EXPT.NO:01	**8051 ASSEMBLY LANGUAGE EXPERIMENTS USING SIMULATOR**
DATE:	

AIM:
To Write and simulate basic arithmetic program using ASM in 8051 microcontroller.

HARDWARE/SOFTWARE REQUIREMENTS:
1) Keil µ vision version 3

PROCEDURE:
1) Open project and go to new Micro Vision Project.
2) Give a name to project and save it.
3) Select a device for project "target 1".
4) Select the device name and category as 89C51 under Atmel Category and click on.
5) The project and its folders can be seen on the left side in the project window.
6) Now go to file and add new file from the menu.
7) Save the file with the specific name with .asm extension.
8) Add this .asm extension file to source group folder in the project window by right clicking on source group folder and selecting existing files to source group 1.
9) Build the project from the build target option in the project tab.
10) Once the project is build with no errors click debug to start debug session.
11) Click on debug mode and then run F5.
12) Give the output in the registered windows and also from peripherals that is I/O ports.

PROGRAM:

1) ADDITIONAL OF TWO 8 BIT NUMBERS

```
MOV A , #05H

MOV R0 , #04H

ADD A , R0
```

2) **SUBTRACTION OF TWO 8 BIT NUMBERS**

```
MOV A , #09H

MOV R0 , #06H

SUBB A , R0
```

3) MULTIPLICATION OF TWO 8 BIT NUMBERS

```
MOV A , #02H

MOV R0 , #04H

MUL A  R0
```

4) DIVISION OF TWO 8 BIT NUMBERS

```
MOV A , #0AH

MOV R0 , #02H

DIV A, R0
```

5) AND OPERATION

```
MOV A , #0FH
MOV R0 , #0FH
ANL A , R0
```

6) OR OPERATION

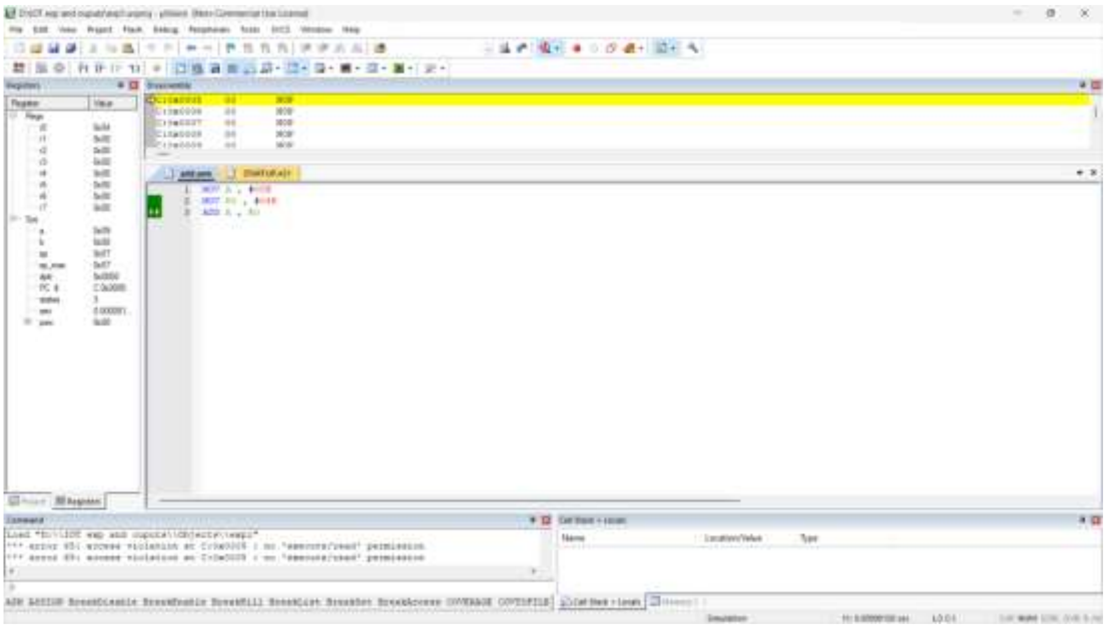
```
MOV A , #00H
MOV R0 , #0FH
ORL A , R0
```

7) **XOR OPERATION**

```
MOV A , #0FH
MOV R0 , #00H
XRL A , R0
```

OUTPUT:

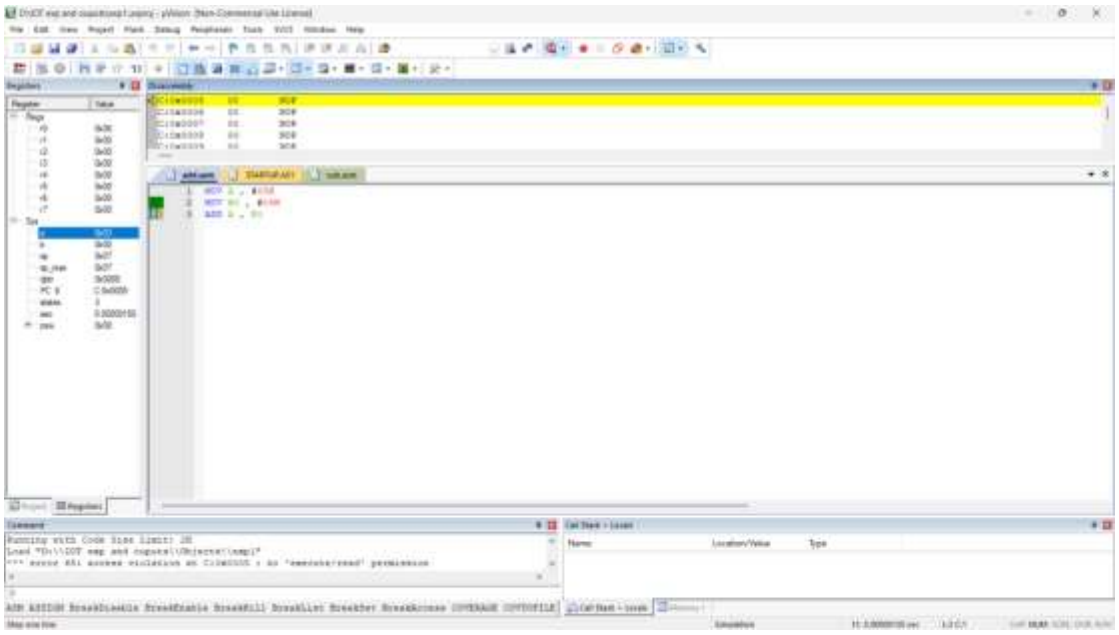
1) **ADDITIONAL OF TWO 8 BIT NUMBERS**



```
MOV A , #[illegible]
MOV R0 , #[illegible]
ADD A , R0
```

```
Load "[illegible]"
*** error 65: access violation at C:0x0005 : no 'execute/read' permission
*** error 65: access violation at C:0x0005 : no 'execute/read' permission
```

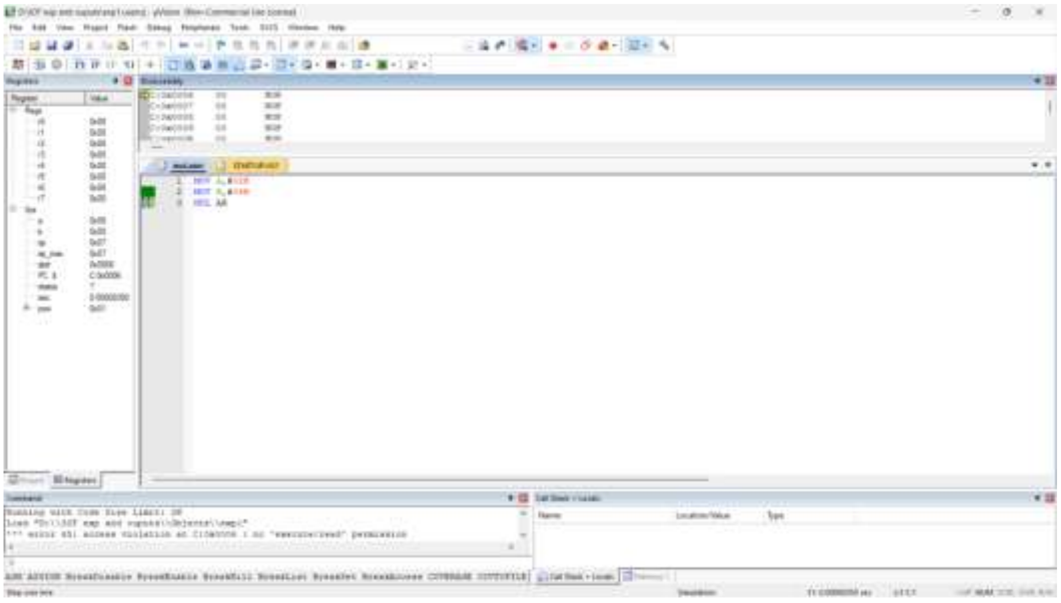
2) **SUBTRACTION OF TWO 8 BIT NUMBERS**



```
MOV A , #[illegible]
MOV R0 , #[illegible]
ADD A , R0
```

```
Running with Code Size Limit: 2K
Load "[illegible]"
*** error 65: access violation at C:0x0005 : no 'execute/read' permission
```

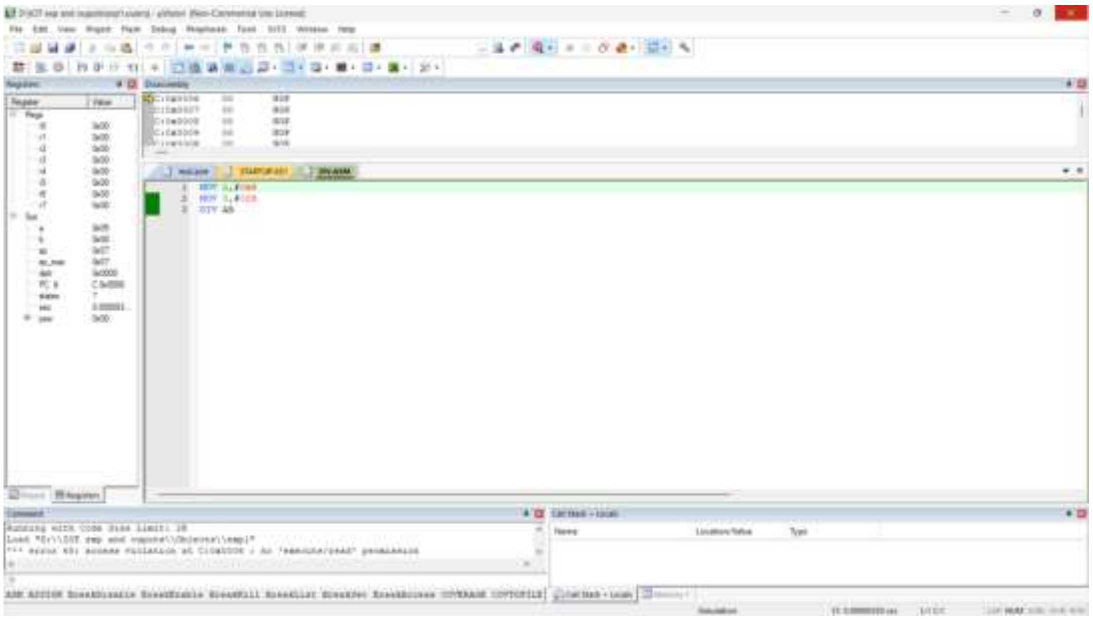
3) MULTIPLICATION OF TWO 8 BIT NUMBERS



```
1  MOV A,#[illegible]
2  MOV B,#[illegible]
3  MUL AB
```

```
Running with Code Size Limit: 2K
Load "D:\\IOT exp and outputs\\Objects\\exp1"
*** error 65: access violation at C:0x0006 : no 'execute/read' permission
```

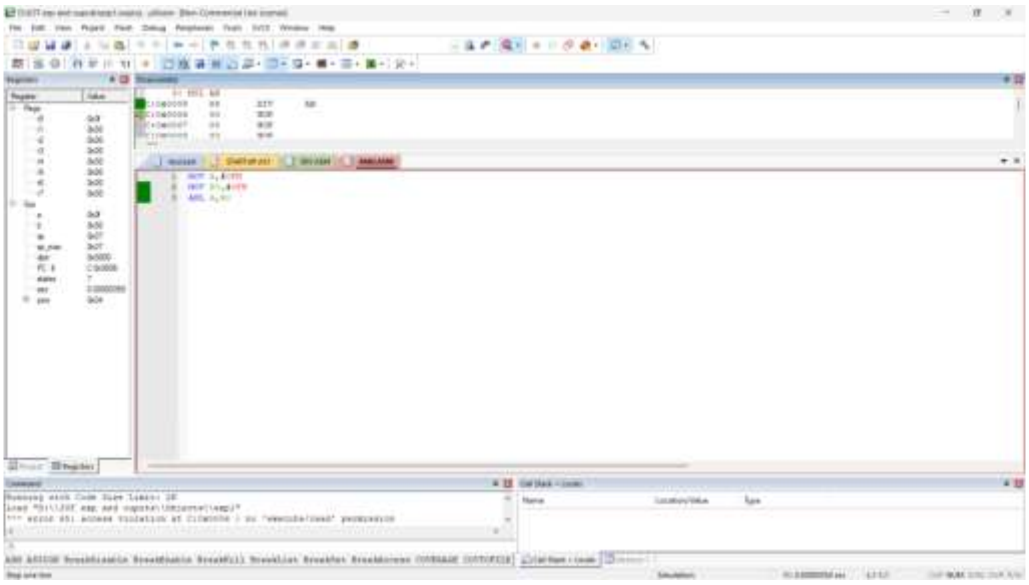
4) DIVISION OF TWO 8 BIT NUMBERS



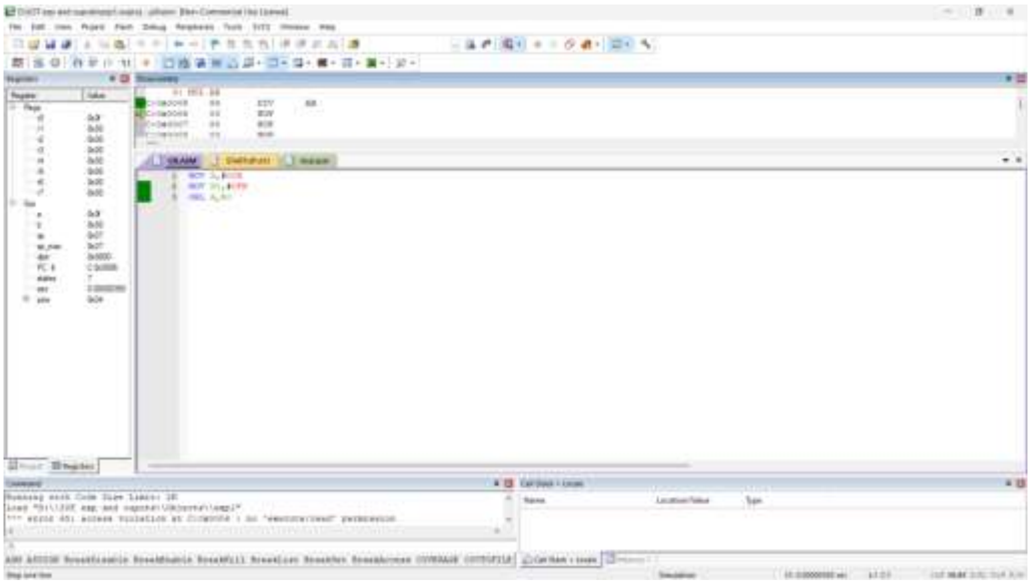
```
1  MOV A,#[illegible]
2  MOV B,#[illegible]
3  DIV AB
```

```
Running with Code Size Limit: 2K
Load "D:\\IOT exp and outputs\\Objects\\exp1"
*** error 65: access violation at C:0x0006 : no 'execute/read' permission
```

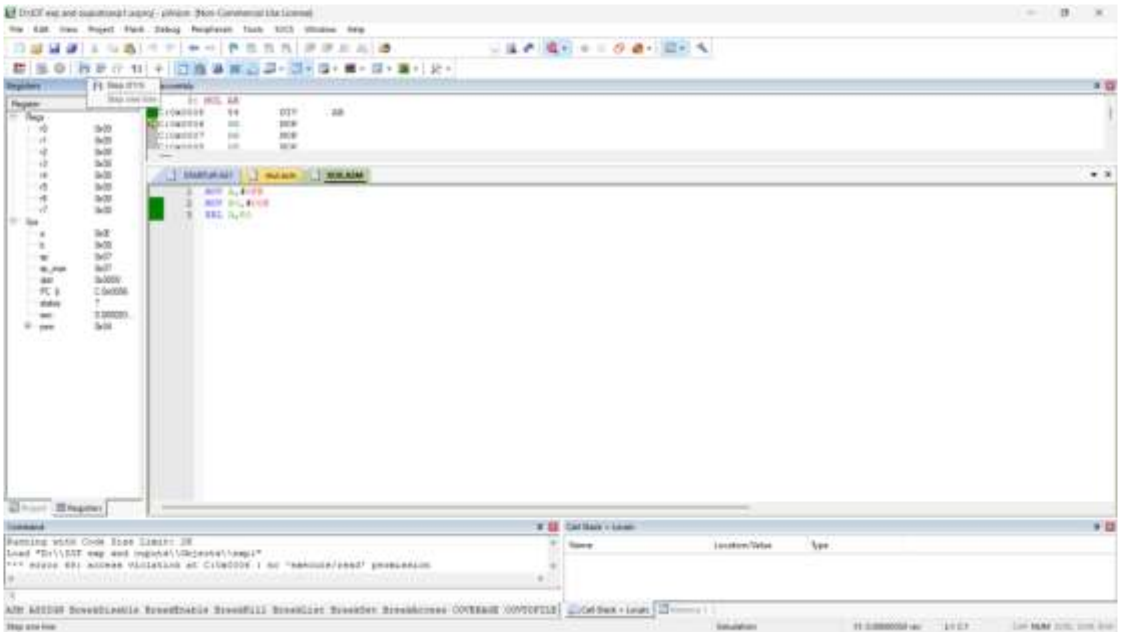
5) AND OPERATION



6) OR OPERATION



7) **XOR OPERATION**



```
MOV A,#[illegible]
MOV R1,#[illegible]
XRL A,R1
```

```
Running with Code Size Limit: 2K
Load "D:\\[illegible]\\Objects\\[illegible]"
*** error 65: access violation at C:0x0006 : no 'execute/read' permission
```

RESULT: Thus the basic arithmetic program using ASM in 8051 microcontroller has been executed.

EXPT.NO:02	BASIC ARITHMETIC PROGRAM USING EMBEDDED C
DATE:	

AIM:
To Write and simulate basic arithmetic program using embedded C in 8051 microcontroller.

HARDWARE/SOFTWARE REQUIREMENTS:

1) Keil µ vision version 3

PROCEDURE:

1) Open project and go to new Micro Vision Project.
2) Give a name to project and save it.
3) Select a device for project “target 1”.
4) Select the device name and category as 89C51 under Atmel Category and click on.
5) The project and its folders can be seen on the left side in the project.
6) Now go to file and add new file from the menu.
7) Type the embedded C program to be simulated.
8) Save the file with a specific name with .c extension.
9) Add this .c extension file to source group folder in the project window by right clicking on source group 1 folder and selecting the existing files to source group 1.
10) Built the project from the build target option in the project tab. Once the project is built with no errors click debug to start debug session.
11) Click on debug mode and then run F5.
12) Give the output in the registered windows and also from peripherals that is I/O ports.

PROGRAM:

a) DISPLAY THE GIVEN VALUE TO ALL THE PORT

```
#include <reg51.h>
void main()
 {
    P1 = 0 x 01;
    P2 = 0 x 02;
    P3 = 0 x 03;
 }
```

b) SIMPLE ADDITION PROGRAM

```
#include <reg51.h>
void main()
{
   int a, b, c;
   a = 0 x 01;
   b = 0 x 02;
   c = a + b;
   P1 = c;
 }
```

c) ADDITION WITH CARRY

```
#include <reg51.h>
void main()
 {
  int a, b, c;
   a = 0 x FF;
   b = 0 x 90;
  c = a + b;
  P1 = c;
  P2 = c >> 8;
   }
```

d) SIMPLE SUBTRACTION PROGRAM

```
#include <reg51.h>
void main()
{
int a, b, c;
a = 0 x 02;
b = 0 x 01;
c = a - b;
P1 = c;
}
```

e) SIMPLE MULTIPLICATION PROGRAM

```
#include <reg51.h>
void main()
{
int a, b, c;
a = 0 x 06;
b = 0 x 03;
c = a * b;
P1 = c;
}
```

f) SIMPLE DIVISION PROGRAM

```
#include <reg51.h>
void main()
{
int a, b, c;
a = 0 x 06;
b = 0 x 03;
c = a / b;
P1 = c;
}
```

g) SIMPLE MODULO DIVISION PROGRAM

```
#include <reg51.h>
void main()
{
int a, b, c;
a = 0 x 07;
b = 0 x 02;
c = a % b;
P1 = c;
}
```

h) SIMPLE PROGRAM FOR PRE INCREMENT

```
#include <reg51.h>
void main()
{
int a, c;
a = 0 x 07;
c = ++a;
P1 = c;
}
```

i) SIMPLE PROGRAM FOR POST INCREMENT

```
#include <reg51.h>
void main()
{
int a, c;
a = 0 x 07;
c = a++;
P1 = c;
}
```

j) SIMPLE PROGRAM FOR PRE DECREMENT

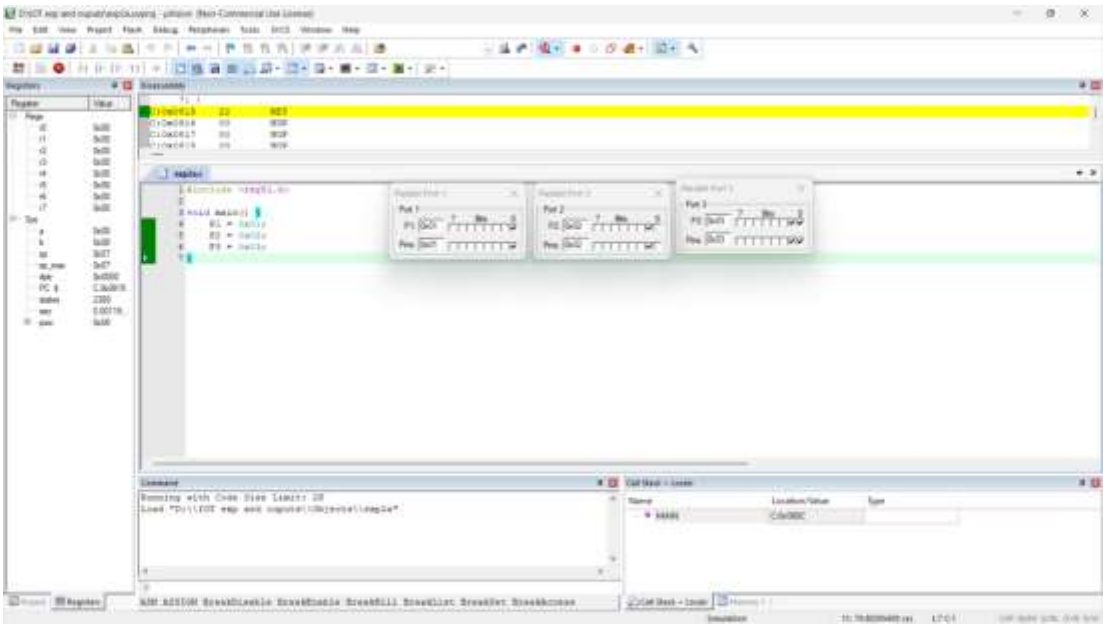
```
#include <reg51.h>
void main()
{
int a, c;
a = 0 x 07;
c = --a;
P1 = c;
}
```

k) SIMPLE PROGRAM FOR POST DECREMENT

```
#include <reg51.h>
void main()
{
int a, c;
a = 0 x 07;
c = a--;
P1 = c;
}
```

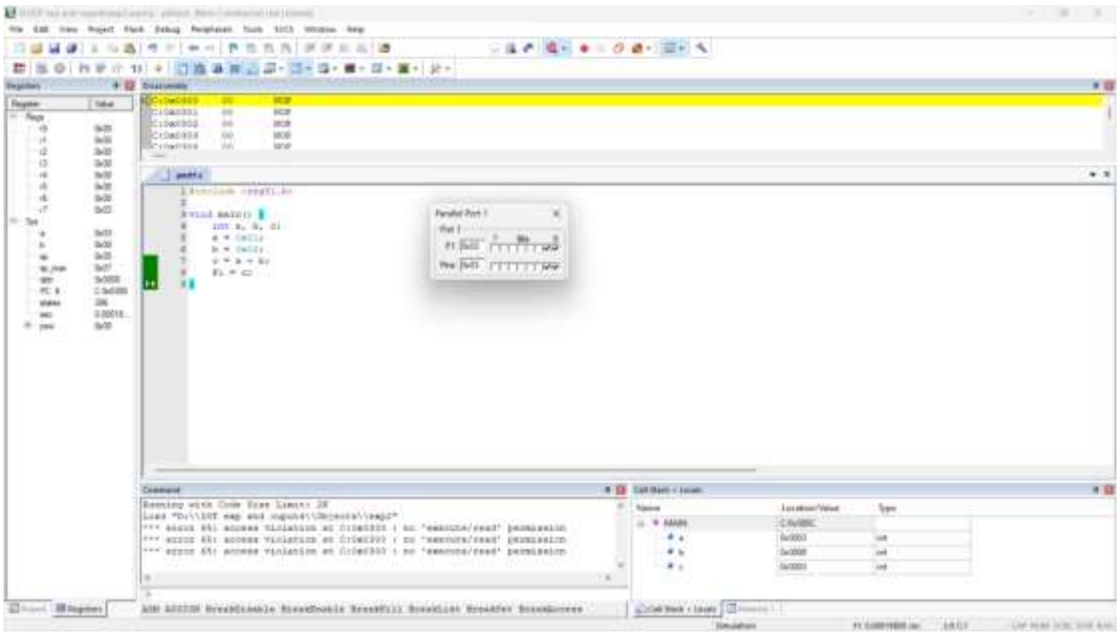
OUTPUT:

a) DISPLAY THE GIVEN VALUE TO ALL THE PORT



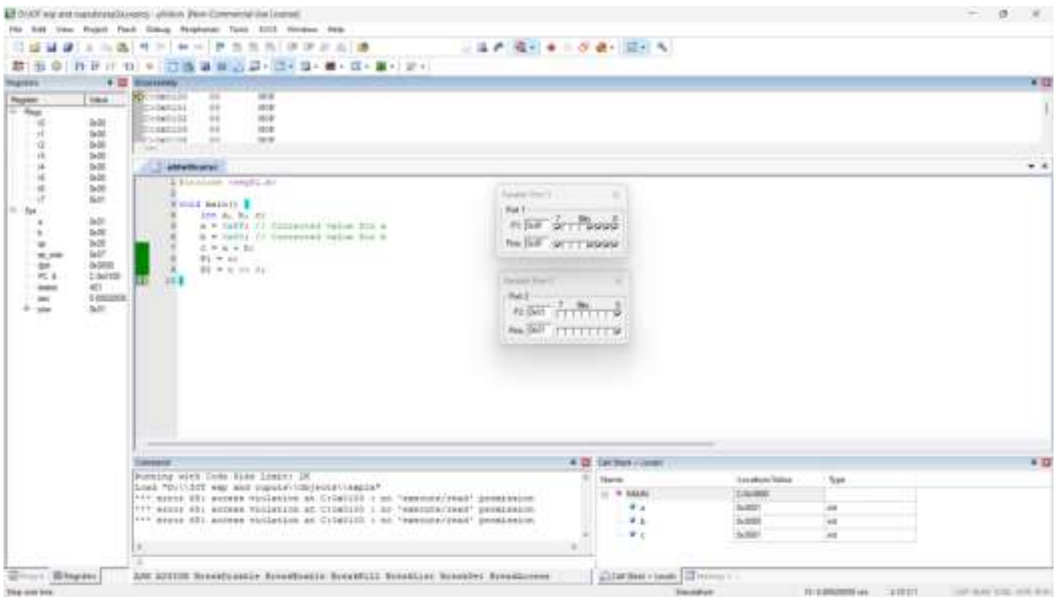
```
#include <reg51.h>

void main() {
    P1 = [illegible];
    P2 = [illegible];
    P3 = [illegible];
}
```

b) SIMPLE ADDITION PROGRAM



```
#include <reg51.h>

void main() {
    int a, b, c;
    a = [illegible];
    b = [illegible];
    c = a + b;
    P1 = c;
}
```

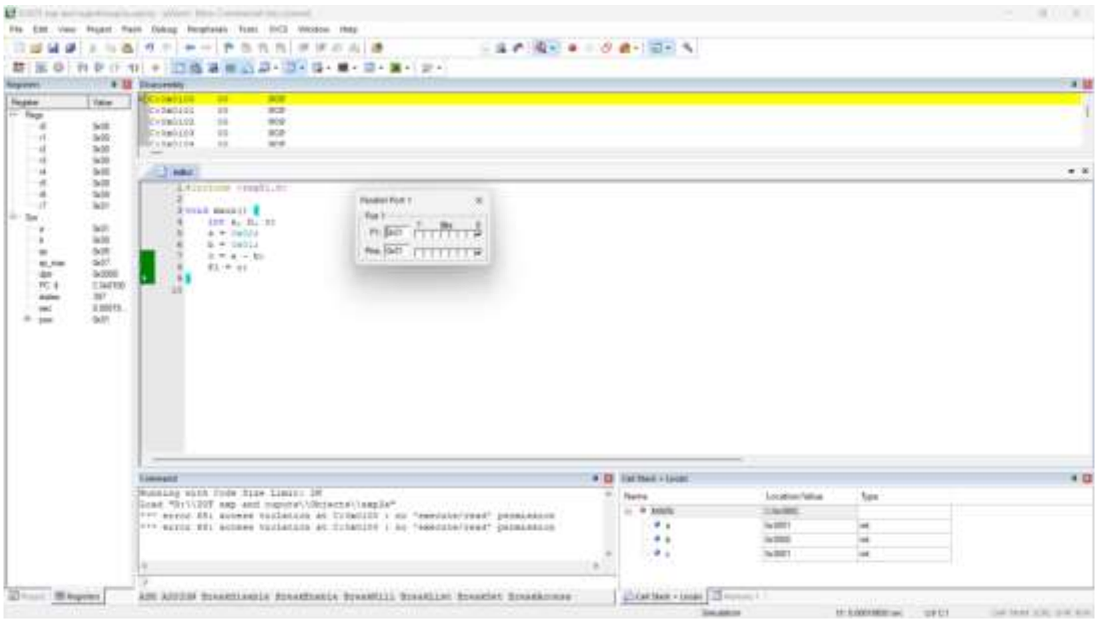
c) ADDITION WITH CARRY



```
#include <reg51.h>

void main() {
    int a, b, c;
    a = 0xFF; // Corrected value for a
    b = 0x01; // Corrected value for b
    c = a + b;
    P1 = c;
    P2 = c >> 8;
}
```

```
Running with Code Size Limit: 2K
Load "D:\\IOT map and inputs\\Objects\\sample"
*** error 65: access violation at C:0x0100 : no 'execute/read' permission
*** error 65: access violation at C:0x0100 : no 'execute/read' permission
*** error 65: access violation at C:0x0100 : no 'execute/read' permission
```

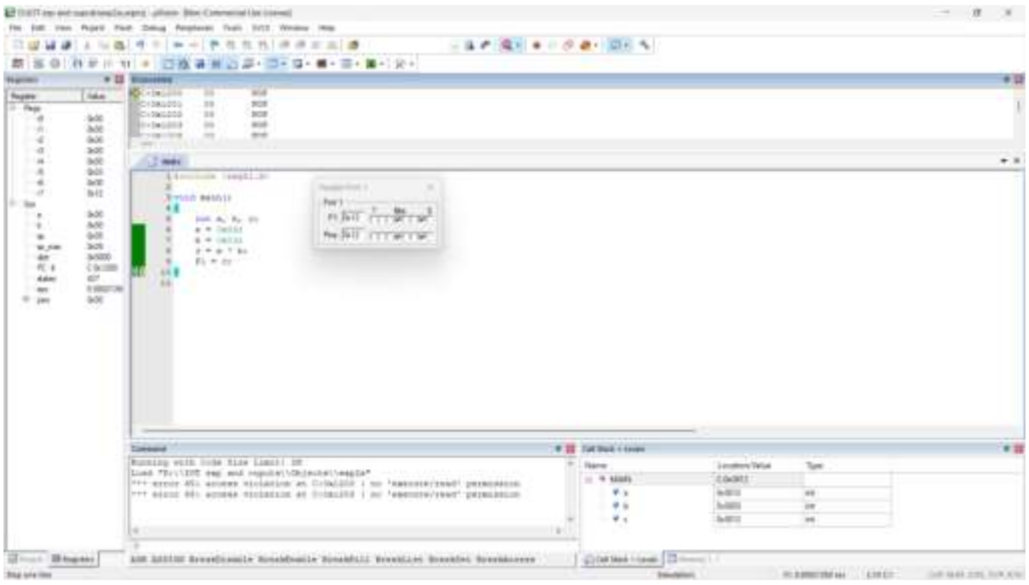
d) SIMPLE SUBTRACTION PROGRAM



```
#include <reg51.h>

void main() {
    int a, b, c;
    a = [illegible];
    b = [illegible];
    c = a - b;
    P1 = c;
}
```

```
Running with Code Size Limit: 2K
Load "D:\\IOT map and inputs\\Objects\\sample"
*** error 65: access violation at C:0x0100 : no 'execute/read' permission
*** error 65: access violation at C:0x0104 : no 'execute/read' permission
```

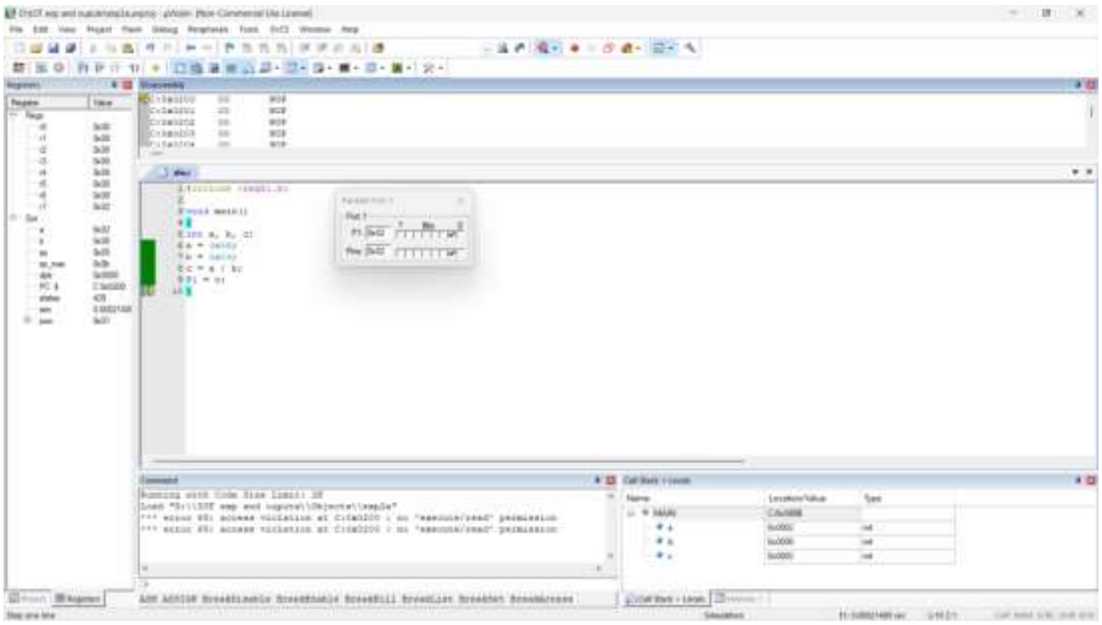
e) SIMPLE MULTIPLICATION PROGRAM



```
#include <reg51.h>

void main()
{
    int a, b, c;
    a = [illegible];
    b = [illegible];
    c = a * b;
    P1 = c;
}
```

```
Running with Code Size Limit: 2K
Load "D:\\IOT exp and [illegible]\\Objects\\[illegible]"
*** error 65: access violation at C:0x[illegible] : no 'execute/read' permission
*** error 65: access violation at C:0x[illegible] : no 'execute/read' permission
```

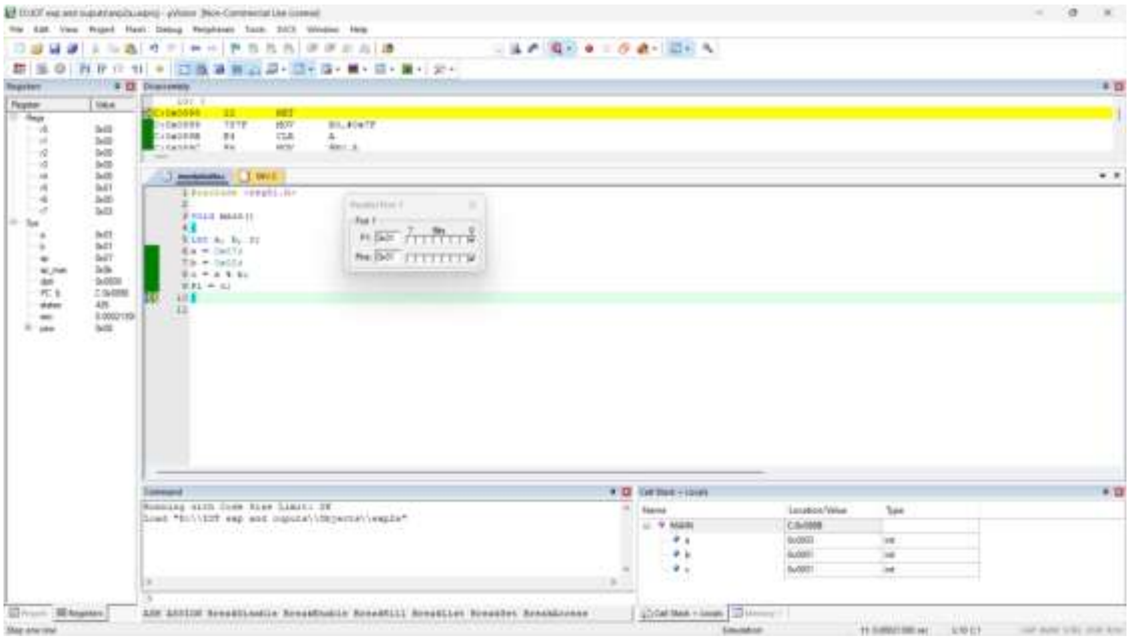
f) SIMPLE DIVISION PROGRAM



```
#include <reg51.h>

void main()
{
int a, b, c;
a = [illegible];
b = [illegible];
c = a / b;
P1 = c;
}
```

```
Running with Code Size Limit: 2K
Load "D:\\IOT exp and [illegible]\\Objects\\[illegible]"
*** error 65: access violation at C:0x[illegible] : no 'execute/read' permission
*** error 65: access violation at C:0x[illegible] : no 'execute/read' permission
```

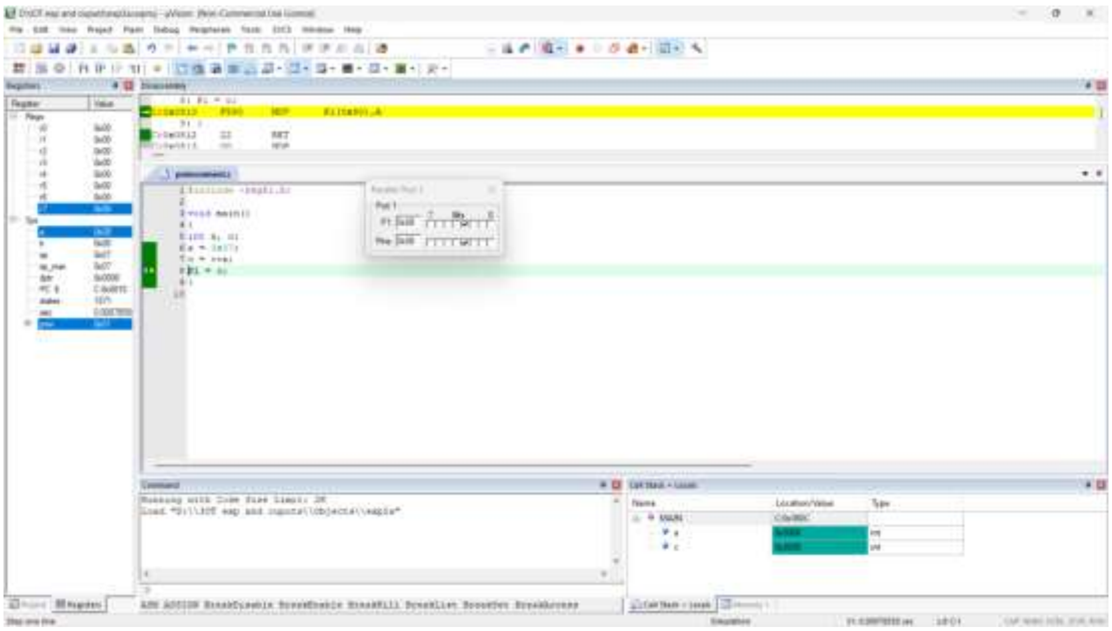
g) SIMPLE MODULO DIVISION PROGRAM



```
1 #include <reg51.h>
2
3 void main()
4 {
5 int a, b, c;
6 a = [illegible];
7 b = [illegible];
8 c = a % b;
9 P1 = c;
10 }
11
```

```
Running with Code Size Limit: 2K
Load "D:\\IOT exp and ouputs\\Objects\\exple"
```

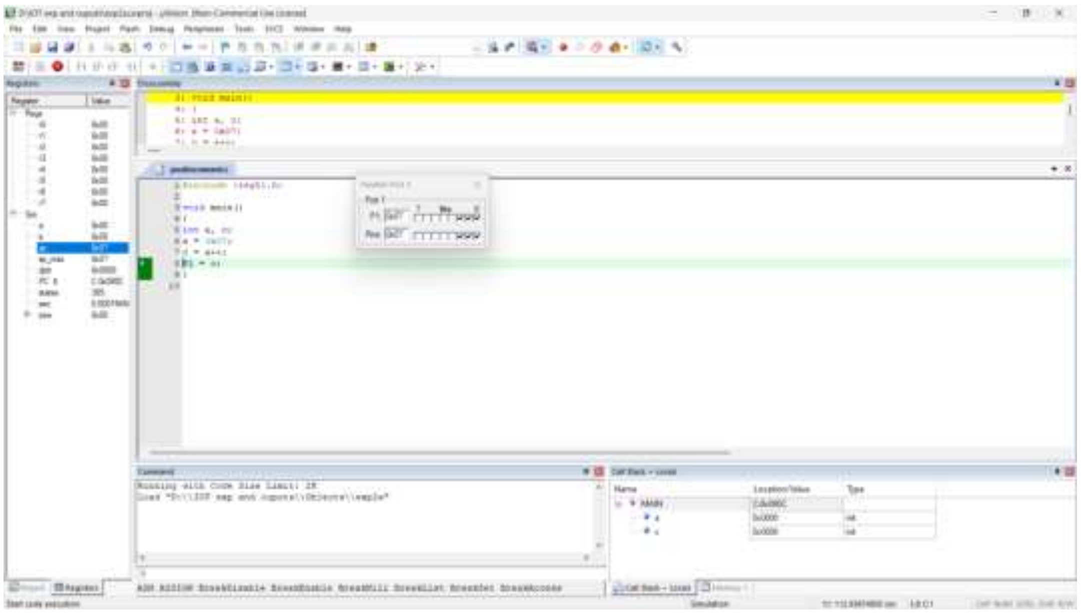
h) SIMPLE PROGRAM FOR PRE INCREMENT



```
1 #include <reg51.h>
2
3 void main()
4 {
5 int a, c;
6 a = [illegible];
7 c = ++a;
8 P1 = c;
9 }
10
```

```
Running with Code Size Limit: 2K
Load "D:\\IOT exp and ouputs\\Objects\\exple"
```

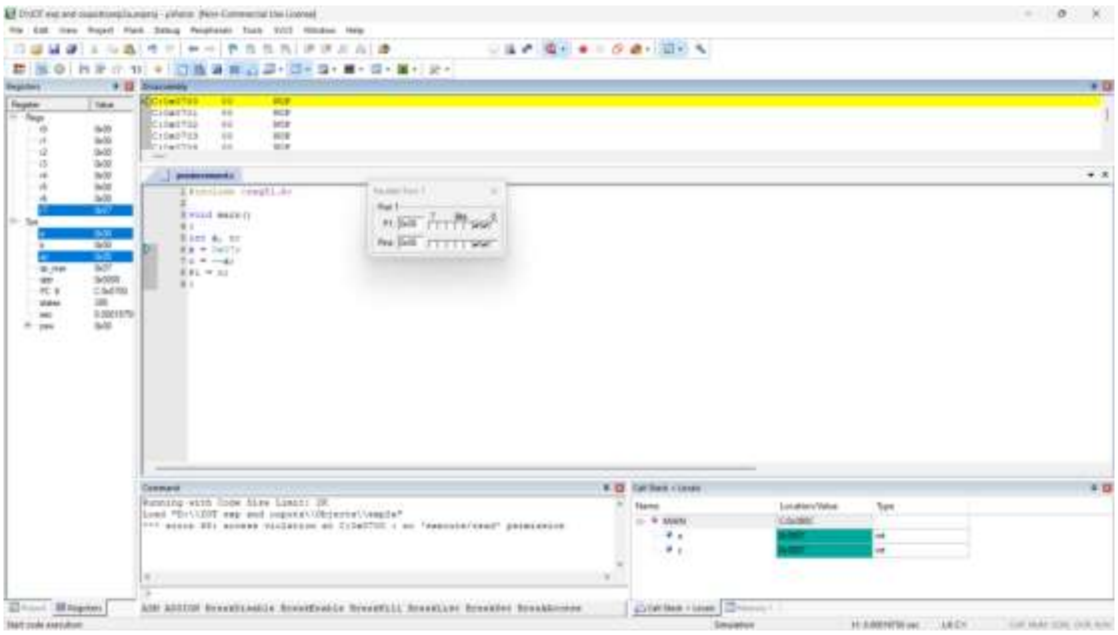
i) **SIMPLE PROGRAM FOR POST INCREMENT**



```
#include <reg51.h>

void main()
{
int a, c;
a = 0x07;
c = a++;
P1 = c;
}
```

```
Running with Code Size Limit: 2K
Load "D:\\IOT exp and outputs\\Objects\\exmple"
```

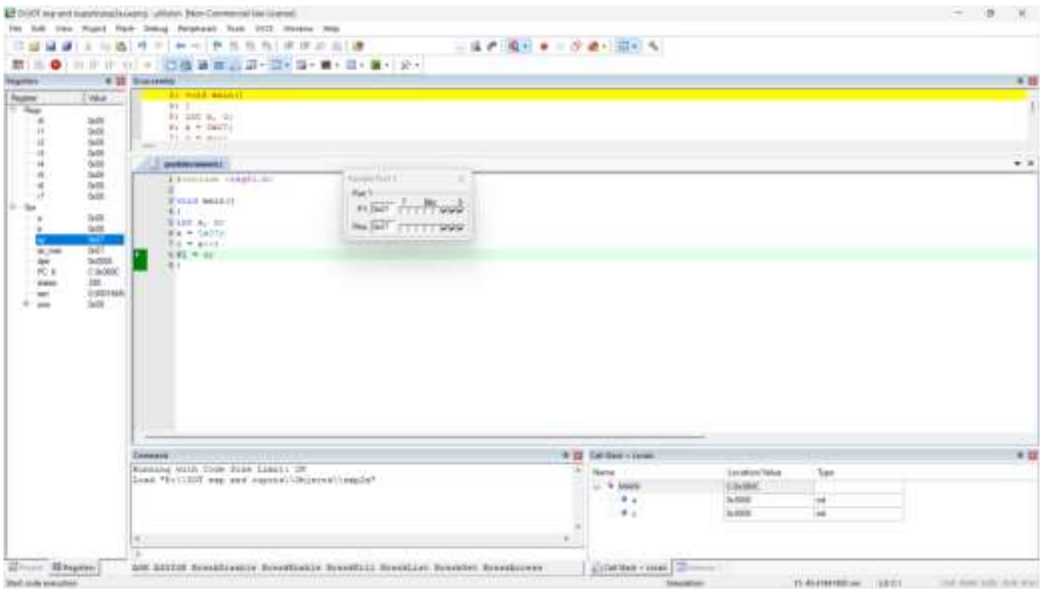
j) **SIMPLE PROGRAM FOR PRE DECREMENT**



```
#include <reg51.h>

void main()
{
int a, c;
a = 0x07;
c = --a;
P1 = c;
}
```

```
Running with Code Size Limit: 2K
Load "D:\\IOT exp and outputs\\Objects\\exmple"
*** error 65: access violation at C:0x0700 : no 'execute/read' permission
```

k) SIMPLE PROGRAM FOR POST DECREMENT



```
#include <reg51.h>

void main()
{
int a, c;
a = 0x07;
c = a--;
P1 = c;
}
```

```
Running with Code Size Limit: 2K
Load "E:\\IOT exp and ouputs\\Objects\\exp2a"
```

RESULT:

Thus the Embedded C program simulated using 8051 microcontroller in Keil µ vision version.

EXPT.NO:03	WRITE 8051 ASSEMBLY LANGUAGE EXPERIMENTS USING SIMULATOR
DATE:	

AIM:
To Write and simulate basic ascending, descending, smallest, largest, comparison and swapping program using ASM in 8051 microcontroller.

HARDWARE/SOFTWARE REQUIREMENTS:
Keil µ vision version 3

PROCEDURE:
1) Open project and go to new Micro Vision Project.
2) Give a name to project and save it.
3) Select a device for project "target 1".
4) Select the device name and category as 89C51 under Atmel Category and click on.
5) Now go to file and add new file from the menu.
6) Save the file with the specific name with .asm extension.
7) Add this .asm extension file to source group folder in the project window by right clicking on source group folder and selecting existing files to source group 1.
8) Built the project from the build target option in the project tab. Once the project is built with no errors click debug to start debug session.
9) Click on debug mode and then run F5.
10) Give the output in the registered windows and also from peripherals that is I/O ports.

PROGRAM:

a) LARGEST NUMBER FROM AN ARRAY OF 10 NUMBERS

```
          ORG 0000H;
          MOV R0, #50H;
          MOV A, @R0;
          MOV R2, A;
          DEC R2;
          INC R0;
          MOV B, @R0;
          INC R0;
BACK:     MOV A, @R0;
          CJNE A, B, LOOP;
          SJMP LOOP 1;
LOOP:     JC LOOP 1;
          MOV B, A;
LOOP 1:   INC R0;
          DJNZ R2, BACK;
NEXT:     MOV 60H, B;
          END;
```

b) SMALLEST NUMBER FROM AN ARRAY OF 10 NUMBERS

```
                ORG 0000H;
                MOV R0, #50H;
                MOV A, @R0;
                MOV R2, A;
                DEC R2;
                INC R0;
                MOV B, @R0;
                INC R0;
BACK:    MOV A, @R0;
               CJNE A, B, LOOP;
               SJMP LOOP 1;
LOOP:    JNC LOOP 1;
               MOV B, A;
LOOP 1:  INC R0;
               DJNZ R2, BACK;
NEXT:   MOV 60H, B;
               END;
```

c) ARRANGE THE NUMBERS IN ASCENDING ORDER

```
          ORG 0000h
          MOV R2, #0Ah
AGAIN1: MOV R1, #0Ah
          DEC R1
          MOV R0, #30h
AGAIN2: MOV a,@R0
          INC R0
          MOV b,@R0
          CJNE a, b, nex1
NEX1:     JC Next
          XCH a, b
         MOV @R0, b
         DEC R0
         MOV @R0, a
         INC R0
NEXT:     DJNZ R1, Again2
         DJNZ R2, Again1
         SJMP $
         END
```

d) ARRANGE THE NUMBERS IN DESCENDING ORDER

```
                 ORG 0000h
                 MOV R2, #0Ah
AGAIN 1: MOV R1, #0Ah
                 DEC R1
                 MOV R0, #30H
AGAIN 2: MOV a,@R0
                 INC R0
                 MOV b,@R0
                 CJNE a, b, nex1
 NEX 1:    JNC Next
                XCH a, b
                MOV @R0, b
                DEC R0
                MOV @R0,a
                INC R0
NEXT:     DJNZ R1, Again2
               DJNZ R2, Again1
               SJMP $
               END
```

e) Assume that P1 is an input port connected to a temperature sensor. Write a program to read the temperature and test it for the value of 75. According to the test results, place the temperature value into the registers indicated by the following.

If T = 75	**then A = 75**
If T < 75	**then R1 = T**
If T >75	**then R2 = T**

SOLUTION:

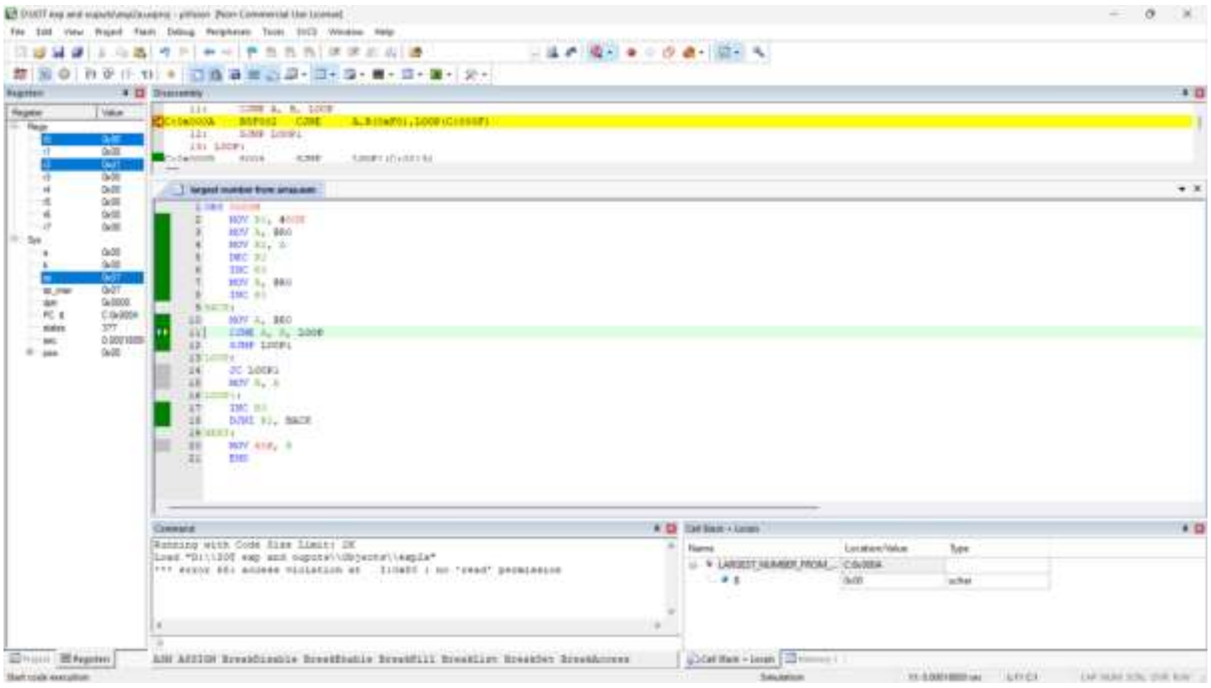
```
               MOV P1, #0FFH
               MOV A, P1
               CJNE A, #75,OVER
               SJMP EXIT
OVER:   JNC NEXT
               MOV R1, A
               SJMP EXIT
NEXT:   MOV R2, A
EXIT:    ……
```

f) PROGRAM TO SWAP TWO NOS

```
MOV R0, #OOH;
MOV A, #01H;
MOV B, #054;
MOV RO, B;
MOV B, A;
MOV A, R0;
END;
```

OUTPUT:

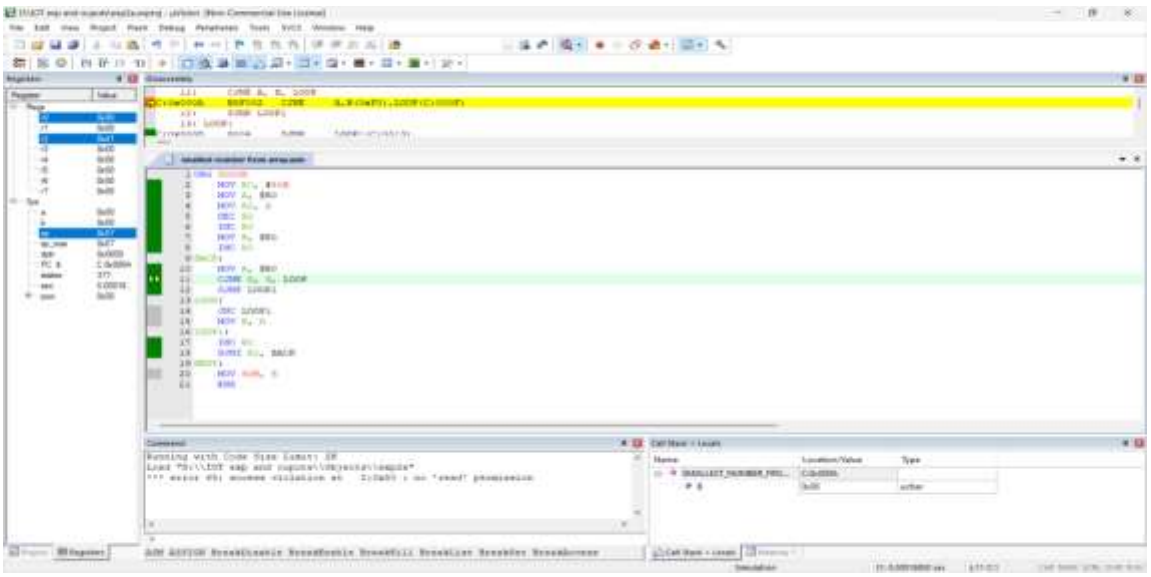
a) LARGEST NUMBER FROM AN ARRAY OF 10 NUMBERS



```
ORG [illegible]
    MOV R1, #[illegible]
    MOV A, @R0
    MOV R2, A
    DEC R2
    INC R0
    MOV B, @R0
    INC R0
BACK:
    MOV A, @R0
    CJNE A, B, LOOP
    SJMP LOOP1
LOOP:
    JC LOOP1
    MOV B, A
LOOP1:
    INC R0
    DJNZ R2, BACK
NEXT:
    MOV [illegible], B
    END
```

```
Running with Code Size Limit: 2K
Load "D:\\IOT exp and ouputs\\Objects\\expla"
*** error 65: access violation at   I:0x80 : no 'read' permission
```

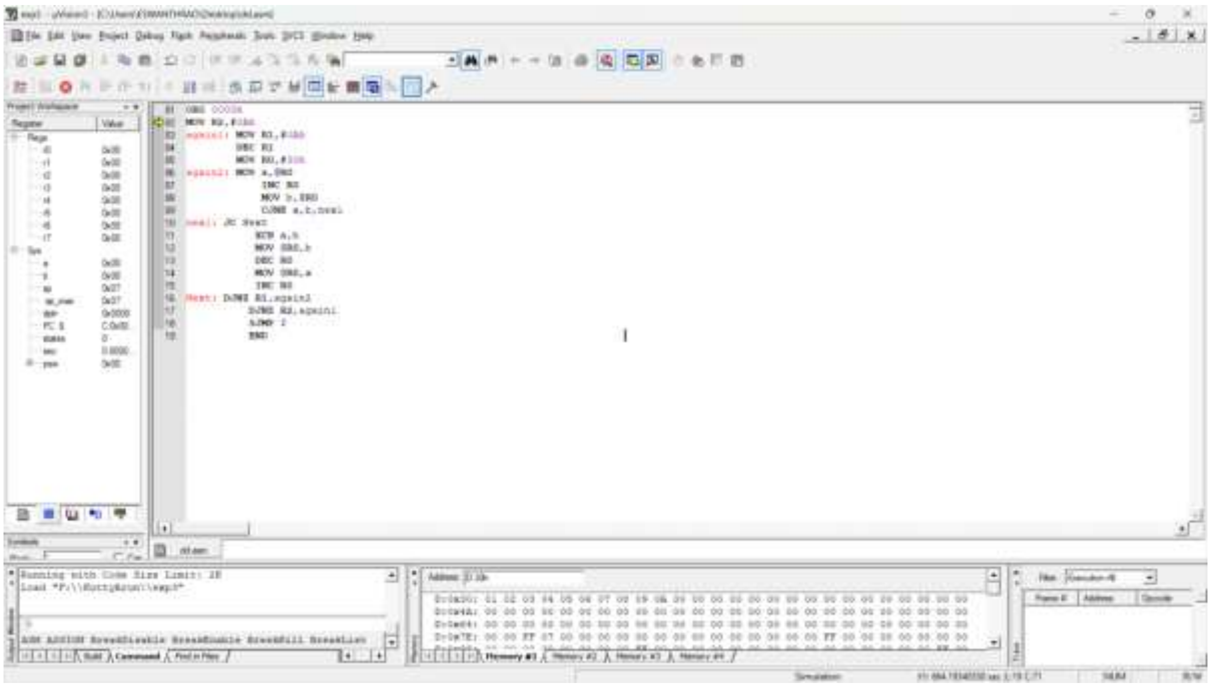
b) SMALLEST NUMBER FROM AN ARRAY OF 10 NUMBERS



```
ORG [illegible]
    MOV R1, #[illegible]
    MOV A, @R0
    MOV R2, A
    DEC R2
    INC R0
    MOV B, @R0
    INC R0
BACK:
    MOV A, @R0
    CJNE A, B, LOOP
    SJMP LOOP1
LOOP:
    JNC LOOP1
    MOV B, A
LOOP1:
    INC R0
    DJNZ R2, BACK
NEXT:
    MOV [illegible], B
    END
```

```
Running with Code Size Limit: 2K
Load "D:\\IOT exp and ouputs\\Objects\\expla"
*** error 65: access violation at   I:0x80 : no 'read' permission
```

c) ARRANGE THE NUMBERS IN ASCENDING ORDER

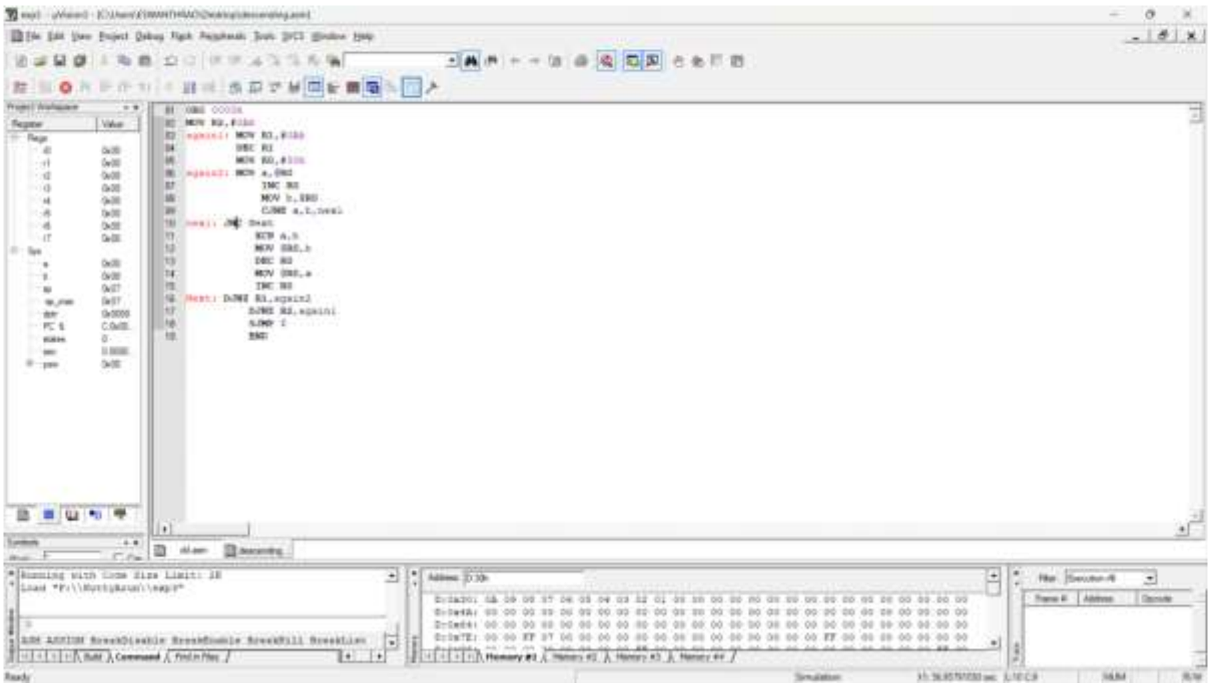


```
ORG 0000h
MOV R2,#10h
again1: MOV R1,#10h
        DEC R1
        MOV R0,#30h
again2: MOV a,@R0
          INC R0
          MOV b,@R0
          CJNE a,b,next
next: JC Next
         XCH a,b
         MOV @R0,b
         DEC R0
         MOV @R0,a
         INC R0
Next: DJNZ R1,again2
       DJNZ R2,again1
       SJMP $
       END
```

```
Running with Code Size Limit: 2K
Load "F:\\KuttyArun\\exp3"
```

```
Address: D:30h
D:0x30: 01 02 03 04 05 06 07 08 09 0A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x4A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x64: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x7E: 00 00 FF 07 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF 00 00 00 00 00 00 00
```

d) ARRANGE THE NUMBERS IN DESCENDING ORDER

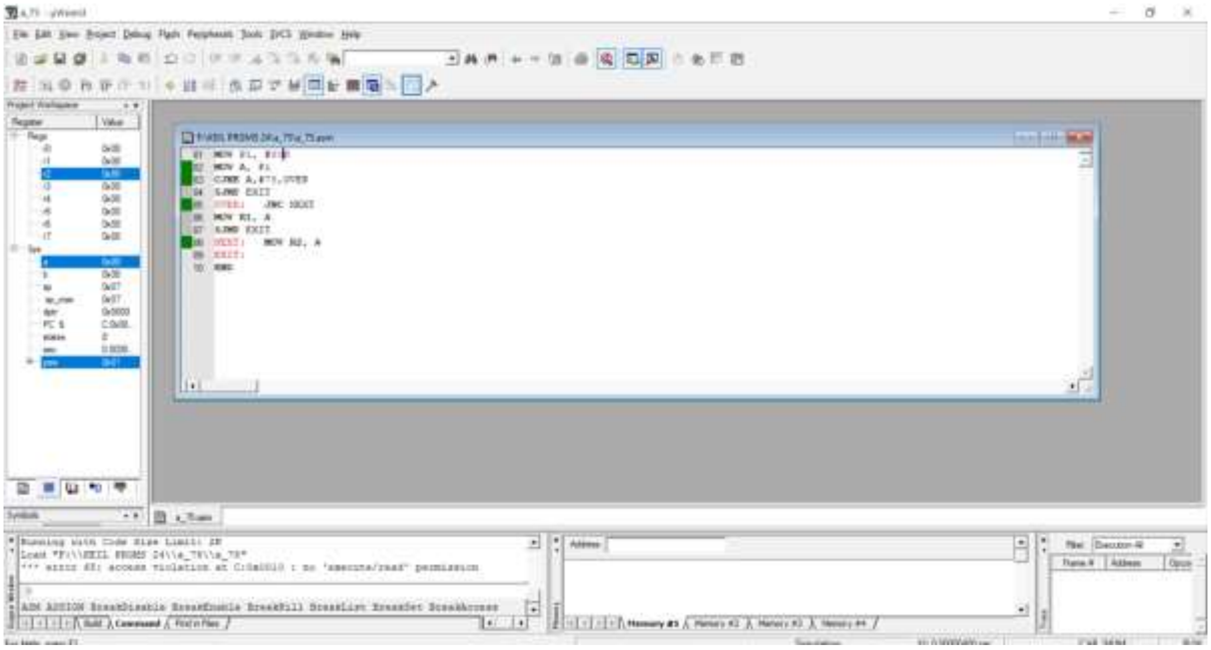


```
ORG 0000h
MOV R2,#10h
again1: MOV R1,#10h
        DEC R1
        MOV R0,#30h
again2: MOV a,@R0
          INC R0
          MOV b,@R0
          CJNE a,b,next
next: JNC Next
         XCH a,b
         MOV @R0,b
         DEC R0
         MOV @R0,a
         INC R0
Next: DJNZ R1,again2
       DJNZ R2,again1
       SJMP $
       END
```

```
Running with Code Size Limit: 2K
Load "F:\\KuttyArun\\exp3"
```

```
Address: D:30h
D:0x30: 0A 09 08 07 06 05 04 03 02 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x4A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x64: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x7E: 00 00 FF 07 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF 00 00 00 00 00 00 00
```

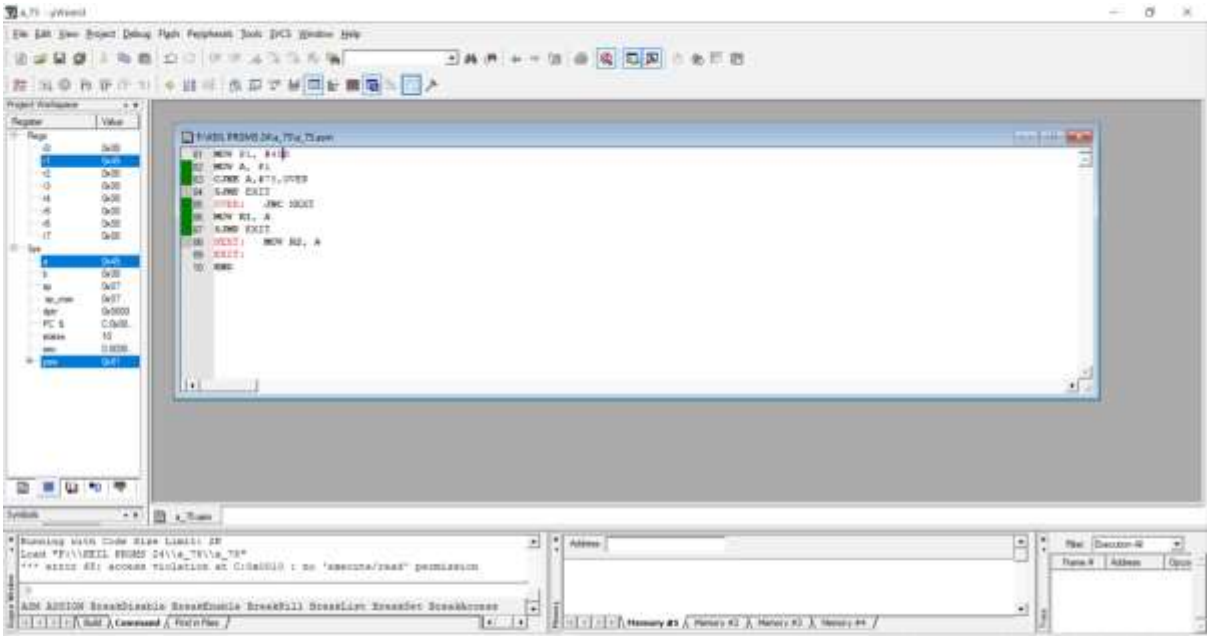
e) COMPARISON



```
MOV R1, #[illegible]
MOV A, R1
CJNE A,#71,OVER
SJMP EXIT
OVER:   JNC NEXT
MOV R1, A
SJMP EXIT
NEXT:   MOV R2, A
EXIT:
END
```

```
Running with Code Size Limit: 2K
Load "F:\\KEIL PRGMS 24\\a_7E\\a_7E"
*** error 65: access violation at C:0x0010 : no 'execute/read' permission
```

```
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess
```

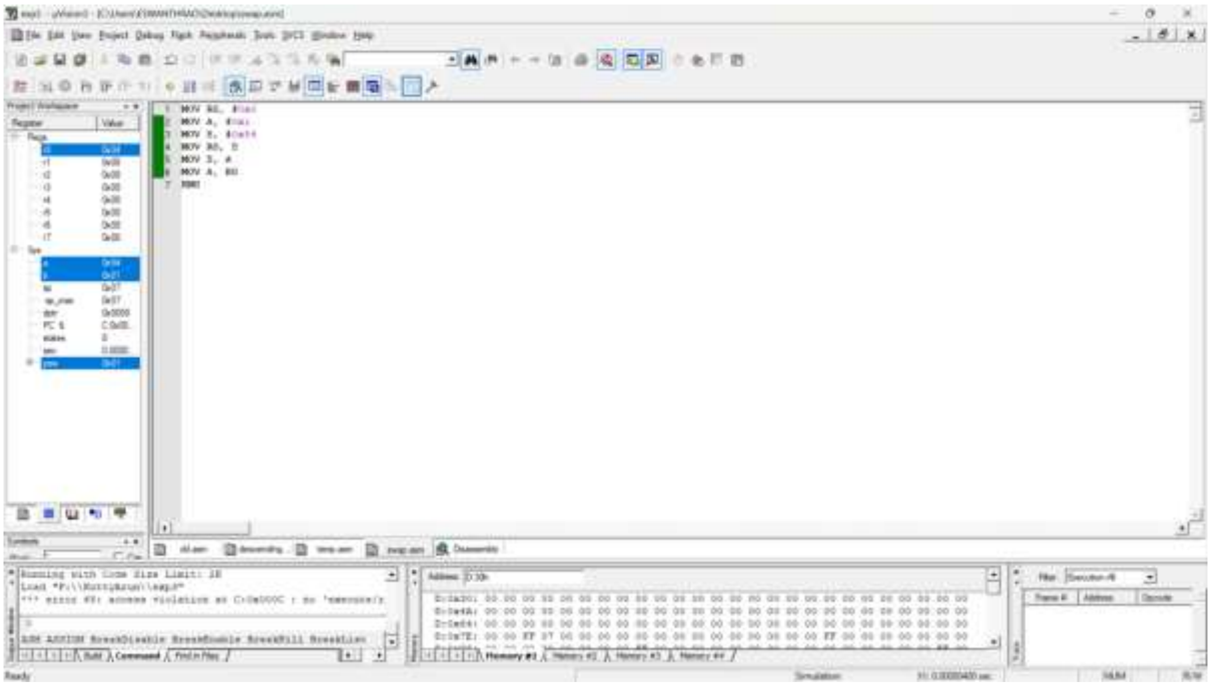


```
MOV R1, #[illegible]
MOV A, R1
CJNE A,#71,OVER
SJMP EXIT
OVER:   JNC NEXT
MOV R1, A
SJMP EXIT
NEXT:   MOV R2, A
EXIT:
END
```

```
Running with Code Size Limit: 2K
Load "F:\\KEIL PRGMS 24\\a_7E\\a_7E"
*** error 65: access violation at C:0x0010 : no 'execute/read' permission
```

```
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess
```

f) PROGRAM TO SWAP TWO NOS



```
MOV R0, #[illegible]
MOV A, #[illegible]
MOV B, #[illegible]
MOV R0, B
MOV B, A
MOV A, R0
END
```

```
Running with Code Size Limit: 2K
Load "F:\\[illegible]\\swap.o"
*** error 65: access violation at C:0x000C : no 'execute/r
```

```
D:0x30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x4A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x64: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x7E: 00 00 FF 07 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF 00 00 00 00 00 00
```

RESULT:

Thus the basic arithmetic program using ASM in 8051 microcontroller has been executed.

EXPT.NO:04	WRITE BASIC AND ARITHMETIC PROGRAMS USING EMBEDDED C
DATE:	

AIM:
To Write and simulate basic arithmetic program using embedded C in 8051 microcontroller.

HARDWARE/SOFTWARE REQUIREMENTS:
1) Keil µ vision version 3

PROCEDURE:
1) Open project and go to new Micro Vision Project.
2) Give a name to project and save it.
3) Select a device for project "target 1".
4) Select the device name and category as 89C51 under Atmel Category and click on.
5) The project and its folders can be seen on the left side in the project.
6) Now go to file and add new file from the menu.
7) Type the embedded C program to be simulated.
8) Save the file with a specific name with .c extension.
9) Add this .c extension file to source group folder in the project window by right clicking on source group 1 folder and selecting the existing files to source group 1.
10) Built the project from the build target option in the project tab or by pressing epsilon on the keyboard.
11) Once the project is built with no errors click debug to start debug session.
12) Click on debug mode and then run F5.
13) Give the output in the registered windows and also from peripherals that is I/O ports.

PROGRAM:

a) WRITE AN 8051 C PROGRAM TO SEND VALUES 00-FF TO PORT P

```
#include  <reg51.h>
Void main (void)
{
  Unsigned char  z;
  For (z=0; z<=255; z++)
{
  P1= z;
}
}
```

b) WRITE AN 8051 C PROGRAM TO SEND HEX VALUES FOR ASCII CHARACTERS OF 0,1,2,3,4,5,A,B,C AND D TO PORT P1.

```
#include  <reg51.h>
Void main (void)
{
  Unsigned char my num[] = "012345ABCD";
  Unsigned char  z;
  For (z=0; z<=10; z++){
```

```
               P1= mynum[z];
              }
               }
```

c) WRITE AN 8051 C PROGRAM TO TOGGLE ALL THE BITS OF P1 CONTINUOUSLY.

```
#include  <reg51.h>
Void main (void)
{
   For (; ;);
       {
         P1 = 0 x 55;
         P2 = 0 x AA;
        }
}
```

d) WRITE AN 8051 C PROGRAM TO SEND VALUES OF -4 TO +4 TO PORT P1.

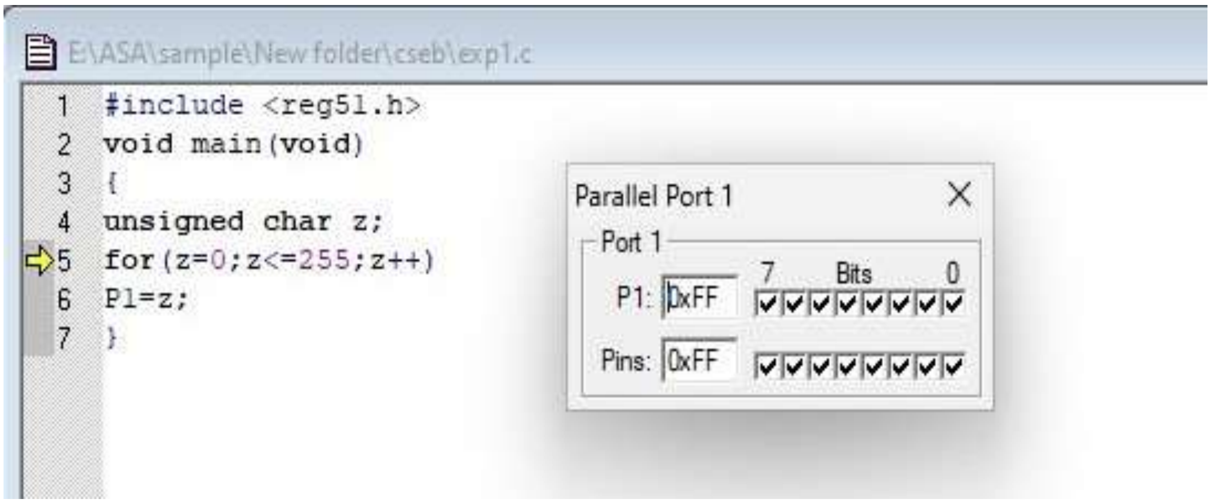
```
#include  <reg51.h>
Void main (void)
{
   char my num[] = {+1,-1,+2,-2,+3,-3,+4,-4};
   Unsigned char  z;
   For (z=0; z<=8; z++){
   P1= mynum[z];
}}
```

e) WRITE AN 8051 C PROGRAM TO TOGGLE BIT D0 OF THE PORT P1(P1.0) 50,000 TIMES.

```
#include <reg51.h>
sbit MYBIT = P1^0;
void main (void)
{
      Unsigned int  z;
      For (z=0; z<=50000; z++)
           {
                MYBIT = 0;
                MYBIT = 1; } }
```

OUTPUT:

a) 8051 C PROGRAM TO SEND VALUES 00-FF TO PORT P



E:\ASA\sample\New folder\cseb\exp1.c

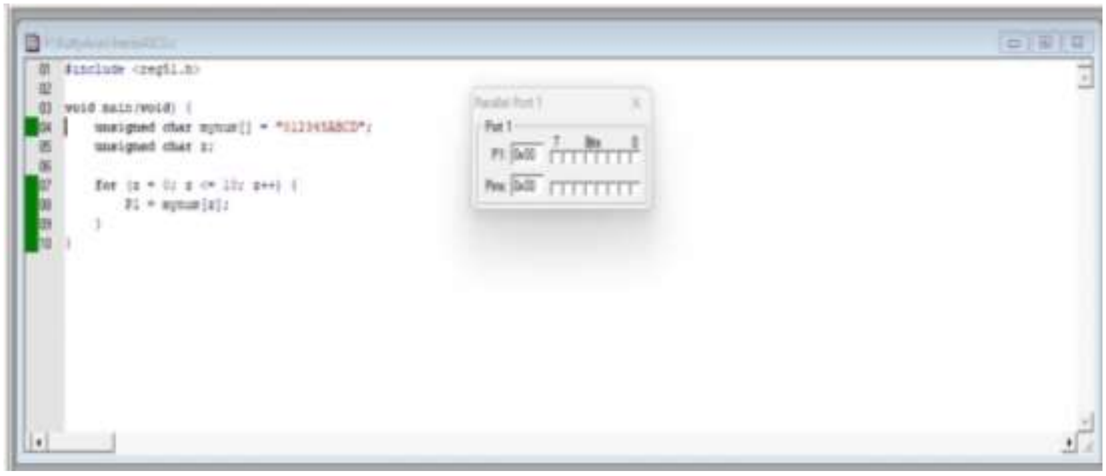
```
#include <reg51.h>
void main(void)
{
unsigned char z;
for(z=0;z<=255;z++)
P1=z;
}
```

Parallel Port 1

Port 1

```
P1:   0xFF
Pins: 0xFF
```

b) 8051 C PROGRAM TO SEND HEX VALUES FOR ASCII CHARACTERS



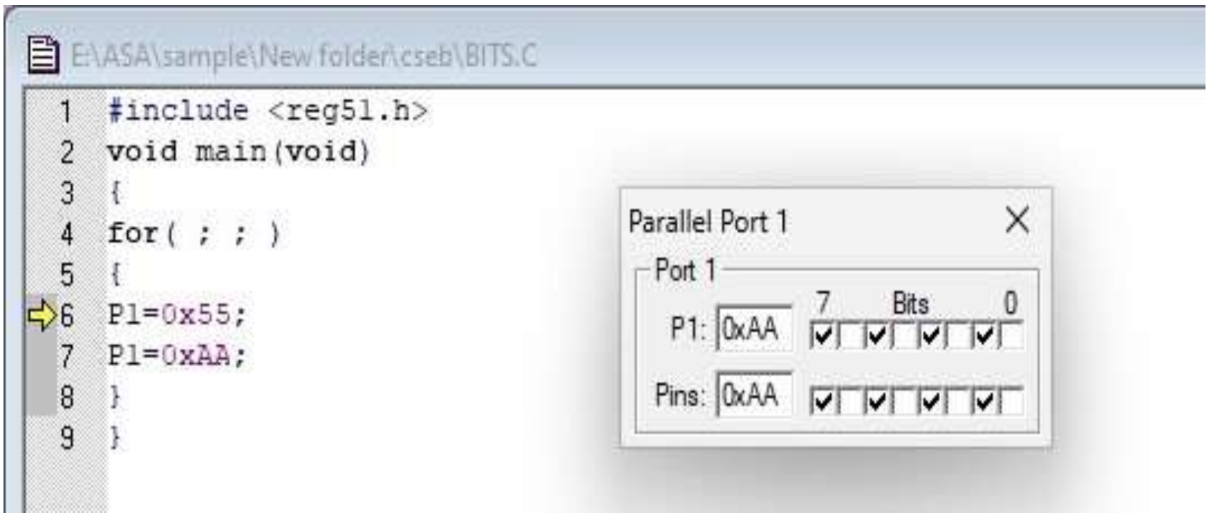
```
#include <reg51.h>

void main(void) {
    unsigned char mynum[] = "012345ABCD";
    unsigned char z;

    for (z = 0; z <= 10; z++) {
        P1 = mynum[z];
    }
}
```

Parallel Port 1

Port 1

```
P1:   0x00
Pins: 0x00
```

c) 8051 C PROGRAM TO TOGGLE ALL THE BITS OF P1 CONTINUOUSLY.


E:\ASA\sample\New folder\cseb\BITS.C

```c
#include <reg51.h>
void main(void)
{
for( ; ; )
{
P1=0x55;
P1=0xAA;
}
}
```

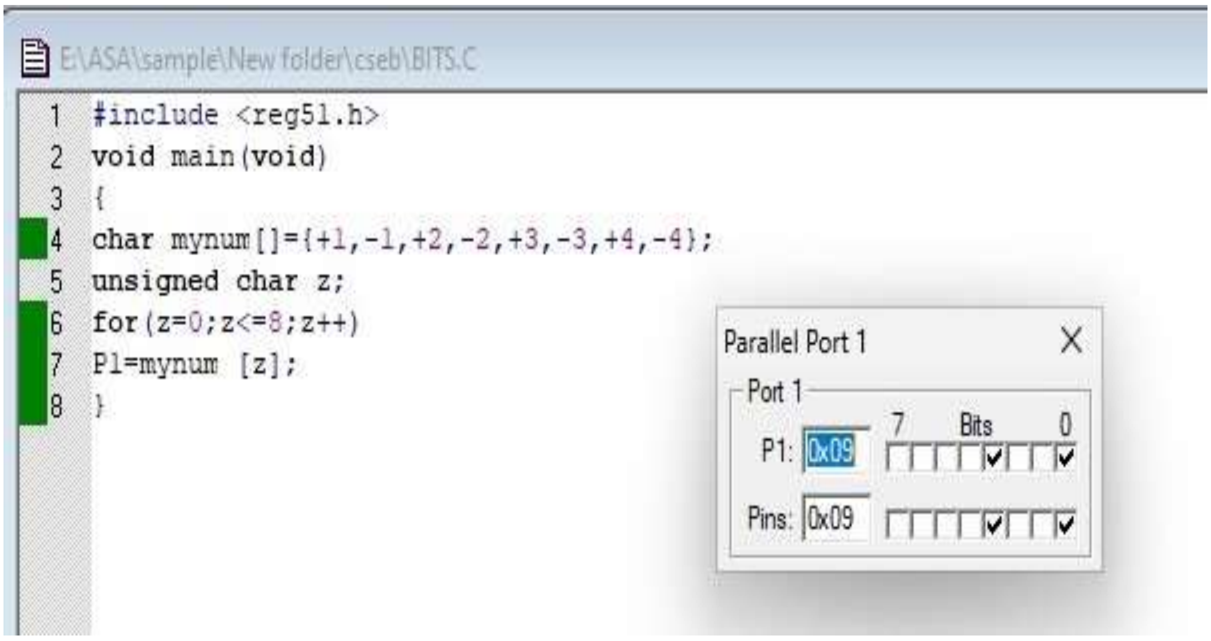
Parallel Port 1

Port 1

P1: 0xAA

Pins: 0xAA

d) 8051 C PROGRAM TO SEND VALUES OF -4 TO +4 TO PORT P1.


E:\ASA\sample\New folder\cseb\BITS.C

```c
#include <reg51.h>
void main(void)
{
char mynum[]={+1,-1,+2,-2,+3,-3,+4,-4};
unsigned char z;
for(z=0;z<=8;z++)
P1=mynum [z];
}
```

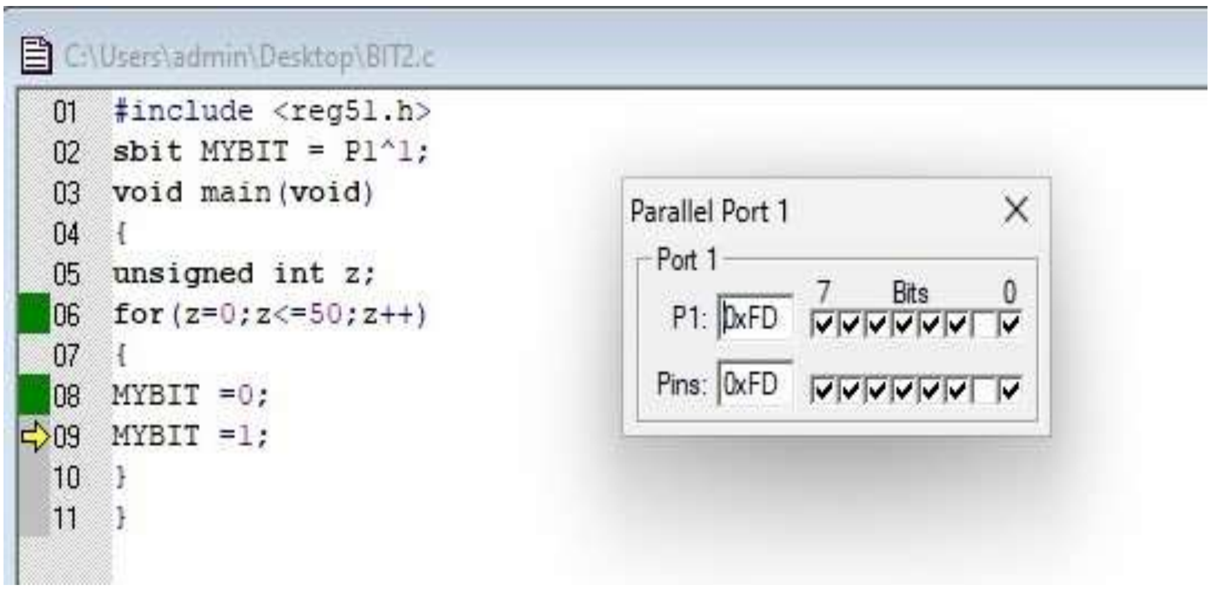
Parallel Port 1

Port 1

P1: 0x09

Pins: 0x09

e) 8051 C PROGRAM TO TOGGLE BIT D0 OF THE PORT P1(P1.0) 50,000 TIMES.



C:\Users\admin\Desktop\BIT2.c

```
#include <reg51.h>
sbit MYBIT = P1^1;
void main(void)
{
unsigned int z;
for(z=0;z<=50;z++)
{
MYBIT =0;
MYBIT =1;
}
}
```

Parallel Port 1

Port 1

P1: 0xFD

Pins: 0xFD

RESULT:

Thus the Embedded C program simulated using 8051 microcontroller in Keil µ vision version 3.

EXPT.NO 5	INTRODUCTION TO ARDUINO PLATFORM AND PROGRAMMING
DATE:	

AIM:

To upload and execute basic programs on Arduino board using Arduino IDE.

HARDWARE / SOFTWARE REQUIREMENTS:

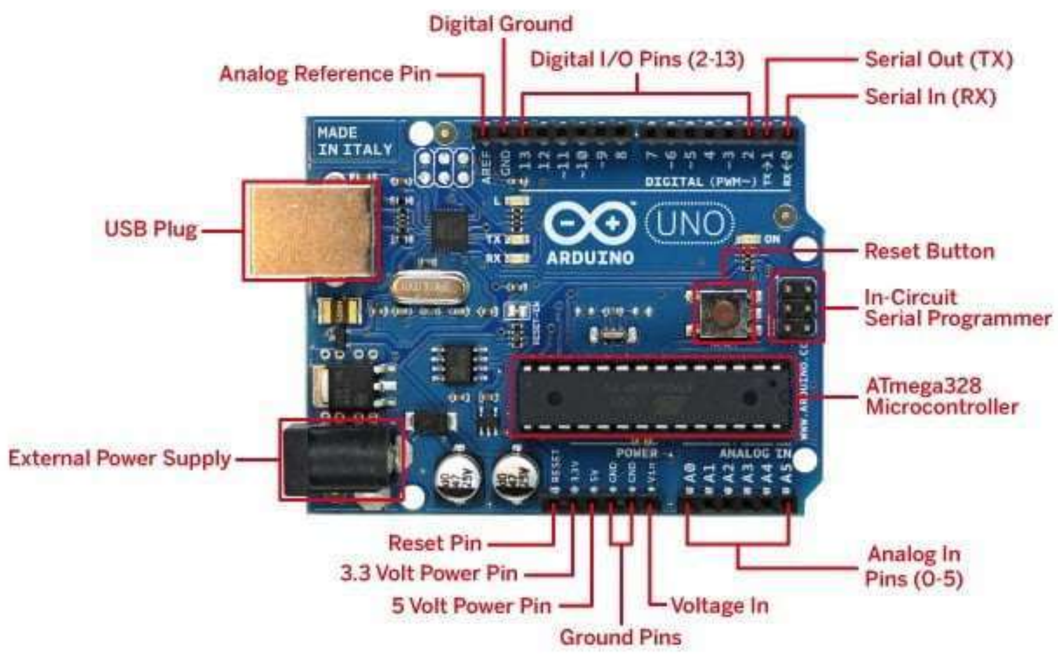
1. Arduino Uno board
2. Arduino IDE

THEORY:

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments.



Advantages:

- Inexpensive
- Cross-platform
- Simple, clear programming environment
- Open source and extensible software
- Open source and extensible hardware

PROCEDURE:

1. Open Arduino IDE
2. Click File → New Sketch
3. Connect Arduino Uno using USB-serial communication cable
4. In Tools select Boards →Arduino Uno and also select the port to which the board is connected
5. Type the program and verify
6. Upload the program to the board
7. Print a string: Check for the string in Output window of Arduino IDE
8. Blink LED, Blink LED without delay function: Check for the built-in LED on the Arduino Uno board to go ON and OFF repeatedly.

PROGRAM:

1. **Print a string**

```
// the setup function runs once when you press reset or power the board
void setup() {
  Serial.begin(9600);
  Serial.println("hello world");
}

// the loop function runs over and over again forever
void loop() {

}
```

..

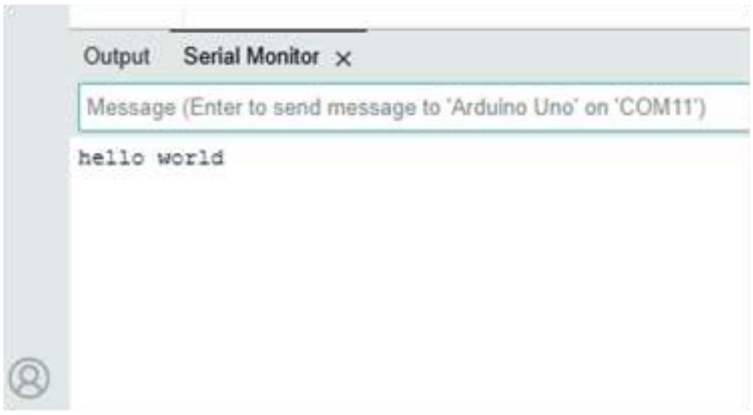
2. **Blink LED**

```
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(1000);                      // wait for a second
  digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
  delay(1000);                      // wait for a second
}
```

..

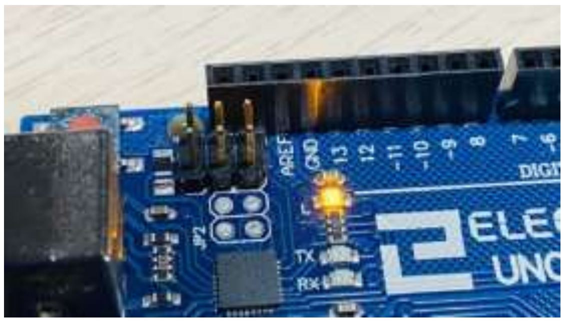
OUTPUT:

1. **Print a String:**



```
hello world
```

2. **Blink LED, Blink LED without Delay function:**



RESULT:

Thus some basic Arduino Programming has been done and executed in Arduino Uno board using Arduino IDE.

EXPT.NO 6 a	EXPLORE DIFFERENT COMMUNICATION METHODS WITH IOT DEVICES-ZIGBEE
DATE:	

AIM:

To perform an experiment to configure two ZigBee XBee S2C modules as coordinator and router and communicate between them using XCTU software.

HARDWARE / SOFTWARE REQUIREMENTS:

1. ZigBee XBee S2C – 2 no.s
2. XBee USB adapter – 2 no.s
3. USB cable – 2 no.s
4. XCTU software

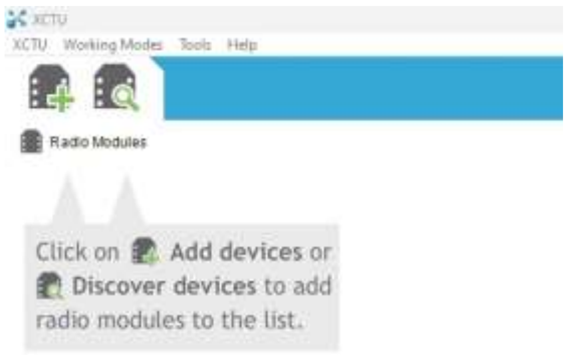
THEORY:

ZigBee/XBee PRO S2C 802.15.4 RF Modules are an embedded solutions providing wireless end-point connectivity to devices. These devices use the IEEE 802.15.4 networking protocol for fast point-to-multi point or peer-to-peer networking. They designed for high-throughput applications requiring low latency and predictable communication timing.

XBee and XBee-PRO ZigBee modules are ideal for hobbyists and educational research as well as applications in the energy and control markets where manufacturing efficiencies are critical. The Serial Peripheral Interface (SPI) provides a high-speed interface and optimizes the integration with embedded microcontrollers, lowering development costs and reducing time to market. Digi's ZigBee compatible module is based on the Ember EM35x (EM357 and EM3587) system on chip (SoC) radio ICs from Silicon Labs, utilizing 32-bit ARM Cortex M3 processor.

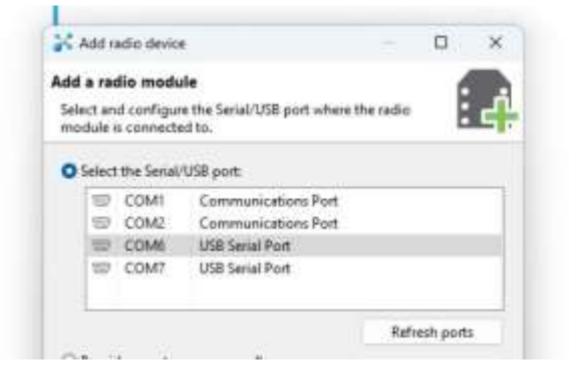
PROCEDURE:

1. XBee connection with USB adapter: Connect the XBee module to XBee USB adapter
2. Open XCTU software 6.1.1 software. Click on the search button of XCTU. It directs the XBee connected to the USB adapter. Start the configuration of the XBee module.
3. Open the XCTU software& then click on the search on the top.



Click on Add devices or Discover devices to add radio modules to the list.

4. To know the com port where radios are connected open the device manager and note down the com port. Select the ports and click on next.


Add radio device

Add a radio module

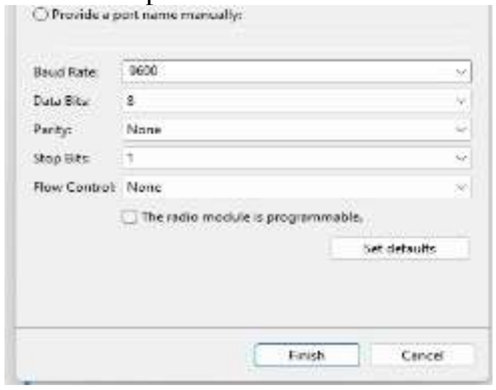
Select and configure the Serial/USB port where the radio module is connected to.

Select the Serial/USB port:

COM1	Communications Port
COM2	Communications Port
COM6	USB Serial Port
COM7	USB Serial Port

5. Leave the port parameters to default. (i.e. 9600-8-N-1) and click on finish

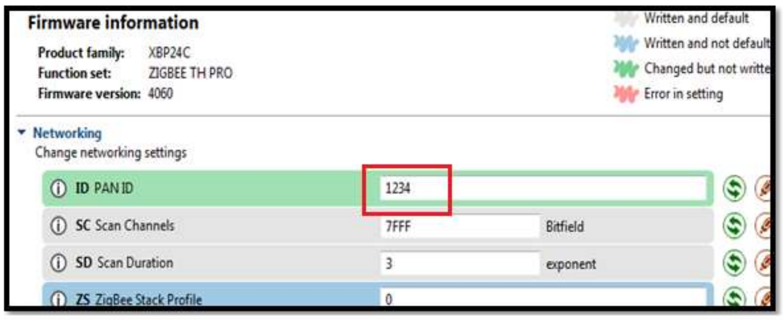
- 9600- Baud rate
- 8- Data bit
- Parity- None and 1- stop bit


Provide a port name manually:

Baud Rate:	9600
Data Bits:	8
Parity:	None
Stop Bits:	1
Flow Control:	None

The radio module is programmable.

6. The radios (XBees) are discovered by XCTU& click on Add selected device to see the XBee listed on the left side of the XCTU.
7. Click on the First radio to load the setting. You can see the product family as XBP24C and Function set as 802.15.4 TH PRO.
8. To upgrade the latest firmware, click on the Upgrade Firmware and leave the function set to default and select the Newest version. Click on the finish to upgrade.
9. Now configure the first radio as COORDINATOR. First set Pan ID for the network. This can be any Hex value from 0 to FFFF. (Set it as 1234).

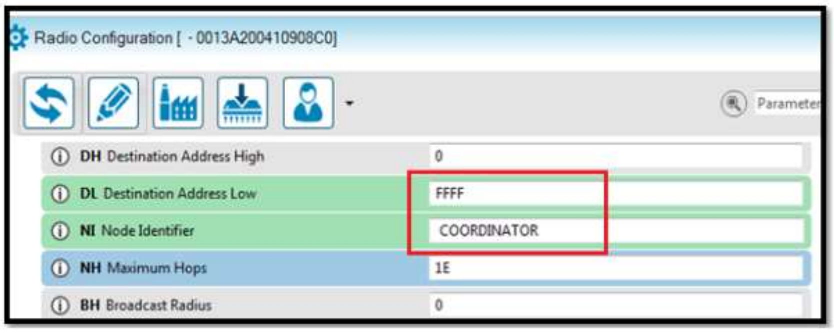

Firmware information

Product family: XBP24C
Function set: ZIGBEE TH PRO
Firmware version: 4060

Networking
Change networking settings

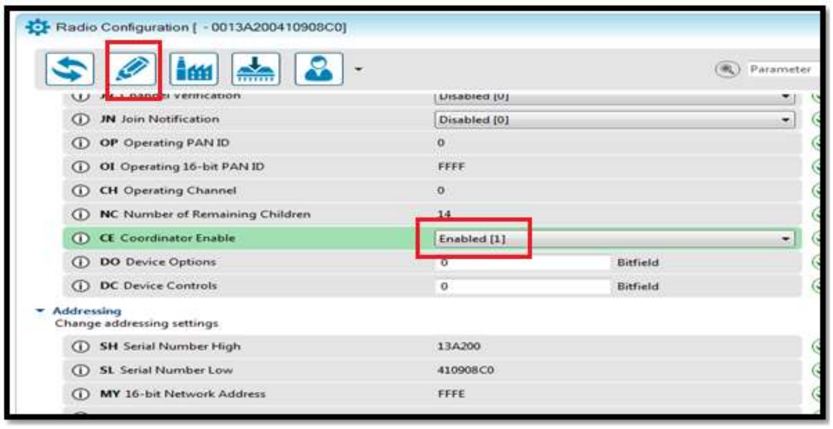
ID PAN ID	1234	
SC Scan Channels	7FFF	Bitfield
SD Scan Duration	3	exponent
ZS ZigBee Stack Profile	0	

10. To operate the Radio in the broadcast mode set the DL (Destination address) as FFFF. Type the name of the first radio in NI (Node Identifier). Any name can be provided. Here the NI set as COORDINATOR.


Radio Configuration [- 0013A200410908C0]

DH Destination Address High	0
DL Destination Address Low	FFFF
NI Node Identifier	COORDINATOR
NH Maximum Hops	1E
BH Broadcast Radius	0

11. To make the Radio as COORDINATOR, Set the CE parameter to Enable and click on the Write button (Pencil Icon) to save the information.


Radio Configuration [- 0013A200410908C0]

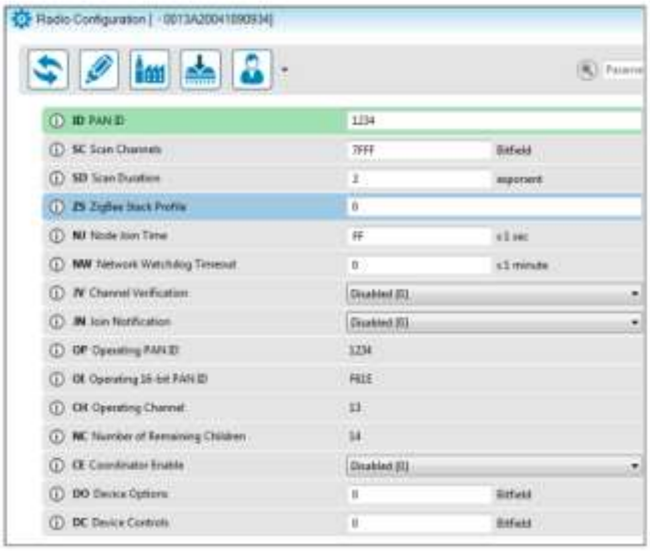
JV Channel Verification	Disabled [0]	
JN Join Notification	Disabled [0]	
OP Operating PAN ID	0	
OI Operating 16-bit PAN ID	FFFF	
CH Operating Channel	0	
NC Number of Remaining Children	14	
CE Coordinator Enable	Enabled [1]	
DO Device Options	0	Bitfield
DC Device Controls	0	Bitfield

Addressing
Change addressing settings

SH Serial Number High	13A200
SL Serial Number Low	410908C0
MY 16-bit Network Address	FFFE

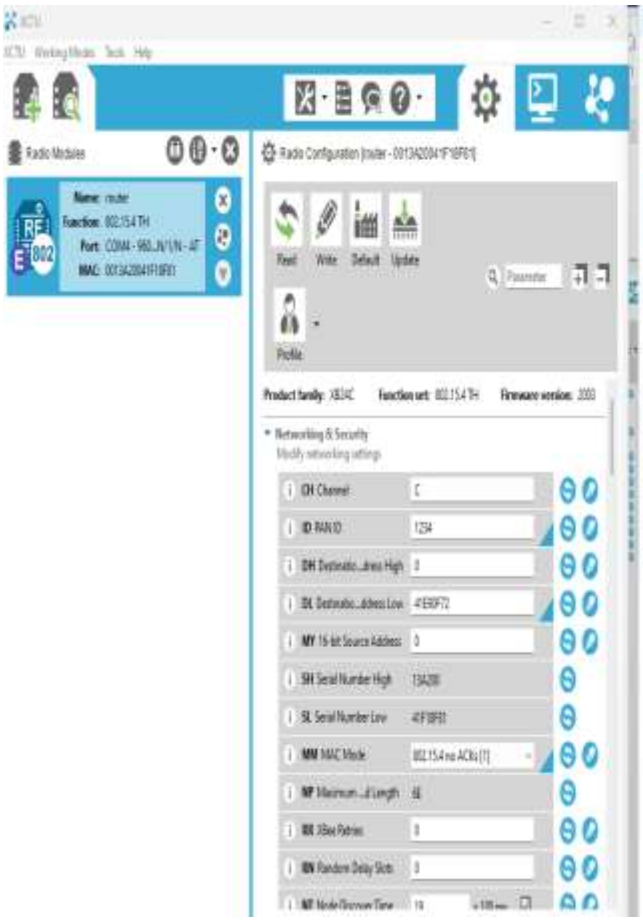
12. Now click on the second radio and configure this as ROUTER.

13. Enter the Pan Id as 1234 which is the same as COORDINATOR. Set the JV channel Verification to enable so that it joins the COORDINATOR. Leave the CE to Disable and click on the Write to save the changes.


Radio Configuration [- 0013A20041090934]

ID PAN ID	1234	
SC Scan Channels	7FFF	Bitfield
SD Scan Duration	2	exponent
ZS ZigBee Stack Profile	0	
NJ Node Join Time	FF	x 1 sec
NW Network Watchdog Timeout	0	x 1 minute
JV Channel Verification	Disabled [0]	
JN Join Notification	Disabled [0]	
OP Operating PAN ID	1234	
OI Operating 16-bit PAN ID	FB1E	
CH Operating Channel	13	
NC Number of Remaining Children	14	
CE Coordinator Enable	Disabled [0]	
DO Device Options	0	Bitfield
DC Device Controls	0	Bitfield

14. Communication between Coordinator and Router: The Radios are configured, now you can do the communication between the COORDINATOR and Router, Click on the search button of the second window of XCTU. Select the com port of the second radio. That is the ROUTER radio. Click on the router radio to load the settings.



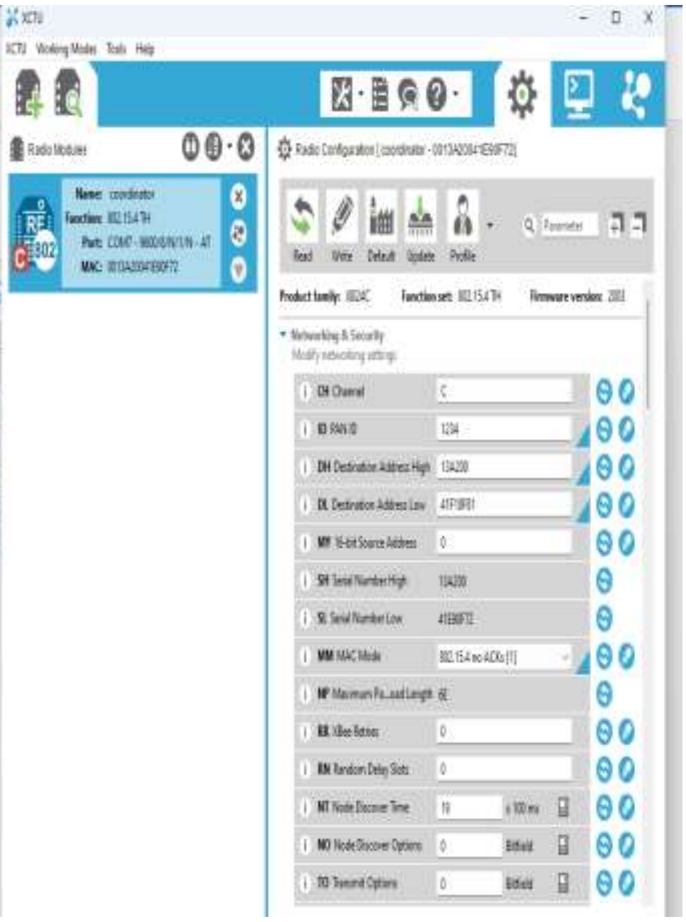
Radio Configuration [router - 0013A20041F18F81]

Name: router
Function: 802.15.4 TH
Port: COM4 - 960...N/1/N - AT
MAC: 0013A20041F18F81

Product family: XB24C Function set: 802.15.4 TH Firmware version: 2003

Networking & Security
Modify networking settings

CH Channel	C
ID PAN ID	1234
DH Destinatio...dress High	0
DL Destinatio...ddress Low	41E90F72
MY 16-bit Source Address	0
SH Serial Number High	13A200
SL Serial Number Low	41F18F81
MM MAC Mode	802.15.4 no ACKs [1]
NP Maximum...d Length	6E
RR XBee Retries	0
RN Random Delay Slots	0
NT Node Discover Time	19 x 100 ms


XCTU
XCTU Working Modes Tools Help
Radio Modules
Read Write Default Update Profile
Parameter

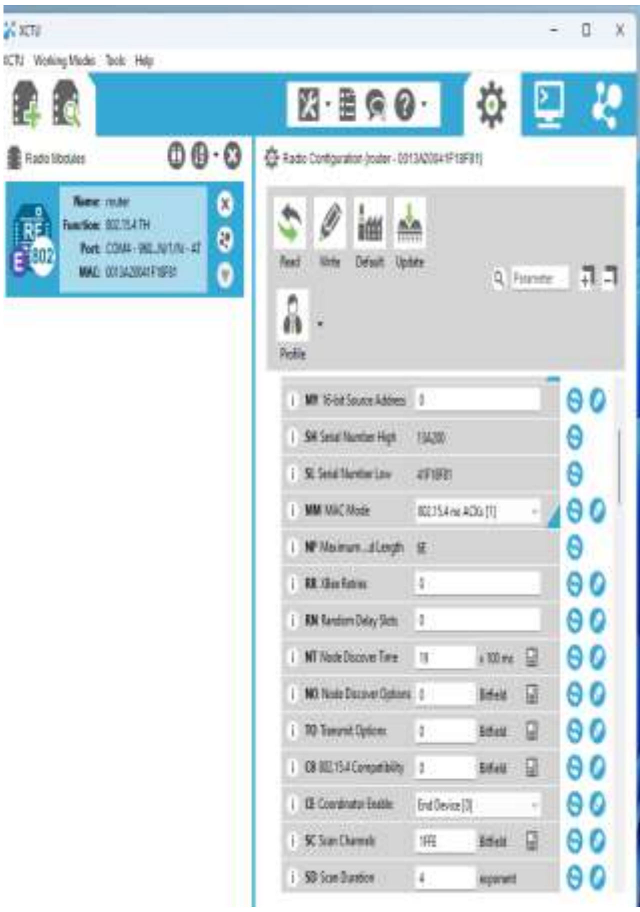
Radio Configuration [coordinator - 0013A20041E90F72]

Name: coordinator
Function: 802.15.4 TH
Port: COM7 - 9600/8/N/1/N - AT
MAC: 0013A20041E90F72

Product family: XB24C Function set: 802.15.4 TH Firmware version: 2003

Networking & Security
Modify networking settings

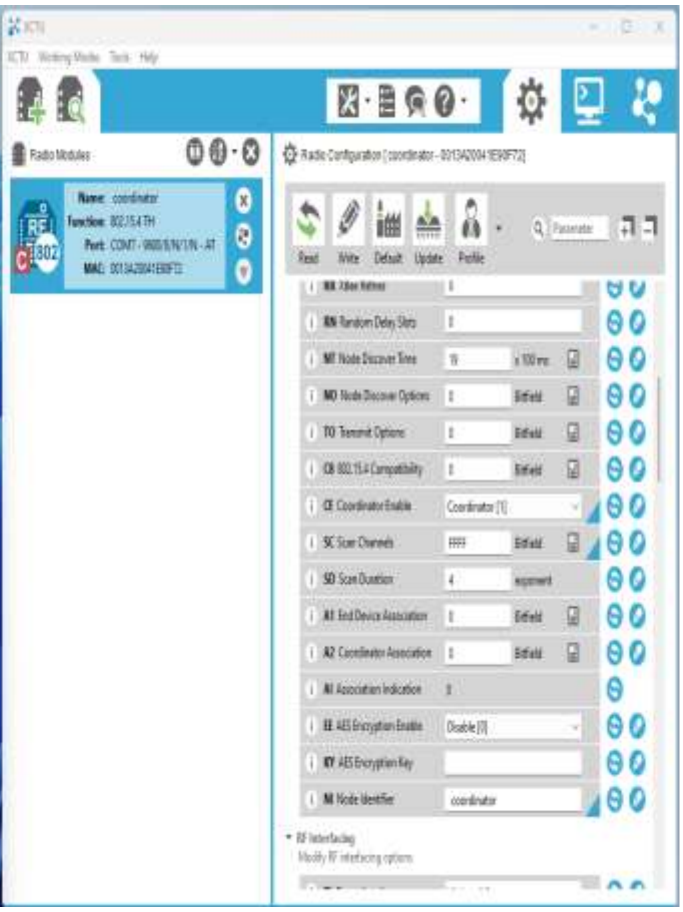
CH Channel	C
ID PAN ID	1234
DH Destination Address High	13A200
DL Destination Address Low	41F18F81
MY 16-bit Source Address	0
SH Serial Number High	13A200
SL Serial Number Low	41E90F72
MM MAC Mode	802.15.4 no ACKs [1]
NP Maximum Pa...load Length	6E
RR XBee Retries	0
RN Random Delay Slots	0
NT Node Discover Time	19 x 100 ms
NO Node Discover Options	0 Bitfield
TO Transmit Options	0 Bitfield



Radio Configuration [router - 0013A20041F18F81]

Name: router
Function: 802.15.4 TH
Port: COM4 - 960...N/1/N - AT
MAC: 0013A20041F18F81

MY 16-bit Source Address	0
SH Serial Number High	13A200
SL Serial Number Low	41F18F81
MM MAC Mode	802.15.4 no ACKs [1]
NP Maximum...d Length	6E
RR XBee Retries	0
RN Random Delay Slots	0
NT Node Discover Time	19 x 100 ms
NO Node Discover Options	0 Bitfield
TO Transmit Options	0 Bitfield
C8 802.15.4 Compatibility	0 Bitfield
CE Coordinator Enable	End Device [0]
SC Scan Channels	1FFE Bitfield
SD Scan Duration	4 exponent

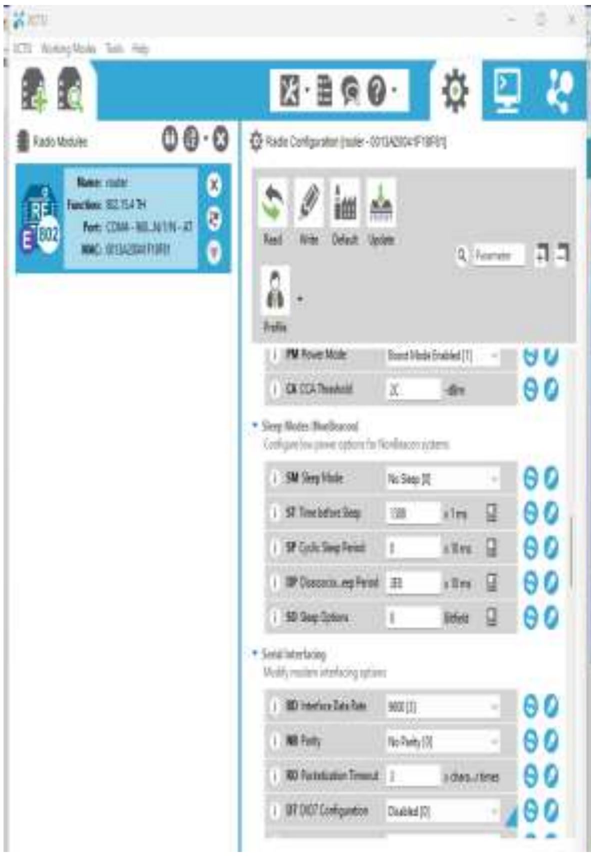

XCTU
XCTU Working Modes Tools Help
Radio Modules
Read Write Default Update Profile
Parameter

Radio Configuration [coordinator - 0013A20041E90F72]

Name: coordinator
Function: 802.15.4 TH
Port: COM7 - 9600/8/N/1/N - AT
MAC: 0013A20041E90F72

RR XBee Retries	0
RN Random Delay Slots	0
NT Node Discover Time	19 x 100 ms
NO Node Discover Options	0 Bitfield
TO Transmit Options	0 Bitfield
C8 802.15.4 Compatibility	0 Bitfield
CE Coordinator Enable	Coordinator [1]
SC Scan Channels	FFFF Bitfield
SD Scan Duration	4 exponent
A1 End Device Association	0 Bitfield
A2 Coordinator Association	0 Bitfield
AI Association Indication	0
EE AES Encryption Enable	Disable [0]
KY AES Encryption Key	
NI Node Identifier	coordinator

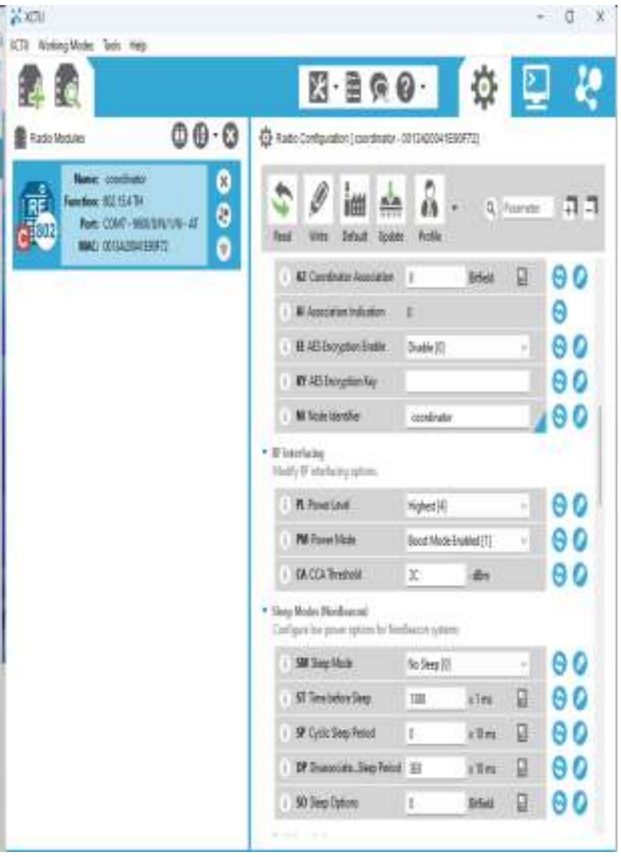
RF Interfacing
Modify RF interfacing options


XCTU
XCTU Working Modes Tools Help
Radio Modules
Read Write Default Update Profile
Parameter

Name: router
Function: 802.15.4 TH
Port: COM4 - 9600/8/N/1/N - AT
MAC: 0013A20041F18F81

Radio Configuration [router - 0013A20041F18F81]

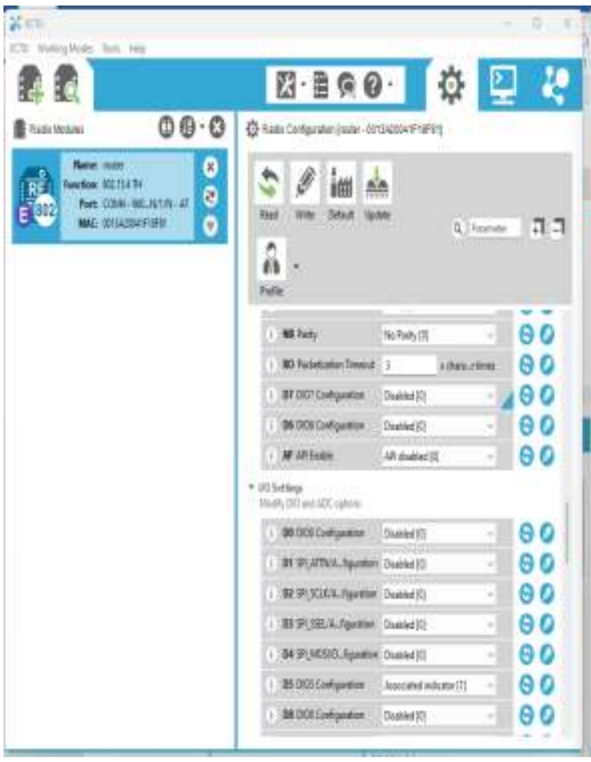
```
PM Power Mode               Boost Mode Enabled [1]
CA CCA Threshold            2C          -dBm
Sleep Modes (NonBeacon)
Configure low power options for NonBeacon systems
SM Sleep Mode               No Sleep [0]
ST Time before Sleep        1388        x 1 ms
SP Cyclic Sleep Period      0           x 10 ms
DP Disassociat...eep Period 3E8         x 10 ms
SO Sleep Options            0           Bitfield
Serial Interfacing
Modify modem interfacing options
BD Interface Data Rate      9600 [3]
NB Parity                   No Parity [0]
RO Packetization Timeout    3           x chara...r times
D7 DIO7 Configuration       Disabled [0]
```


XCTU
XCTU Working Modes Tools Help
Radio Modules
Read Write Default Update Profile
Parameter

Name: coordinator
Function: 802.15.4 TH
Port: COM7 - 9600/8/N/1/N - AT
MAC: 0013A20041E90F72

Radio Configuration [coordinator - 0013A20041E90F72]

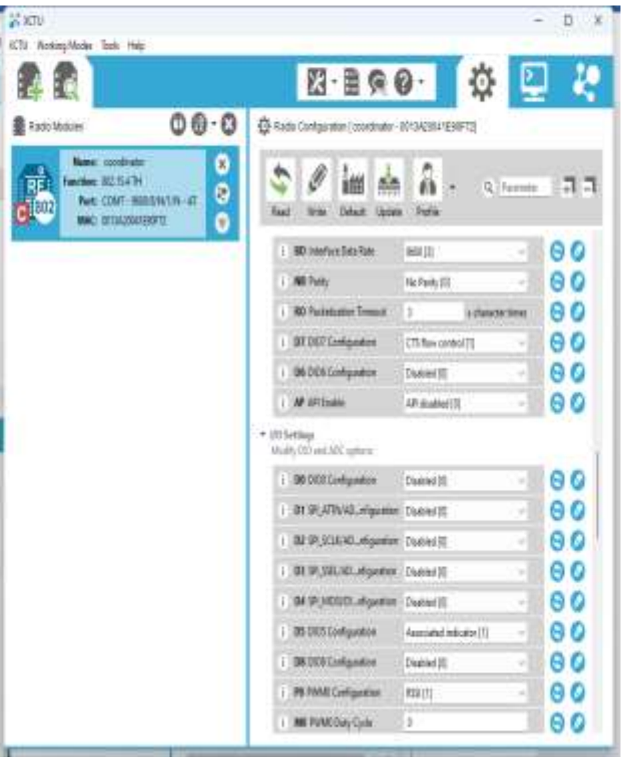
```
A2 Coordinator Association  0           Bitfield
AI Association Indication   0
EE AES Encryption Enable    Disable [0]
KY AES Encryption Key
NI Node Identifier          coordinator
RF Interfacing
Modify RF interfacing options
PL Power Level              Highest [4]
PM Power Mode               Boost Mode Enabled [1]
CA CCA Threshold            2C          -dBm
Sleep Modes (NonBeacon)
Configure low power options for NonBeacon systems
SM Sleep Mode               No Sleep [0]
ST Time before Sleep        1388        x 1 ms
SP Cyclic Sleep Period      0           x 10 ms
DP Disassociat...Sleep Period 3E8       x 10 ms
SO Sleep Options            0           Bitfield
```


XCTU
XCTU Working Modes Tools Help
Radio Modules
Read Write Default Update Profile
Parameter

Name: router
Function: 802.15.4 TH
Port: COM4 - 9600/8/N/1/N - AT
MAC: 0013A20041F18F81

Radio Configuration [router - 0013A20041F18F81]

```
NB Parity                   No Parity [0]
RO Packetization Timeout    3           x chara...r times
D7 DIO7 Configuration       Disabled [0]
D6 DIO6 Configuration       Disabled [0]
AP API Enable               API disabled [0]
I/O Settings
Modify DIO and ADC options
D0 DIO0 Configuration       Disabled [0]
D1 SPI_ATTN/A...figuration  Disabled [0]
D2 SPI_SCLK/A...figuration  Disabled [0]
D3 SPI_SSEL/A...figuration  Disabled [0]
D4 SPI_MOSI/D...figuration  Disabled [0]
D5 DIO5 Configuration       Associated indicator [1]
D8 DIO8 Configuration       Disabled [0]
```


XCTU
XCTU Working Modes Tools Help
Radio Modules
Read Write Default Update Profile
Parameter

Name: coordinator
Function: 802.15.4 TH
Port: COM7 - 9600/8/N/1/N - AT
MAC: 0013A20041E90F72

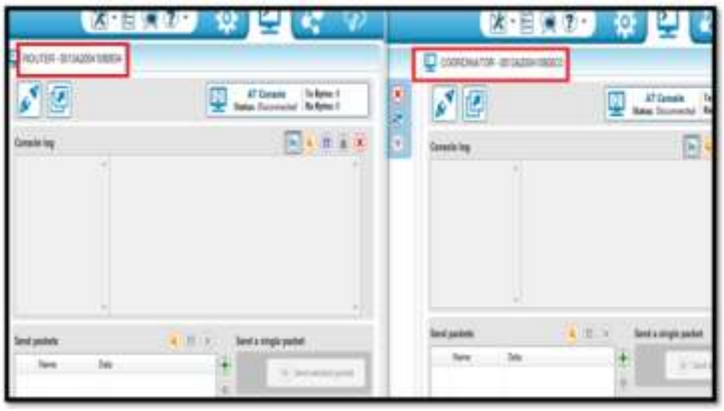
Radio Configuration [coordinator - 0013A20041E90F72]

```
BD Interface Data Rate      9600 [3]
NB Parity                   No Parity [0]
RO Packetization Timeout    3           x character times
D7 DIO7 Configuration       CTS flow control [1]
D6 DIO6 Configuration       Disabled [0]
AP API Enable               API disabled [0]
I/O Settings
Modify DIO and ADC options
D0 DIO0 Configuration       Disabled [0]
D1 SPI_ATTN/AD...nfiguration Disabled [0]
D2 SPI_SCLK/AD...nfiguration Disabled [0]
D3 SPI_SSEL/AD...nfiguration Disabled [0]
D4 SPI_MOSI/DI...nfiguration Disabled [0]
D5 DIO5 Configuration       Associated indicator [1]
D8 DIO8 Configuration       Disabled [0]
P0 PWM0 Configuration       RSSI [1]
M0 PWM0 Duty Cycle          0
```

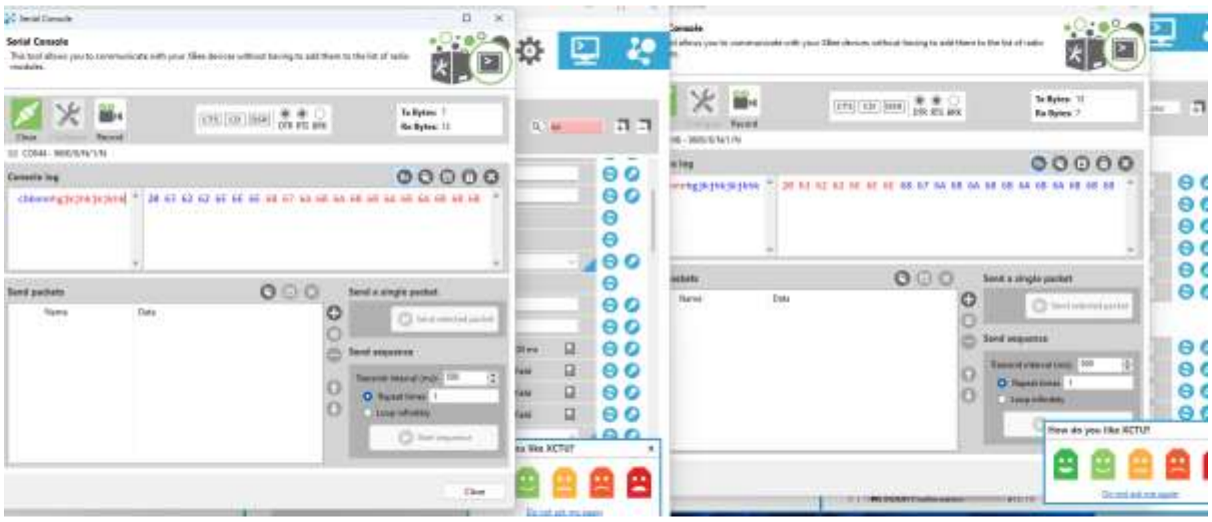
OUTPUT:

1. The new window has a ROUTER and the old window has a COORDINATOR.

- Left side- ROUTER
- Right side- COORDINATOR



2. Click on the Console terminal window of both radio and Serial close button.
3. Now type the message in the Console log window to see received by other radio. The transmitter message in BLUE and Received message in RED.



RESULT:

Thus two ZigBee XBee S2C modules have been configured as coordinator and router and communicate between them has been established between them using XCTU software.

EXPT.NO. 7.a	INTRODUCTION TO RASPBERRY PI PLATFORM
DATE:	

AIM:

To study about Raspberry Pi Pico RP2040 microcontroller board and Thonny IDE.

INTRODUCTION:

Designed by Raspberry Pi, RP2040 features a dual-core Arm Cortex-M0+ processor with 264kB internal RAM and support for upto 16MB of off-chip flash. A wide range of flexible I/O options includes I2C, SPI, and - uniquely - Programmable I/O (PIO). These support endless possible applications for this small and affordable package.

Whether you have a Raspberry Pi Pico or another RP2040-based microcontroller board, everything you need to get started is here. You'll find support for getting started with C/C++ or MicroPython on Raspberry Pi Pico, and links to resources for other boards that use RP2040. There are also links to the technical documentation for both the Raspberry Pi Pico microcontroller board and our RP2040 microcontroller chip.

RP2040 is the debut microcontroller from Raspberry Pi. It brings our signature values of high performance, low cost, and ease of use to the microcontroller space. With a large on-chip memory, symmetric dual-core processor complex, deterministic bus fabric, and rich peripheral set augmented with our unique Programmable I/O (PIO) subsystem, it provides professional users with unrivalled power and flexibility. With detailed documentation, a polished MicroPython port, and a UF2 bootloader in ROM, it has the lowest possible barrier to entry for beginner and hobbyist users. RP2040 is a stateless device, with support for cached execute-in-place from external QSPI memory. This design decision allows you to choose the appropriate density of non-volatile storage for your application, and to benefit from the low pricing of commodity Flash parts.

RP2040 is manufactured on a modern 40nm process node, delivering high performance, low dynamic power consumption, and low leakage, with a variety of low-power modes to support extended-duration operation on battery power.

Key features:

- Dual ARM Cortex-M0+ @ 133MHz
- 264kB on-chip SRAM in six independent banks
- Support for up to 16MB of off-chip Flash memory via dedicated QSPI bus
- DMA controller
- Fully-connected AHB crossbar
- Interpolator and integer divider peripherals
- On-chip programmable LDO to generate core voltage
- 2 on-chip PLLs to generate USB and core clocks
- 30 GPIO pins, 4 of which can be used as analogue inputs

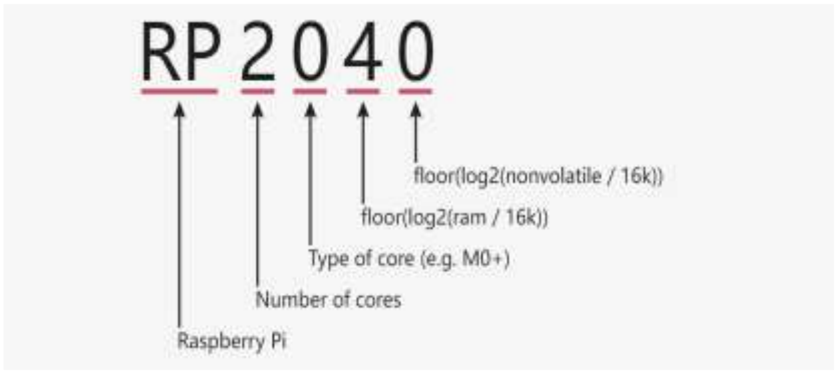
Peripherals:

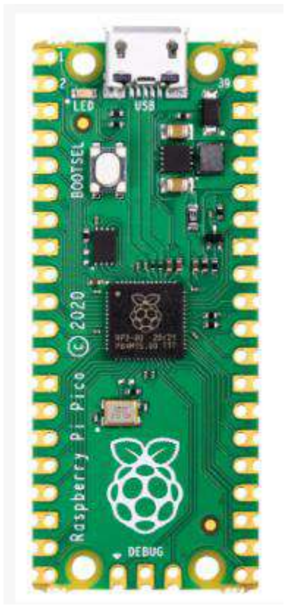
- 2 UARTs
- 2 SPI controller

- 2 I2C controllers
- 16 PWM channels
- USB 1.1 controller and PHY, with host and device support
- 8 PIO state machines

Features of RP2040 Chip — High performance, Low cost, Small package:

The RP2040 features a dual-core Arm Cortex-M0+ processor clocked at 133MHz with 264KB internal SRAM and 2MB internal flash storage and can be programmed in both C/C++ and the beginner-friendly MicroPython.





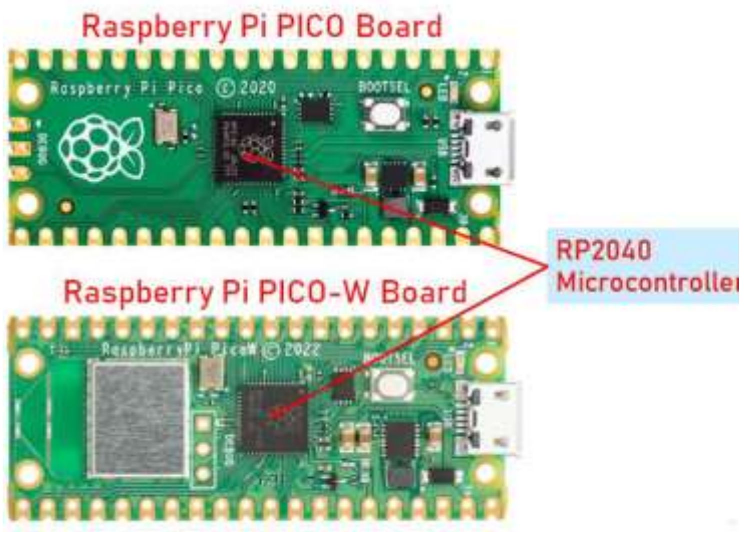
Raspberry pi Pico

Raspberry Pi Pico W and Pico WH

Raspberry Pi Pico W adds on-board single-band 2.4GHz wireless interfaces (802.11n) using the Infineon CYW43439 while retaining the Pico form factor. The on-board 2.4GHz wireless interface has the following features:

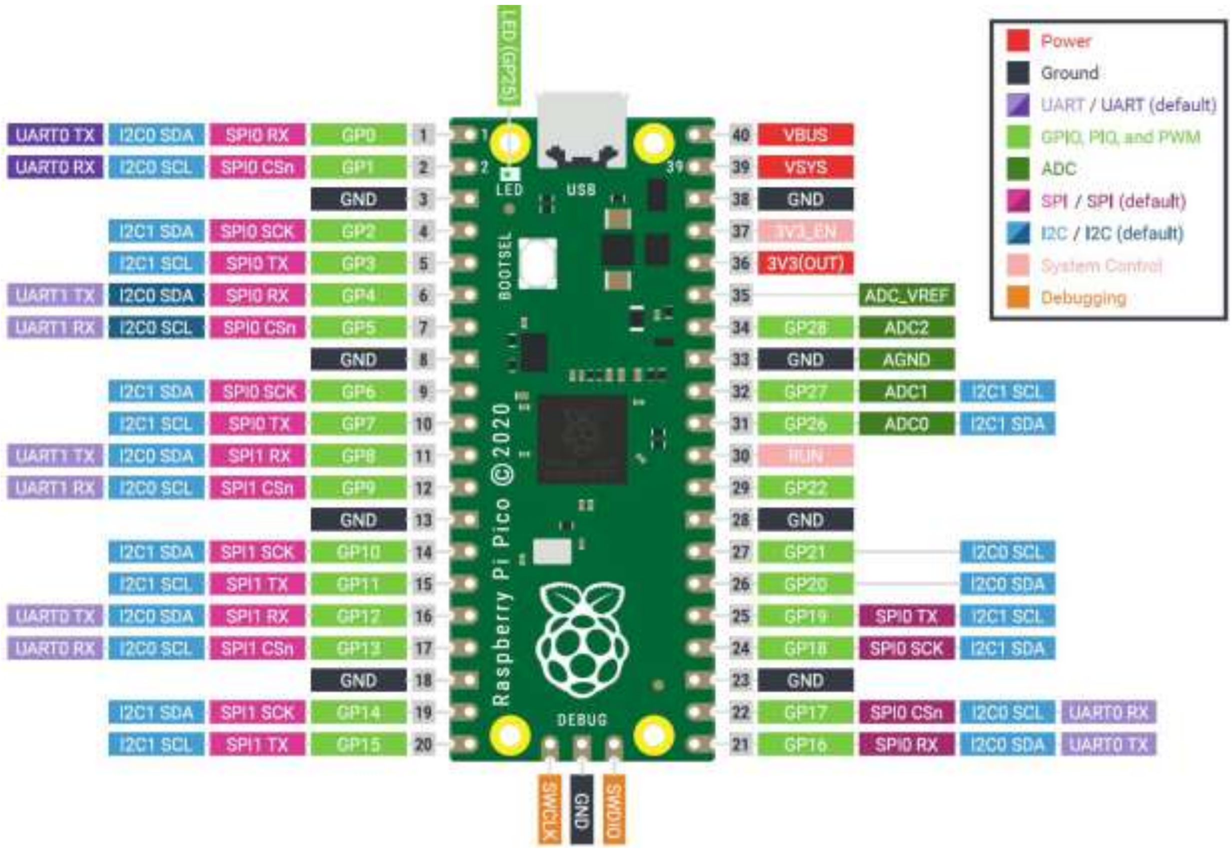
- Wireless (802.11n), single-band (2.4 GHz)
- WPA3
- Soft access point supporting up to four clients

The antenna is an onboard antenna licensed from ABRACON (formerly ProAnt). The wireless interface is connected via SPI to the RP2040 microcontroller. Due to pin limitations, some of the wireless interface pins are shared. The CLK is shared with VSYS monitor, so only when there isn't an SPI transaction in progress can VSYS be read via the ADC. The Infineon CYW43439 DIN/DOUT and IRQ all share one pin on the RP2040. Only when an SPI transaction isn't in progress is it suitable to check for IRQs. The interface typically runs at 33MHz. For best wireless performance, the antenna should be in free space. For instance, putting metal under or close by the antenna can reduce its performance both in terms of gain and bandwidth. Adding grounded metal to the sides of the antenna can improve the antenna's bandwidth.



RASPBERRY PI PICO PINOUT FEATURES:

- Micro-USB B port for power and data (and for reprogramming the Flash)
- 12MHz Crystal Oscillator with two PLLs provide a system clock up to 133MHz, and a fixed 48MHz clock for USB or ADC.
- Code may be executed directly from 2MB of on-board Flash memory through a dedicated SPI
- Raspberry Pi Pico is a microcontroller board based on the Raspberry Pi RP2040 - 32 bit microcontroller chip
- 3-pin ARM Serial Wire Debug (SWD) port.



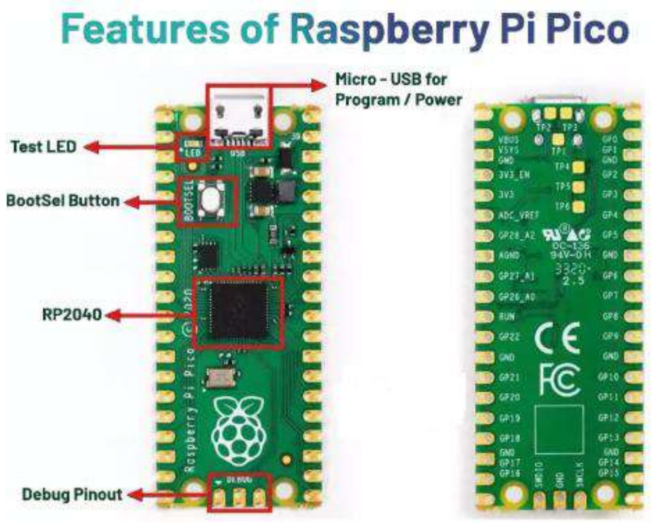
- FourRP2040 GPIO pins are used for internal board functions, these are:

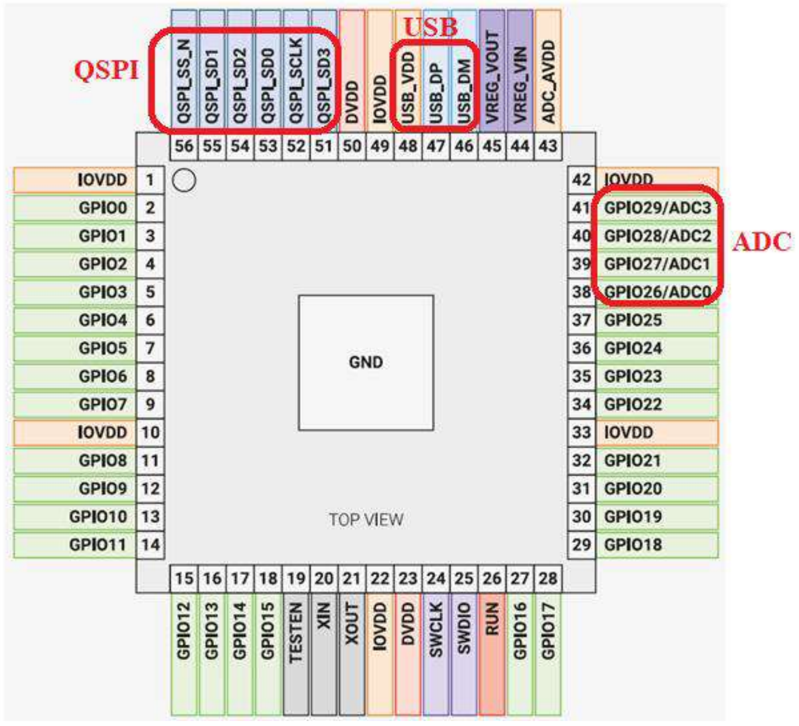
- GPIO29 IP (ADC3) Used to measure VSYS/3
- GPIO25 OP Connected to onboard LED
- GPIO24 IP VBUS sense - high if VBUS is present, else low
- GPIO23 OP Controls the on-board SMPS Power Save pin

RP2040 MICROCONTROLLER BASIC ARCHITECTURE:

Power Supply

At its simplest, RP2040 requires two different voltage supplies, 3.3V (for the IO) and 1.1V (for the chip's digital core).


Features of Raspberry Pi Pico
Micro - USB for Program / Power
Test LED
BootSel Button
RP2040
Debug Pinout



CORTEX-M0 PROCESSOR FEATURES:

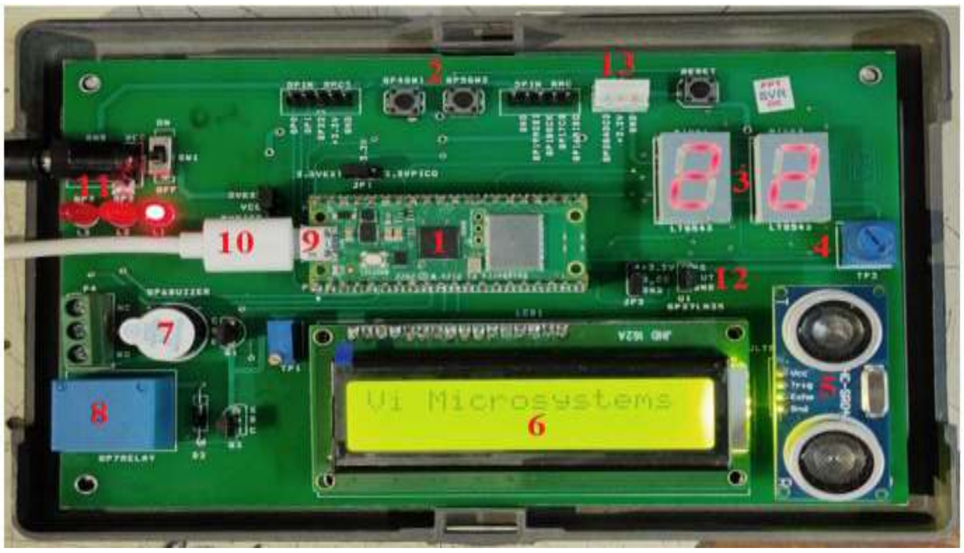
- The ARM Cortex-M0+ processor brings 32-bit power at an 8-bit Cost.
- Has the Smallest Footprint & Lowest Power requirements, consumes just 9µA/MHz on a low-cost 90nm LP process, around one third of the energy of any 8- or 16-bit processor available today,

while delivering significantly higher performance.

- The low-power processor is suitable for a wide variety of applications, including sensors and wearables.
- Single-cycle IO to speed access to GPIO and peripherals.
- Improved debug and trace capability and a 2-stage pipeline.
- The exceptional code density of Cortex-M0+ significantly reduces memory requirements, ideal for use in wearables for healthcare, fitness, and more.
- With its three highly optimized low-power modes, the processor conserves energy to match processing demands.

The Following Onboard peripherals available in the carrier board

1. RP2040 Microcontroller
2. Bush Button
3. Seven Segment display
4. Potentiometer
5. Ultrasonic Sensor
6. 16*2 LCD Display
7. Buzzer
8. Relay
9. USB Connector
10. USB Cable (B-Type)
11. LED
12. Temperature sensor
13. ADC



SOFTWARE REQUIRED & PROGRAMMING LANGUAGE:

Thonny

MicroPython

Raspberry Pi Pico Programming – Overview – GPIO Access

The Raspberry Pi Pico accepts programming with the following programming languages: C/C++, MicroPython, assembly language. Although the Pico by default is set up for use with the powerful and popular C/C++ language, many beginners find it easier to use MicroPython, which is a version of the Python programming language developed specifically for microcontrollers. The Thonny text editor which has been developed specifically for Python programs can be used to run python programs

RESULT:

Thus the basic architecture, pin details of Raspberry Pi Pico RP2040 microcontroller board and Thonny IDE for python programming has been studied.

EXPT.NO 7.b	Python Programming in Raspberry Pi
DATE:	

AIM:

To execute basic python programs in Raspberry Pi Pico board

HARDWARE / SOFTWARE REQUIREMENTS:

1. Raspberry Pi Pico W
2. Thonny IDE

PROCEDURE:

1. Connect Raspberry Pi board to PC using USB-serial communication cable
2. Type the program in Thonny IDE.
3. Button LED – check for LED to glow on pressing the switch
4. Buzzer – check for the buzzer to beep
5. LED – check for LEDs connected to pin 2 and pin 3 to glow
6. POT – connect the potentiometer to the Raspberry Pi board (Vcc-3.3V, Gnd-Gnd, Data in- GPIO 28) and run the program. Check for the potentiometer values displayed in Thonny IDE.

PROGRAM:

1.Button_LED:

```
from machine import Pin
from utime import sleep_ms

button1 = Pin(5, Pin.IN,Pin.PULL_UP)#Internal pull-up
button2 = Pin(4, Pin.IN,Pin.PULL_UP)
led1 = Pin(2, Pin.OUT)
led2 = Pin(3, Pin.OUT) #0 means that the light is currently off

while True:

    if button1.value() == 0:       #key press
        led1.value(1)

    else:
        led1.value(0)

    if button2.value() == 0:       #key press

        led2.value(1)

    else:
        led2.value(0)
```

....

2.Buzzer:

```
from machine import Pin
import time

Buzzer = Pin(6, Pin.OUT)

while True:
   Buzzer.value(1)
        print(" Buzzer ON")
   time.sleep(0.2)
   Buzzer.value(0)
        print(" Buzzer OFF")
   time.sleep(1)
```

....

3.LED:

```
from machine import Pin
from time import *

led2=Pin(2,Pin.OUT)
led3=Pin(3,Pin.OUT)

while True:
   led2.value(1)
   sleep_ms(200)
   led3.value(1)
   led2.value(0)
   sleep_ms(200)
   led3.value(0)
```

....

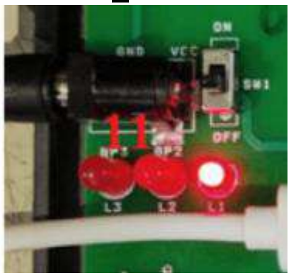
4.POT:

```
from machine import Pin
import utime

POT_Value = machine.ADC(28)
conversion_factor = 3.3/(65535)

while True:
  #print(POT_Value.read_u16() )
  print(POT_Value.read_u16() * conversion_factor)
  utime.sleep(1)
```

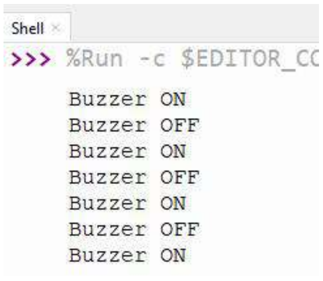
....

OUTPUT:

1.Button_LED:

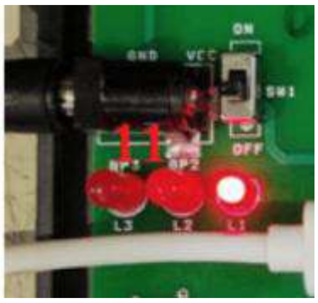


2.Buzzer:

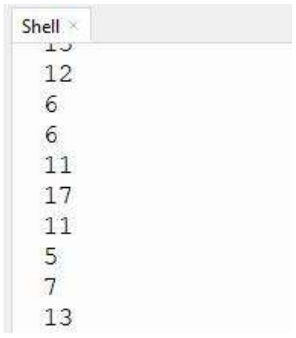


```
Shell
>>> %Run -c $EDITOR_CC

    Buzzer ON
    Buzzer OFF
    Buzzer ON
    Buzzer OFF
    Buzzer ON
    Buzzer OFF
    Buzzer ON
```

3.LED:



4.POT:



```
Shell
  12
  6
  6
  11
  17
  11
  5
  7
  13
```

RESULT:

Thus the python programs to have been executed in Raspberry Pi Pico board.

EXPT.NO:08	
DATE:	**INTERFACING SENSORS WITH RASPBERRY PI**

AIM:

To execute programs for interfacing sensors with Raspberry Pi Pico.

HARDWARE/SOFTWARE REQUIREMENTS:

1. Raspberry Pi Pico W
2. Thonny IDE

PROCEDURE:

1. Connect Raspberry Pi board to PC using USB-serial communication cable
2. Type the program in Thonny IDE.
3. TEMP – connect the temperature sensor LM35 to the Raspberry Pi board (Vcc-3.3V, Gnd-Gnd, Data in- GPIO 28) and run the program. Check for the room temperature values displayed in Thonny IDE.
4. ULTRASONIC – the Ultrasonic distance sensor is HC-SR04 interfaced to pin 14, pin 15 on the Raspberry Pi board. Run the program and check for the distance values displayed in Thonny IDE.
5. LIGHT DETECTOR – connect the Photo detector LM35 to the Raspberry Pi board (Vcc-3.3V, Gnd-Gnd, Data in- GPIO 28) and run the program. In Thonny IDE check for the voltage values corresponding to the light received on the sensor.
6. IR SENSOR – connect the InfraRed Sensor to the Raspberry Pi board (Vcc-3.3V, Gnd-Gnd, Data in- GP28) and run the program. In Thonny IDE check for '1' and '0'values corresponding to open and close of IR receiver.

PROGRAM:

1.TEMP:

```
from machine import Pin
import time
conversion_factor = 3.3/(65536)
adc2= machine.ADC(28)

while True:
    val2 = adc2.read_u16()
    temp = (val2 * conversion_factor)*100
    print("==============================")
    print("temperature: ",temp)
time.sleep(0.8)
```

....

2.ULTRASONIC:

```
from hcsr04 import HCSR04
from time import sleep

sensor = HCSR04(trigger_pin=15, echo_pin=14, echo_timeout_us=10000)

while True:
    distance = sensor.distance_cm()
    D = int(distance)
    print('Distance:', distance, 'cm')
    sleep(0.4)
```

....

3.LIGHT DETECTOR

```
from machine import Pin
import time
import utime
conversion_factor = 3.3/(65536)
adc2 = machine.ADC(28)

while True:
    val2 = adc2.read_u16()
ldr = (val2 * conversion_factor)
    print("==============================")
    print("LDR Voltage : ",ldr)
time.sleep(0.8)
```

....

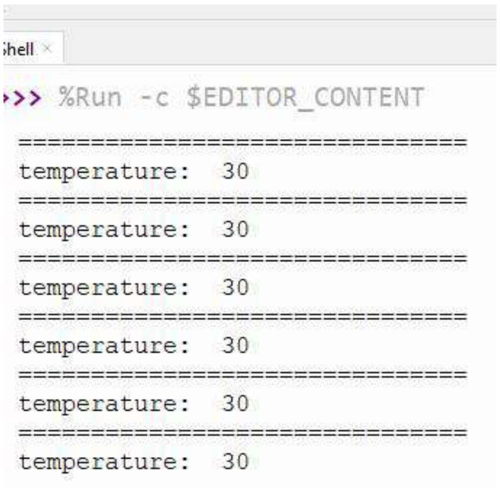
4.IR SENSOR

```
from machine import Pin
import time
ir_pin = Pin(28, Pin.IN)
while True:
    print("========================")
ir_data=ir_pin.value()
    print("IR_SENSOR: ",ir_data)
time.sleep(0.5)
```

....

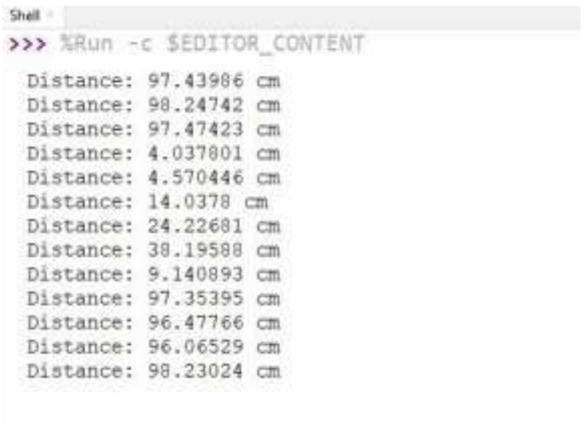
OUTPUT:

1.TEMP:



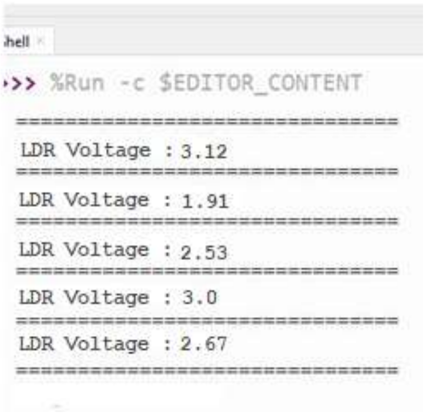
```
>>> %Run -c $EDITOR_CONTENT
 ================================
 temperature:  30
 ================================
 temperature:  30
 ================================
 temperature:  30
 ================================
 temperature:  30
 ================================
 temperature:  30
 ================================
 temperature:  30
```

2. ULTRASONIC:



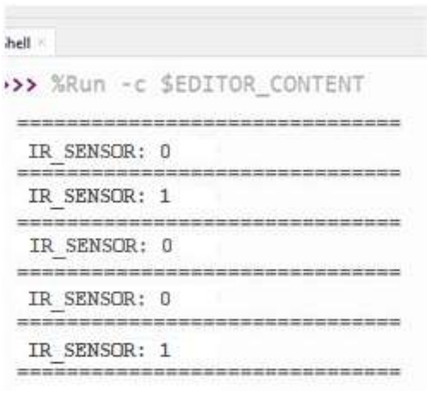
```
>>> %Run -c $EDITOR_CONTENT
 Distance: 97.43986 cm
 Distance: 98.24742 cm
 Distance: 97.47423 cm
 Distance: 4.037801 cm
 Distance: 4.570446 cm
 Distance: 14.0378 cm
 Distance: 24.22681 cm
 Distance: 38.19588 cm
 Distance: 9.140893 cm
 Distance: 97.35395 cm
 Distance: 96.47766 cm
 Distance: 96.06529 cm
 Distance: 98.23024 cm
```

3. LIGHT DETECTOR:



```
>>> %Run -c $EDITOR_CONTENT
 ================================
 LDR Voltage : 3.12
 ================================
 LDR Voltage : 1.91
 ================================
 LDR Voltage : 2.53
 ================================
 LDR Voltage : 3.0
 ================================
 LDR Voltage : 2.67
 ================================
```

4. IR SENSOR:



```
Shell
>>> %Run -c $EDITOR_CONTENT
================================
 IR_SENSOR: 0
================================
 IR_SENSOR: 1
================================
 IR_SENSOR: 0
================================
 IR_SENSOR: 0
================================
 IR_SENSOR: 1
================================
```

RESULT:

Thus the python programs for interfacing sensors with Raspberry Pi Pico have been executed.

EXPT.NO:09	COMMUNICATE BETWEEN ARDUINO AND RASPBERRY PI USING ANY WIRELESS MEDIUM
DATE:	

AIM:

To Communicate between Arduino and Raspberry PI usingwireless medium-ESP01(WIFI Module)

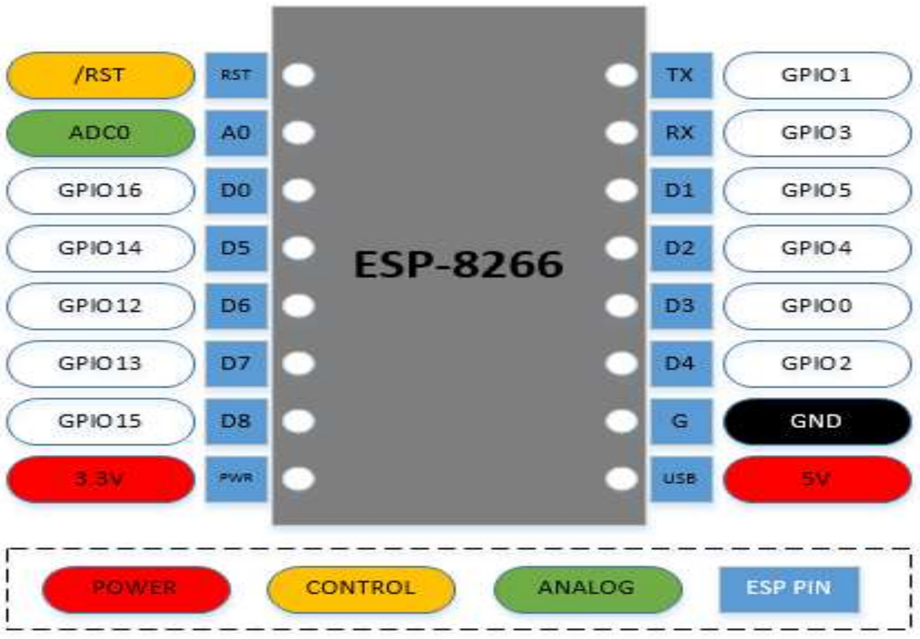
HARDWARE/SOFTWARE REQUIREMENTS:

1. Raspberry PI
2. ESP01
3. Arduino UNO

PROCEDURE:

1. Connect pin TX of ESP0I to RX (set as 3) of Arduino UNO.
2. Connect pin RX of ESP0I to TX (set as 2) of Arduino UNO.
3. Connect pin ENABLE of ESP01 to 3.3v(high).
4. Connect pin GND of ESP01 to GND in Arduino UNO.
5. Connect VCC of ESP01 to 3.3v in Arduino UNO.
6. Create a Wifi hotspot to get data connectivity for ESP01.
7. Connect Raspberry pi and Arduino to PC using USB-serial communication cable.
8. Type and run the program.
9. After execution copy the IP address from Thonny IDE (Raspberry pi environment) to Arduino IDE program.
10. Run the Arduino program.
11. Check the output received in Thonny IDE (Raspberry pi environment) for random number generated using Arduino program.

DIAGRAM:



PROGRAM:

1. Program to send data from Arduino

```
#include <SoftwareSerial.h>
#define RX 3
#define TX 2
String AP = "A54";       // AP NAME
String PASS = "12345678"; // AP PASSWORD
String HOST = "192.168.253.207";
String PORT = "1234";
intcountTrueCommand;
intcountTimeCommand;
boolean found = false;
intvalSensor = 1;
SoftwareSerial esp8266(RX,TX);



void setup() {
Serial.begin(9600);
  esp8266.begin(115200);
sendCommand("AT",5,"OK");
sendCommand("AT+CWMODE=1",5,"OK"); //AT+CWMODE=1 TO SET MODULE AS
STATION
sendCommand("AT+CWJAP=\""+ AP +"\",\""+ PASS +"\"",20,"OK"); //CONNECT TO
ARDUINO
 String ip = "AT+CIFSR"; //CONNECT TO IP ADDRESS
sendCommand(ip,5,"OK");
  while(esp8266.available())
{
      // The esp has data so display its output to the serial window
      char c = esp8266.read(); // read the next character.
Serial.write(c);
      break;
}*/
}

void loop() {
valSensor = getSensorData();
 String getData = String(valSensor);
sendCommand("AT+CIPMUX=1",5,"OK"); // TO ENABLE MULTIPLE CONNECTIONS
sendCommand("AT+CIPSTART=0,\"UDP\",\""+ HOST +"\","+ PORT,15,"OK");
sendCommand("AT+CIPSEND=0," +String(getData.length()),4,">");
 esp8266.println(getData);delay(1500);countTrueCommand++;
sendCommand("AT+CIPCLOSE=0",5,"OK");
}

intgetSensorData(){
  return random(1000); // Replace with your own sensor code
}
```

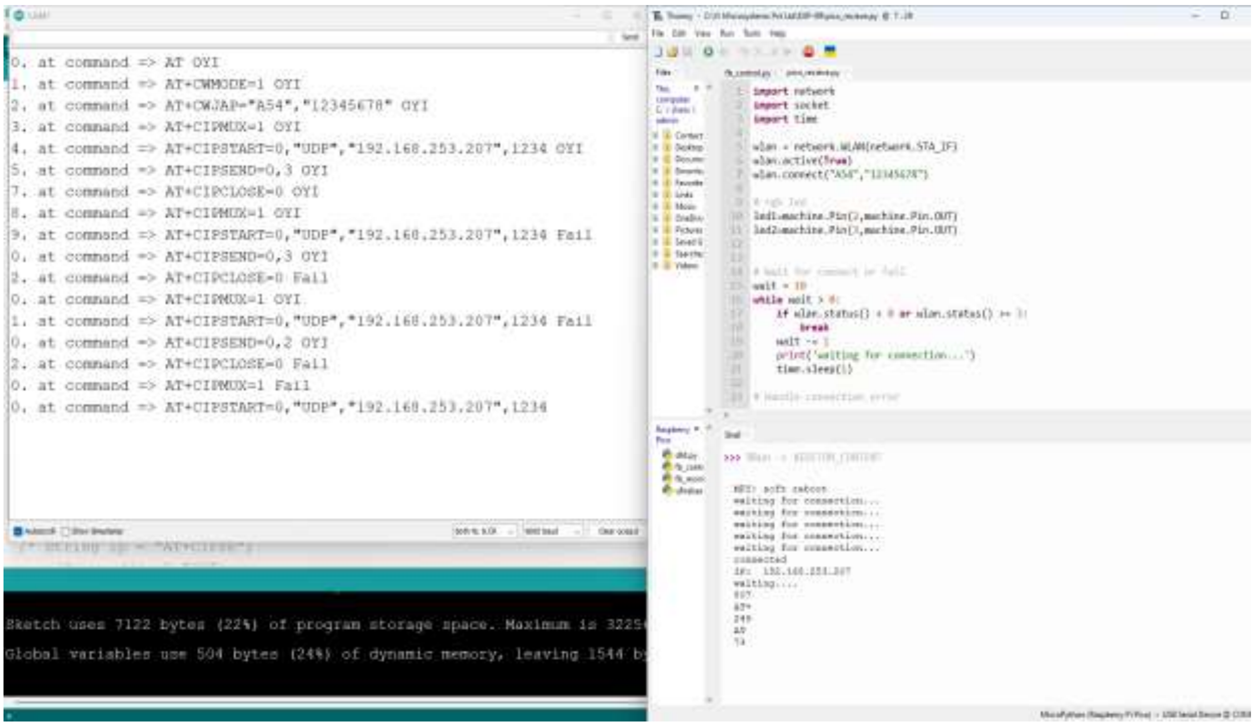
```
void sendCommand(String command, intmaxTime, char readReplay[]) {
Serial.print(countTrueCommand);
Serial.print(". at command => ");
Serial.print(command);
Serial.print(" ");
  while(countTimeCommand< (maxTime*1))
  {
    esp8266.println(command);//at+cipsend
    if(esp8266.find(readReplay))//ok
    {
      found = true;
      break;
    }

countTimeCommand++;
  }

  if(found == true)
  {
Serial.println("OYI");
countTrueCommand++;
countTimeCommand = 0;
  }

  if(found == false)
  {
Serial.println("Fail");
countTrueCommand = 0;
countTimeCommand = 0;
  }

  found = false;
 }
```

2. Program to receive data using Raspberry Pi:

```
import network
import socket
import time

wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect("A54","12345678")

# rgb led
led1=machine.Pin(2,machine.Pin.OUT)
led2=machine.Pin(3,machine.Pin.OUT)
```

```
# Wait for connect or fail
wait = 10
while wait > 0:
   if wlan.status() < 0 or wlan.status() >= 3:
      break
   wait -= 1
   print('waiting for connection...')
time.sleep(1)

# Handle connection error
if wlan.status() != 3:
   raise RuntimeError('wifi connection failed')
else:
   print('connected')
ip=wlan.ifconfig()[0]
   print('IP: ', ip)


localIP  = wlan.ifconfig()[0]
localPort   = 1234
bufferSize  = 1024

# Create a datagram socket
UDPServerSocket = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)


# Bind to address and ip
UDPServerSocket.bind((localIP, localPort))
print('waiting....')

while True:
bytesAddressPair = UDPServerSocket.recvfrom(bufferSize)
   message = bytesAddressPair[0]
   address = bytesAddressPair[1]
   m=str(message.decode())
   m=m.strip()
   print(m)
```

OUTPUT:



```
0. at command => AT OYI
1. at command => AT+CWMODE=1 OYI
2. at command => AT+CWJAP="A54","12345678" OYI
3. at command => AT+CIPMUX=1 OYI
4. at command => AT+CIPSTART=0,"UDP","192.168.253.207",1234 OYI
5. at command => AT+CIPSEND=0,3 OYI
7. at command => AT+CIPCLOSE=0 OYI
8. at command => AT+CIPMUX=1 OYI
9. at command => AT+CIPSTART=0,"UDP","192.168.253.207",1234 Fail
0. at command => AT+CIPSEND=0,3 OYI
2. at command => AT+CIPCLOSE=0 Fail
0. at command => AT+CIPMUX=1 OYI
1. at command => AT+CIPSTART=0,"UDP","192.168.253.207",1234 Fail
0. at command => AT+CIPSEND=0,2 OYI
2. at command => AT+CIPCLOSE=0 Fail
0. at command => AT+CIPMUX=1 Fail
0. at command => AT+CIPSTART=0,"UDP","192.168.253.207",1234
```

```
Sketch uses 7122 bytes (22%) of program storage space. Maximum is 3225
Global variables use 504 bytes (24%) of dynamic memory, leaving 1544 b
```

```python
import network
import socket
import time

wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect("A54","12345678")

# rgb led
led1=machine.Pin(2,machine.Pin.OUT)
led2=machine.Pin(3,machine.Pin.OUT)


# Wait for connect or fail
wait = 10
while wait > 0:
    if wlan.status() < 0 or wlan.status() >= 3:
        break
    wait -= 1
    print('waiting for connection...')
    time.sleep(1)

# Handle connection error
```

```
>>> [illegible]

MPY: soft reboot
waiting for connection...
waiting for connection...
waiting for connection...
waiting for connection...
waiting for connection...
connected
ip:  192.168.253.207
waiting....
[illegible]
AT+
[illegible]
AT
[illegible]
```

RESULT:

Thus the communication between Arduino and Raspberry PI using wireless medium-ESP01(WIFI Module) has been done.

EXPT.NO:10	SETUP A CLOUD PLATFORM TO LOG THE DATA
DATE:	

AIM:

To Setup a cloud platform to log the datafrom Arduino UNO using ESP01(WIFI Module).

HARDWARE/SOFTWARE REQUIREMENTS:

1. ESP01
2. Arduino UNO
3. Think speak cloud

PROCEDURE:

1. Connect pin TX of ESP0I to RX (set as 3) of Arduino UNO.
2. Connect pin RX of ESP0I to TX (set as 2) of Arduino UNO.
3. Connect pin ENABLE of ESP01 to 3.3v(high).
4. Connect pin GND of ESP01 to GND in Arduino UNO.
5. Connect VCC of ESP01 to 3.3v in Arduino UNO.
6. Create a Wifi hotspot to get data connectivity for ESP01.
7. Connect Arduino UNO to PC using USB-serial communication cable.
8. Create an account in think speak cloud and then create a new channel.
9. Give a name to the channel and select the field(s) by tick mark and save.
10. Copy the write key from API key and paste in the program in string API section.
11. Type and run the program.

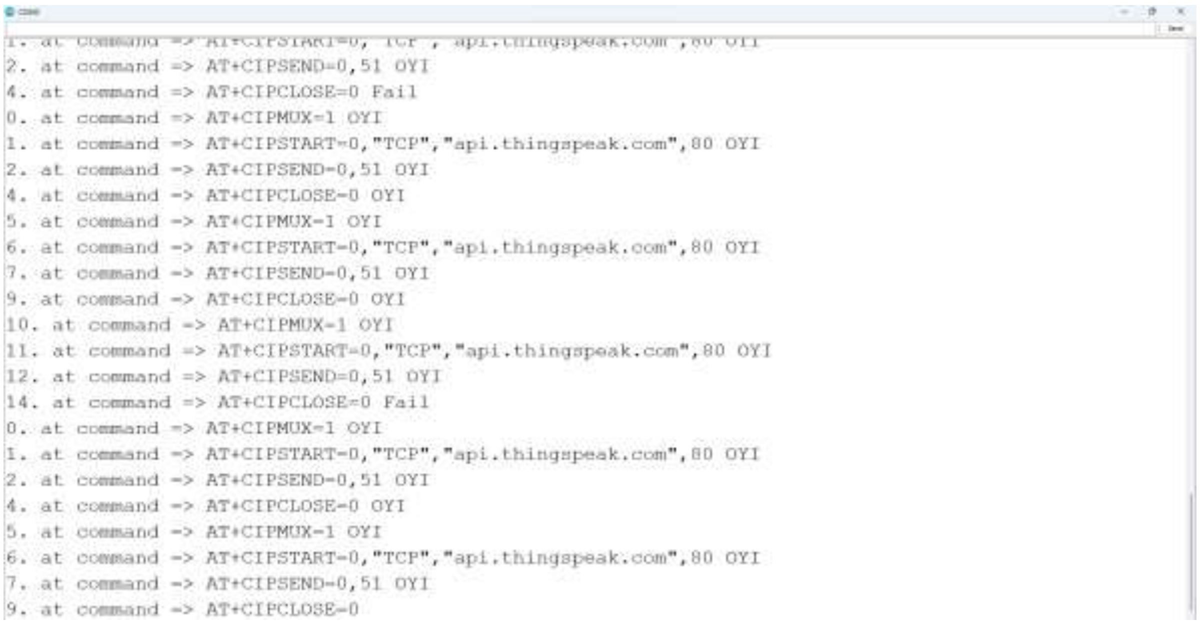
PROGRAM:

```
#include <SoftwareSerial.h>
#define RX 3
#define TX 2
String AP = "A54";          // AP NAME
String PASS = "0123456789";    // AP PASSWORD
String API = "N2UXO5PAQ1BQWD9S";   // Write API KEY
String HOST = "api.thingspeak.com";
String PORT = "80";
String field = "field1";
intcountTrueCommand;
intcountTimeCommand;
boolean found = false;
intvalSensor = 1;
SoftwareSerial esp8266(RX,TX);


void setup() {
Serial.begin(9600);
  esp8266.begin(115200);
```

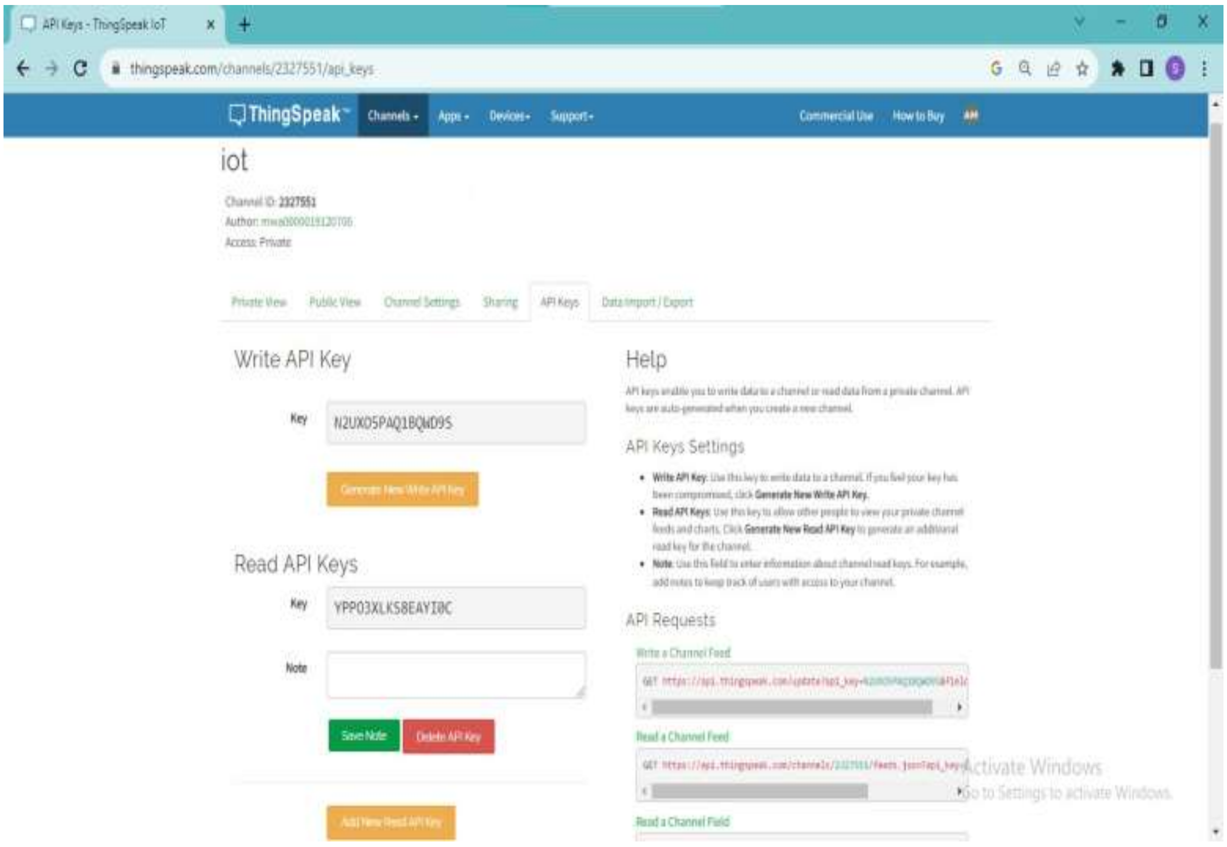
```
sendCommand("AT",5,"OK");
sendCommand("AT+CWMODE=1",5,"OK");
sendCommand("AT+CWJAP=\""+ AP +"\",\""+ PASS +"\"",20,"OK");
}

void loop() {
valSensor = getSensorData();
 String getData = "GET /update?api_key="+ API +"&"+ field +"="+String(valSensor);
sendCommand("AT+CIPMUX=1",5,"OK");
sendCommand("AT+CIPSTART=0,\"TCP\",\""+ HOST +"\","+ PORT,15,"OK");
sendCommand("AT+CIPSEND=0," +String(getData.length()+4),4,">");
 esp8266.println(getData);delay(1500);countTrueCommand++;
sendCommand("AT+CIPCLOSE=0",5,"OK");
}

intgetSensorData(){
  return random(1000); // Replace with your own sensor code
}

void sendCommand(String command, intmaxTime, char readReplay[]) {
Serial.print(countTrueCommand);
Serial.print(". at command => ");
Serial.print(command);
Serial.print(" ");
  while(countTimeCommand< (maxTime*1))
  {
    esp8266.println(command);//at+cipsend
    if(esp8266.find(readReplay))//ok
    {
      found = true;
      break;
    }

countTimeCommand++;
```

```
  }

  if(found == true)
  {
Serial.println("OYI");
countTrueCommand++;
countTimeCommand = 0;
  }

  if(found == false)
  {
Serial.println("Fail");
countTrueCommand = 0;
countTimeCommand = 0;
  }
  found = false;
 }
```

OUTPUT:



```
1. at command => AT+CIPSTART=0,"TCP","api.thingspeak.com",80 OYI
2. at command => AT+CIPSEND=0,51 OYI
4. at command => AT+CIPCLOSE=0 Fail
0. at command => AT+CIPMUX=1 OYI
1. at command => AT+CIPSTART=0,"TCP","api.thingspeak.com",80 OYI
2. at command => AT+CIPSEND=0,51 OYI
4. at command => AT+CIPCLOSE=0 OYI
5. at command => AT+CIPMUX=1 OYI
6. at command => AT+CIPSTART=0,"TCP","api.thingspeak.com",80 OYI
7. at command => AT+CIPSEND=0,51 OYI
9. at command => AT+CIPCLOSE=0 OYI
10. at command => AT+CIPMUX=1 OYI
11. at command => AT+CIPSTART=0,"TCP","api.thingspeak.com",80 OYI
12. at command => AT+CIPSEND=0,51 OYI
14. at command => AT+CIPCLOSE=0 Fail
0. at command => AT+CIPMUX=1 OYI
1. at command => AT+CIPSTART=0,"TCP","api.thingspeak.com",80 OYI
2. at command => AT+CIPSEND=0,51 OYI
4. at command => AT+CIPCLOSE=0 OYI
5. at command => AT+CIPMUX=1 OYI
6. at command => AT+CIPSTART=0,"TCP","api.thingspeak.com",80 OYI
7. at command => AT+CIPSEND=0,51 OYI
9. at command => AT+CIPCLOSE=0
```



API Keys - ThingSpeak IoT

thingspeak.com/channels/2327551/api_keys

iot

Channel ID: 2327551
Author: mwa0000019120706
Access: Private

Write API Key

Key: N2UX05PAQ1BQWD9S

Read API Keys

Key: YPP03XLKS8EAYI0C

Note:

Help

API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.

API Keys Settings

- **Write API Key**: Use this key to write data to a channel. If you feel your key has been compromised, click **Generate New Write API Key**.
- **Read API Keys**: Use this key to allow other people to view your private channel feeds and charts. Click **Generate New Read API Key** to generate an additional read key for the channel.
- **Note**: Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.

API Requests

Write a Channel Feed

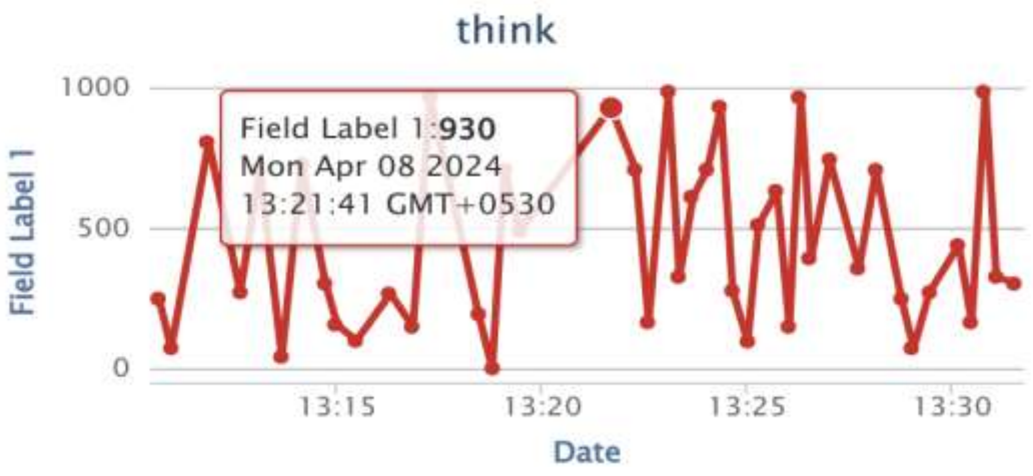
```
GET https://api.thingspeak.com/update?api_key=N2UX05PAQ1BQWD9S&field
```

Read a Channel Feed

```
GET https://api.thingspeak.com/channels/2327551/feeds.json?api_key=
```

Read a Channel Field

Activate Windows
Go to Settings to activate Windows.



RESULT:

Thus, a cloud platform has been set to log the data from Arduino UNO using ESP01(WIFI Module).

EXPT.NO:11	LOG DATA USING RASPBERRY PI AND UPLOAD TO THE CLOUD PLATFORM
DATE:	

AIM:

To Log Data using Raspberry Pi and upload to the cloud platform (Fire base cloud).

HARDWARE/SOFTWARE REQUIREMENTS:

1. Raspberry pi
2. Fire base cloud

PROCEDURE:

1. Login to a firebase cloud and build a real time database.
2. Copy the urlfrom the cloud and paste in the program for both control and monitor.
3. Run the monitor program to view data in cloud (a random value).
4. Similarly run the control program to set keys value as controls -> led as on and off in the cloud.
5. The led in the Raspberry Pico W can found on or off depending on the control values given in the cloud.

PROGRAM:

1.Program to monitor Raspberry pi through firebase cloud

```
from machine import Pin, I2C, ADC

import time

import utime

import network

import urequests

import _thread

import ufirebase as firebase

adc2 = machine.ADC(28)

adc=""

firebase.setURL("https://project-2731037790370440547-default-rtdb.firebaseio.com")

ssid = 'Xperia_6305'

password ='0123456789'

wlan = network.WLAN(network.STA_IF)

wlan.active(True)

wlan.connect(ssid, password)

wait = 10

while wait > 0:

    if wlan.status() < 0 or wlan.status() >= 3:

        break
    wait -= 1
    print('waiting for connection...')
time.sleep(1)
 # Handle connection error
if wlan.status() != 3:
    raise RuntimeError('network connection failed')
time.sleep(2)
 else:
    print('connected')
    status = wlan.ifconfig()
    print( 'ip = ' + status[0] )
time.sleep(2)
    #lcd.clear()
while True:
time.sleep(3)
    print("put")
    val1 = adc2.read_u16()
adc = str(val1)
firebase.put("adc", adc, bg=0, id=3)
```

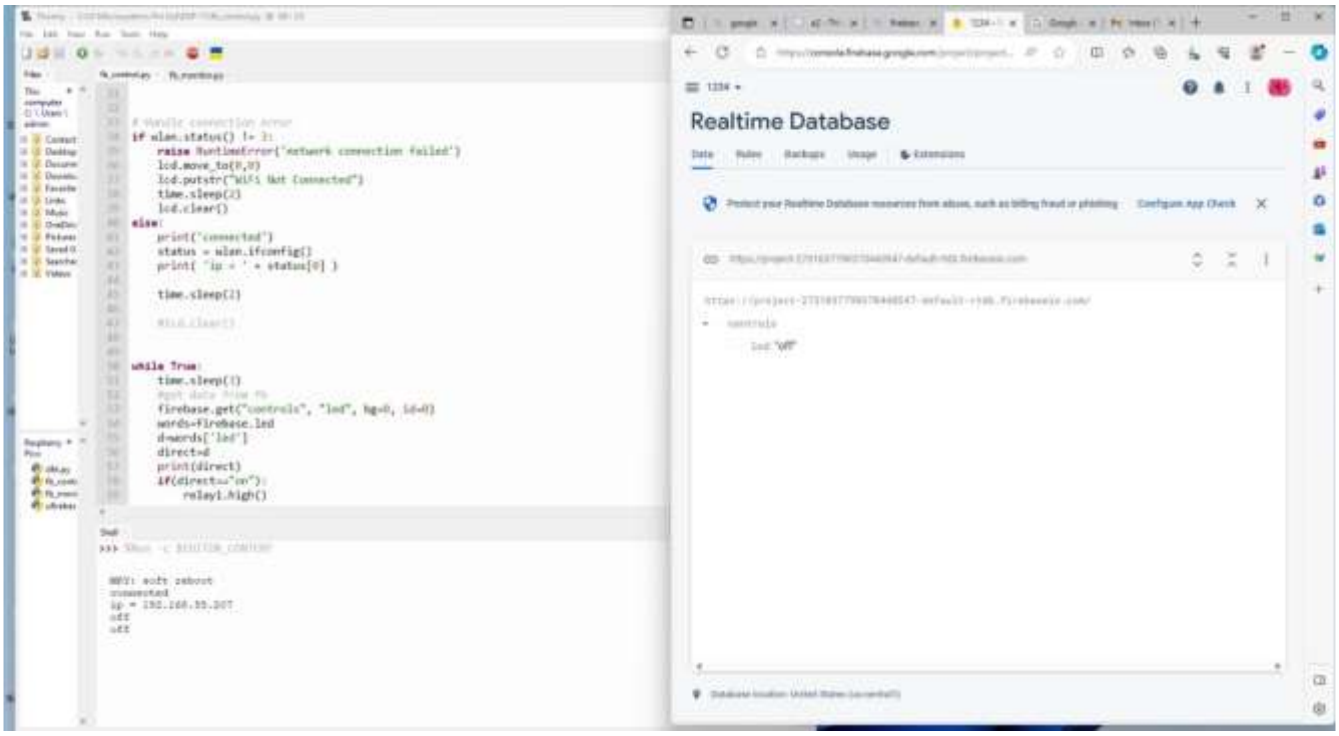
2.Program to control Raspberry pi from firebase cloud

```
from machine import Pin, ADC
import time
import utime
import network
import urequests
import _thread
import ufirebase as firebase
firebase.setURL("https://project-2731037790370440547-default-rtdb.firebaseio.com")
relay1=Pin(2,Pin.OUT) #7
relay1.low()
ssid = 'Xperia_6305'
```

```
password = '0123456789'
wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect(ssid, password)
wait = 10
while wait > 0:
    if wlan.status() < 0 or wlan.status() >= 3:
        break
    wait -= 1
    print('waiting for connection...')
time.sleep(1)
  # Handle connection error
if wlan.status() != 3:
    raise RuntimeError('network connection failed')
lcd.move_to(0,0)
lcd.putstr("WiFi Not Connected")
time.sleep(2)
lcd.clear()
else:
    print('connected')
    status = wlan.ifconfig()
    print( 'ip = ' + status[0] )
time.sleep(2)
    #lcd.clear()
while True:
time.sleep(3)
    #get data from fb
firebase.get("controls", "led", bg=0, id=0)
    words=firebase.led
    d=words['led']
    direct=d
    print(direct)
```

```
if(direct=="on"):
   relay1.high()
if(direct=="off"):
   relay1.low()
```

OUTPUT:
Control:

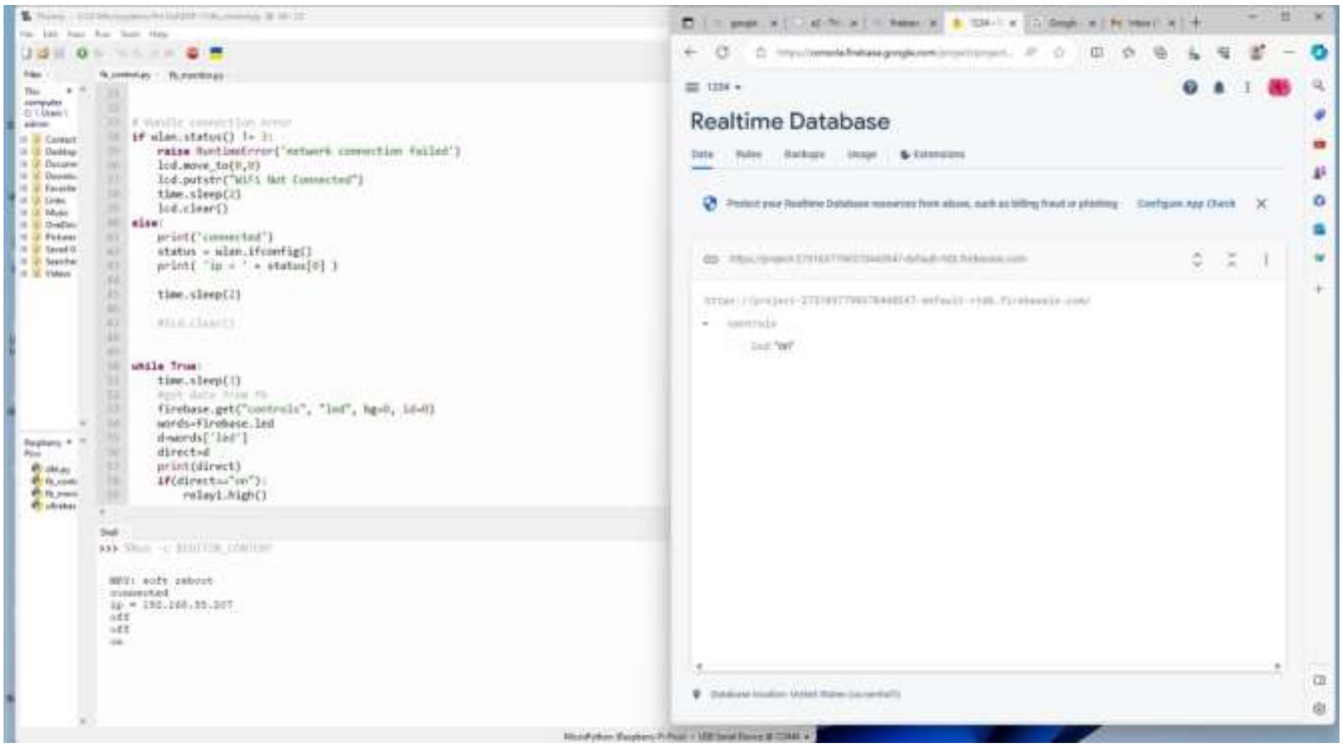


```
# Handle connection error
if wlan.status() != 3:
    raise RuntimeError('network connection failed')
    lcd.move_to(0,0)
    lcd.putstr("WiFi Not Connected")
    time.sleep(2)
    lcd.clear()
else:
    print('connected')
    status = wlan.ifconfig()
    print( 'ip = ' + status[0] )

    time.sleep(2)

    #lcd.clear()


while True:
    time.sleep(1)
    #get data from fb
    firebase.get("controls", "led", bg=0, id=0)
    words=firebase.led
    d=words['led']
    direct=d
    print(direct)
    if(direct=="on"):
        relay1.high()
```

```
>>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
connected
ip = 192.168.55.207
off
off
```

Realtime Database

```
https://project-2731837799378448547-default-rtdb.firebaseio.com/
- controls
    led: "off"
```



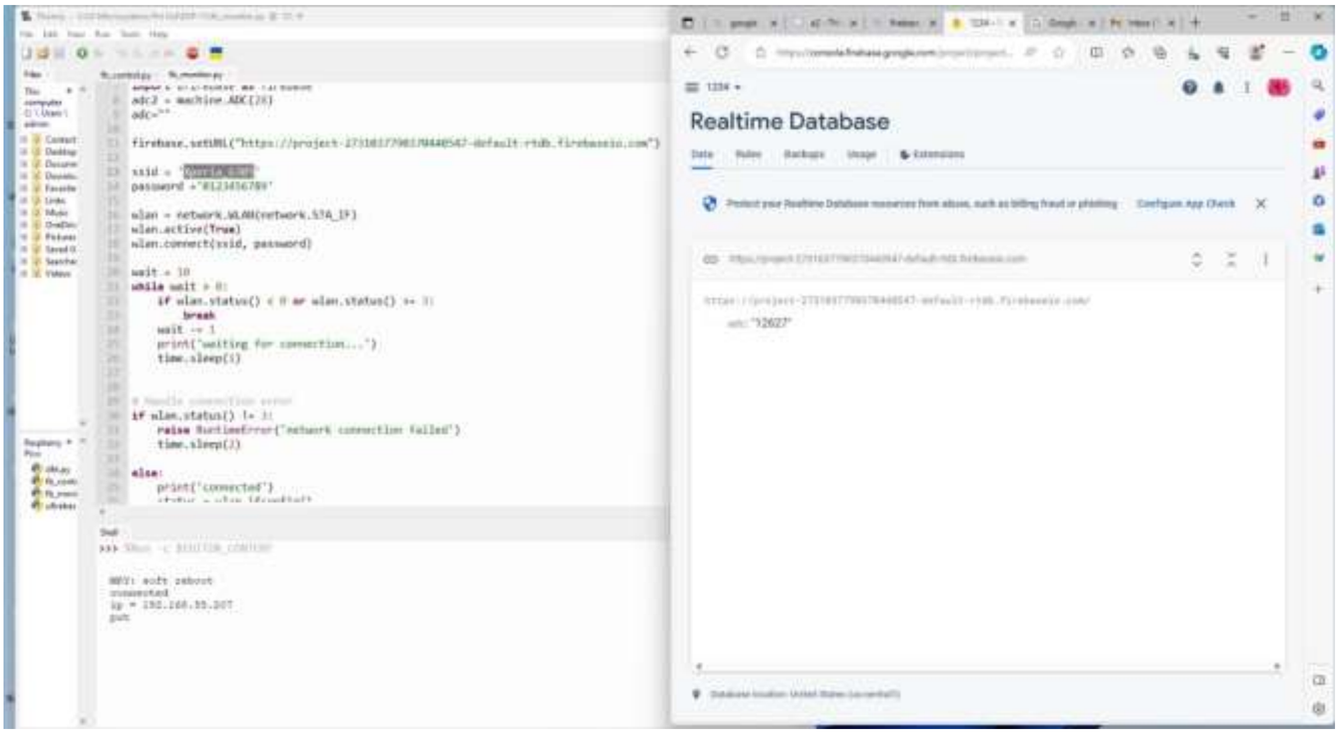
```
# Handle connection error
if wlan.status() != 3:
    raise RuntimeError('network connection failed')
    lcd.move_to(0,0)
    lcd.putstr("WiFi Not Connected")
    time.sleep(2)
    lcd.clear()
else:
    print('connected')
    status = wlan.ifconfig()
    print( 'ip = ' + status[0] )

    time.sleep(2)

    #lcd.clear()


while True:
    time.sleep(1)
    #get data from fb
    firebase.get("controls", "led", bg=0, id=0)
    words=firebase.led
    d=words['led']
    direct=d
    print(direct)
    if(direct=="on"):
        relay1.high()
```

```
>>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
connected
ip = 192.168.55.207
off
off
on
```

Realtime Database

```
https://project-2731837799378448547-default-rtdb.firebaseio.com/
- controls
    led: "on"
```

OUTPUT:
Monitor



```
adc2 = machine.ADC(28)
adc=""

firebase.setURL("https://project-2731837798178440547-default-rtdb.firebaseio.com")

ssid = '[illegible]'
password ='0123456789'

wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect(ssid, password)

wait = 10
while wait > 0:
    if wlan.status() < 0 or wlan.status() >= 3:
        break
    wait -= 1
    print('waiting for connection...')
    time.sleep(1)


# Handle connection error
if wlan.status() != 3:
    raise RuntimeError('network connection failed')
    time.sleep(2)

else:
    print('connected')
```

```
>>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
connected
ip = 192.168.55.207
put
```

Realtime Database

```
https://project-2731837798178440547-default-rtdb.firebaseio.com/
    adc: "12627"
```

Data
Rules
Backups
Usage
Extensions
Protect your Realtime Database resources from abuse, such as billing fraud or phishing
Configure App Check

RESULT:

Thus the program to Log Data using Raspberry Pi and uploading it in to the cloud platform (Fire base cloud) has been done.

EXPT.NO12	DESIGN AN IOT BASED SYSTEM
DATE:	

AIM:
To Design an IOT based system (Home Automation).

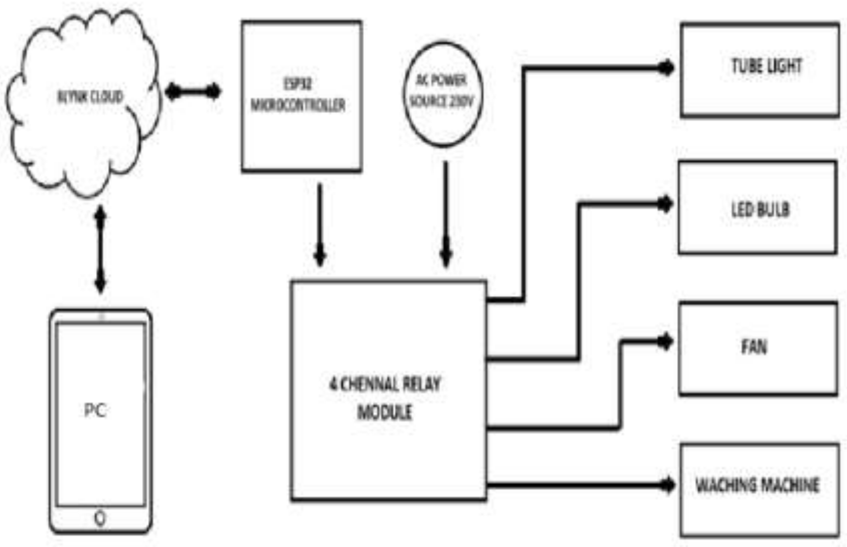
HARDWARE/SOFTWARE REQUIREMENTS:

1. Arduino board
2. Blynk cloud

PROCEDURE:

1. Login to Blynkcloud and to build a home automation system.
2. Select and install ESP32 by expressif system in board manager.
3. In tools select ESP32 Arduino-> DOITESP32 DEVKIT V1.
4. To include the Blynk_authentication token and Blynk_device_name, Blynk_template_id., go to the "Device info" and copy and paste the Blynkauth token andBlynk_device_name, and Blynk_template_idto the program, verify and upload it.
5. Select the widget that need to be controlled in cloud dashboard.

BLOCK DIAGRAM:

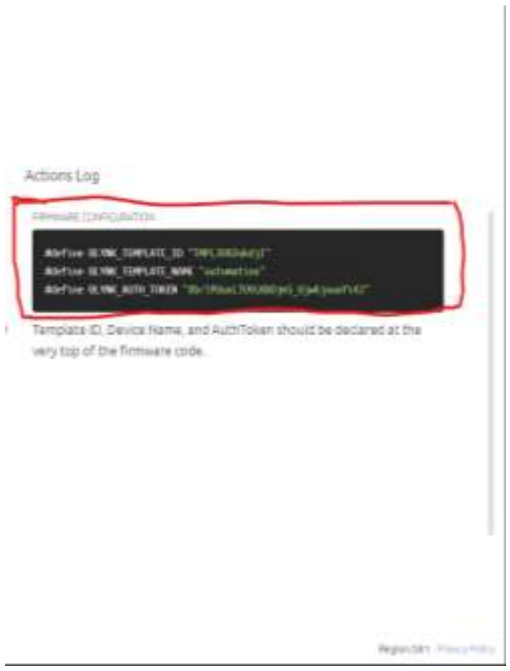


PROGRAM:

```
#define BLYNK_TEMPLATE_ID "TMPL3V82ukdjI"
#define BLYNK_TEMPLATE_NAME "automation"
#define BLYNK_AUTH_TOKEN "XbrlPUuxL7O9LKBOjm5_Ujwhjxuofi4J"

// Comment this out to disable prints and save space
#define BLYNK_PRINT Serial



#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

char auth[] = BLYNK_AUTH_TOKEN;
```

```
// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "A54";              //wifi name
char pass[] = "0123456789";        //password of wifi

int Relay1=15;   //GPIO15 connected with Relay1
int Relay2=13;   //GPIO13 connected with Relay2
int Relay3=14;   //GPIO14 connected with Relay3
int Relay4=27;   //GPIO27 connected with Relay4

BlynkTimer timer;

BLYNK_WRITE(V0)        //check virtual pin V0
{
 intpinValue=param.asInt();   //read vitual pin V0
 if(pinValue==0)
 {
 digitalWrite(Relay1,HIGH);
 }
 else
 {
 digitalWrite(Relay1,LOW);
 }
 }

BLYNK_WRITE(V1)       //check virtual pin V1
{
 intpinValue=param.asInt();   //read vitual pin V1
 if(pinValue==0)
 {
 digitalWrite(Relay2,HIGH);
 }
 else
 {
 digitalWrite(Relay2,LOW);
 }
 }

BLYNK_WRITE(V2)      //check virtual pin V2
{
 intpinValue=param.asInt();   //read vitual pin V2
 if(pinValue==0)
 {
 digitalWrite(Relay3,HIGH);
 }
 else
 {
 digitalWrite(Relay3,LOW);
 }
 }
BLYNK_WRITE(V3)      //check virtual pin V3
{
 intpinValue=param.asInt();   //read vitual pin V3
 if(pinValue==0)
 {
 digitalWrite(Relay4,HIGH);
 }
 else
 {
 digitalWrite(Relay4,LOW);
 }
 }
void setup()
{
 pinMode(Relay1,OUTPUT);
 pinMode(Relay2,OUTPUT);
 pinMode(Relay3,OUTPUT);
 pinMode(Relay4,OUTPUT);

 digitalWrite(Relay1,HIGH);
 digitalWrite(Relay2,HIGH);
 digitalWrite(Relay3,HIGH);
 digitalWrite(Relay4,HIGH);


 // Debug console
 Serial.begin(115200);

 Blynk.begin(auth, ssid, pass);

}

void loop()
{
 Blynk.run();
}
```

OUTPUT:

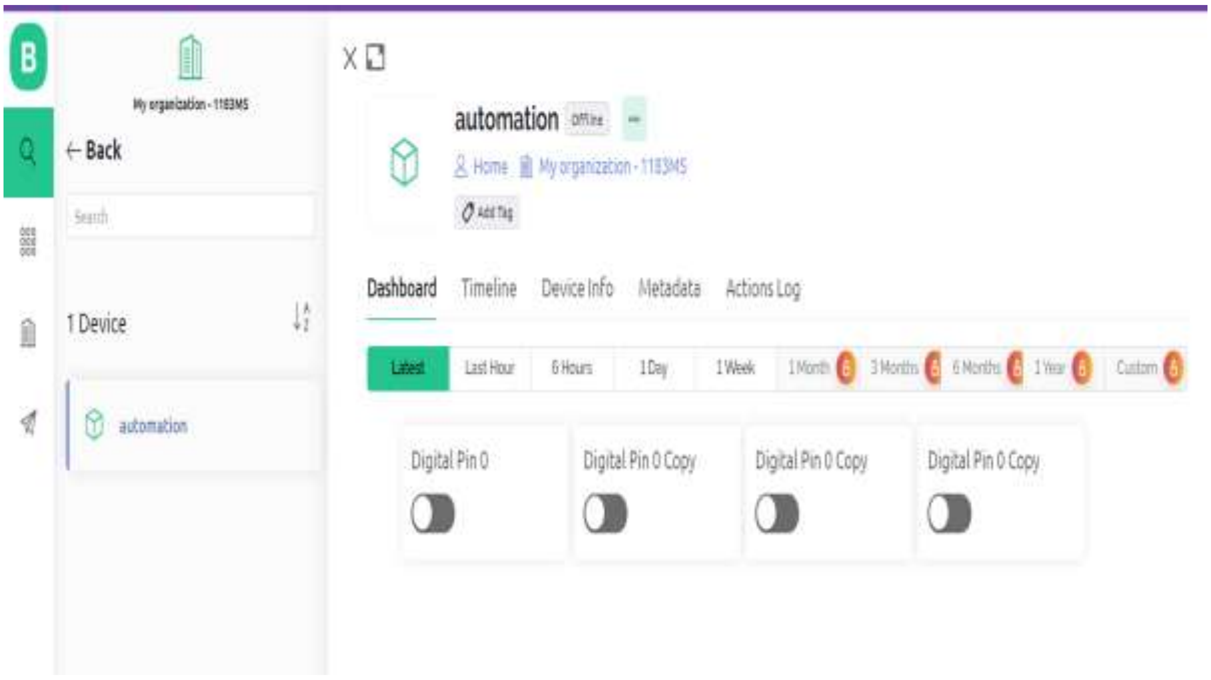


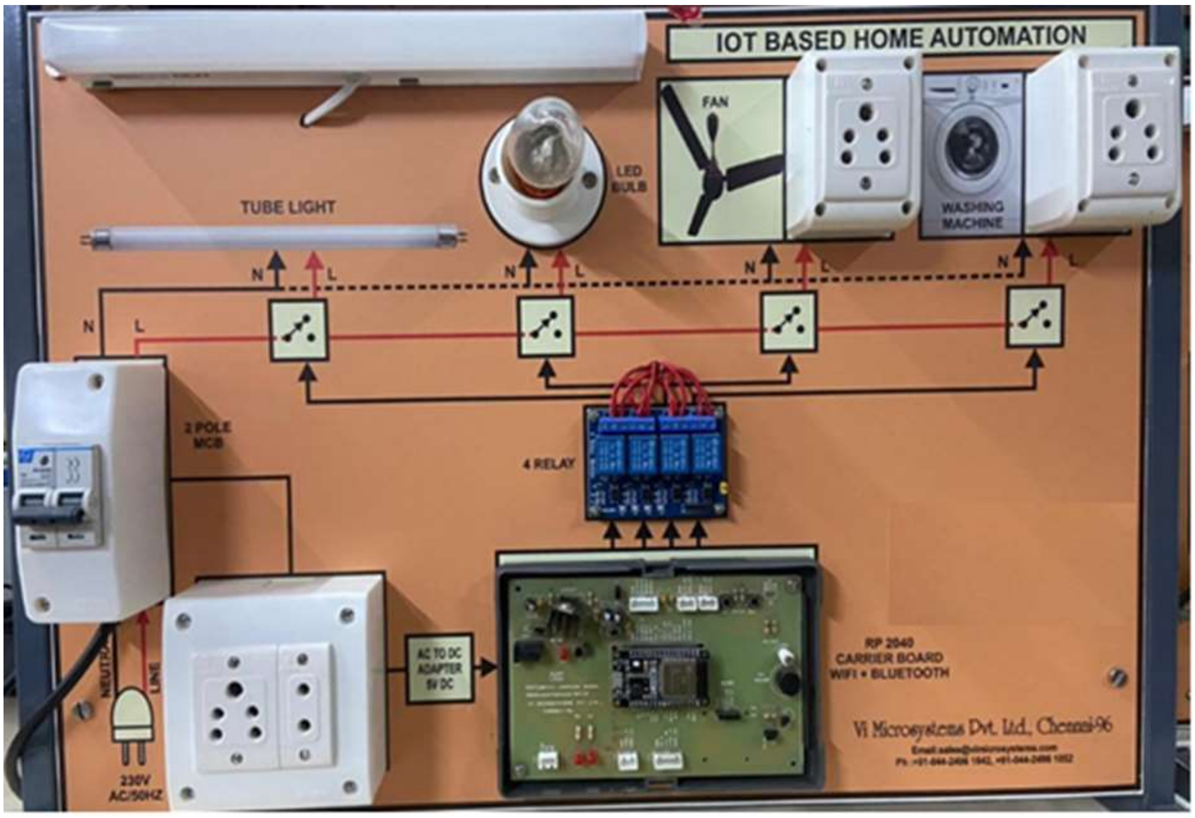
Actions Log

FIRMWARE CONFIGURATION

```
#define BLYNK_TEMPLATE_ID "[illegible]"
#define BLYNK_TEMPLATE_NAME "automation"
#define BLYNK_AUTH_TOKEN "[illegible]"
```

Template ID, Device Name, and AuthToken should be declared at the very top of the firmware code.




IOT BASED HOME AUTOMATION
TUBE LIGHT
LED BULB
FAN
WASHING MACHINE
N
L
2 POLE MCB
4 RELAY
AC TO DC ADAPTER 5V DC
RP 2040 CARRIER BOARD WIFI + BLUETOOTH
NEUTRAL
LINE
230V AC/50HZ
Vi Microsystems Pvt. Ltd., Chennai-96

RESULT:

Thus an IOT based system (Home Automation) was designed.