SRI RAMAKRISHNA ENGINEERING COLLEGE

[Educational Service: SNR Sons Charitable Trust]
[Autonomous Institution, Reaccredited by NAAC with 'A+' Grade]
[Approved by AICTE and Permanently Affiliated to Anna University, Chennai]
[ISO 9001-2015 Certified and all eligible programmes Accredited by NBA]
VATTAMALAIPALAYAM, N.G.G.O. COLONY POST,
COIMBATORE – 641 022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

20EC276– EMBEDDED SYSTEMS AND INTERNET OF THINGS LABORATORY

LAB RECORD

ACADEMIC YEAR: 2023-2024

BATCH: 2021-2026

MAY 2024

SRI RAMAKRISHNA ENGINEERING COLLEGE

[Educational Service: SNR Sons Charitable Trust]
[Autonomous Institution, reaccredited by NAAC with 'A+' Grade]
[Approved by AICTE and Permanently Affiliated to Anna University, Chennai]
[ISO 9001:2015 Certified and all eligible programmes Accredited by NBA]
VATTAMALAIPALAYAM, N.G.G.O. COLONY POST, COIMBATORE – 641 022.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this is the bonafide record of works done by Mr./Ms. NAVYA S K in 20EC276– EMBEDDED SYSTEMS AND INTERNET OF THINGS LABORATORY of this Institution for VI Semester during the Academic Year 2023 – 2024.

Faculty In-Charge | **HOD – CSE**

Mrs.N.Alagusundari, AP/CSE | Dr. A. GRACE SELVARANI, Prof/Head(CSE)

Date:

Register Number: _____

Submitted for the VI Semester M.Tech.-CSE Practical Examination held on _____ during the Academic Year 2023 – 2024.

Internal Examiner | **Subject Expert**

INDEX

Exp. No	Date	Title of the Experiment	Page No	Faculty signature
1.	09.01.2024	Basic Linux Commands	1	
		Task Using Basic Linux Commands	7	
		Linux Kernal Compilation	8	
2.	23.01.2024	Utilization Of Gnu Toolchains For Effective System Programming	13	
3.	30.01.2024	Debugging Programs Using Cscope	20	
4.	06.02.2024	Construct Character Oriented Device Drivers	25	
5.	13.02.2024	Implementation Of Task Management In Real-Time Operating Systems (Rtos) Using Microc/Os-Ii	30	
6.	17.02.2024	Implementation Of Interuppt Management In Real-Time Operating Systems (Rtos) Using Microc/Os-Ii	33	
7.	20.02.2024	Development Of Bluetooth Interfacing Using Msp430 Launchpad	38	
8.	19.03.2024	Development Of Esp8266 Interfacing (Wifi) Using Msp430 Launchpad	40	
9.	26.03.2024	Multiple Led Blinking Using TI CC3200 Launchpad	44	
10.	02.04.2024	Interfacing Push Button Using TI CC3200 Launchpad	46	
11.	09.04.2024	Design Of Iot Application To Sense Nearby Objects Using Pir Sensor With TI CC3200 Launchpad	48	
12.	09.04.2024	Design Of Iot Applications With Sensors To Scan Networks Using TI Cc3200 Launchpad	50	
CONTENT BEYOND SYLLABUS				
09.04.2024		Demonstration of MISRA and Cert C Coding Standards	53	
CAPSTONE PROJECT				
16.04.2024		Iot-based asset tracking and management system	57	

EX.NO: 1	LINUX KERNAL COMPILATION
DATE:	

Aim:

To upgrade the kernel file of the linux operating system using kernel compilation process.

Apparatus Required:

1. Linux OS
2. Virtual Machine

Description:

Linux kernel compilation provides the user to unlock the features which are not available for the standard users. Linux compilation process allows the users to modify their kernel depending on their hardware and software application environment. Following points shows the steps involved in compiling a kernel.

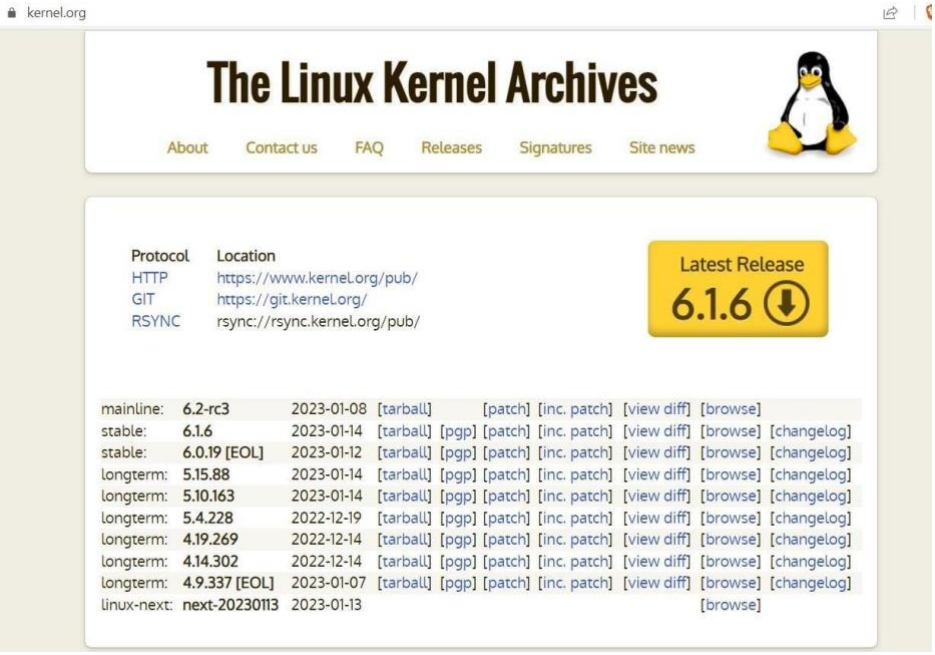
Building Linux Kernel

The process of building a Linux kernel can be performed in seven easy steps. However, the procedure may require a significant amount of time to complete, depending on the system speed.

Follow the steps below to build the latest Linux kernel.

Step 1: Download the Source Code

1. Visit the official kernel website and download the latest kernel version. The downloaded file contains a compressed source code.



2. Open the terminal and use the wget command to download the Linux kernel source code:

wget https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.0.7.tar.xz

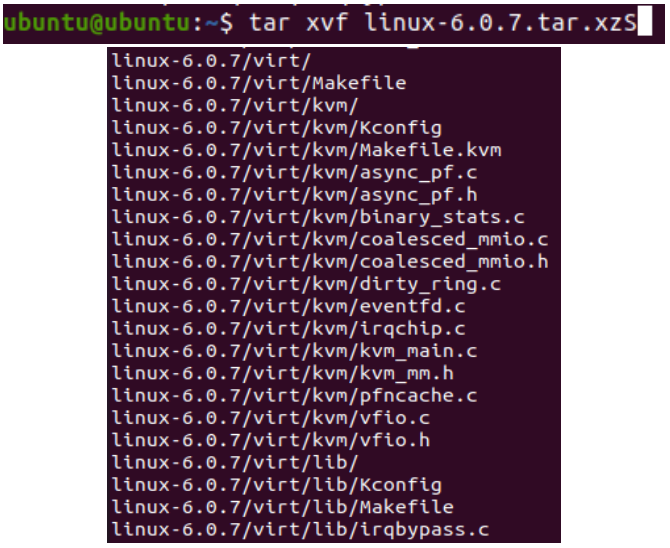
The output shows the “saved” message when the download completes.

Step 2: Extract the Source Code

When the file is ready, run the tar command to extract the source code:

tar xvf linux-6.0.7.tar.xz

The output displays the extracted kernel source code:

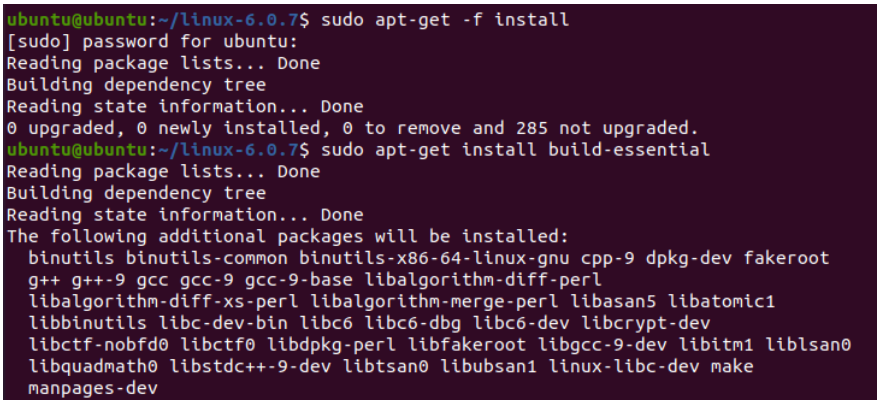


Step 3: Install Required Packages

Install additional packages before building a kernel. To do so, run this command:

sudo apt-get install git fakeroot build-essential ncurses-dev xz-utils libssl-dev bc flex libelf-dev bison

The command we used above installs the essential packages for performing Linux kernel compilation



```
ubuntu@ubuntu:~/linux-6.0.7$ sudo apt-get install git fakeroot build-essential ncurses-dev xz-utils libssl-dev bc flex libelf-dev bison
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'libncurses-dev' instead of 'ncurses-dev'
bc is already the newest version (1.07.1-2build1).
bc set to manually installed.
fakeroot is already the newest version (1.24-1).
fakeroot set to manually installed.
build-essential is already the newest version (12.8ubuntu1.1).
The following additional packages will be installed:
  git-man liberror-perl libfl-dev libfl2 libsigsegv2 libssl1.1 m4 zlib1g
  zlib1g-dev
Suggested packages:
  bison-doc flex-doc git-daemon-run | git-daemon-sysvinit git-doc git-el
  git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn ncurses-doc
  libssl-doc m4-doc
The following NEW packages will be installed:
  bison flex git git-man libelf-dev liberror-perl libfl-dev libfl2
  libncurses-dev libsigsegv2 libssl-dev m4 zlib1g-dev
The following packages will be upgraded:
```

Step 4: Configure Kernel

The Linux kernel source code comes with the default configuration. However, you can adjust it to your needs. To do so, follow the steps below:

1. Navigate to the linux-6.0.7 directory using the cd command:

cd linux-6.0.7

2. Copy the existing configuration file using the cp command:

cp -v /boot/config-$(uname -r) .config

```
marko@pnap:~$ cd linux-6.0.7/
marko@pnap:~/linux-6.0.7$ cp -v /boot/config-$(uname -r) .config
'/boot/config-5.15.0-52-generic' -> '.config'
marko@pnap:~/linux-6.0.7$
```

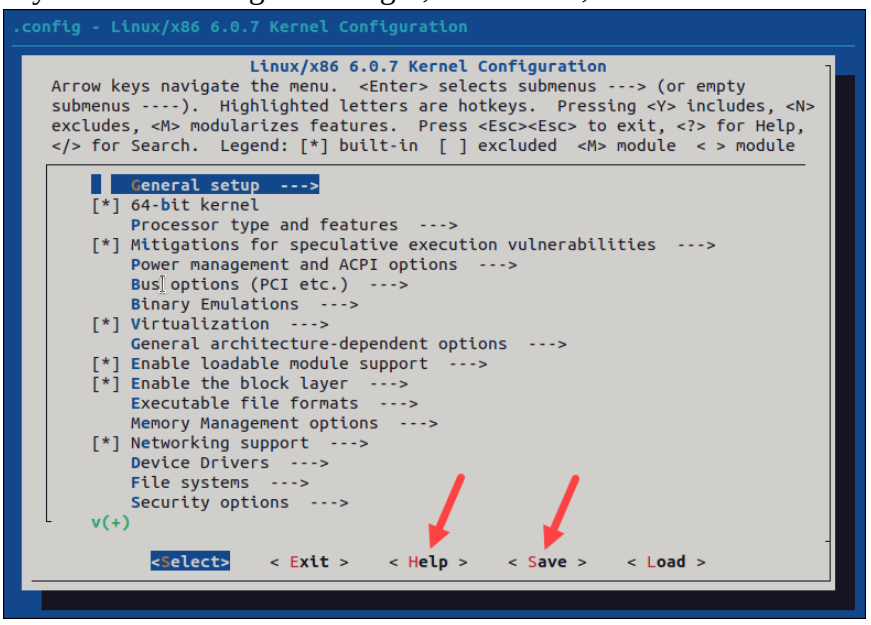
3. To make changes to the configuration file, run the make command:

make menuconfig

The command launches several scripts that open the configuration menu:

```
ubuntu@ubuntu:~/linux-6.0.7$ make menuconfig
  UPD     scripts/kconfig/mconf-cfg
  HOSTCC  scripts/kconfig/mconf.o
  HOSTCC  scripts/kconfig/lxdialog/checklist.o
  HOSTCC  scripts/kconfig/lxdialog/inputbox.o
  HOSTCC  scripts/kconfig/lxdialog/menubox.o
  HOSTCC  scripts/kconfig/lxdialog/textbox.o
  HOSTCC  scripts/kconfig/lxdialog/util.o
  HOSTCC  scripts/kconfig/lxdialog/yesno.o
  HOSTCC  scripts/kconfig/confdata.o
  HOSTCC  scripts/kconfig/expr.o
  LEX     scripts/kconfig/lexer.lex.c
  YACC    scripts/kconfig/parser.tab.[ch]
  HOSTCC  scripts/kconfig/lexer.lex.o
  HOSTCC  scripts/kconfig/menu.o
  HOSTCC  scripts/kconfig/parser.tab.o
  HOSTCC  scripts/kconfig/preprocess.o
  HOSTCC  scripts/kconfig/symbol.o
  HOSTCC  scripts/kconfig/util.o
  HOSTLD  scripts/kconfig/mconf
.config:434:warning: symbol value 'm' invalid for I8K
.config:1998:warning: symbol value 'm' invalid for MCTP
.config:8858:warning: symbol value 'm' invalid for VIDEO_ZORAN_DC30
.config:8859:warning: symbol value 'm' invalid for VIDEO_ZORAN_ZR36060
.config:8860:warning: symbol value 'm' invalid for VIDEO_ZORAN_BUZ
.config:8861:warning: symbol value 'm' invalid for VIDEO_ZORAN_DC10
.config:8862:warning: symbol value 'm' invalid for VIDEO_ZORAN_LML33
.config:8863:warning: symbol value 'm' invalid for VIDEO_ZORAN_LML33R10
.config:8864:warning: symbol value 'm' invalid for VIDEO_ZORAN_AVS6EYES
.config:9955:warning: symbol value 'm' invalid for ANDROID_BINDER_IPC
.config:9956:warning: symbol value 'm' invalid for ANDROID_BINDERFS
configuration written to .config
```

4. The configuration menu includes options such as firmware, file system, network, and memory settings. Use the arrows to make a selection or choose Help to learn more about the options. When you finish making the changes, select Save, and then exit the menu.



Note:

Changing settings for some options can lead to a non-functional kernel. If you are unsure what to change, leave the default settings.

Step 5: Build the Kernel

1. Start building the kernel by running the following command:

make

- The process of building and compiling the Linux kernel takes some time to complete.
- The terminal lists all Linux kernel components: memory management, hardware device drivers, filesystem drivers, network drivers, and process management.

```
marko@pnap:~/linux-6.0.7$ make
  SYNC    include/config/auto.conf.cmd
  HOSTCC  scripts/kconfig/conf.o
  HOSTLD  scripts/kconfig/conf
  SYSHDR  arch/x86/include/generated/uapi/asm/unistd_32.h
  SYSHDR  arch/x86/include/generated/uapi/asm/unistd_64.h
  SYSHDR  arch/x86/include/generated/uapi/asm/unistd_x32.h
  SYSTBL  arch/x86/include/generated/asm/syscalls_32.h
  SYSHDR  arch/x86/include/generated/asm/unistd_32_ia32.h
  SYSHDR  arch/x86/include/generated/asm/unistd_64_x32.h
  SYSTBL  arch/x86/include/generated/asm/syscalls_64.h
  HYPERCALLS arch/x86/include/generated/asm/xen-hypercalls.h
  HOSTCC  arch/x86/tools/relocs_32.o
  HOSTCC  arch/x86/tools/relocs_64.o
  HOSTCC  arch/x86/tools/relocs_common.o
  HOSTLD  arch/x86/tools/relocs
  HOSTCC  scripts/genksyms/genksyms.o
  YACC    scripts/genksyms/parse.tab.[ch]
```

If you are compiling the kernel on Ubuntu, you may receive the following error that interrupts the building process:

No rule to make target 'debian/canonical-certs.pem

Disable the conflicting security certificates by executing the two commands below:

scripts/config --disable SYSTEM_TRUSTED_KEYS

scripts/config --disable SYSTEM_REVOCATION_KEYS

The commands return no output. Start the building process again with make, and press Enter repeatedly to confirm the default options for the generation of new certificates.

2. Install the required modules with this command:

sudo make modules_install

```
marko@pnap:~/linux-6.0.7$ sudo make modules_install
  INSTALL sound/usb/line6/snd-usb-line6.ko
  INSTALL sound/usb/line6/snd-usb-pod.ko
  INSTALL sound/usb/line6/snd-usb-podhd.ko
  INSTALL sound/usb/line6/snd-usb-toneport.ko
  INSTALL sound/usb/line6/snd-usb-variax.ko
  INSTALL sound/usb/misc/snd-ua101.ko
  INSTALL sound/usb/snd-usb-audio.ko
  INSTALL sound/usb/snd-usbmidi-lib.ko
  INSTALL sound/usb/usx2y/snd-usb-us122l.ko
  INSTALL sound/usb/usx2y/snd-usb-usx2y.ko
  INSTALL sound/x86/snd-hdmi-lpe-audio.ko
  INSTALL sound/xen/snd_xen_front.ko
  DEPMOD  6.0.7
marko@pnap:~/linux-6.0.7$
```

3. Finally, install the kernel by typing:

sudo make install

The output shows done when finished:

```
marko@pnap:~/linux-6.0.7$ sudo make install
sh ./arch/x86/boot/install.sh 6.0.7 arch/x86/boot/bzImage \
        System.map "/boot"
run-parts: executing /etc/kernel/postinst.d/apt-auto-removal 6.0.7 /boot/vmlinuz-6.0.7
run-parts: executing /etc/kernel/postinst.d/dkms 6.0.7 /boot/vmlinuz-6.0.7
 * dkms: running auto installation service for kernel 6.0.7                  [ OK ]
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 6.0.7 /boot/vmlinuz-6.0.7
update-initramfs: Generating /boot/initrd.img-6.0.7
run-parts: executing /etc/kernel/postinst.d/update-notifier 6.0.7 /boot/vmlinuz-6.0.7
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 6.0.7 /boot/vmlinuz-6.0.7
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
done
marko@pnap:~/linux-6.0.7$
```

Step 6: Update the Bootloader (Optional)

The GRUB bootloader is the first program that runs when the system powers on.

The make install command performs this process automatically, but you can also do it manually.

1. Update the initramfs to the installed kernel version:

sudo update-initramfs -c -k 6.0.7

2. Update the GRUB bootloader with this command:

sudo update-grub

The terminal prints out the process and confirmation message:

```
marko@pnap:~/linux-6.0.7$ sudo update-initramfs -c -k 6.0.7
update-initramfs: Generating /boot/initrd.img-6.0.7
marko@pnap:~/linux-6.0.7$ sudo update-grub
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-6.0.7
Found initrd image: /boot/initrd.img-6.0.7
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
done
```

Step 7: Reboot and Verify Kernel Version

When you complete the steps above, reboot the machine.

When the system boots up, verify the kernel version using the uname command:

uname -mrs

The terminal prints out the current Linux kernel version.

```
marko@pnap:~$ uname -mrs
Linux 6.0.7 x86_64
marko@pnap:~$
```

Result:

Thus, the procedure was followed and the kernel file of Linux OS was updated.

EX.NO: 2	UTILIZATION OF GNU TOOLCHAINS FOR EFFECTIVE SYSTEM PROGRAMMING
DATE:	

Aim:

- To develop a C program and execute various GNU Bin Utils tools.
- To debug a C program using GDB tools in a Linux system.

Software requirement:

- Ubuntu Linux distros
- Text editors like gedit, vi in Linux with gcc

Procedure:

- Develop a C program main.c in gedit text editor.
- Create C files mul.c, div.c in gedit text editor which contains the prototype of the main program module main.c
- Create a header with .h in gedit and insert the prototype signatures into the header file head.h.
- Include the header head.h in main.c program.
- After the creation of these files, start building the executable files as follows in the terminal.

// Creation of object files

$ gcc –c mul.c

$ gcc –c div.c

$ gcc –i main.c

- Create a static library using GNU Bin Utils tool (ar)

$ ar rs libhead.a mul.o div.o

- Link the object files to create executable file

$ gcc -o main main.o libhead.a (or)

$ gcc -o pattern -L . pattern.o -I pattern.a

- Apply different GNU tool chains [GDB, Bin Utils (objdump, nm, strings, strips)] and observe the generated output from the terminal window.

Program:

Program for Head.h

```
int mul(int,int);
int div(int,int);
```

Program for mul.c

```
int mul(a,b)
{
        return(a*b);
}
```

Program for div.c

```
int div(c,d)
{
        return(c/d);
}
```

Program for main.c

```
# include <stdio.h>
# include “Head.h”
void main()
{
        int x;
        int y;
        printf(“Enter x,y \n”);
        scanf(“%d %d”, &x,&y);
        printf(“%d”,mul(x,y));
        printf(“%d”,div(x,y));
}
```

Commands:

- $ gcc -c mul.c
- $ gcc -c div.c
- $ gcc -c main.c
- $ ar rs libhead.a mul.o div.o
- $ gcc -o main main.o libhead.a
- objdump main.o
- strings main.o
- size main.o
- nm main.o
- strip main.o

Program to reverse the number:

```
#include<stdio.h>
int main()
{
int n,rev=0;
printf(“Enter the number:\n”);
scanf(“%d”,&n);
while(n!=0)
{
        rev=rev *10 + (n%10);
        n=n/10;
}
printf(“Enter the reverse number=%d\n”,rev);
return 0;
}
```

Compile the program and execute the following steps:

- gcc -g filename.c
- gdb -q a.out

Output:

head.h

```
1 int mul(int,int);
2 int div(int,int);
3
```

mul.c

```
1 int mul(int a,int b)
2 {
3         return(a*b);
4 }
5
```

div.c

```
1 int div(int c,int d)
2 {
3         return(c/d);
4 }
5
```

```
ubuntu@ubuntu:~$ strip main.o
ubuntu@ubuntu:~$ size main.o
   text    data     bss     dec     hex filename
    335       0       0     335     14f main.o
```

main.c

```
1 #include <stdio.h>
2 #include "head.h"
3 void main()
4 {
5         int x; int y;
6         printf("Enter x,y:");
7         scanf("%d %d", &x,&y);
8         printf("Multiplication of x and y is %d\n",mul(x,y));
9         printf("Division of x and y is %d\n",div(x,y));
10 }
11
```

```
ubuntu@ubuntu:~$ gedit head.h
ubuntu@ubuntu:~$ gedit mul.c
ubuntu@ubuntu:~$ gedit div.c
ubuntu@ubuntu:~$ gedit main.c
ubuntu@ubuntu:~$ gcc -c mul.c
ubuntu@ubuntu:~$ gcc -c div.c
ubuntu@ubuntu:~$ gcc -c main.c
ubuntu@ubuntu:~$ ar rs libhead.a mul.o div.o
ubuntu@ubuntu:~$ gcc -o main main.o libhead.a
```

```
ubuntu@ubuntu:~$ strings main.o
Enter x,y:
%d %d
Multiplication of x and y is %d
Division of x and y is %d
GCC: (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0
main.c
main
_GLOBAL_OFFSET_TABLE_
printf
__isoc99_scanf
__stack_chk_fail
.symtab
.strtab
.shstrtab
.rela.text
.data
.bss
.rodata
.comment
.note.GNU-stack
.note.gnu.property
.rela.eh_frame
ubuntu@ubuntu:~$ nm main.o
                 U div
                 U _GLOBAL_OFFSET_TABLE_
                 U __isoc99_scanf
0000000000000000 T main
                 U mul
                 U printf
                 U __stack_chk_fail
```

```
ubuntu@ubuntu:~$ objdump -t main

main:     file format elf64-x86-64

SYMBOL TABLE:
0000000000000318 l    d  .interp        0000000000000000              .interp
0000000000000338 l    d  .note.gnu.property     0000000000000000              .note.gnu.property
0000000000000358 l    d  .note.gnu.build-id     0000000000000000              .note.gnu.build-id
000000000000037c l    d  .note.ABI-tag  0000000000000000              .note.ABI-tag
00000000000003a0 l    d  .gnu.hash      0000000000000000              .gnu.hash
00000000000003c8 l    d  .dynsym        0000000000000000              .dynsym
00000000000004b8 l    d  .dynstr        0000000000000000              .dynstr
0000000000000574 l    d  .gnu.version   0000000000000000              .gnu.version
0000000000000588 l    d  .gnu.version_r 0000000000000000              .gnu.version_r
00000000000005c8 l    d  .rela.dyn      0000000000000000              .rela.dyn
0000000000000688 l    d  .rela.plt      0000000000000000              .rela.plt
0000000000001000 l    d  .init  0000000000000000              .init
0000000000001020 l    d  .plt   0000000000000000              .plt
0000000000001060 l    d  .plt.got       0000000000000000              .plt.got
0000000000001070 l    d  .plt.sec       0000000000000000              .plt.sec
00000000000010a0 l    d  .text  0000000000000000              .text
00000000000012d8 l    d  .fini  0000000000000000              .fini
0000000000002000 l    d  .rodata        0000000000000000              .rodata
000000000000205c l    d  .eh_frame_hdr  0000000000000000              .eh_frame_hdr
00000000000020b0 l    d  .eh_frame      0000000000000000              .eh_frame
0000000000003da8 l    d  .init_array    0000000000000000              .init_array
0000000000003db0 l    d  .fini_array    0000000000000000              .fini_array
0000000000003db8 l    d  .dynamic       0000000000000000              .dynamic
0000000000003fa8 l    d  .got   0000000000000000              .got
0000000000004000 l    d  .data  0000000000000000              .data
0000000000004010 l    d  .bss   0000000000000000              .bss
0000000000000000 l    d  .comment       0000000000000000              .comment
0000000000000000 l    df *ABS*  0000000000000000              crtstuff.c
00000000000010d0 l     F .text  0000000000000000              deregister_tm_clones
0000000000001100 l     F .text  0000000000000000              register_tm_clones
0000000000001140 l     F .text  0000000000000000              __do_global_dtors_aux
0000000000004010 l     O .bss   0000000000000001              completed.8061
```

reverse.c

```
1 #include<stdio.h>
2 int main()
3 {
4         int n,rev=0;
5         printf("Enter the number:");
6         scanf("%d",&n);
7         while(n!=0)
8         {
9                 rev=rev*10+(n%10);
10                n/=10;
11        }
12        printf("Enter the reverse number=%d\n",rev);
13        return 0;
14 }
```

```
ubuntu@ubuntu:~$ gedit reverse.c
ubuntu@ubuntu:~$ gcc -g reverse.c
ubuntu@ubuntu:~$ gdb -q a.out
Reading symbols from a.out...
(gdb) r
Starting program: /home/ubuntu/a.out
Enter the number:1234
Enter the reverse number=4321
[Inferior 1 (process 4751) exited normally]
(gdb)
```

Result:

Thus, different GNU tools were applied for the developed ‘C’ program and the output from the terminal window was observed and verified successfully.

EX.NO: 3	DEBUGGING PROGRAMS USING CSCOPE
DATE:	

Aim:

To familiarize basic Cscope tools and experiment it using a C program.

Software requirement:

- Ubuntu Linux distros.
- Text editor like gedit
- Cscope tools

Procedure:

- Develop a c program in gedit text editor.
- Compile the developed c program in terminal window.
- Enter the command Cscope in terminal window.
- If not installed Cscope, install using sudo apt-get install Cscope.
- Use Cscope options for the developed C program and observe the results in terminal. window.

Theory:

- Cscope is an interactive,screen oriented tool that allows the user to browse through c source files for specified elements of code.
- By default, Cscope examines the C (.c & .h) source files in the current directory. Cscope may also be invoked for source files named on the command line. In either case, Cscope searches the standard directories for the #include files that it does not find in the current directory. Cscope uses a symbol reference cross-reference, called Cscope, out by default,to locate functions, function calls, Macros, variables and pre-processors symbols in the files.
- Cscope builds the symbol cross-reference for the first time it is used on the source files for the program being browsed. On a subsequent invocation, Cscope rebuilds the cross-reference only if a source file has changed or the list of source files is different. When the cross-reference is re-built, the data for the unchanged files from the cross- reference are copied, which makes rebuilding faster than the initial build.

Requesting the initial search:

After the cross-reference is ready, cscope will display this menu:

- Find this C symbol:
- Find this function definition:
- Find functions called by this function:
- Find functions calling this function:
- Find this text string:
- Change this text string:
- Find this egrep pattern:

- Find this file:
- Find files #including this file:
- Press the <Up> or <Down> keys repeatedly to move to the desired input field, type the text to search for, and then press the <Return> key.

Issuing subsequent requests using Cscope options:

- If the search is successful, any of these single-character commands can be used:

0-9a-zA-Z

- Edit the file referenced by the given line number.

<Space>

- Display next set of matching lines.

<Tab>

- Alternate between the menu and the list of matching lines

<Up>

- Move to the previous menu item (if the cursor is in the menu) or move to the previous matching line (if the cursor is in the matching line list.)

<Down>

- Move to the next menu item (if the cursor is in the menu) or move to the next matching line (if the cursor is in the matching line list.)

+

- Display next set of matching lines.

-

- Display previous set of matching lines.

^e

- Edit displayed files in order.

>

- Write the displayed list of lines to a file.

>>

- Append the displayed list of lines to a file.

<

- Read lines from a file that is in symbol reference format (created by > or >>), just like the -F option.

^

- Filter all lines through a shell command and display the resulting lines, replacing the lines that were already there.

|

- Pipe all lines to a shell command and display them without changing them. At any time these single-character commands can also be used:

<Return>

- Move to next input field.

^n

- Move to previous input field.

^y

- Search with the last text typed.

^b

- Move to previous input field and search pattern.

^f

- Move to next input field and search pattern.

^c

- Toggle ignore/use letter case when searching. (When ignoring letter case, search for ``FILE'' will match ``File'' and ``file''.)

^r

- Rebuild the cross-reference.

!

- Start an interactive shell (type ^d to return to cscope).

^l

- Redraw the screen.

?

- Give help information about cscope commands.

^d

- Exit Cscope.

Note:

If the first character of the text to be searched for matches one of the above commands, escape it by typing a (backslash) first. Substituting new text for old text.

After the text to be changed has been typed, Cscope will prompt for the new text, and then it will display the lines containing the old text. Select the lines to be changed with these single- character commands:

0-9a-zA-Z

Mark or unmark the line to be changed.

*

Mark or unmark all displayed lines to be changed.

<Space>

Display next set of lines.

+

Display next set of lines.

-

Display previous set of lines.

a

Mark or unmark all lines to be changed.

^d

Change the marked lines and exit.

<Esc>

Exit without changing the marked lines.

!

Start an interactive shell (type ^d to return to Cscope).

^l

Redraw the screen.

?

Give help information about Cscope commands.

Special keys

If your terminal has arrow keys that work in vi, you can use them to move around the input fields. The up-arrow key is useful to move to the previous input field instead of using the <Tab> key repeatedly. If you have <CLEAR>, <NEXT>, or <PREV> keys they will act as the ^l, +, and - commands, respectively.

Output:

```
┌──(kali㉿kali)-[~]
└─$ sudo apt install gedit
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following additional packages will be installed:
  docbook-xml fonts-dejavu gedit-common gir1.2-gtksource-4 gir1.2-peas-1.0 libamtk-5-0 libamtk-5-comm
  libjavascriptcoregtk-4.1-0 libpeas-1.0-0 libpeas-common libpython3.11 libpython3.11-minimal libpyth
  python3.11 python3.11-minimal sgml-base sgml-data xml-core yelp yelp-xsl
Suggested packages:
  docbook docbook-dsssl docbook-xsl docbook-defguide gedit-plugins glibc-doc libnss-nis libnss-nisplu
  perlsgml w3-recs opensp debhelper
Recommended packages:
  manpages-dev libc-devtools
The following NEW packages will be installed:
  docbook-xml fonts-dejavu gedit gedit-common gir1.2-gtksource-4 gir1.2-peas-1.0 libamtk-5-0 libamtk-
  libpython3.11 libpython3.11-minimal libpython3.11-stdlib libssl3 libtepl-6-2 libtepl-common libwebk
  yelp-xsl
The following packages will be upgraded:
  libc-bin libc-dev-bin libc-l10n libc6 libc6-dev libc6-i386 locales
7 upgraded, 28 newly installed, 0 to remove and 1759 not upgraded.
Need to get 34.0 MB/59.1 MB of archives.
After this operation, 185 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

```
┌──(kali㉿kali)-[~]
└─$ sudo apt install cscope
[sudo] password for kali:
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
Suggested packages:
  cscope-el
The following NEW packages will be installed:
  cscope
0 upgraded, 1 newly installed, 0 to remove and 1759 not upgraded.
Need to get 223 kB of archives.
After this operation, 1,260 kB of additional disk space will be used.
Get:1 http://kali.download/kali kali-rolling/main amd64 cscope amd64 15.9-1 [223 kB]
Fetched 223 kB in 20s (11.0 kB/s)
Selecting previously unselected package cscope.
(Reading database ... 301968 files and directories currently installed.)
Preparing to unpack .../cscope_15.9-1_amd64.deb ...
Unpacking cscope (15.9-1) ...
Setting up cscope (15.9-1) ...
Processing triggers for kali-menu (2022.2.0) ...
Processing triggers for man-db (2.10.2-1) ...

┌──(kali㉿kali)-[~]
└─$ cscope -v
```

Open

```
1 #include<stdio.h>
2 void sum(int,int);
3 void main()
4 {
5         int a,b;
6         printf("Enter two numbers : ");
7         scanf("%d %d",&a,&b);
8         sum(a,b);
9 }
10
11 void sum(int a,int b)
12 {
13         int c;
14         c=a+b;
15         printf("The sum=%d",c);
16 }
```

```
┌──(kali㉿kali)-[~]
└─$ gedit sum_of_nos.c
^C

┌──(kali㉿kali)-[~]
└─$ gcc sum_of_nos.c

┌──(kali㉿kali)-[~]
└─$ ./a.out
Enter two numbers : 14 26
The sum=40
```

```
File  Actions  Edit  View  Help
#include<stdio.h>
int main()
{
        int num,temp,fact=1;
        printf("Enter a number:");
        scanf("%d",&num);
        temp=num;
        while(num≠0)
        {
                fact*=num;
                num--;
        }
        printf("The factorial of %d is %d",temp,fact);
        return 0;
}
```

```
┌──(kali㉿kali)-[~]
└─$ vim factorial.c

┌──(kali㉿kali)-[~]
└─$ gcc factorial.c

┌──(kali㉿kali)-[~]
└─$ ./a.out
Enter a number:7
The factorial of 7 is 5040
```

```
C symbol: a

  File    Function Line
0 cprog.c main     3 int a = 12;
1 cprog.c main     5 printf("%d",a+b);

Find this C symbol:
Find this global definition:
Find functions called by this function:
Find functions calling this function:
Find this text string:
Change this text string:
Find this egrep pattern:
Find this file:
Find files #including this file:
Find assignments to this symbol:
```

```
Files #including this file: stdio.h

  File          Line
0 cprog.c         1 #include <stdio.h>
1 factorial.c     1 #include <stdio.h>
2 odd_or_even.c   1 #include <stdio.h>
3 sum_of_nos.c    1 #include <stdio.h>
4 stdio.h       902 #include <bits/stdio.h>

Find this C symbol:
Find this global definition:
Find functions called by this function:
Find functions calling this function:
Find this text string:
Change this text string:
Find this egrep pattern:
Find this file:
Find files #including this file:
Find assignments to this symbol:
```

```
File: odd_or_even

  File
0 odd_or_even.c

Find this C symbol:
Find this global definition:
Find functions called by this function:
Find functions calling this function:
Find this text string:
Change this text string:
Find this egrep pattern:
Find this file:
Find files #including this file:
Find assignments to this symbol:
```

Result:

Thus, we have experimented Cscope tools with an example C program.

EX.NO: 4	CONSTRUCT CHARACTER ORIENTED DEVICE DRIVERS
DATE:	

Aim:

To construct a simple character device driver program in Linux.

Description:

- The device drivers are embedded software modules that contain the functionality to operate the individual hardware devices.
- The reason for the device driver software is to remove the need for the application to know how to control each piece of hardware.
- Each individual device driver would typically need to know only how to control its hardware device
- Character device drivers are used for driving sequential access devices. The amount of data accessed is not of fixed size.
- The character device drivers are accessed by the application using the standard calls such as open, read, write.
- The role of a driver is to provide mechanisms which allow normal users to access protected parts of its system, in particular ports, registers and memory addresses normally managed by the operating system.
- One of the good features of Linux is the ability to extend at runtime the set of the features offered by the kernel. Users can add or remove functionalities to the kernel while the system is running.
- These \programs" that can be added to the kernel at runtime are called \module" and built into individuals with .ko (Kernel object) extension.

The Linux kernel takes advantage of the possibility to write kernel drivers as modules which can be uploaded on request.

Commands:

Command	*Description*
$ uname –r	Returns a string naming the current system
$ ls	To check object file created or not in the specified directory
$ sudo dmesg	To see the message communicated by modules to the kernel
$ sudo dmesg –C	To clear the communicated message
$ sudo dmesg	To check message communication
$ lsmod	List all the modules running in the systems

$ sudo insmod simpleDriver.ko	(here simpleDriverf is user defined file.. ko kernel object file) It inserts the simpleDriver module in the list
$ sudo rmmod simpleDriver.k	To remove kernel object (now the module is removed successfully check the command

Code:

hello.c

```
#include <linux/module.h>
#include <linux/init.h>

/*META INFORMATION*/
MODULE_LICENSE("GPL");
MODULE_AUTHOR("Raghav 4 GNU/Linux");
MODULE_DESCRIPTION("A hello world Linux kernal module");

// @brief This function is called, when the module is loaded into the kernel
static int __init hello_start(void)
{
        printk ("Hello, I'm here to help\n");
        return 0;
}

// @brief This function is called, when the module is removed into the kernel
static void __exit hello_end(void)
{
        printk("Goodbye, I hope I was helpful\n");
}

module_init(hello_start);
module_exit(hello_end);
```

Makefile:

```
obj-m += hello.o
KVERSION = $(shell uname -r)
all:
        make -C /lib/modules/$(KVERSION)/build M=$(PWD) modules
clean:
        make -C /lib/modules/$(KVERSION)/build M=$(PWD) clean
```

Output:

```
ubuntu@ubuntu:~$ mkdir character_device_driver
ubuntu@ubuntu:~$ cd character_device_driver
ubuntu@ubuntu:~/character_device_driver$ gedit hello.c
ubuntu@ubuntu:~/character_device_driver$ gedit Makefile
ubuntu@ubuntu:~/character_device_driver$ make
make -C /lib/modules/5.15.0-58-generic/build M=/home/ubuntu/character_device_driver modules
make[1]: Entering directory '/usr/src/linux-headers-5.15.0-58-generic'
  CC [M]  /home/ubuntu/character_device_driver/hello.o
  MODPOST /home/ubuntu/character_device_driver/Module.symvers
  CC [M]  /home/ubuntu/character_device_driver/hello.mod.o
  LD [M]  /home/ubuntu/character_device_driver/hello.ko
  BTF [M] /home/ubuntu/character_device_driver/hello.ko
Skipping BTF generation for /home/ubuntu/character_device_driver/hello.ko due to unavailability of vmlinux
make[1]: Leaving directory '/usr/src/linux-headers-5.15.0-58-generic'
ubuntu@ubuntu:~/character_device_driver$ ls
hello.c   hello.mod    hello.mod.o  Makefile       Module.symvers
hello.ko  hello.mod.c  hello.o      modules.order
ubuntu@ubuntu:~/character_device_driver$ sudo insmod hello.ko
[sudo] password for ubuntu:
```

Open — hello.c — ~/character_device_driver

```c
#include <linux/module.h>
#include <linux/init.h>

/*META INFORMATION*/
MODULE_LICENSE("GPL");
MODULE_AUTHOR("Raghav 4 GNU/Linux");
MODULE_DESCRIPTION("A hello world Linux kernal module");

// @brief This function is called, when the module is loaded into the kernel
static int __init hello_start(void)
{
        printk ("Hello, I'm here to help\n");
        return 0;
}

// @brief This function is called, when the module is removed into the kernel
static void __exit hello_end(void)
{
        printk("Goodbye, I hope I was helpful\n");
}

module_init(hello_start);
module_exit(hello_end);

```

Open — Makefile — ~/character_device_driver

```
obj-m += hello.o
KVERSION = $(shell uname -r)
all:
	make -C /lib/modules/$(KVERSION)/build M=$(PWD) modules
clean:
	make -C /lib/modules/$(KVERSION)/build M=$(PWD) clean

```

```
ubuntu@ubuntu:~/character_device_driver$ lsmod
Module                  Size  Used by
hello                  16384  0
vsock_loopback         16384  0
vmw_vsock_virtio_transport_common    40960  1 vsock_loopback
vmw_vsock_vmci_transport    32768  2
vsock                  45056  7 vmw_vsock_virtio_transport_common,vsock_loopback,vmw_vsock_vmci_transport
nls_iso8859_1          16384  1
snd_ens1371            32768  2
snd_ac97_codec        155648  1 snd_ens1371
binfmt_misc            24576  1
gameport               24576  1 snd_ens1371
ac97_bus               16384  1 snd_ac97_codec
intel_rapl_msr         20480  0
intel_rapl_common      40960  1 intel_rapl_msr
snd_pcm               135168  2 snd_ac97_codec,snd_ens1371
snd_seq_midi           20480  0
crct10dif_pclmul       16384  1
snd_seq_midi_event     16384  1 snd_seq_midi
ghash_clmulni_intel    16384  0
snd_rawmidi            49152  2 snd_seq_midi,snd_ens1371
vmw_balloon            24576  0
aesni_intel           376832  0
crypto_simd            16384  1 aesni_intel
cryptd                 24576  2 crypto_simd,ghash_clmulni_intel
snd_seq                77824  2 snd_seq_midi,snd_seq_midi_event
snd_seq_device         16384  3 snd_seq,snd_seq_midi,snd_rawmidi
snd_timer              40960  2 snd_seq,snd_pcm
joydev                 32768  0
```

```
ubuntu@ubuntu:~/character_device_driver$ lsmod | grep hello
hello                  16384  0
```

```
ubuntu@ubuntu:~/character_device_driver$ dmesg | tail
[   12.935300] rfkill: input handler disabled
[   83.441481] rfkill: input handler enabled
[   90.273891] rfkill: input handler disabled
[  878.621137] perf: interrupt took too long (2768 > 2500), lowering kernel.perf_event_max_sample_rate to 72250
[ 1014.491648] hello: loading out-of-tree module taints kernel.
[ 1014.491683] hello: module verification failed: signature and/or required key missing - tainting kernel
[ 1014.492388] Hello, I'm here to help
[ 1100.525380] Goodbye, I hope I was helpful
[ 1230.154669] perf: interrupt took too long (3519 > 3460), lowering kernel.perf_event_max_sample_rate to 56750
[ 1433.640951] Hello, I'm here to help
ubuntu@ubuntu:~/character_device_driver$ sudo rmmod hello
ubuntu@ubuntu:~/character_device_driver$ dmesg | tail
[   83.441481] rfkill: input handler enabled
[   90.273891] rfkill: input handler disabled
[  878.621137] perf: interrupt took too long (2768 > 2500), lowering kernel.perf_event_max_sample_rate to 72250
[ 1014.491648] hello: loading out-of-tree module taints kernel.
[ 1014.491683] hello: module verification failed: signature and/or required key missing - tainting kernel
[ 1014.492388] Hello, I'm here to help
[ 1100.525380] Goodbye, I hope I was helpful
[ 1230.154669] perf: interrupt took too long (3519 > 3460), lowering kernel.perf_event_max_sample_rate to 56750
[ 1433.640951] Hello, I'm here to help
[ 1539.310597] Goodbye, I hope I was helpful
```

```
ubuntu@ubuntu:~/character_device_driver$ modinfo hello.ko
filename:       /home/ubuntu/character_device_driver/hello.ko
description:    A hello world Linux kernal module
author:         Raghav 4 GNU/Linux
license:        GPL
srcversion:     7CC42D0B45E4422A5624400
depends:
retpoline:      Y
name:           hello
vermagic:       5.15.0-58-generic SMP mod_unload modversions
```

Result:

The C program is written to create Character Device Driver program and output is verified successfully.

EX.NO: 5	IMPLEMENTATION OF TASK MANAGEMENT IN REAL-TIME OPERATING SYSTEMS (RTOS) USING MICROC/OS-II
DATE:	

Aim:

To develop a C program for creating tasks using FreeRTOS APIs.

Software Requirement:

- Ubuntu Linux distros
- Text editors like gedit, vi in linux with gcc

Description:

- In FreeRTOS, an application can consist of many tasks. If the processor running the application contains a single core, then only one task can be executing at any given time. This implies that a task can exist in one of two states, Running and Not Running.
- When a task is in the Running state the processor is executing the task's code. When a task is in the Not Running state, the task is dormant, its status having been saved ready for it to resume execution the next time the scheduler decides it should enter the Running state.
- The FreeRTOS scheduler is the only entity that can switch a task in and out.

Creating Tasks: The xTaskCreate() API Function

- Tasks are created using the FreeRTOS xTaskCreate() API function.

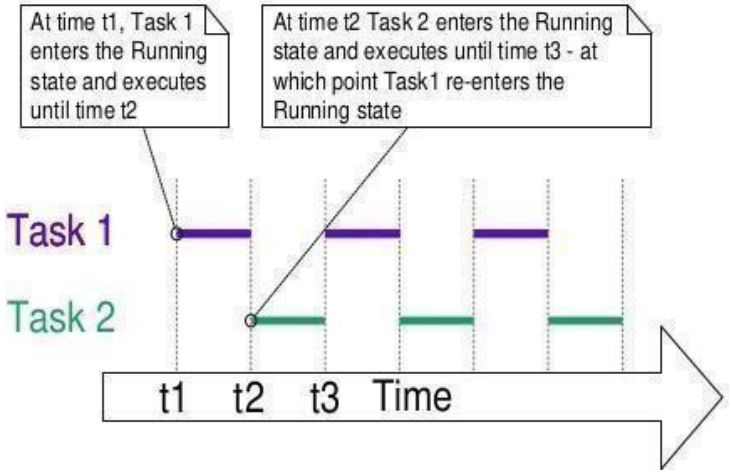
```
BaseType_t xTaskCreate( TaskFunction_t pvTaskCode,
                        const char * const pcName,
                        uint16_t usStackDepth,
                        void *pvParameters,
                        UBaseType_t uxPriority,
                        TaskHandle_t *pxCreatedTask );
```

- pvTaskCode parameter is simply a pointer to the function that implements the task (in effect, just the name of the function).
- pcName A descriptive name for the task. this is not used by FreeRTOS in any way. It is included purely as a debugging aid. Identifying a task by a human readable name is much simpler than attempting to identify it by its handle.
- usStackDepth Each task has its own unique stack that is allocated by the kernel to the task when the task is created. The usStackDepth value tells the kernel how large to make the stack.
- pvParameters Task functions accept a parameter of type pointer to void (void*). The value assigned to pvParameters is the value passed into the task.
- uxPriority Defines the priority at which the task will execute. Priorities can be assigned from 0, which is the lowest priority, to (configMAX_PRIORITIES – 1), which is the highest priority.
- pxCreatedTask This can be used to pass out a handle to the task being created. This handle can then be used to reference the task in API calls that, for example, change the task priority

or delete the task. If your application has no use for the task handle, then pxCreatedTask can be set to NULL.

Returned value

- pdPASS This indicates that the task has been created successfully.
- pdFAIL This indicates that the task has not been created because there is insufficient heap memory available for FreeRTOS to allocate enough RAM to hold the task data structures and stack.
- In the below program, two tasks (Task 1 & Task 2) are created as follows:



Procedure:

- Install the dependencies for Ubuntu

  sudo apt-get install libcs6-dev-i386
- Navigate to the C source code

  $cd Project/main.c
- Compile the Makefile using make command

  $make

Program:

```
// Task Creation
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>
#include<FreeRTOS.h>
#include<task.h>
void vTask1(void*);
void vTask2(void*);
void vApplicationIdleHook(void);
int main(void)
{
        xTaskCreate(vTask1,"Task1",1000,NULL,1,NULL);
        xTaskCreate(vTask2,"Task2",1000,NULL,1,NULL);
        vTaskStartScheduler();
        return 0;
}
```

```
void vAssertCalled( unsigned long ulLine, const char * const pcFileName )
{
	taskENTER_CRITICAL();
	{
		printf("[ASSERT] %s:%lu\n", pcFileName, ulLine);
		flush(stdout);
	}
	taskEXIT_CRITICAL();
	exit(-1);
}
void vTask1(void* parameter)
{
	while(1)
	{
		printf("Task1\n");
		sleep(500);
	}
}
void vTask2(void* parameter)
{
	while(1)
	{
		printf("Task2\n");
		sleep(500);
	}
}
void vApplicationIdleHook(void)
{
	//printf("Idle\r\n");
}
```

Output:

```
Task1
Task2
Task1
Task2
Task1
Task2
Task1
Task2
Task1
Task2
Task1
Task2
Task1
Task2
Task1
Task2
Task1
Task2
Task1
Task2
Task1
Task2
Task1
Task2
Task1
Task2
Task1
Task2
Task1
Task2
Task1
Task2
```

Result:

Thus, task creation using FreeRTOS API functions is executed and the output is verified successfully.

EX.NO: 6	IMPLEMENTATION OF INTERUPPT MANAGEMENT IN REAL-TIME OPERATING SYSTEMS (RTOS) USING MICROC/OS-II
DATE:	

Aim:

To develop a C program for scheduling tasks based on "Round Robin algorithm" using FreeRTOS APIs.

Software Requirement:

- Ubuntu Linux distros
- Text editors like gedit, vi in linux with gcc

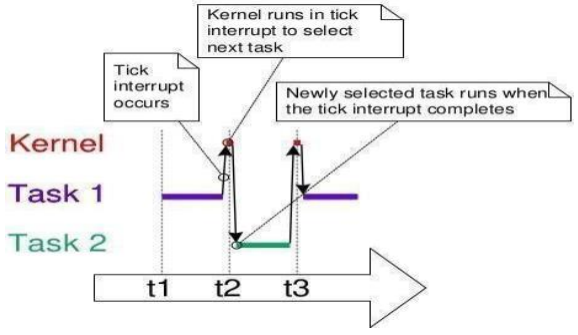
Description:

Task Priorities:

- The priority can be changed after the scheduler has been started by using the vTaskPrioritySet() API function.
- Priorities are defined in configMAX_PRIORITIES compile time configuration constant within FreeRTOSConfig.h.
- Therefore, the range of available priorities is 0 to (configMAX_PRIORITIES – 1).
- The FreeRTOS scheduler will always ensure that the highest priority task that is able to run is the task selected to enter the Running state. Where more than one task of the same priority is able to run, the scheduler will transition each task into and out of the Running state, in turn.

Time Measurement and the Tick Interrupt:

- Scheduling Algorithms, describes an optional feature called 'time slicing' to be able to select the next task to run, the scheduler itself must execute at the end of each time slice 1.
- A periodic interrupt, called the 'tick interrupt', is used for this purpose.
- configTICK_RATE_HZ compile time configuration constant within FreeRTOSConfig.h.
- configTICK_RATE_HZ is set to 100 (Hz), then the time slice will be 10 milliseconds. The time between two tick interrupts is called the 'tick period'. One time slice equals one tick period.
- The optimal value for configTICK_RATE_HZ is dependent on the application being developed, although a value of 100 is typical.

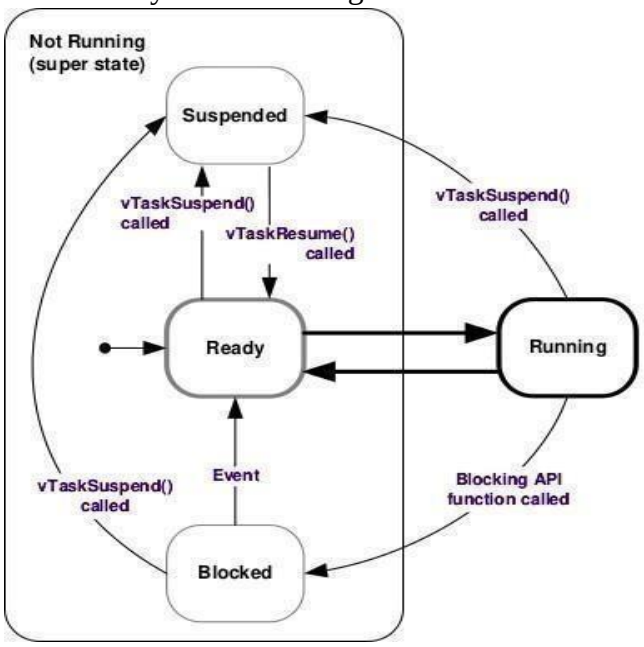


- FreeRTOS API calls always specify time in multiples of tick periods, which are often referred to simply as 'ticks'. The pdMS_TO_TICKS() macro converts a time specified in milliseconds into a time specified in ticks.

TickType_t xTimeInTicks = pdMS_TO_TICKS(200);

Expanding the 'Not Running' State:

- To make the tasks useful they must be re-written to be event-driven. An event-driven task has work (processing) to perform only after the occurrence of the event that triggers it, and is not able to enter the Running state before that event has occurred.
- A task that is waiting for an event is said to be in the 'Blocked' state, which is a sub-state of the Not Running state.
- Temporal (time-related) events—the event being either a delay period expiring, or an absolute time being reached. For example, a task may enter the Blocked state to waitfor 10 milliseconds to pass.
- Synchronization events—where the events originate from another task or interrupt. For example, a task may enter the Blocked state to wait for data to arrive on a queue. Synchronization events cover a broad range of event types.
- The Suspended State Suspended' is also a sub-state of Not Running. Tasks in the Suspended state are not available to the scheduler. The only way into the Suspended state is through a call to the vTaskSuspend() API function, the only way out being through a call to the vTaskResume() or xTaskResumeFromISR() API functions.
- The Ready State Tasks that are in the Not Running state but are not Blocked or Suspended are said to be in the Ready state. They are able to run, and therefore 'ready' to run, but are not currently in the Running state.



- ***vTaskDelay()*** places the calling task into the Blocked state for a fixed number of tick interrupts.

void vTaskDelay(TickType_t xTicksToDelay);

- vTaskDelay(pdMS_TO_TICKS(100)) will result in the calling task remaining in the Blocked state for 100 milliseconds.

The vTaskDelayUntil() API Function

- The parameters to vTaskDelayUntil() specify, instead, the exact tick count value at which the calling task should be moved from the Blocked state into the Ready state.
- vTaskDelayUntil() is the API function that should be used when a fixed execution period is required (where you want your task to execute periodically with a fixed frequency), as the time at which the calling task is unblocked is absolute, rather than relative to when the function was called (as is the case with vTaskDelay()).
- void vTaskDelayUntil(TickType_t * pxPreviousWakeTime, TickType_t xTimeIncrement);
- pxPreviousWakeTime : This parameter is named on the assumption that vTaskDelayUntil() is being used to implement a task that executes periodically and with a fixed frequency. In this case, pxPreviousWakeTime holds the time at which the task last left the Blocked state (was 'woken' up). This time is used as a reference point to calculate the time at which the task should next leave the Blocked state.
- xTimeIncrement This parameter is also named on the assumption that vTaskDelayUntil() is being used to implement a task that executes periodically and with a fixed frequency—the frequency being set by the xTimeIncrement value.
- The xLastWakeTime variable needs to be initialized with the current tick count. Note that this is the only time the variable is explicitly written to. After this xLastWakeTime is managed automatically by the vTaskDelayUntil() API function.

The Idle Task and the Idle Task Hook

There must always be at least one task that can enter the Running state. To ensure this is the case, an Idle task is automatically created by the scheduler when **vTaskStartScheduler**() is called.

- The idle task has the lowest possible priority (priority zero), to ensure it never prevents a higher priority application task from entering the Running state.

Idle Task Hook Functions

- To add application specific functionality directly into the idle task through the use of an idle hook (or idle callback) function—a function that is called automatically by the idle task once per iteration of the idle task loop.
- Placing the processor into a low power mode.
- An Idle task hook function must never attempt to block or suspend.
- Idle task is responsible for cleaning up kernel resources after a task has been deleted. If the idle task remains permanently in the Idle hook function, then this clean-up cannot occur.

void vApplicationIdleHook(void);

Procedure:

- Install the dependencies for Ubuntu

  sudo apt-get install libcs6-dev-i386
- Navigate to the C source code

  $cd Project/main.c
- Compile the Makefile using make command

  $make

Program:

```
// Task Scheduling using Round Robin algorithm
#include <stdio.h>
#include <stdlib.h>
#include <FreeRTOS.h>
#include <task.h>
#include <timers.h>
#define TASKSCHEDULER
#ifdef TASKSCHEDULER
void vTask1(void*);
void vTask2(void*);
void vTask3(void*);
void vTask4(void*);
#endif
void vApplicationIdleHook(void);
int main(void)
{
	#ifdef TASKSCHEDULER
	xTaskCreate( vTask1, "Task 1", 1000, NULL, 1, NULL );
	xTaskCreate( vTask2, "Task 2", 1000, NULL, 1, NULL );
	xTaskCreate( vTask3, "Task 3", 1000, NULL, 1, NULL );
	xTaskCreate( vTask4, "Task 4", 1000, NULL, 1, NULL );
	#endif
	vTaskStartScheduler();
	return 0;
}
void vAssertCalled( unsigned long ulLine, const char * const pcFileName )
{
	taskENTER_CRITICAL();
	{
		printf("[ASSERT] %s:%lu\n", pcFileName, ulLine);
		flush(stdout);
	}
	taskEXIT_CRITICAL();
	exit(-1);
}
#ifdef TASKSCHEDULER
void vTask1(void* parameter)
{
	while(1)
	{
		printf("Task 1\n");
		vTaskDelay(pdMS_TO_TICKS(250));
	}
}
```

```
void vTask2(void* parameter)
{
	while(1)
	{
		printf("Task 2\n");
		vTaskDelay(pdMS_TO_TICKS(250));
	}
}
void vTask3(void* parameter)
{
	TickType_t xLastWaketime = xTaskGetTickCount(); while(1)
	{
		printf("Task 3 with 250ms\n");
		vTaskDelayUntil(&xLastWaketime, pdMS_TO_TICKS(250));
	}
}
void vTask4(void* parameter)
{
	TickType_t xLastWaketime = xTaskGetTickCount();
	while(1)
	 {
		printf("Task 4 with 500ms\n");
		vTaskDelayUntil(&xLastWaketime, pdMS_TO_TICKS(500));
	}
}
#endif

void vApplicationIdleHook(void)
{
//	printf("Idle\r\n");
}
```

Output:

```
BUILD COMPLETE: FreeRTOS-Sim
--------------------------
est@est:~/1802258/pmvankerFreeRTOS$ ./FreeRTOS-Sim
Running as PID: 14919
 Timer Resolution for Run TimeStats is 100 ticks per second.
Task 1
Task 2
Task 3 with 250ms
Task 4 with 500ms
Task 1
Task 2
Task 3 with 250ms
Task 4 with 500ms
Task 1
Task 2
Task 3 with 250ms
Task 1
Task 2
Task 3 with 250ms
Task 4 with 500ms
Task 1
Task 2
Task 3 with 250ms
Task 1
Task 2
Task 3 with 250ms
Task 4 with 500ms
Task 1
Task 2
Task 3 with 250ms
Task 1
Task 2
Task 3 with 250ms
Task 4 with 500ms
Task 1
Task 2
Task 3 with 250ms
Task 1
Task 2
```

Result:

Thus, tasks were created and scheduled based on “Round Robin algorithm” using FreeRTOS APIs and the output is verified successfully.

EX.NO: 7	DEVELOPMENT OF BLUETOOTH INTERFACING USING MSP430 LAUNCHPAD
DATE:	

Aim:

To write a sketch program to connect the Bluetooth Module with MSP430G2553 to control a LED.

Apparatus Required:

- MSP430G2553 Launchpad
- Energia IDE
- HC-05 Bluetooth module.

Procedure:

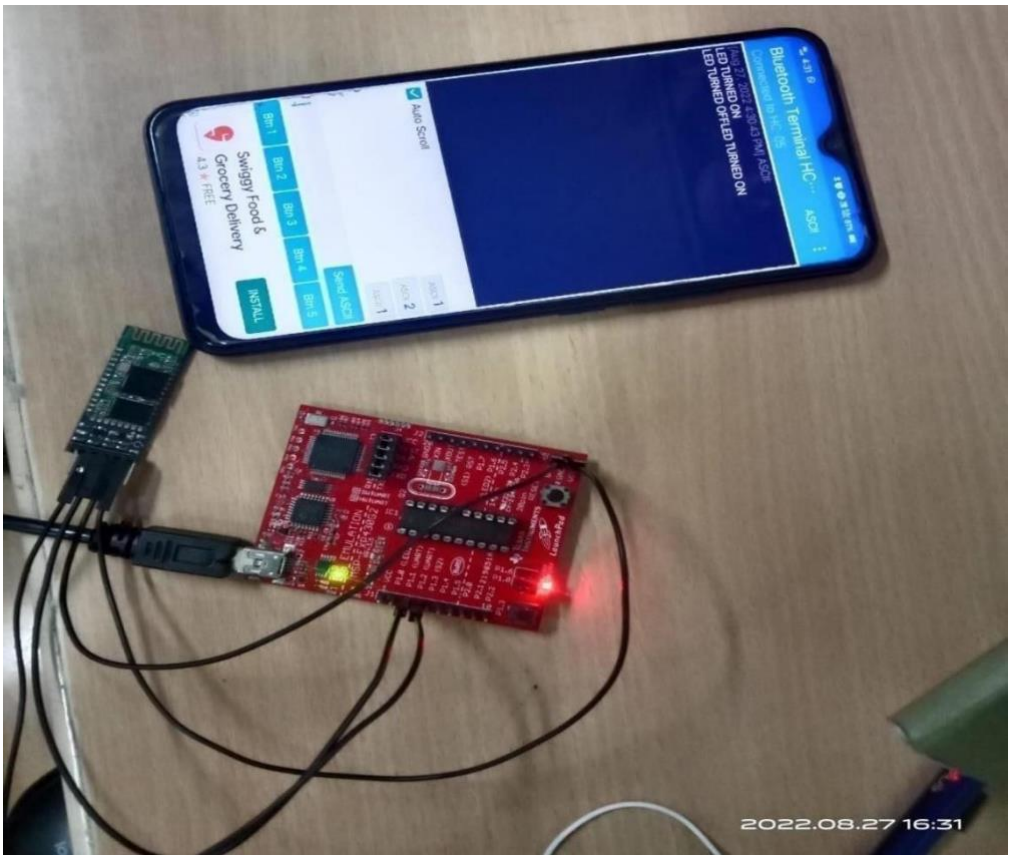
- Attach the MSP430G2553 board with the system.
- Attach the Bluetooth Module with MSP430G2553 board.
- Double click on Energia IDE on the desktop.
- Select the board type as MSP430G2553 Launchpad from Tool.
- Create a new program on Energia IDE and save it.
- Compile the program and upload it to the MSP430G2553 Launchpad board.
- Run the program and verify the output by controlling the LED using an Android application on mobile.

Program:

```
#define LED RED_LED
void setup()
{
	Serial.begin(9600);
	pinMode(2, OUTPUT);
}
void loop()
{
	if (Serial.available())
	{
		char data_received;
		data_received = Serial.read();
		if (data_received == '1')
		{
			digitalWrite(LED, HIGH);
			Serial.write("LED turned ON\n");
		}
```

```
            if (data_received == '2')
            {
                digitalWrite(LED, LOW);
                Serial.write("LED turned OFF\n");
            }
        }
}
```

Output:



Result:

Thus, the sketch program to connect the Bluetooth Module with MSP430G2553 to control a led was implemented successfully.

EX.NO: 8	DEVELOPMENT OF ESP8266 INTERFACING (WIFI) USING MSP430 LAUNCHPAD
DATE:	

Aim:

To write a sketch program to connect the ESP8266 WiFi Module with MSP430G2553 to send a data to browser.

Apparatus Required:

- MSP430G2553 Launchpad
- Energia IDE
- ESP8266 WiFi module.

Procedure:

- Attach the MSP430G2553 board with the system.
- Attach the WiFi Module with MSP430G2553 board.
- Double click on Energia IDE on the desktop.
- Select the board type as MSP430G2553 Launchpad from Tool.
- Create a new program on Energia IDE and save it.
- Compile the program and upload it to the MSP430G2553 Launchpad board.
- Run the program and verify the output by sending data to browser.

Program:

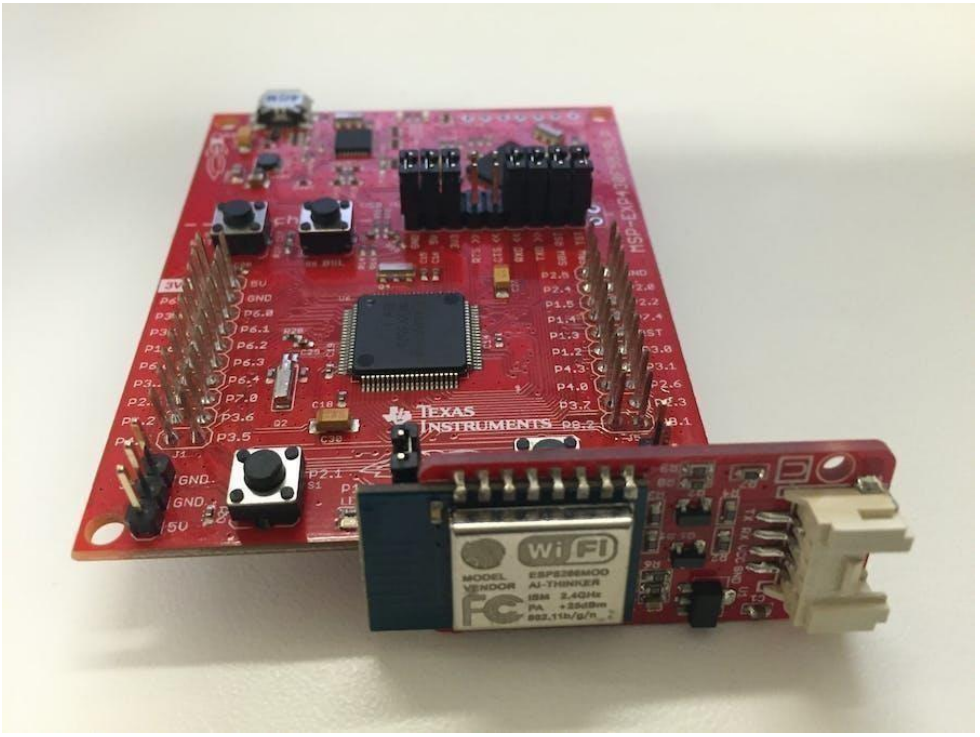
```
#define SSID "RAGHAV"
#define PASS "12345678"
#define DST_IP "things.ubidots.com"
#define idvariable "569fc4ba76254229c49896a6"
int len;

void setup()
{
        // Open serial communications and wait for port to open:
        char cmd[254];
        Serial.begin(9600);
        Serial.setTimeout(5000);
        //test if the module is ready
        Serial.println("AT+RST");
        delay(1000);
        if (Serial.find("ready"))
        {
                Serial.println("Module is ready");
        }
```

```
        else
        {
                Serial.println("Module have no response.");
                while (1);
        }

        delay (1000);
        //connect to the wifi
        boolean connected = false;
        for (int i = 0; i < 5; i++)
        {
                if (connectWiFi())
                {
                        connected = true;
                        break;
                }
        }
        if (!connected) {
                while (1);
        }
        delay(5000);
        Serial.println("AT+CIPMUX=0");
}

void loop()
{
        int value = analogRead(A0); //you can change ir to another pin
        int num=0;
        String var = "{\"value\":"+ String(value) + "}";
        num = var.length();
        String cmd = "AT+CIPSTART=\"TCP\",\"";
        cmd += DST_IP;
        cmd += "\",80";
        Serial.println(cmd);
        if (Serial.find("Error"))
                return;
        len=strlen ("POST /api/v1.6/datasources/");
        len=len+strlen (idvariable);
        len=len+strlen ("/values HTTP/1.1\nContent-Type: application/json\nContent-Length:
        ");
        char numlength[4]; // this will hold the length of num which is the length of the JSON
        element
        sprintf(numlength, "%d", num); // saw this clever code off the net; works yay
        len=len+strlen (numlength);
        len=len + num; //fixed length of the string that will print as Content-Length: in the
        POST
        len=len+strlen ("\nX-Auth-Token: ");
```

```
        len=len+strlen (token);
        len=len+strlen ("\nHost: things.ubidots.com\n\n");
        len=len+strlen ("\n\n");
        Serial.print("AT+CIPSEND=");
        Serial.println (len); //length of the entire data POST for the CIPSEND command of
        ESP2866
        //Serial.println(cmd.length());
        if (Serial.find(">"))
        {
                //Serial.print(">");
        }
        else
        {
                Serial.println("AT+CIPCLOSE");
                delay(1000);
                return;
        }

        Serial.print ("POST /api/v1.6/variables/");
        Serial.print (idvariable);
        Serial.print ("/values HTTP/1.1\nContent-Type: application/json\nContent-Length: ");
        Serial.print (num);
        Serial.print ("\nX-Auth-Token: ");
        Serial.print (token);
        Serial.print ("\nHost: things.ubidots.com\n\n");
        Serial.print (var);
        Serial.println ("\n\n");
        delay(9000);
        //Serial.find("+IPD"); clear the input buffer after the web site responds to the POST
        while (Serial.available())
        {
                char c = Serial.read();
        }
        delay(1000);
}
boolean connectWiFi()
{
        Serial.println("AT+CWMODE=1");
        String cmd = "AT+CWJAP=\"";
        cmd += SSID;
        cmd  += "\",\"";
        cmd +=  PASS;
        cmd += "\"";
        Serial.println(cmd);
        delay(2000);
        if (Serial.find("OK"))
        {
```

```
                return true;
        }
        else
        {
                return false;
        }
}
```

Output:



Result:

Thus, the sketch program to connect a ESP8266 Wifi Module with MSP430G2553 to send a data to browser was implemented successfully.

EX.NO: 9	MULTIPLE LED BLINKING USING TI CC3200 LAUNCHPAD
DATE:	

Aim:

To write a CC3200 sketch for blinking (ON/OFF) of inbuilt LED using CC3200.

Apparatus Required:

- Energia IDE
- CC3200 board
- LED
- Bread board
- 220 ohm resistor
- Jumper wires

Procedure:

- Attach the CC3200 board with the system.
- Interface LED circuit with CC3200 board.
- Double click on Energia on the desktop.
- Select the board type as CC3200 from Tools-Board and also select COM port number from the PORT option
- Create a new program in the Energia IDE software and save it.
- Compile the program and upload it to the CC3200 board.
- Run the program and verify the output.

Code:

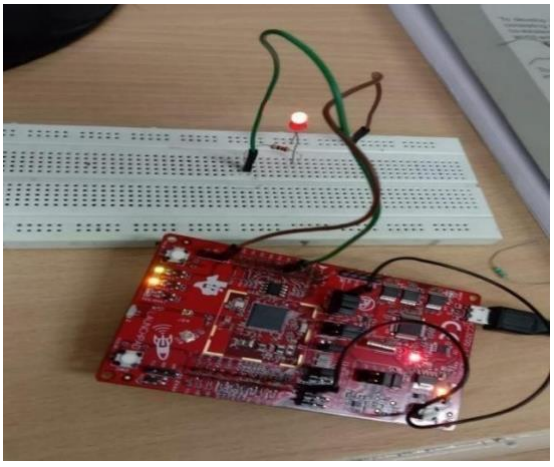
For Single LED Bulb:

```
#define LED 5
void setup()
{
        pinMode(LED,OUTPUT);
}
void loop()
{
        digitalWrite(LED, HIGH);
        delay(1000):
        digitalWrite(LED,LOW);
        delay(1000);
}
```

For Multiple LED Bulb:

```
#define RLED 9
#define GLED 10
#define YLED 29
void setup()
{
        pinMode(RLED,OUTPUT);
        pinMode(GLED,OUTPUT);
        pinMode(YLED,OUTPUT);
}
void loop()
{
        digitalWrite(RLED, HIGH);
        digitalWrite(GLED, HIGH);
        digitalWrite(YLED, HIGH);
        delay(1000):
        digitalWrite(RLED,LOW);
        digitalWrite(GLED,LOW);
        digitalWrite(YLED,LOW);
        delay(1000);
}
```

Output:



Result:

Thus, the Energia sketch to ON/OFF of built-in LEDs was executed successfully.

EX.NO: 10	INTERFACING PUSH BUTTON USING TI CC3200 LAUNCHPAD
DATE:	

Aim:

To write a CC3200 sketch to turn on and off a light emitting diode (LED) connected to a digital Pin when pressing a push button attached to a digital pin.

Apparatus Required:

- Energia
- CC3200 Board
- Push Button
- 10K ohm resistor
- Breadboard

Procedure:

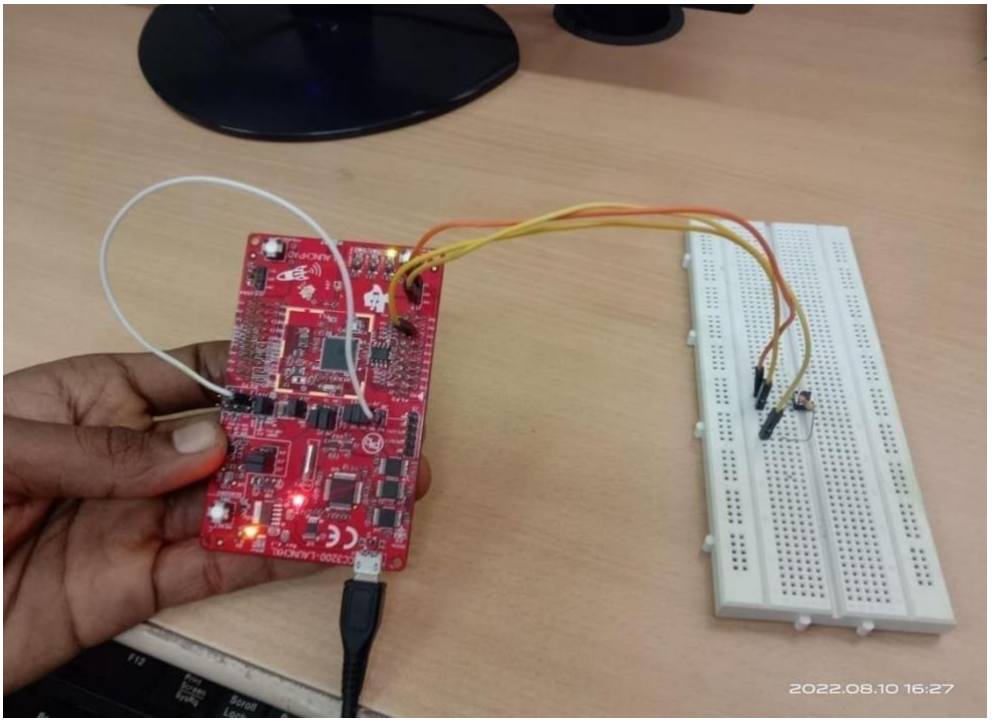
- Attach the CC3200 board to the system.
- Connect Push Button to digital pin 8 and LED with the digital pin 2 of the CC3200 board.
- Double-click on Energia on the desktop.
- Select the board type as CC3200 from Tools-Board and also select the COM port number from the PORT option.
- Create a new program in the Energia software and save it.
- Compile the program and upload it to the CC3200 board.
- Run the program and verify the output.

Code:

```
const int buttonPin = 8;
const int ledPin = 2;
int buttonState = 0;

void setup()
{
        pinMode(ledPin,OUTPUT);
        pinMode(buttonPin, INPUT);
        Serial.begin(9600);
}
```

```
void loop()
{
        buttonState = digitalRead(buttonPin);
        if(buttonState == HIGH)
        {
                digitalWrite(ledPin,HIGH);
                Serial.println("LEDglows");
        }
        else
        {
                digitalWrite(ledPin, LOW);
        }
}
```

Output:



Result:

Thus, the Energia sketch to interface pushbutton and LED with CC3200 was executed and the output was verified successfully.

EX.NO: 11	DESIGN OF IOT APPLICATION TO SENSE NEARBY OBJECTS USING PIR SENSOR WITH TI CC3200 LAUNCHPAD
DATE:	

Aim:

To write a program in Energia to check whether any live object traces are present by using PIR Sensor using CC3200.

Apparatus Required:

- Energia IDE
- CC3200 Board
- LED
- PIR Sensor
- Breadboard

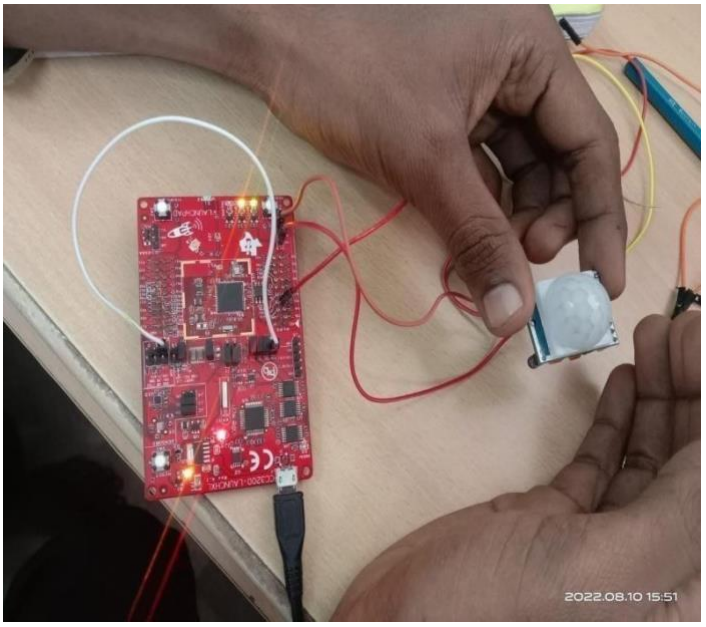
Procedure:

- Attach the CC3200 board to the system.
- Connect the PIR sensor with the digital pin of the CC3200 board.
- Double-click on Energia on the desktop.
- Select the board type as CC3200 from Tools-Board and also select the COM port number from the PORT option.
- Create a new program in the Energia software and save it.
- Compile the program and upload it to the CC3200 board.
- Run the program and verify the output.

Program:

```
int pir=4;
int val = LOW;
void setup()
{
        pinMode(pir, INPUT);
        Serial.begin(9600);
}
void loop() {
        val = digitalRead(pir);
        if (val = = HIGH)
        {
                Serial.println("Motion Detected");
        }
        else
        {
                Serial.println("Motion NOT Detected");
        }
}
```

Output:



Result:

Thus, the Energia sketch to interface the PIR sensor with CC3200 was executed and the output was verified successfully.

EX.NO: 12	DESIGN OF IOT APPLICATIONS WITH SENSORS TO SCAN NETWORKS USING TI CC3200 LAUNCHPAD
DATE:	

Aim:

To write a Energia sketch program to scan for available Wifi networks and to print its Wifi MAC address using CC3200.

Apparatus Required:

- CC3200 Wifi LaunchPad

Procedure:

- Attach the CC3200 board with the system.
- Double click on Energia IDE on the desktop.
- Select the board type as CC3200 Launchpad from Tool.
- Go to files select examples in that choose Wifi
- Create a new program on Energia IDE and save it.
- Compile the program and upload it to the CC3200 Launchpad board.
- Run the program and verify the output turning on the nearby hotspots.

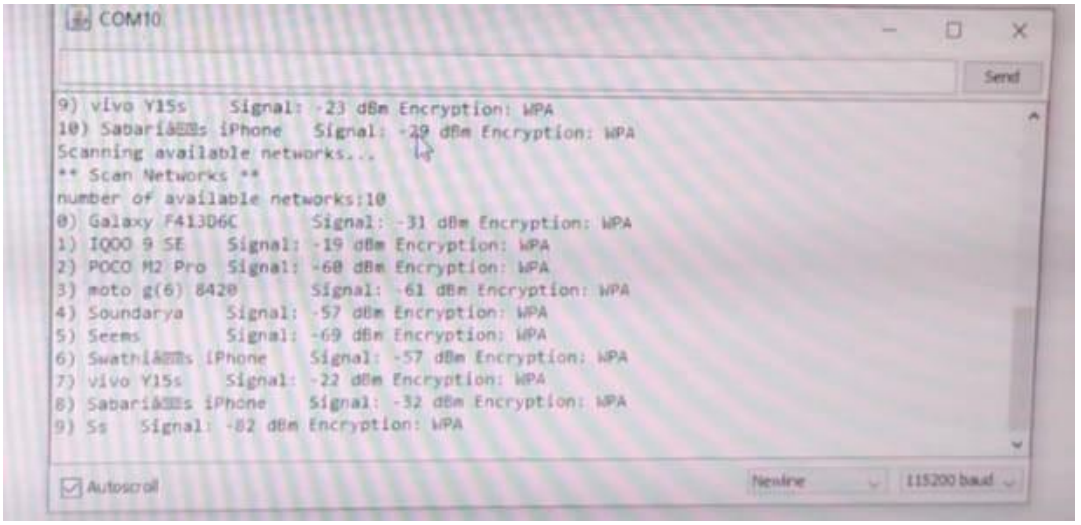
Code:

```
#ifndef CC3200R1M1RGC
// Do not include SPI for CC3200LaunchPad
#include <SPI.h>
#endif
#include <WiFi.h>
void setup() {
        //Initialize serial and wait for the port to open:
        Serial.begin(115200);
        WiFi.init();
        Serial.println(WiFi.firmwareVersion());
        // Print WiFi MAC address:
        printMacAddress();
        // scan for existing networks:
        Serial.println("Scanning available networks...");
        listNetworks();
}
void loop()
{
        delay (10000);
        // scan for existing networks:
        Serial.println("Scanning available networks...");
        listNetworks();
```

```
}
void printMacAddress()
{
	// the MAC address of your Wifi
	byte mac[6];
	// print your MAC address:
	WiFi.macAddress(mac);
	Serial.print("MAC: ");
	Serial.print(mac[5], HEX);
	Serial.print(":");
	Serial.print(mac[4], HEX);
	Serial.print(":");
	Serial.print(mac[3], HEX);
	Serial.print(":");
	Serial.print(mac[2], HEX);
	Serial.print(":");
	Serial.print(mac[1], HEX);
	Serial.print(":");
	Serial.println(mac[0], HEX);
}
void listNetworks()
{
	// scan for nearby networks:
	Serial.println("** Scan Networks**");
	int numSsid = WiFi.scanNetworks();
	if (numSsid == -1)
	{
		Serial.println("Couldn't get a wificonnection");
		while (true);
	}
	// print the list of networks seen:
	Serial.print("number of available networks:");
	Serial.println(numSsid);
	// print the network number and name for each network found:
	for (int thisNet = 0; thisNet < numSsid; thisNet++)
	{
		Serial.print(thisNet);
		Serial.print(") ");
		Serial.print(WiFi.SSID(thisNet));
		Serial.print("\tSignal: ");
		Serial.print(WiFi.RSSI(thisNet));
		Serial.print(" dBm");
		Serial.print("\tEncryption: ");
		printEncryptionType(WiFi.encryptionType(thisNet));
	}
}
```

```
void printEncryptionType(int thisType) {
        // read the encryption type and print out the name:
        switch (thisType)
        {
                case ENC_TYPE_WEP:
                        Serial.println("WEP");
                        break;
                case ENC_TYPE_TKIP:
                        Serial.println("WPA");
                        break;
                case ENC_TYPE_CCMP:
                        Serial.println("WPA2");
                        break;
                case ENC_TYPE_NONE:
                        Serial.println("None");
                        break;
                case ENC_TYPE_AUTO:
                        Serial.println("Auto");
                        break;
        }
}
```

Output:



Result:

Thus, the study on scanned networks was executed using CC3200 and implemented successfully.

DATE:	CONTENT BEYOND SYLLABUS DEMONSTRATION OF MISRA C AND CERT C CODING STANDARDS

Aim:

To analyse and adopt various MISRA standards by comparing it with C program.

Description:

- MISRA C is a set of software development guidelines for the C programming language developed by the MISRA Consortium.
- Its aims are to facilitate code safety, security, portability and reliability in the context of embedded systems, specifically those systems programmed in ISO C / C90 / C99.
- The CERT C and CERT C++ coding standards are secure coding practices for the C and C++ languages.
- Security vulnerabilities in embedded software increase chances of attacks from malicious actors.
- These attacks inject malware, steal information, or perform other unauthorized tasks. Secure coding practices plug these vulnerabilities and effectively reduce the surface of attack.

Code:

C program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
   int num,rev=0;
   printf("Enter a number:");
   scanf("%d",&num);
   while(num>0)
   {
      rev = rev*10 + num%10;
      num/=10;
   }
   printf("%d",rev);
}
```

MISRA C Program:

```
#include "stdio.h"
void main()
{
   unsigned int num;
   unsigned int rev=0;
   unsigned int const TEN=10;
   scanf("%d",&num);
   if(num>0 && num/10>0)
   {
      printf("Reverse of the number is:");
```

```
    while(num>0)
    {
       rev = (rev*TEN) + (num%TEN);
       num = num/TEN;
    }
    printf("%d",rev);
  }
  else
  {
    printf("Number of digits should be 2 and greater:\n");
  }
}
```

MISRA C Standards Followed in the Code:

Rule 1.1(required): All code shall confirm to ISO/IEC 9898:1990

The below rules have been adopted in the coding practice.

Line #1: Rule 19.1(A): #include statement in a file should only be preceded by other preprocessor or directories or comments.

#include<stdio.h>

Line#3: Rule 16.1(R): Function shall not be defined with variable number of arguments. Rule 16.2(R): Function shall not call themselves either directly or indirectly.

int main()

Line#4,5: Rule 6.2(R): Unsigned character type shall be used only for numeric value.

unsigned int num;

unsigned int rev=0;

Line#6: Rule 1.8b(R): The const keyword shall be used whenever appropriate.

unsigned int const TEN=10;

Line#9: Rule 4.1(R): Only escape sequences that are defined in ISO standard are permitted. All hexadecimal escape sequences are not permitted.

printf("\nReverse of the number is:");

C program:

```
#include<stdio.h>
int main()
{
  int n, a, b, nt, i;
  printf("Enter a number:");
  scanf("%d",&n);
  a=0;
  b=1;
  if(n>0 && n<=2)
  {
    printf("%d %d",a,b);
    return 0;
  }
  else
```

```
    {
        printf("%d %d",a,b);
        for(i=2;i<n;i++)
        {
            nt = a+b;
            a = b;
            b = nt;
            printf("%d",nt);
        }
    }
    printf("\n");
    return 0;
}
```

CERT C Code:

```
#include<stdio.h>
#include<stddef.h>
#define SIZE 7
unsigned int main()
{
    unsigned int n, a=0, b=1, nt, i;
    if(SIZE > 0 && SIZE <= 2)
    {
        printf("%d %d ",a,b);
        return 0;
    }
    else
    {
        printf("The series is\n");
        printf("%d %d ",a,b);
        for(i=2;i<SIZE;i++)
        {
            nt = a + b;
            a = b;
            b = nt;
            printf("%d ",nt);
        }
    }
}
```

CERT C Standards:

Line #1: PRE04-A. Do not reuse a standard header file name. If a file with the same name as a standard file name is placed in the search path for included source files, the behaviour is undefined. PRE31-C. Guarantee header file names are unique Guarantee header file names are unique, all included files should differ (in a case insensitive manner) in their firsteight characters or in their (one character) file extension.

#include<stdio.h>

Line #4: DCL02-A. Use visually distinct identifiers

Use visually distinct identifiers to eliminate errors resulting from misrecognizing the spelling of an identifier during the development and review of code.

DCL04-A. Take care when declaring more than one variable per declaration

Declaring multiple variables in a single declaration can cause confusion regarding the types of the variables and their initial values. If more than one variable is declared in a declaration, care must be taken that the actual type and initialized value of the variable is known.

int n, a, b, nt, i;

Line #7: DCL00-A. Declare immutable values using const or enum

Immutable (constant values) should be declared as const-qualified objects (unmodifiable Ivalues), enumerations values, or as a last resort, a #define.

DCL06-A. Use meaningful symbolic constants to represent literal values in program logic Avoid the use of magic numbers in code when possible. Magic numbers are constant values that represent an arbitrary value, such as a determined appropriate buffer size.

a = 0;

b = 1;

Line #17: INT01-A. Use size_t for all integer values representing the size of an object. The size_t type is the unsigned integer type of the result of the sizeof operator. The underlying representation of variables of type size_t is guaranteed to be of sufficient precision to represent the size of an object.

for (i=2; i<n;i++)

Result:

Thus, the MISRA and CERT C coding standard is understood and used in practice of embedded programming development.