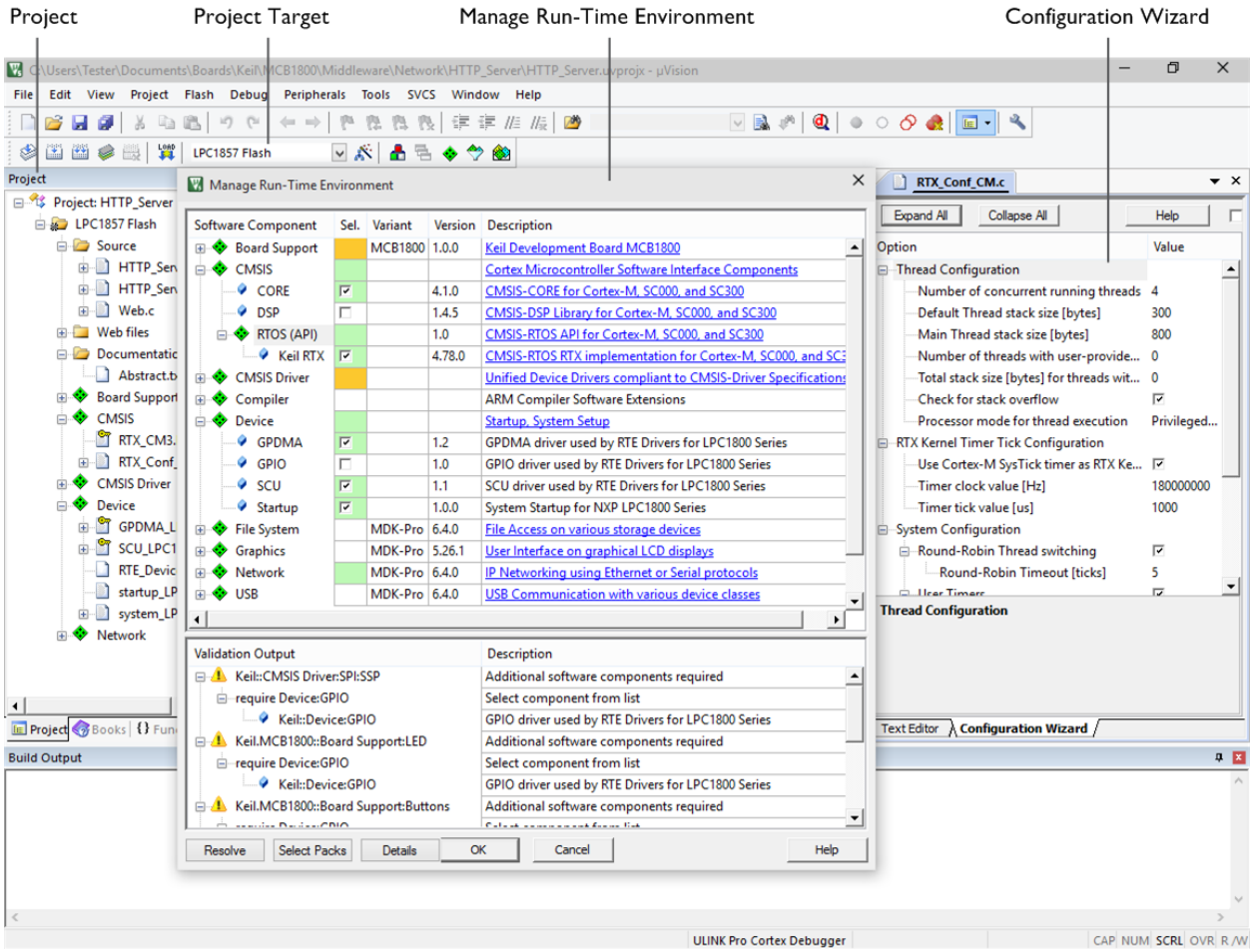
HARDWARE USED: Microcontroller kit, Power supply, connecting wires.

SOFTWARE USED: Keil μVision 4, Flash Magic.

WORKING OF SOFTWARE:

1. Keil µVision:

The µVision IDE combines project management, run-time environment, build facilities, source code editing, and program debugging in a single powerful environment. µVision is easy-to-use and accelerates your embedded software development. µVision supports multiple screens and allows you to create individual window layouts anywhere on the visual surface.



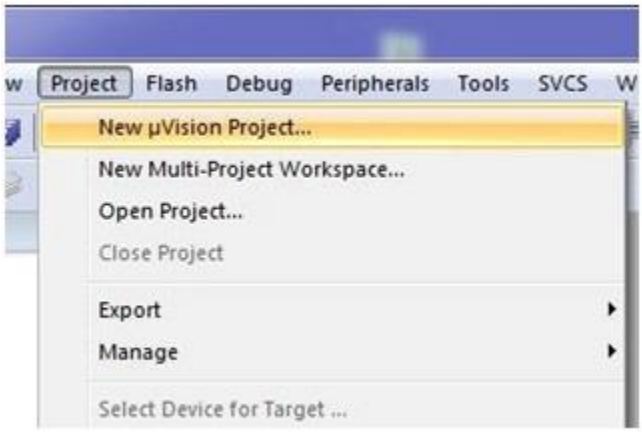
Procedure of using Keil:

Step 1: Download the Keil Uvision IDE

For learning purposes, you can try the evaluation version of Keil which has code limitation of 2K bytes. You must download the C51 version for programming on 8051 microcontroller architecture.

Step 2: To initiate the programming you must create a project using the keil Uvision IDE

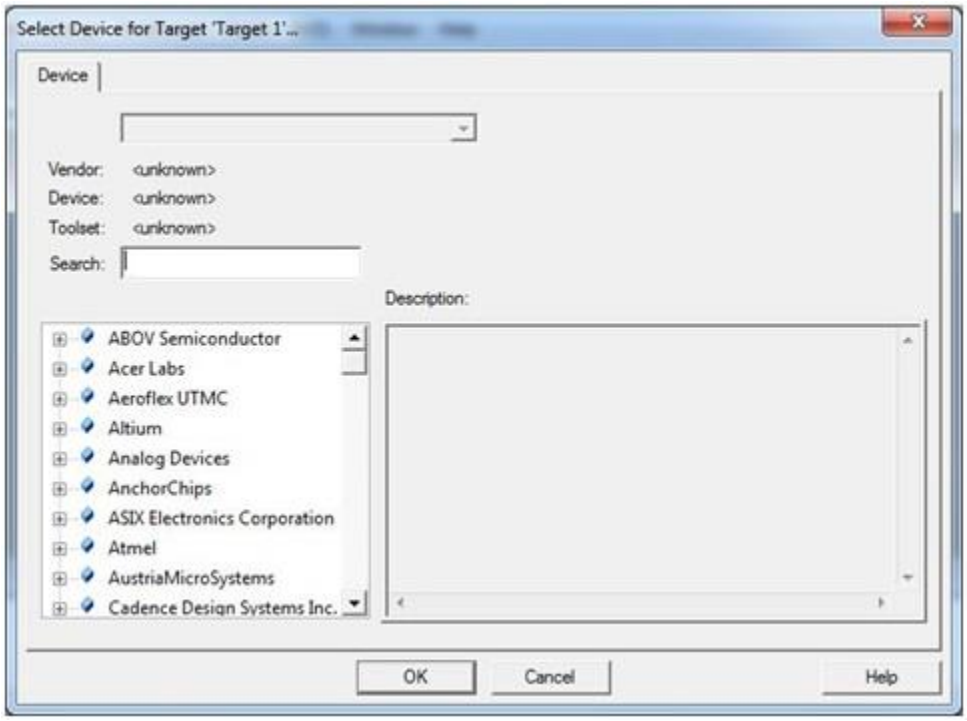
The option to create a new project will be available under the project tab in the toolbar. Next, you have to store the project in a folder and give a suitable name to it.



Step 3: Selecting the type of device you are working with

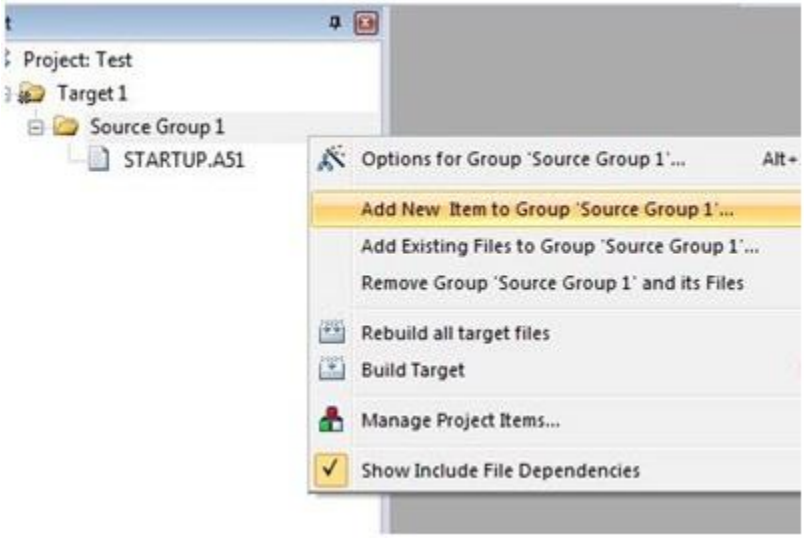
The device selection dialog provides you with the option to select the 8051 derivatives for which you want to develop the program. If you are not sure about your device you can refer to

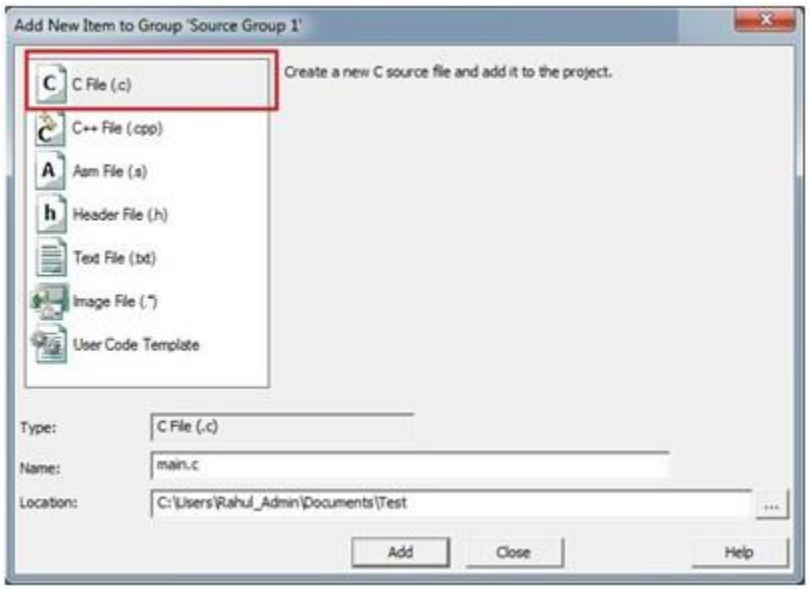
the description of the devices which is displayed on the left pane of the dialog. Accordingly, select your device and click OK to confirm.



Step 4: Adding C files to your project

You must add C file to your project before you begin coding. This can be done by right-clicking on your project from the project pane and selection “Add a new item to source group 1”. In the next dialog box, you will be given with the choice on what type of file you want to add such as C, C++, header, text etc. Since here we are dealing with Embedded C programming, select C File (.c) option. Provide the necessary name, location and click on add.





Step 5: Coding in C

The main part has finally arrived, so now you can go along with programming in C with your respective microcontroller.

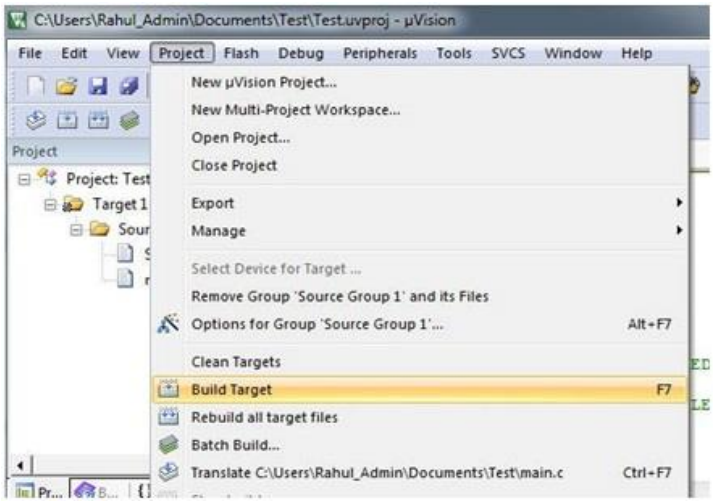
Example: Blinking of LED connected to Port 1 of 8051(or any other Intel family controller like Atmel controller in laboratory).

```
#include <reg51.h>
//delay function declaration
void delay(void);
```

```
void main(void)
{
	//an infinite loop
	while(1)
	{
		// Turn ON all LED's connected to Port1
		P1 = 0xFF;
		delay();
		// Turn OFF all LED's connected to Port1
		P1 = 0x00;
		delay();
	}
}
//delay function definition
void delay(void)
{
	int i,j;
	for(i=0;i<0xff;i++)
		for(j=0;j<0xff;j++);
}
```

Step 6 : Compiling and building the C project using Keil Uvision IDE

In order to build recently created C program go to Project tab and click on Build Target on the menu bar. An alternate way to do this is by pressing the F7 key. If the code that you have written is correct, the code will successfully compile without any errors. You can check your output in the Build Output pane.



```
 'Target 1'
TARTUP.A51...
in.c...

: data=9.0 xdata=0 code=56
est" - 0 Error(s), 0 Warnir
lapsed:  00:00:01
```

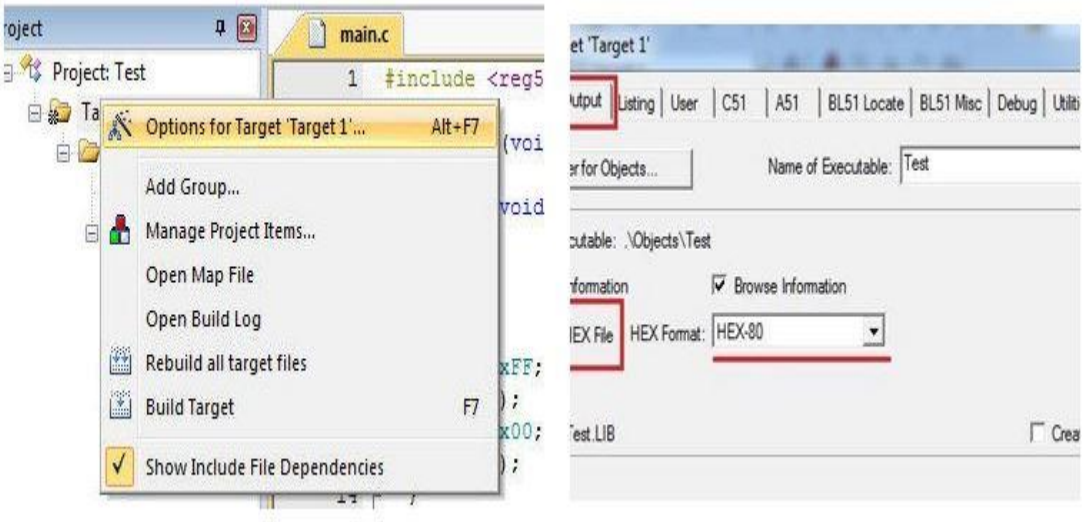
Step 7: Generating the hex file using Keil Uvision IDE

The code you compiled cannot be directly fed to the microcontroller, it is not possible. For that purpose, we have to generate the hex code for your respective file.

In order to generate the hex code, right click on the 'Target 1' folder and select options for target 'Target 1'. Select the Output tab in the target 'Target 1' dialog box. Make sure Create Hex File option is checked and the HEX format should be HEX-80. Click OK.

Again rebuild your project by pressing F7. Your required hex file would have been generated with the same as your project in the Objects folder.

If you wish you can also view your hex code by using a notepad.



Step 8: Burning the hex code into 8051 microcontroller

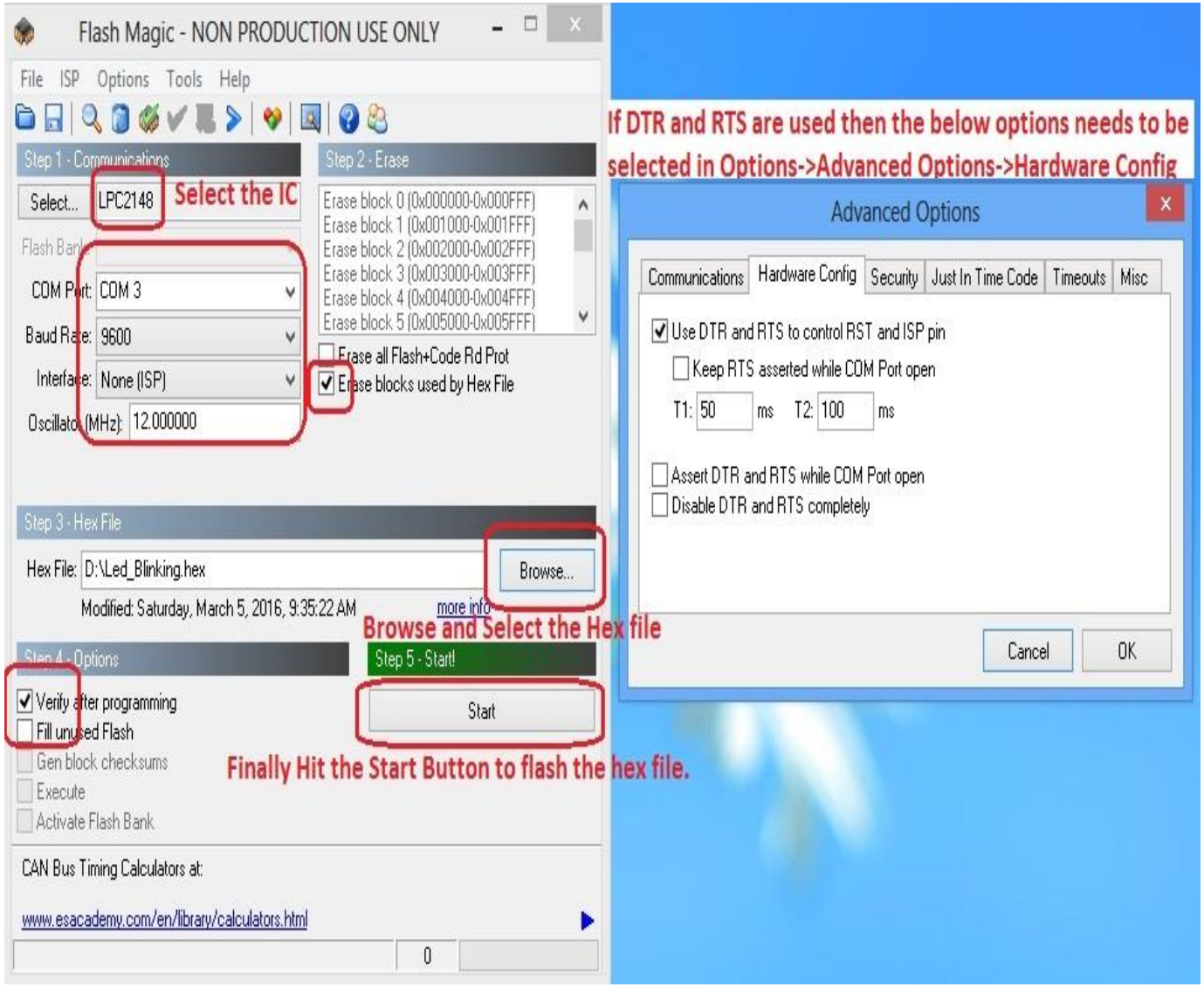
In order to burn the hex code to your microcontroller, there are two ways which are specific to the device you are working with. Some devices, for example, P89V51 they have their own built-in bootloader and you can burn the hex code directly through the serial port. Mostly you will require a specific programmer for your 8051 microcontroller through which you can easily upload your hex code by connecting the programmer via normal USB port.

2. Flash Magic

Flash Magic is a PC tool for programming flash based microcontrollers from NXP using a serial or Ethernet protocol while in the target hardware.

Procedure for using Flash Magic:

1. Select the IC from Select Menu.
2. Select the COM Port. Check the device manager for detected Com port.
3. Select Baud rate from 9600-115200
4. Select None Isp Option.
5. Oscillator Freq 12.000000(12Mhz).
6. Check the Erase blocks used by Hex file option
7. Browse and Select the hex file.
8. Check the Verify After Programming Option.
9. If DTR and RTS are used then go to Options->Advanced Options-> Hardware Configuration and select the Use DTR and RTS Option.
10. Hit the Start Button to flash the hex file.
11. Once the hex file is flashed, Reset the board. Now the controller should run your application code.


Flash Magic - NON PRODUCTION USE ONLY
File ISP Options Tools Help
Step 1 - Communications
Select...
LPC2148
Select the IC
COM Port: COM 3
Baud Rate: 9600
Interface: None (ISP)
Oscillator (MHz): 12.000000
Step 2 - Erase
Erase block 0 (0x000000-0x000FFF)
Erase block 1 (0x001000-0x001FFF)
Erase block 2 (0x002000-0x002FFF)
Erase block 3 (0x003000-0x003FFF)
Erase block 4 (0x004000-0x004FFF)
Erase block 5 (0x005000-0x005FFF)
Erase all Flash+Code Rd Prot
Erase blocks used by Hex File
Step 3 - Hex File
Hex File: D:\Led_Blinking.hex
Browse...
Modified: Saturday, March 5, 2016, 9:35:22 AM
more info
Browse and Select the Hex file
Step 4 - Options
Step 5 - Start!
Verify after programming
Fill unused Flash
Gen block checksums
Execute
Activate Flash Bank
Start
Finally Hit the Start Button to flash the hex file.
CAN Bus Timing Calculators at:
www.esacademy.com/en/library/calculators.html
0
If DTR and RTS are used then the below options needs to be selected in Options->Advanced Options->Hardware Config
Advanced Options
Communications
Hardware Config
Security
Just In Time Code
Timeouts
Misc
Use DTR and RTS to control RST and ISP pin
Keep RTS asserted while COM Port open
T1: 50 ms T2: 100 ms
Assert DTR and RTS while COM Port open
Disable DTR and RTS completely
Cancel
OK

PROGRAM:

```
#include<reg51.h>
void main()
{
unsigned int x;

for(;;)
{
P2=0x55;
for( x=0;x<400;x++);
{
}
P2=0xAA;

for( x=0;x<400;x++);
{
}
}
}
```

OUTPUT- After the execution of the program the Led start blinking alternatively at odd and even positions, after some delay.

EXPERIMENT 2

AIM: Write a program to generate a 10KHz square wave using 8051.

OUTCOME: Students will learn the fundamentals related to calculations for waveform generations of different shapes and duty cycles.

HARDWARE USED: Microcontroller kit, Power supply, connecting wires.

SOFTWARE USED: Keil µvision4, Flash magic

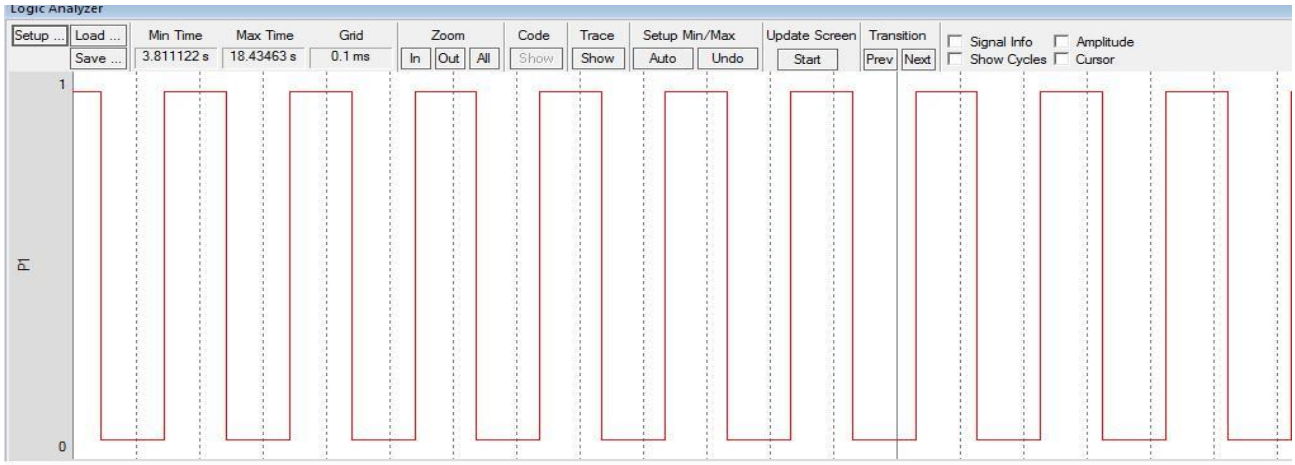
// The Keil will be used. Since the waveform generation can be through compiler the flash magic need not to be used.

Entire port or just a particular pin can be used to generate the waveforms.

PROGRAM:

```
#include<reg51.h>
void main()
{
unsigned int x;
for(;;)
{
P1=0X80;
for(x=0;x<40000;x++)
{
P1=0X00;
}
}
}
```

OUTPUT:



Square Wave Display

EXPERIMENT 3

AIM: Write a program to show the use of INT0 & INT1 of 8051.

OUTCOME: Students will learn to use interrupts for fast processing using flash magic for onboard switch and leds control and Keil for programming.

SOFTWARE USED: Keil µvision4, Flash magic

HARDWARE USED: Micro controller kit 8051, connecting wires and power supply

PROGRAM:

```
//interrupt INT0 & INT1
#include <reg51.h>
sbit LED1 =P1^0;

sbit LED2 =P1^1;

//Function Declaration
port_init();
InitINT0();

InitINT1();
//Main Function
main()
{
port_init();
InitINT0();
InitINT1();
while(1)
}

port_Init()
{
P0=0X00;
P1=0X00;
P2=0X00;
P3=0X0C;
}
InitINT0() //int0 ISR
{
IT0=1;
EX0=1;
EA=1;
}
InitINT1()//INT1 ISR
{
```

```
IT1=1;
EX1=1;

EA=1;
}
external0_isr()interrupt 0
{
LED1 = ~LED1;
}
external1_isr()interrupt 2

{
LED2 = ~LED2;
}
```

OUTPUT:

The interrupts given at PORT 3 (P2.2 & P2.3) using PULL-UP KEYS (K5 & K6) have been studied successfully:-

- When K5 pressed - LED on.
- When K5 pressed again - LED off.

Similarly,

- When K6 pressed - LED on.
- When K6 pressed again - LED off

EXPERIMENT 4

OBJECTIVE: Write a program to display temperature using internal sensor of MSP430 and display the same in CCS.

OUTCOME: Students will learn using Temperature sensor and ADC in MSP430 and defining interrupts on MSP430.

SOFTWARE USED: CCS (Code Composer Studio)

HARDWARE USED: MSP430F5529 LaunchPad

// The MSP430 board has internal temperature sensor and to use it ADC and Interrupts need to programmed. Also there are some calibrations calculations required for which the manuals have to be referred.

PROGRAM:

```
#include <msp430.h>
#define CALADC12_15V_30C *((unsigned int *)0x1A1A) // Temperature Sensor Calibration-30 C
//See device datasheet for TLV table memory mapping
#define CALADC12_15V_85C *((unsigned int *)0x1A1C)      // Temperature Sensor Calibration-
85 C

unsigned long temp;
volatile float temperatureDegC;
volatile float temperatureDegF;

int main(void)
{
WDTCTL = WDTPW + WDTHOLD;       // Stop WDT
REFCTL0 &= ~REFMSTR;         // Reset REFMSTR to hand over control to
ADC12_A ref control registers
ADC12CTL0 = ADC12SHT0_8 + ADC12REFON + ADC12ON;
Internal ref = 1.5V
ADC12CTL1 = ADC12SHP;       // enable sample timer
ADC12MCTL0 = ADC12SREF_1 + ADC12INCH_10; // ADC i/p ch A10 = temp sense i/p
ADC12IE = 0x001; // ADC_IFG upon conv result-ADCMEMO __delay_cycles(100); // delay to
allow Ref to settle
ADC12CTL0 |= ADC12ENC;

while(1)
{
ADC12CTL0 &= ~ADC12SC; ADC12CTL0 |= ADC12SC;
// Sampling and conversion start
__bis_SR_register(LPM0_bits + GIE); // LPM0 with interrupts enabled
 __no_operation();
```

Temperature in Celsius. See the Device Descriptor Table section in the System Resets,

Interrupts, and Operating Modes, System Control Module chapter in the device user's guide for background information on the used formula.

```
temperatureDegC = (float)(((long)temp - CALADC12_15V_30C) * (85 - 30)) /
(CALADC12_15V_85C - CALADC12_15V_30C) + 30.0f;

//Temperature in Fahrenheit
Tf = (9/5)*Tc + 32 temperatureDegF = temperatureDegC * 9.0f / 5.0f + 32.0f;

__no_operation();      // SET BREAKPOINT HERE

}
}
#pragma vector=ADC12_VECTOR
__interrupt void ADC12ISR (void)

{
switch(__even_in_range(ADC12IV,34))
{
case 0: break;  // Vector 0: No interrupt
case 2: break;  // Vector 2: ADC overflow
case 4: break;  // Vector 4: ADC timing overflow
case 6:            // Vector 6: ADC12IFG0
temp = ADC12MEM0; // Move results, IFG is cleared
__bic_SR_register_on_exit(LPM0_bits); // Exit active CPU
break;
case 8: break;  // Vector         8: ADC12IFG1
case 10: break;          // Vector         10: ADC12IFG2
case 12: break;          // Vector         12: ADC12IFG3
case 14: break;          // Vector         14: ADC12IFG4
case 16: break;          // Vector         16: ADC12IFG5
case 18: break;          // Vector         18: ADC12IFG6
case 20: break;          // Vector         20: ADC12IFG7
case 22: break;          // Vector         22: ADC12IFG8
case 24: break;          // Vector         24: ADC12IFG9
case 26: break;          // Vector         26: ADC12IFG10
case 28: break;          // Vector         28: ADC12IFG11
case 30: break;          // Vector         30: ADC12IFG12
case 32: break;          // Vector         32: ADC12IFG13
case 34: break;          // Vector        34: ADC12IFG14
default: break;
}
}
```

EXPERIMENT 5

AIM: Generation of ramp waveform using DAC and 8051 microcontroller.

OUTCOME: Students will be able to learn using DAC in 8051 family available of microcomputers available.

SOFTWARE USED: Keil µvision4, Flash magic

HARDWARE USED: Micro controller kit 8051, connecting wires and power supply

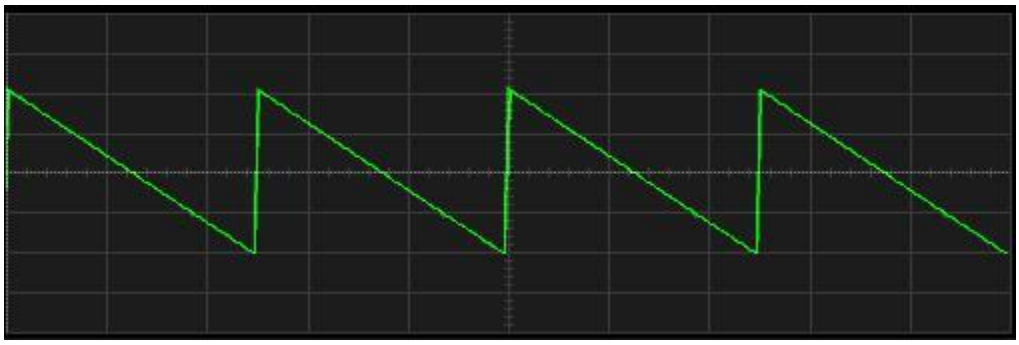
// The DAC is responsible for visualization of waveforms as the programming is done with digital values but the waveform is constructed using predefined formulae considering the applied supply.

Ex: For generation of Sinusoidal waveform the table has to be prepared for all possible angles. The smallest the angles , the smoothen will be the waveform.

PROGRAM:

```
#include<reg51.h>
void main()
{unsigned int i;
while(1)
{for(i=0;i<256;i++)
{
P2=i;
}
}
}
```

OUTPUT:



Ramp Waveform

EXPERIMENT 6(A)

OBJECTIVE: To interface GPIO ports with pushbutton and Leds.

OUTCOME: Students will be able to learn configuring on-board switches and leds on MSP430 Boards.

HARDWARE USED: MSP430F5529 LaunchPad

SOFTWARE USED: CCS

PROGRAM:

```
#include <msp430.h>
void main(void)
{
unsigned int i=0;
WDTCTL = WDTPW | WDTHOLD;
P1DIR |= 0x01; // Led is attached to P1.0
P4DIR |= 0x80;// Led is attched to P1.7
for(;;)
{
P1OUT ^= 0x01; //Toggle led p1.0
P4OUT ^= 0x80; //Toggle led p1.7
for(i=0;i<20000;i++)
{}
}
return 0;
}
```

EXPERIMENT 6(B)

PUSH BUTTON_LED

```
#include <msp430.h>
void main( void )
{
unsigned int i=0;
WDTCTL = WDTPW | WDTHOLD;

SW1 -P1.1, SW2- P2.1, LED1- P1.0, LED2- P4.7 P1REN |= 0x02; // Enable resistor on P1.1 P1OUT
= 0x03; //Set resistor to pull-up, P1.0 high P1DIR = 0x01; //P1.0 as output & P1.1 as intput

P2REN |= 0x02; // Enable resistor on P2.1
P2OUT = 0x02; // Set resistor to pull-up
P2DIR = 0x01;

P4OUT = 0x80; // Set P4.7 high
P4DIR = 0x80; // Set P4.7 as output

while(1)
```

```
{
if(!(P1IN & 0x02)) // If push button is pressed
{
P4OUT ^= 0x80;
}
if(!(P2IN & 0x02))
{
P1OUT ^= 0x01;
}

for(i=0;i<20000;i++)
{}
}
}
```

EXPERIMENT 7

OBJECTIVE: To Interface Potentiometer with GPIO.

OUTCOME: Students will be able to learn using analog sensors with MSP430 boards and to display their values using serial monitor application available in Energia.

HARDWARE USED: MSP-EXP430G2 LaunchPad, 10-kilohm Potentiometer, hook-up wire.

SOFTWARE USED: ENERGIA

// ANALOGREADSERIAL: Reads an analog input on pin A3, prints the result to the serial monitor. Attach the center pin of a potentiometer to pin A3, and the outside pins to ~3V and ground.

```
//the setup routine runs once when you press reset:
void setup()
 {                      // initialize serial communication at 9600 bits per second:
Serial.begin(9600);     // msp430g2231 must use 4800
}                               //the loop routine runs over and over again forever:

void loop()
 {
read the input on analog pin A3:
int sensorValue = analogRead(A3);
print out the value you read.
Serial.println(sensorValue);
delay(1); // delay in between reads for stability
}
```

OUTPUT: The continues analog values can be seen on serial monitor.

EXPERIMENT 8

AIM: Write a program for PWM based LED intensity controlled by potentiometer connected to GPIO.

OUTCOME: Students will be able to use analog sensor like potentiometer to control any output (led intensity in this case) through the application of pulse width modulation techniques using ADC.

SOFTWARE USED: ENERGIA

HARDWARE USED: Micro controller board MSP430, connecting wires, power supply, potentiometer.

PROGRAM:

```
const int analogInPin = A0; // Analog input pin that the potentiometer is attached to const int analogOutPin =RED_LED; // Analog output pin that the LED is attached to

int sensorValue = 0;    // value read from the pot
int outputValue = 0;    // value output to the PWM (analog out)

void setup() {
//initialize serial communications at 9600 bps
Serial.begin(9600);
}

void loop() {
// read the analog in value
sensorValue = analogRead(analogInPin);

//map it to the range of the analog out:
outputValue = map(sensorValue, 0, 1023, 0, 255);

//change the analog out value:
analogWrite(analogOutPin, outputValue);

//print the results to the serial monitor
Serial.print("sensor = " );
Serial.print(sensorValue);
Serial.print("\t output = ");
Serial.println(outputValue);

//wait 10 milliseconds before the next loop for the analog-to-digital converter to settle after the last reading
delay(10);
}
```

OUTPUT:

The output will be intensity variations in on-board leds. Or if the variation is not seen easily, one can program another led connected to any GPIO pin to visualize the impact of potentiometer.

EXPERIMENT 9

AIM: Write a program for PWM generation for MSP430 using Timers.

OUTCOME: Students will be able to learn to use timers for PWM generation.

SOFTWARE USED: Code Composer Studio and Tera term.

HARDWARE USED: Micro controller board MSP430, connecting wires ,power supply

WORKING OF SOFTWARE:

Tera Term (alternatively TeraTerm) is an open-source, free, software implemented, terminal emulator (communications) program. It emulates different types of computer terminals, from DEC VT100 to DEC VT382. It supports telnet, SSH 1 & 2 and serial port connections.

TeraTerm (as all terminal programs) connects to a COM port on your PC. It may be a real COM port, or a virtual one such as the application UART of the LaunchPad, or the virtual COM port of an USB/serial converter.

If using a real COM port or an USB/serial converter, you'll need to connect the MSP on the launchPad correctly (including a level shifter).

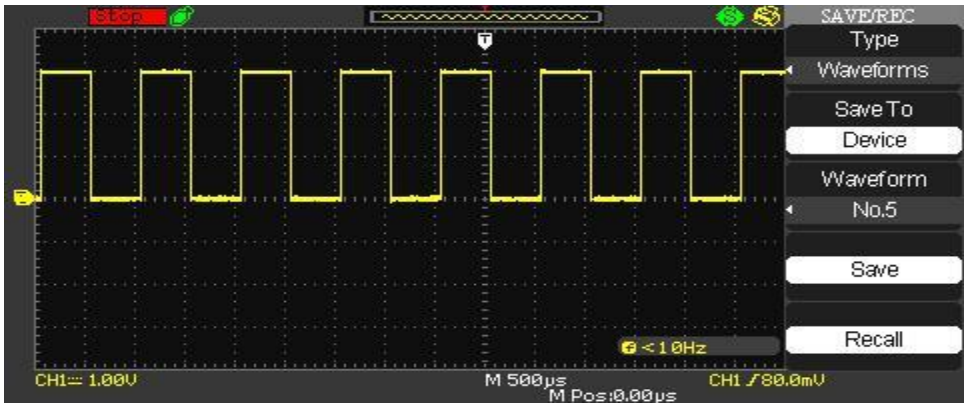
If using the application UART, you have to set the jumpers properly for RX/TX. The maximum baudrate is 9600 then (with the other two options, you can go up to 115200Bd or even higher).

PROGRAM:

```
#include <msp430f5529.h>

void main(void)
{
WDTCTL = WDTPW + WDTHOLD;
// Stop WDT
/*** GPIO Set-Up ***/
P1DIR |= BIT2;                          //
P1SEL |= BIT2;                          //SELECTION OF PERIPHERAL

P1DIR |= BIT3;
P1SEL |= BIT3;
                                        //*** Timer0_A Set-Up ***/
TA0CCR0 |= 200 - 1;
TA0CCTL1 |= OUTMOD_7;
TA0CCR1 |= 100;
TA0CTL |= TASSEL_2 + MC_1;
                                        //*** Timer1_A Set-Up ***/

TA0CCR0 |= 1000 - 1;
TA0CCTL2 |= OUTMOD_7;
TA0CCR2 |= 500;
```

```
TA0CTL |= TASSEL_2 + MC_1;
return 0;      __
}
```

OUTPUT:



EXPERIMENT 10

OBJECTIVE: Setting up CC3100 as an HTTP server.

OUTCOME: Students will be able to create a server and control the onboard leds through it. They will also be able to program in HTML for creation of the page.

HARDWARE USED: CC3100

SOFTWARE USED: ENERGIA

//WiFi Web Server LED Blink

A simple web server that lets you blink an LED via the web. This sketch will print the IP address of your WiFi (once connected) to the Serial monitor. From there, you can open that address in a web browser to turn on and off the LED on pin 9.

If the IP address of your WiFi is your Address:

http://yourAddress/H turns the LED on

http://yourAddress/L turns it off

This example is written for a network using WPA encryption. For WEP or WPA, change the Wifi.begin() call accordingly.

CIRCUIT:

```
*CC3200 WiFi LaunchPad or CC3100 WiFi BoosterPack with TM4C or MSP430 LaunchPad*/
#ifndef __CC3200R1M1RGC__
//Do not include SPI for CC3200 LaunchPad
#include <SPI.h>
#endif

#include <WiFi.h>

//your network name also called SSID
char ssid[] = "Vishal";
your network password char password[] = "12341234";

//your network key Index number (needed only for WEP)
int keyIndex = 0;
WiFiServer server(80);

void setup() {
Serial.begin(9600); // initialize serial communication
pinMode(RED_LED, OUTPUT); // set the LED pin mode attempt to connect to Wifi network
Serial.print("Attempting to connect to Network named: ");

//print the network name (SSID)
```

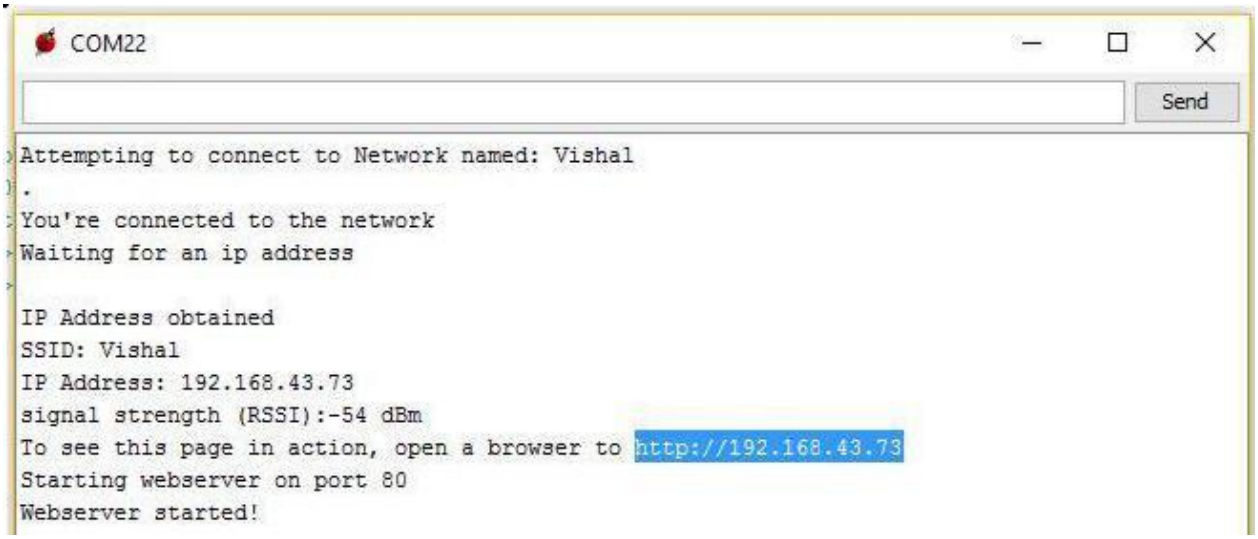
```
Serial.println(ssid);
Connect to WPA/WPA2 network. Change this line if using open or WEP network
WiFi.begin(ssid, password);
while ( WiFi.status() != WL_CONNECTED) {

//print dots while we wait to connect
Serial.print(".");
delay(300);
}

Serial.println("\nYou're connected to the network");
Serial.println("Waiting for an ip address");
while (WiFi.localIP() == INADDR_NONE)
{

//print dots while we wait for an ip addresss
Serial.print(".");
delay(300);
}
Serial.println("\nIP Address obtained");

//you're connected now, so print out the status
printWifiStatus();
Serial.println("Starting webserver on port 80");

server.begin(); // start the web server on port 80 Serial.println("Webserver started!");
}
void loop() {
int i = 0;
WiFiClient client = server.available(); // listen for incoming clients
if (client) {       // if you get a client,
Serial.println("new client");    // print a message out the serial port
char buffer[150] = {0};// make a buffer to hold incoming data
while (client.connected())// loop while the client's connected
 {
if (client.available())// if there's bytes to read from the client
 {          ,
char c = client.read();  // read a byte, then
Serial.write(c);// print it out the serial monitor
if (c == '\n') // if the byte is a newline character
{

//if the current line is blank, you got two newline characters in a row. That's the end of the client HTTP request, so send a response.
```

```
if (strlen(buffer) == 0)
{

//HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK) and a content-type so the client knows what's coming, then a blank line
client.println("HTTP/1.1 200 OK"); client.println("Content-type:text/html");
client.println();

//the content of the HTTP response follows the header
client.println("<html><head><title>CC3100 WiFi Web Server</title></head><body align=center>");
client.println("<h1 align=center><font color=\"red\">Welcome to the CC3100 WiFi Web Server</font></h1>");
client.println("<h1 align=center><font color=\"green\">MIET, MEERUT</font></h1>");
client.println("<h1 align=center><font color=\"blue\">EC-VI A/B/C</font></h1>");
client.print("RED LED <button onclick=\"location.href='/H'\">HIGH</button>");
client.println(" <button onclick=\"location.href='/L'\">LOW</button><br>");


//The HTTP response ends with another blank line.
client.println();
//break out of the while loop
break;
}
else // if you got a newline, then clear the buffer
{          memset(buffer, 0, 150);
i = 0;
}
}
else if (c != '\r') { buffer[i++] = c;
}
//if you got anything else but a carriage return character, add it to the end of the currentLine
Check to see if the client request was "GET /H" or "GET /L".
if (endsWith(buffer, "GET /H"))
{
digitalWrite(RED_LED, HIGH);         // GET /H turns the LED on
}
if (endsWith(buffer, "GET /L"))
{
digitalWrite(RED_LED, LOW);        // GET /L turns the LED off
}
}
}
```

```
// close the connection:
client.stop();
Serial.println("client disonnected");
}
}
//a way to check if one array ends with another array boolean endsWith(char* inString, char* compString)
{ int compLength = strlen(compString);
int strLength = strlen(inString);
//compare the last "compLength" values of the inString
int i;
for (i = 0; i < compLength; i++)
{
char a = inString[(strLength - 1) - i];
char b = compString[(compLength - 1) - i];
if (a != b)
{
return false;
}
}
return true;
}

void printWifiStatus()
{
// print the SSID of the network you're attached to.
Serial.print("SSID: ");
Serial.println(WiFi.SSID());

//print your WiFi IP address
IPAddress ip = WiFi.localIP();
Serial.print("IP Address: ");
Serial.println(ip);

//print the received signal strength
long rssi = WiFi.RSSI();
Serial.print("signal strength (RSSI):");
Serial.print(rssi);
Serial.println(" dBm");

//print where to go in a browser
Serial.print("To see this page in action, open a browser to http://");
Serial.println(ip);}
```

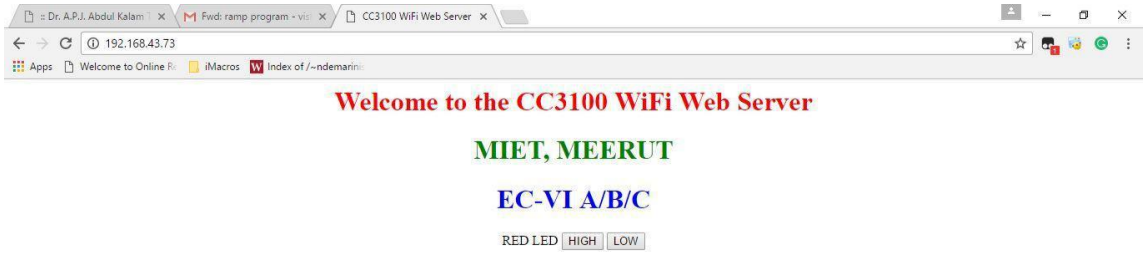
OUTPUT:

1. Serial Monitor Output



Serial Monitor display

2. Result Webpage on CC3100 Server



Webpage on CC3100 Server