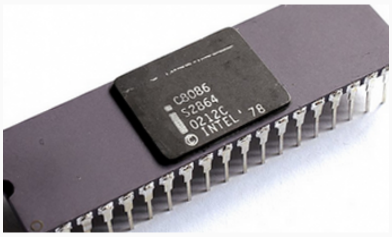
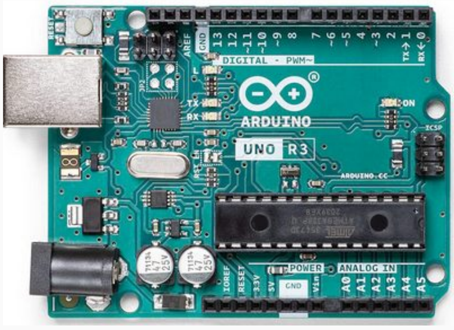
Introduction to Processors and Chips Laboratory(IPCL)

Sem- III, Division :- S.Y - 16 & 17

Lab
Manual

July
2023





Prasenje
et Patil

Digital Signer:Prasenjeet Patil
DN:C=IN, E=prasenjeet.patil@gmail.com, O=MIT ADT Uni, OU=SoC, CN=Prasenjeet Patil
Date:27-08-2024 11:07:57 +05:30

©Prasenjeet Damodar Patil
Associate Professor, Dept. of Computer Science & Engineering,
SoC, MIT ADT University, Pune

Instructions to the students

1. Conduct yourself in a responsible manner at all times in the laboratory
2. Students are required to attend all labs.
3. The write up work of the lab must be completed before entering the lab, it will get checked on the day of performance only.
4. Should take only the lab manual, calculator (if needed) and a pen or pencil to the work area.
5. Should learn the prelab questions. Read through the lab experiment to familiarize themselves with assembly and C++ language .
6. Should utilize 2 hours time properly for understanding, developing and testing the code.
7. If the experiment is not completed in the stipulated time, the pending work has to be carried out in the leisure hours or extended hours.
8. Students are required to maintain discipline in the lab.

Contents

Chapter I

Introduction

1 Experiment No.0-Hello World!! in NASM

```
section .data

msg db 'Hello World',10 ;assign msg variable with your message string
msglen: equ $-msg

section .text
global _start

_start:

mov eax,4 ; invoke SYS_WRITE (kernel opcode 4)
mov ebx,1 ;write to the STDOUT file
mov ecx,msg  ; move the address of our message string into ECX
mov edx,msglen; move the address of our message length into ECX
int 80h

mov eax,1 ; invoke SYS_EXIT (kernel opcode 1)
mov ebx,0 ;return 0 status on exit - 'No Errors
int 80h


;~$ nasm -f elf -o helloworld.o helloworld.asm
;~$ ld -m elf_i386 -o helloworld helloworld.o
;~$ ./helloworld
```

Step 0 :- Installing NASM
$ whereis nasm ; check if nasm is already installed **if not** follow the steps
$ sudo apt-get update ; get updated list of packages available for linux distro
$ sudo apt -y install nasm ; after update, for nasm installing nasm
$ whereis nasm ; to confirm nasm is installed or not

Step 1 :- Assembling of assembly program type in linux terminal
$ nasm -f elf -o helloworld.o helloworld.asm ; for creating object file from assembly file
Step 2 :- linking the object file

$ ld -m elf_i386 -o helloworld helloworld.o ; for creating executable file(intel 32-bit architecture) from object file

Step 3 :- Loading the executable in memory

$ ./helloworld ; loader command ./ name of the output file here helloworld

2 (ALP) to display “Your name”& “Roll number” on screen using macro

Aim :- Write an Assembly Language Program (ALP) to display “Your name”& “Roll number” on screen using macro. Accept user input from keyboard for name and roll number

```
section .data
msg1 db 'Enter your Name',10   ; 10=Ah=0xA as next line operator
len1: equ $-msg1

msg2 db 'Enter your roll no.',10
len2: equ $-msg2

msg3 db 'Entered name & roll no. is',10
len3: equ $-msg3

section .bss
name1 resb 10
rollno resb 10

section .text
global _start

_start:

%macro rw 4
mov eax,%1
mov ebx,%2
mov ecx,%3
mov edx,%4
int 80h
%endmacro

rw 4,1,msg1,len1
rw 3,0,name1,10

rw 4,1,msg2,len2
rw 3,0,rollno,10

rw 4,1,msg3,10
rw 4,1,name1,10
rw 4,1,rollno,10

mov eax,1    ; The system call for exit (sys_exit)
mov ebx,0    ; Exit with return code of 0 (no error)
int 80h;
```

Chapter II

Arithmetic operations

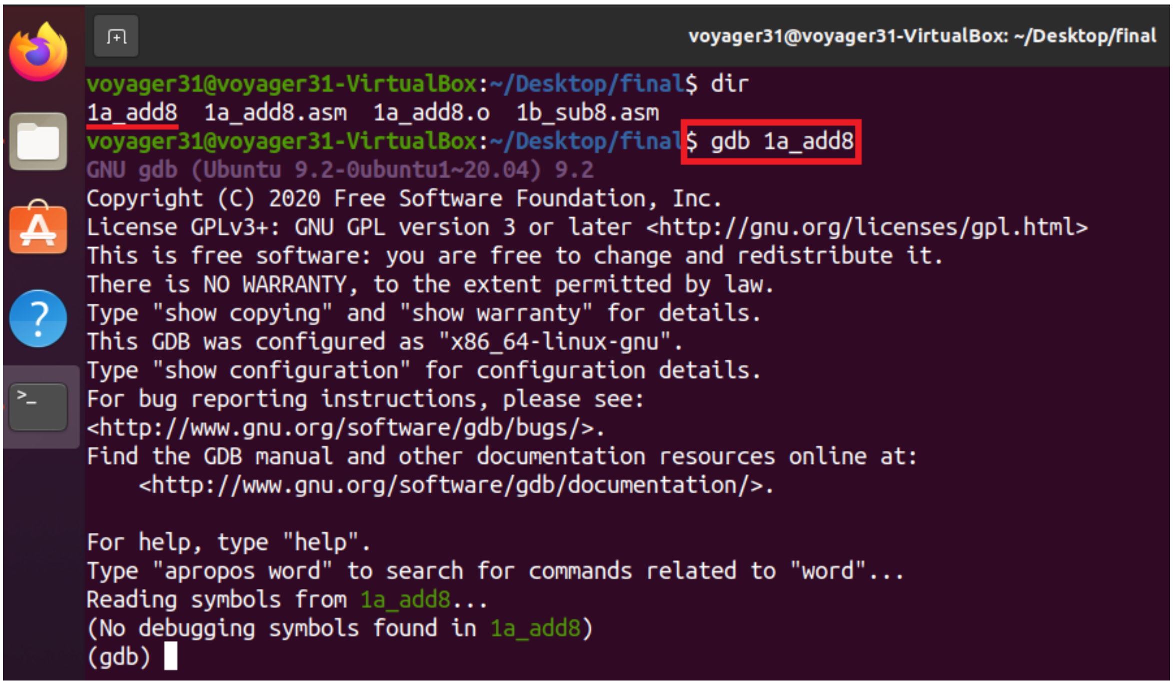
1 Addition of Numbers

1.1 Addition of two 8-bit unsigned numbers with GDB

Aim :- Write an ALP to perform Addition of two 8-bit numbers and show execution with gdb

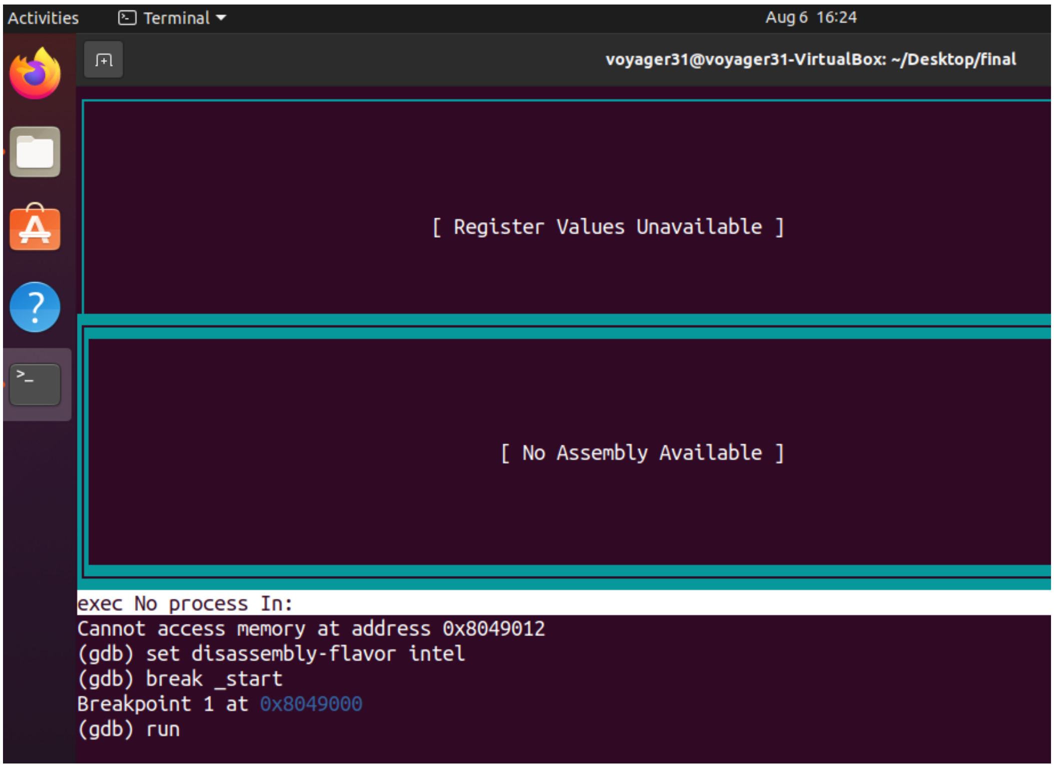
```
section .data
section .bss

section .text
global _start
_start:

mov al,121 ;
mov bl,100
add al,bl

 mov eax,1
 mov ebx,0
 int 80h
```

Step 1:- Enter the name of the executable file to be debugged at terminal prompt by using command
$ **gdb** name of executable file



```
voyager31@voyager31-VirtualBox:~/Desktop/final$ dir
1a_add8  1a_add8.asm  1a_add8.o  1b_sub8.asm
voyager31@voyager31-VirtualBox:~/Desktop/final$ gdb 1a_add8
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from 1a_add8...
(No debugging symbols found in 1a_add8)
(gdb) 
```

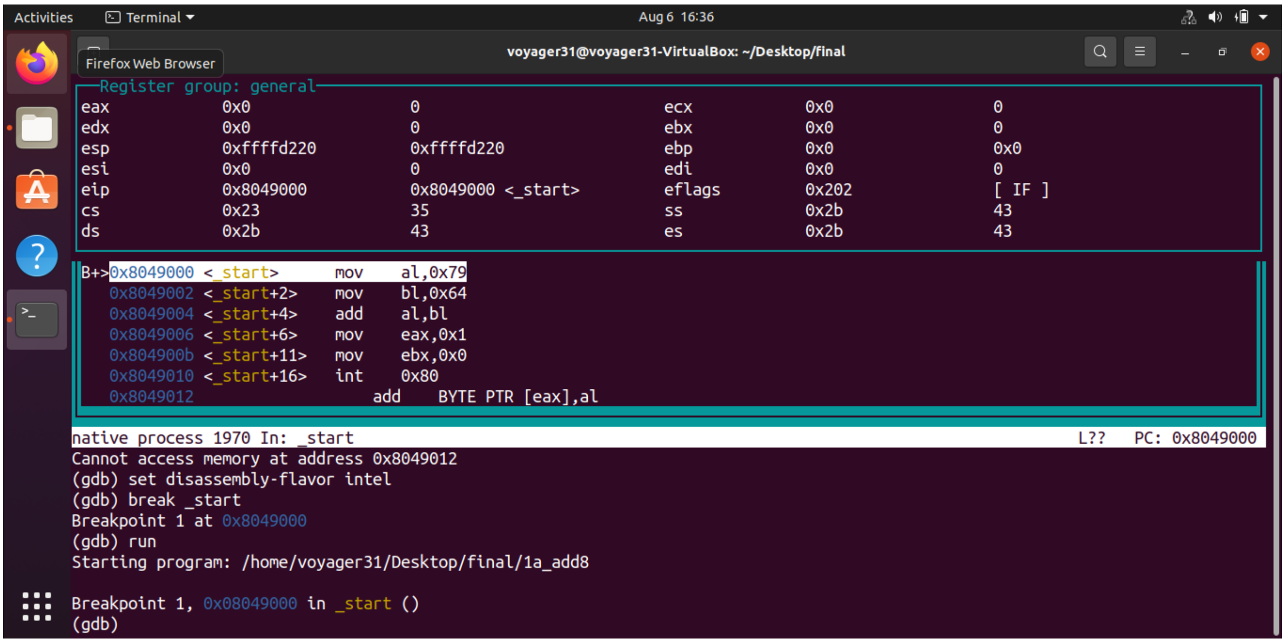
Step 2:- Now, for select the layout for assembly language and registers by typing following instructions at gdb terminal
(gdb) layout asm
(gdb) layout regs
Step 3:- Also, for displaying registers layout in intel format, type
(gdb) set disassembly-flavor intel
In order to set starting line for debugging, select _start, as the programming instructions are written after _start, by typing the instruction
(gdb) break _start
You will get the following layout on your screen



```
[ Register Values Unavailable ]

[ No Assembly Available ]

exec No process In:
Cannot access memory at address 0x8049012
(gdb) set disassembly-flavor intel
(gdb) break _start
Breakpoint 1 at 0x8049000
(gdb) run
```

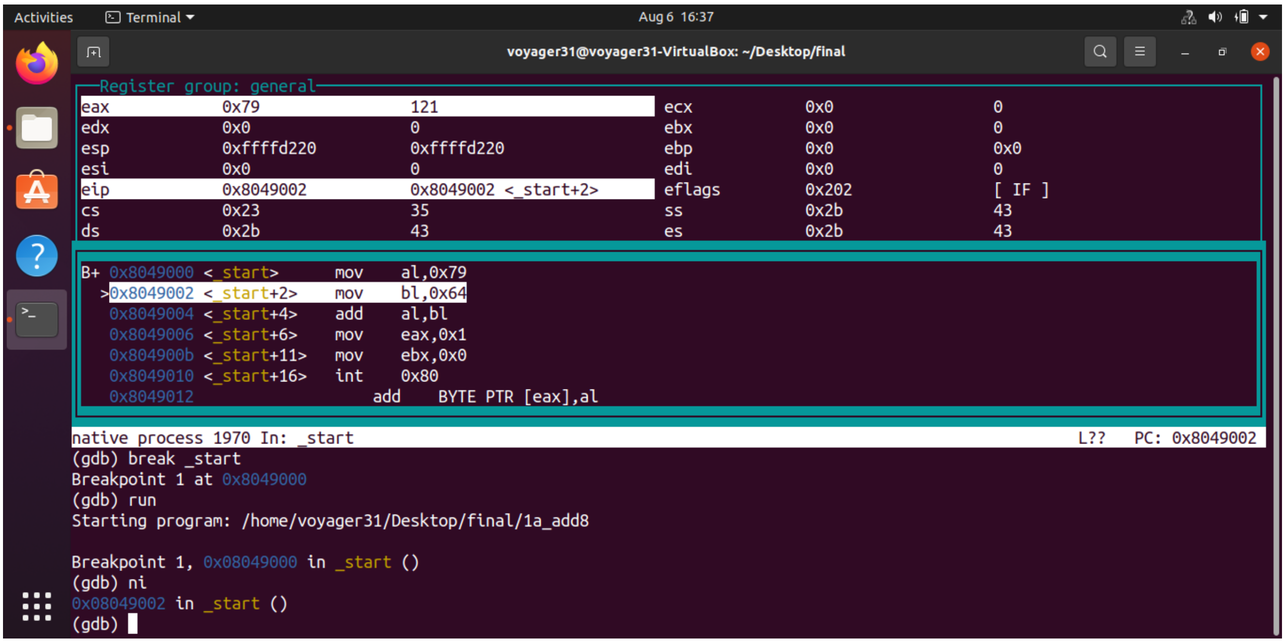
Step 4:- In order to debug the program, type **run**
(gdb) run
You will get the following layout on your screen. Here, in the registers layout, all 32-bit registers along with there content are displayed.

The second window shows the assembly level program that is loaded for debugging. The breakpoint(B+) is set at memory address 0x8049000, which points to the first instruction of the program to be debugged. The Program Counter(PC) is also holds the same value.



```
─Register group: general─
eax            0x0                 0                    ecx            0x0                 0
edx            0x0                 0                    ebx            0x0                 0
esp            0xffffd220          0xffffd220           ebp            0x0                 0x0
esi            0x0                 0                    edi            0x0                 0
eip            0x8049000           0x8049000 <_start>   eflags         0x202               [ IF ]
cs             0x23                35                   ss             0x2b                43
ds             0x2b                43                   es             0x2b                43

B+>0x8049000 <_start>       mov    al,0x79
   0x8049002 <_start+2>     mov    bl,0x64
   0x8049004 <_start+4>     add    al,bl
   0x8049006 <_start+6>     mov    eax,0x1
   0x804900b <_start+11>    mov    ebx,0x0
   0x8049010 <_start+16>    int    0x80
   0x8049012                     add    BYTE PTR [eax],al

native process 1970 In: _start                                          L??   PC: 0x8049000
Cannot access memory at address 0x8049012
(gdb) set disassembly-flavor intel
(gdb) break _start
Breakpoint 1 at 0x8049000
(gdb) run
Starting program: /home/voyager31/Desktop/final/1a_add8

Breakpoint 1, 0x08049000 in _start ()
(gdb)
```

Step 5:- In order to debug the program, step by step i.e one instruction at a time you can either type **ni** (for next instruction) or **stepi**(for single stepping instruction) or **si**
When you type ni, the first line of the code is loaded into the memory, and you can observe the content of registers in register layout as well the first instruction will be highlighted in assembly layout window.



```
Register group: general
eax            0x79                121                    ecx            0x0                 0
edx            0x0                 0                      ebx            0x0                 0
esp            0xffffd220          0xffffd220             ebp            0x0                 0x0
esi            0x0                 0                      edi            0x0                 0
eip            0x8049002           0x8049002 <_start+2>   eflags         0x202               [ IF ]
cs             0x23                35                     ss             0x2b                43
ds             0x2b                43                     es             0x2b                43

B+ 0x8049000 <_start>       mov    al,0x79
  >0x8049002 <_start+2>     mov    bl,0x64
   0x8049004 <_start+4>     add    al,bl
   0x8049006 <_start+6>     mov    eax,0x1
   0x804900b <_start+11>    mov    ebx,0x0
   0x8049010 <_start+16>    int    0x80
   0x8049012                    add    BYTE PTR [eax],al

native process 1970 In: _start                                                     L??   PC: 0x8049002
(gdb) break _start
Breakpoint 1 at 0x8049000
(gdb) run
Starting program: /home/voyager31/Desktop/final/1a_add8

Breakpoint 1, 0x08049000 in _start ()
(gdb) ni
0x08049002 in _start ()
(gdb) 
```

Output:

____

____

____

Click to see video

1.2 ALP to add two single digit nos.

Aim :- Write an Assembly Language Program (ALP) to add two single digit nos.(sum≤9). display result with std_output

```
section .data
msg db 'The sum is:', 0xA
len: equ $-msg

section .bss
 sum resb 1

section .text
   global _start
_start:              ;entry point
   mov  eax, '4'
   sub  eax, '0'; to convert from ASCII to Hex

   mov  ebx, '5'
   sub  ebx, '0'
   add  eax, ebx
   add  eax, '0' to convert back from Hex to ASCII
   mov  [sum], eax

   mov  eax, 4   ;system call number (sys_write)
   mov  ebx, 1   ;file descriptor (stdout)
   mov  ecx, msg
   mov  edx, len
   int  80h            ;call kernel

   mov  eax, 4  ;system call number (sys_write)
   mov  ebx, 1  ;file descriptor (stdout)
   mov  ecx,sum
   mov  edx, 1
   int  80h             ;call kernel

   mov  eax, 1   ;system call number (sys_exit)
   mov  ebx, 0
   int  80h             ;call kernel
```

Output:

Aim :- Write an Assembly Language Program (ALP) to add two single digit nos.(sum$\leq$9) by taking input from keyboard

```
section .data
   msg1 db ”Enter a digit ”, 0xA
   len1: equ $-msg1
   msg2 db ”Please enter a second digit”, 0xA
   len2: equ $-msg2
   msg3 db ”The sum is: ”
   len3: equ $-msg3

section .bss
   num1 resb 2
   num2 resb 2
   res resb 1

section .text
   global _start
_start:        ;entry point
   mov eax, 4
   mov ebx, 1
   mov ecx, msg1
   mov edx, len1
   int 80h

   mov eax, 3
   mov ebx, 0
   mov ecx, num1
   mov edx, 2
   int 80h

   mov eax, 4
   mov ebx, 1
   mov ecx, msg2
   mov edx, len2
   int 80h

   mov eax, 3
   mov ebx, 0
   mov ecx, num2
   mov edx, 2
   int 80h

   mov eax, 4
   mov ebx, 1
   mov ecx, msg3
   mov edx, len3
   int 80h

   mov eax,[num1];moving the 1st number to eax and 2nd no. to ebx
   sub eax,'0';subtracting ascii '0' to convert it into a hex ↩
```

```
    number

    mov ebx, [num2]
    sub ebx, '0';subtracting ascii '0' to convert it into a hex number

    add eax, ebx ;add eax and ebx
    add eax, '0';convert the sum from hexadecimal to ASCII

    mov [res],eax ; storing the sum in memory location res

    ; print the sum
    mov eax, 4
    mov ebx, 1
    mov ecx, res
    mov edx, 1
    int 80h

    mov eax, 1
    mov ebx, 0
    int 80h
```

Output:

1.3 Addition of two 8-bit numbers(2 digit numbers)

Aim :- Write an ALP to perform Addition of two 8-bit numbers(Two digit numbers). Display the result with message in terminal (STDOUT) for addition.

```
section .data
        lower_byte: db 0
        higher_byte: db 0

section .text
global _start
        _start:

        mov al, 12
        mov bl, 25
        add al,bl
        aam
        add al, '0'
        mov [lower_byte],al
        add ah,'0'
        mov [higher_byte],ah

        mov eax,4
        mov ebx,1
        mov ecx,higher_byte
        mov edx,1
        int 80h

        mov eax,4
        mov ebx,1
        mov ecx,lower_byte
        mov edx,1
        int 80h

        mov eax,1
        int 80h
```

Output:

2 Subtraction of Numbers

2.1 Subtraction of two 8-bit unsigned numbers with GDB

Aim :- Write an ALP to perform Subtraction of two 8-bit numbers and show execution with gdb.

```
section .data
section .bss

section .text
global _start
_start:

mov al,255
mov bl,100
sub al,bl

 mov eax,1
 mov ebx,0
 int 80h
```

Output:

2.2 Subtraction of two 8-bit numbers. Result with std_output

Aim :- Write an ALP to perform Subtraction of two 8-bit numbers. Display the result with message in terminal (STDOUT) for Subtraction

```
section .data
msg db "The subtraction of two 8 bit nos is",10,13
len1 equ $-msg

section .bss
res resb 1

section .text
global _start
_start:

mov al,6 ;
mov bl,5
sub al,bl
add al,'0'
mov [res],al

mov eax,4
mov ebx,1
mov ecx,msg
mov ebx,len1
int 80h

mov eax,4
mov ebx,1
mov ecx,res
mov edx,1
int 80h

mov eax,1
mov ebx,0
 int 80h
```

Output:

2.3 Subtraction of two 8-bit numbers(2 digit numbers)

Aim :- Write an ALP to perform Subtraction of two 8-bit numbers(Two digit numbers). Display the result with message in terminal (STDOUT) for addition.

```
section .data
        lower_byte: db 1
        higher_byte: db 1

section .text
global _start
        _start:

        mov al, 25
        mov bl, 01
        sub al,bl
        aam
        add al, '0'
        mov [lower_byte],al
        add ah,'0'
        mov [higher_byte],ah

        mov eax,4
        mov ebx,1
        mov ecx,higher_byte
        mov edx,1
        int 80h

        mov eax,4
        mov ebx,1
        mov ecx,lower_byte
        mov edx,1
        int 80h

        mov eax,1
        int 80h
```

Output:

2.4 Sum of elements in array

Aim :- Write 8086 assembly language program (ALP) to add array of n numbers, 16 bit numbers stored in memory & store the result (32 bit) in memory. Display the result with message in terminal (STDOUT) for array addition

```
section .data
x db 1,2,3,0
section .bss
sum resb 2

section .text
global _start
_start:
mov ax,4
mov bx,0
mov cx,x

t:
add bx,[cx]
add cx,1
dec ax
jnz t

add bx,'0'
mov [sum],ebx

mov eax,4
mov ebx,1
mov ecx,sum
mov edx,2
int 80h

mov eax,1
mov ebx,0
int 80h
```

Output:

3 Multiplication and Division of no.s

3.1 Multiplication of two 8 bit no.s

Aim :- Write an ALP to perform Multiplication of two 8-bit numbers. Display the result with message in terminal (STDOUT) for Multiplication. Display proper strings to prompt the user while accepting the input and displaying the result.

```
section .data
m1 db 'Enter the first no.',10
l1: equ $-m1

m2 db 'Enter the second no.',10
l2: equ $-m2

m3 db 'Multiplication of two no.s is',10
l3: equ $-m3

section .bss
n1 resb 1
n2 resb 1
res resb 1

section .text
global _start
_start:

%macro rw 4
mov eax,%1
mov ebx,%2
mov ecx,%3
mov edx,%4
int 80h
%endmacro

rw 4,1,m1,l1
rw 3,0,n1,2
rw 4,1,m2,l2
rw 3,0,n2,2

mov al,[n1];
sub al,'0'
mov bl,[n2]
sub bl,'0'
mul bl
add al,'0'
mov [res],al

rw 4,1,m3,l3
rw 4,1,res,1

```

```
mov eax,1
mov ebx,0
 int 80h
```

Output:

3.2 Multiplication of two 8 bit no.s

Aim :- Write an ALP to perform Multiplication of two 8-bit numbers(2 digit). Display the result with message in terminal (STDOUT) for Multiplication. Display proper strings to prompt the user while accepting the input and displaying the result.

```
section .data
        lower_byte: db 0
        higher_byte: db 0

section .text
global _start
        _start:

        mov al, 4
        mov bl, 5
        mul bl
        aam
        add al, '0'
        mov [lower_byte],al
        add ah,'0'
        mov [higher_byte],ah

        mov eax,4
        mov ebx,1
        mov ecx,higher_byte
        mov edx,1
        int 80h

        mov eax,4
        mov ebx,1
        mov ecx,lower_byte
        mov edx,1
        int 80h

        mov eax,1
        mov ebx,0
        int 80h
```

Output:

4 Division of two 8 bit no.s

Aim :- Write 8086 assembly language program (ALP) to divide 8 bit number by 8 bit number. Make your program user-friendly by accepting 16 bit dividend & 8 bit divisor from user and display quotient & remainder on screen. Display proper strings to prompt the user while accepting the input and displaying the result

```
section .data
m1 db 'Enter dividend',10
l1: equ $-m1

m2 db 'Enter divisor',10
l2: equ $-m2

m3 db 'Quotient is',10
l3: equ $-m3

m4 db 'Remainder is',10
l4: equ $-m4

section .bss
n1 resb 1
n2 resb 1
q resb 1
r resb 1

section .text
global _start
_start:

%macro rw 4
mov eax,%1
mov ebx,%2
mov ecx,%3
mov edx,%4
int 80h
%endmacro

rw 4,1,m1,l1
rw 3,0,n1,2
rw 4,1,m2,l2
rw 3,0,n2,2

mov al,[n1];
sub al,'0'
mov bl,[n2]
sub bl,'0'
div bl
add al,'0'
mov [q],al
add ah,'0'
```

```
mov [r],ah

rw 4,1,m3,l3
rw 4,1,q,1

rw 4,1,m4,l4
rw 4,1,r,1

mov eax,1
mov ebx,0
int 80h
```

Output:

4.1 Division of 16 bit number by 8 bit number

Aim :- Write 8086 assembly language program (ALP) to divide 16 bit number by 8 bit number. Make your program user-friendly by accepting 16 bit dividend & 8 bit divisor from user and display quotient & remainder on screen. Display proper strings to prompt the user while accepting the input and displaying the result

```
section .data
m1 db 'Enter dividend',10
l1: equ $-m1

m2 db 'Enter divisor',10
l2: equ $-m2

m3 db 'Quotient is',10
l3: equ $-m3

m4 db 'Remainder is',10
l4: equ $-m4

section .bss
n1 resb 2
n2 resb 1; divisor 8 bit
q resb 1
r resb 1

section .text
global _start
_start:

%macro rw 4
mov eax,%1
mov ebx,%2
mov ecx,%3
mov edx,%4
int 80h
%endmacro

rw 4,1,m1,l1
rw 3,0,n1,2
rw 4,1,m2,l2
rw 3,0,n2,2

mov ax,[n1];
sub ax,'0'
mov bl,[n2]
sub bl,'0'

xor ah,ah
div bl

add al,'0'
mov [q],al

add ah,'0'
mov [r],ah

rw 4,1,m3,l3
rw 4,1,q,1
rw 4,1,m4,l4
rw 4,1,r,1

mov eax,1
mov ebx,0
int 80h
```

Output:

Chapter III

String manipulation

1 String manipulation

1.1 Copy of string

Aim :- Write an ASM program to copy the source string to the destination string and display it on the console. (The source string is pointed by DS:SI and the destination string is pointed by ES:DI)

```
section .data
s1 db 'Hello, world!',0 ;string 1
len equ $-s1

section .bss
s2 resb 20          ;destination


section .text
global _start       ;must be declared for using gcc

_start:             ;tell linker entry point
   mov  ecx, len
   mov  esi, s1
   mov  edi, s2
   cld
   rep  movsb

   mov  eax,4        ;system call number (sys_write)
   mov  ebx,1        ;file descriptor (stdout)
   mov  ecx,s2       ;message to write
   mov  edx,20       ;message length
   int  0x80         ;call kernel

   mov  eax,1        ;system call number (sys_exit)
   int  0x80         ;call kernel
```

Output:

1.2 Conversion of String from Uppercase to Lowercase

Aim :- Write an ASM code to convert the Upper-Case String to the Lower-Case String. (eg: “HELLO WORLD” → “hello world”) and display it on the console. (using LODS and STOS instruction)

```
section .data
s1 db 'VOYAGER31'; Uppercase string to be converted
len equ $-s1

section .bss
s2 resb 20    ; bytes reserved for lower case string

section .text
    global _start

_start:
    mov  ecx, len ; length of string1
    mov  esi, s1
    mov  edi, s2

UpptoLow:
    lodsb
    or   al, 20h
    stosb
    loop UpptoLow
    cld
    rep  movsb

    mov  eax,4
    mov  ebx,1
    mov  ecx,s2 ;message to write
    mov  edx,20;message length
    int  80h

    mov  eax,1
    mov  ebx, 0
    int  80h
```

Output:

1.3 Conversion of String from Lowercase to Uppercase

Aim :- Write an ASM code to convert the Lower-Case String to the Upper-Case String. (eg: “HELLO WORLD” → “hello world”) and display it on the console. (without using stringlen function)

```
section .data
string: db "voyager31",10,0
len:    equ $-string

section .text
global _start

_start: mov ecx, string
        call toUpper
        call print
        mov eax,1
        mov ebx,0
        int 80h

toUpper:
        mov al,[ecx]  ; ecx is the pointer, so [ecx] the current char
        cmp al,0x0
        je done
        cmp al,'a'
        jb next_char
        cmp al,'z'
        ja next_char
        sub al,0x20  ; move AL upper case and
        mov [ecx],al ; write it back to string

next_char:
        inc ecx ; not al, that's the character. ecx has to
                ; be increased, to point to next char
        jmp toUpper
done:   ret

print:  mov ecx, string   ; what to print
        mov edx, len      ; length of string to be printed
        mov ebx, 1
        mov eax, 4
        int 80h
        ret
```

Output:

1.4 Character search in a String

Aim :- Write an ASM code to search a particular character or set of characters from given a string and display the result on the console. (Use SCAS instruction)

```
section .data
my_string db 'hello world', 0
len equ $-my_string

msg_found db 'found!', 0xa
len_found equ $-msg_found

msg_notfound db 'not found!'
len_notfound equ $-msg_notfound

section .text
global _start
_start:   ;tell linker entry point

    mov ecx,len
    mov edi,my_string
    mov al , 'p'
    cld
    repne scasb
    je found ; when found
;If not not then the following code
    mov eax,4
    mov ebx,1
    mov ecx,msg_notfound
    mov edx,len_notfound
    int 80h
    jmp exit

found:
    mov eax,4
    mov ebx,1
    mov ecx,msg_found
    mov edx,len_found
    int 80h

exit:
    mov eax,1
    mov ebx,0
    int 80h
```

Output:

1.5 Comparing two Strings

Aim :- Write an ASM program to compare two strings using the CMPS instruction and display the result on the console.

```
section .data
s1 db 'Hello, world!',0 ;our first string
lens1 equ $-s1
s2 db 'Hello, world!', 0;our second string
lens2 equ $-s2
msg_eq db 'Strings are equal!', 0xa
len_eq  equ $-msg_eq

msg_neq db 'Strings are not equal!'
len_neq equ $-msg_neq

section .text
global _start ;must be declared for using gcc
_start:
   mov esi, s1
   mov edi, s2
   mov ecx, lens2
   cld
   repe  cmpsb
   jecxz  equal  ;jump when ecx is zero

   ;If not equal then the following code
   mov eax, 4
   mov ebx, 1
   mov ecx, msg_neq
   mov edx, len_neq
   int 80h
   jmp exit

equal:
   mov eax, 4
   mov ebx, 1
   mov ecx, msg_eq
   mov edx, len_eq
   int 80h

exit:
   mov eax, 1
   mov ebx, 0
   int 80h
```

Output:

1.6 String Reversal

Aim :- Write an Assembly Language Program (ALP) for comparing two strings

```
section .data
msg db ”Voyager31”
len: equ $-msg

section .bss
rstring resb 14

section .code
global _start
_start:
    %macro write 2
    mov eax,4
    mov ebx,1
    mov ecx,%1
    mov edx,%2
    int 80h
    %endmacro

    %macro read 2
    mov eax,3
    mov ebx,0
    mov ecx,%1
    mov edx,%2
    int 80h
    %endmacro

    mov esi,msg
    mov ecx,14
    add esi,len-1
    mov edi,rstring

AGAIN:mov eax,[esi]
    mov [edi],eax
    dec esi
    inc edi
    LOOP AGAIN
    write rstring,14
    mov eax,1
    int 80h
```

Output:

1.7 String Concatenation

Aim :- Write an Assembly Language Program (ALP) for comparing two strings

```
section .data
    msg1:     db 'Enter the first name',10
    msg1len:  equ $-msg1

    msg2:     db 'Enter the middle name',10
    msg2len:  equ $-msg2

    msg3:     db 'Enter the last name',10
    msg3len:  equ $-msg3

    msg4:     db 'Entered name is as follows',10
    msg4len:  equ $-msg4


section .bss
first resb 10
middle resb 10
last resb 10
fullname resb 30

%macro rw 4
mov eax,%1
mov ebx,%2
mov ecx,%3
mov edx,%4
int 80h
%endmacro

section .text
global _start
_start:
        rw 4,1,msg1,msg1len
        rw 3,0,first,10

        rw 4,1,msg2,msg2len
        rw 3,0,middle,10

        rw 4,1,msg3,msg3len
        rw 3,0,last,10

        rw 4,1,msg4,msg4len

        mov edi,fullname

        mov esi,first
        up: mov al,[esi]
        cmp al,10
```

```
je dn
mov [edi],al
inc esi
inc edi
jmp up

dn:
mov[edi],byte ' '
inc edi
mov esi,middle
up1:mov al,[esi]
cmp al,10
je dn1
mov [edi],al
inc esi
inc edi
jmp up1

dn1:
mov[edi],byte ' '
inc edi
mov esi,last
up2: mov al,[esi]
cmp al,10
je dn2
mov [edi],al
inc esi
inc edi
jmp up2

dn2: rw 4,1,fullname,30

mov eax,1
mov ebx,0
int 80h;
```

Output:

Chapter IV

Arduino Programming

1 LED BLINKING

LED BLINKING. Roll No.:-

AIM: Interface Light Emitting Diode (LED) to Arduino UNO board & write the program for blinking LED with a specified delay.

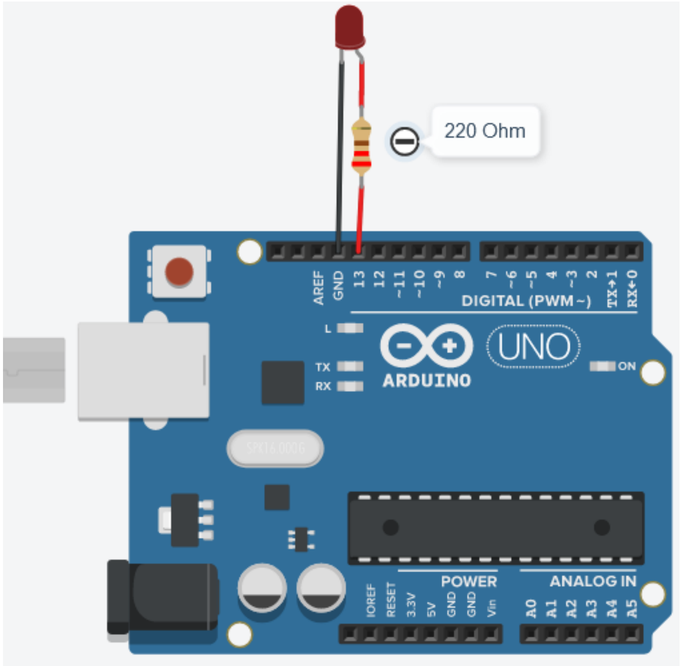
PREREQUISITES:

1. Knowledge of Arduino Uno Board interfaces.
2. Tool- Arduino IDE.

OBJECTIVES:

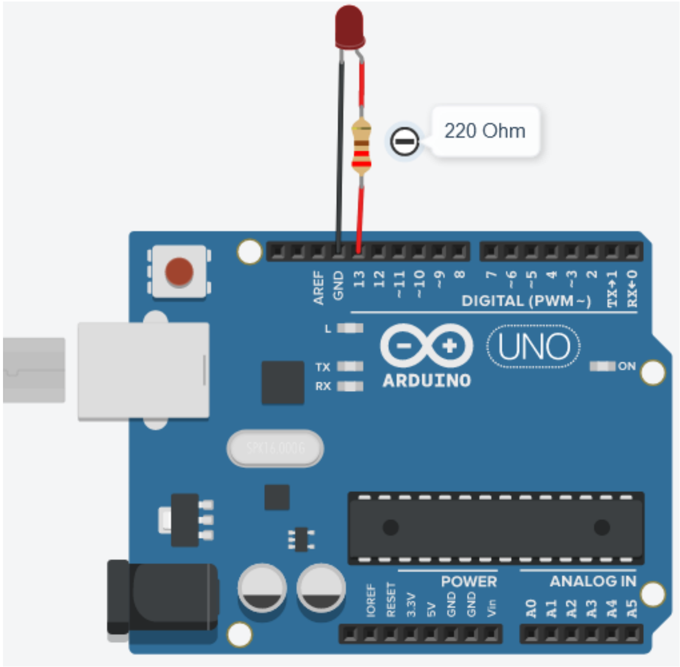
1. To connect & operate LED connected to digital outputs of an Arduino.
2. To understand concept of Interfacing with microcontroller.
3. To understand requirement of microcontroller for interfacing any external devices.

SCHEMATIC DIAGRAM:

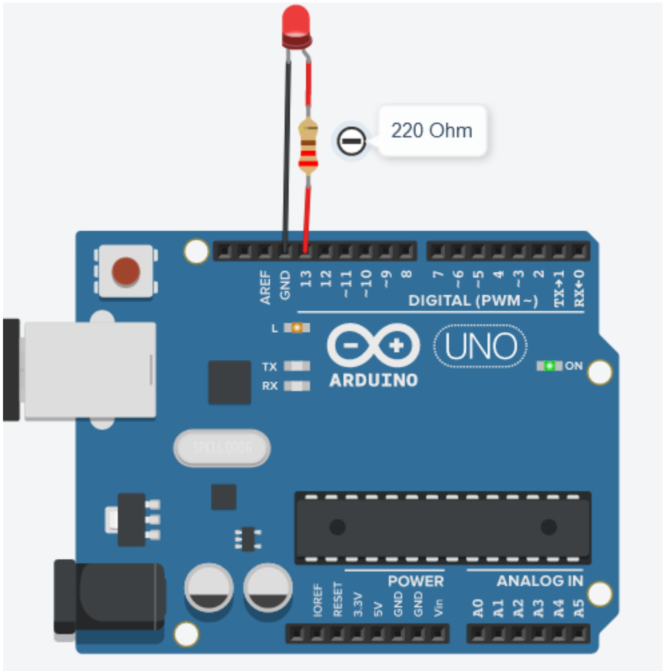


```
int led=13;
void setup()
{
    pinMode(led, OUTPUT);
}
void loop() {
    digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
    delay(1000); // wait for a second
    digitalWrite(led, LOW); // turn the LED off by making the voltage LOW
    delay(1000); // wait for a second
}
```

OUTPUT :- LED OFF



OUTPUT :- LED ON



CONCLUSION:

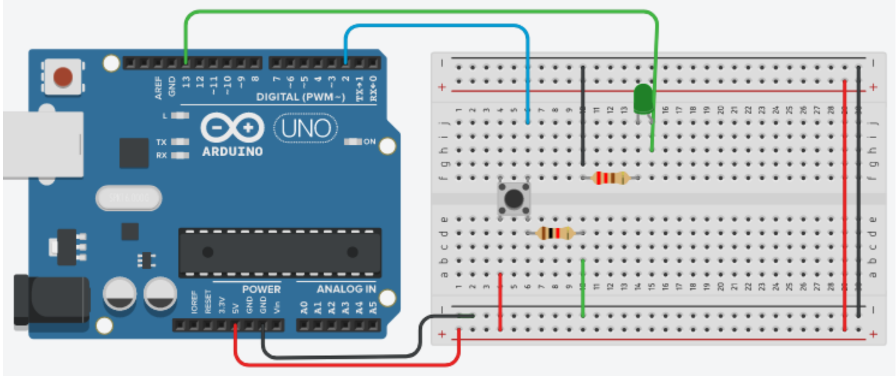
Click to see video

1.1 Turning ON/OFF LED with Push button

Turning ON/OFF LED with Push button. Roll No.:-

AIM: Interface LED and Push Button with Arduino UNO and write a Code to TURN ON the LED by pressing the Push Button

SCHEMATIC DIAGRAM:



```
const int buttonPin = 2;      // the number of the pushbutton pin
const int ledPin =  13;       // the number of the LED pin
int buttonState = 0;          // variable for reading the pushbutton status

void setup() {
    pinMode(ledPin, OUTPUT);
    pinMode(buttonPin, INPUT);
}

void loop() {
    // read the state of the pushbutton value:
    buttonState = digitalRead(buttonPin);
    // check if the pushbutton is pressed.
    // if it is, the buttonState is HIGH:
    if (buttonState == HIGH) {
        // turn LED on:
        digitalWrite(ledPin, HIGH);
    } else {
        // turn LED off:
        digitalWrite(ledPin, LOW);
    }
}
```

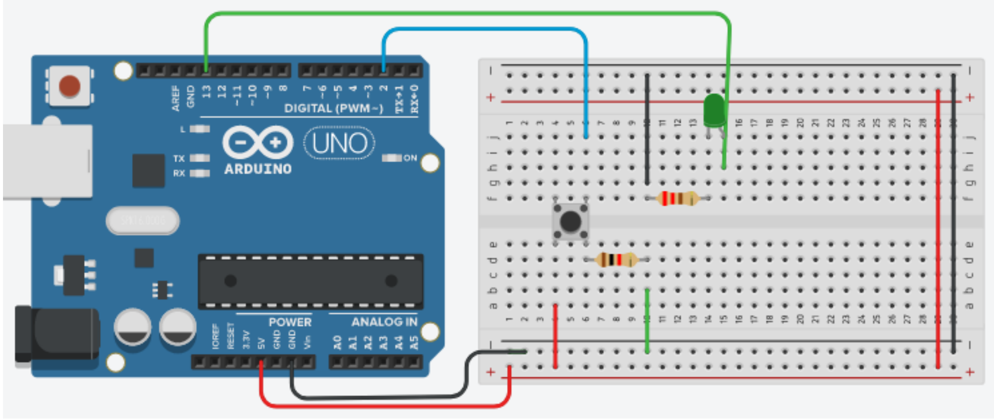
CONCLUSION:

1.2 Turning OFF LED with Push button

Turning OFF LED with Push button with delay. Roll No.:-

AIM: Interface LED and Push Button with Arduino UNO, write a code for KEEP LED ALWAYS ON. If the Push button is pressed, the LED will off for 5Sec , again will go in Always ON Mode.

SCHEMATIC DIAGRAM:



```
const int buttonPin = 2;      // the number of the pushbutton pin
const int ledPin =  13;       // the number of the LED pin
int buttonState = 0;          // variable for reading the pushbutton status

void setup() {
      pinMode(ledPin, OUTPUT);
      pinMode(buttonPin, INPUT);
}

void loop() {
     buttonState = digitalRead(buttonPin);
    if (buttonState == LOW) {
        digitalWrite(ledPin, HIGH);
      }
  else {
        // turn LED off:
        digitalWrite(ledPin, LOW);
        delay(5000);
    }
}
```

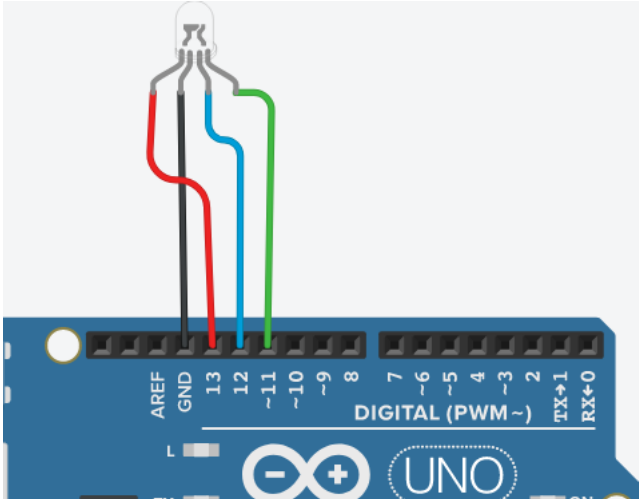
CONCLUSION:

1.3 Turning ON/OFF LED with Push button

LED BLINKING. Roll No.:-

AIM: Interface LED in Common Cathode mode and Push Button with Arduino UNO and write a Code to TURN ON RED, Blue & Green color with delay of 0.5 sec each.

SCHEMATIC DIAGRAM:



```
  const int red =  13;    // for red color
  const int blue =  12;    // for blue color
  const int green =  11;    // for green color

 void setup()
     {
        pinMode(red, OUTPUT);
        pinMode(blue, OUTPUT);
        pinMode(green, OUTPUT);
        }

 void loop()
 {
     digitalWrite(red, HIGH);
     delay(500);
     digitalWrite(blue, HIGH);
     delay(500);
     digitalWrite(green, HIGH);
     delay(500);
 }
```

CONCLUSION:

2 PIR Sensor interfacing

PIR SENSOR INTERFACING Roll No.:-

AIM: Interface the PIR sensor with Arduino UNO board and write the program to control the LED (ON/OFF) on motion and play the buzzer on detection of object.

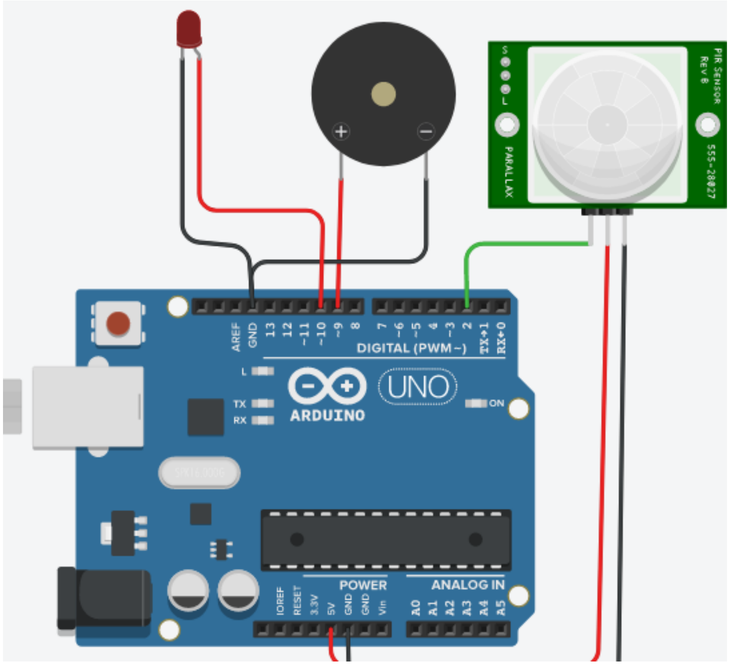
PREREQUISITES:

1. Knowledge of Arduino Uno Board interfaces.
2. Tool- Arduino IDE.

OBJECTIVES:

1. To connect & operate PIR sensor connected to pins of an Arduino.
2. To understand concept of Interfacing with microcontroller.
3. To understand requirement of microcontroller for interfacing any external devices.

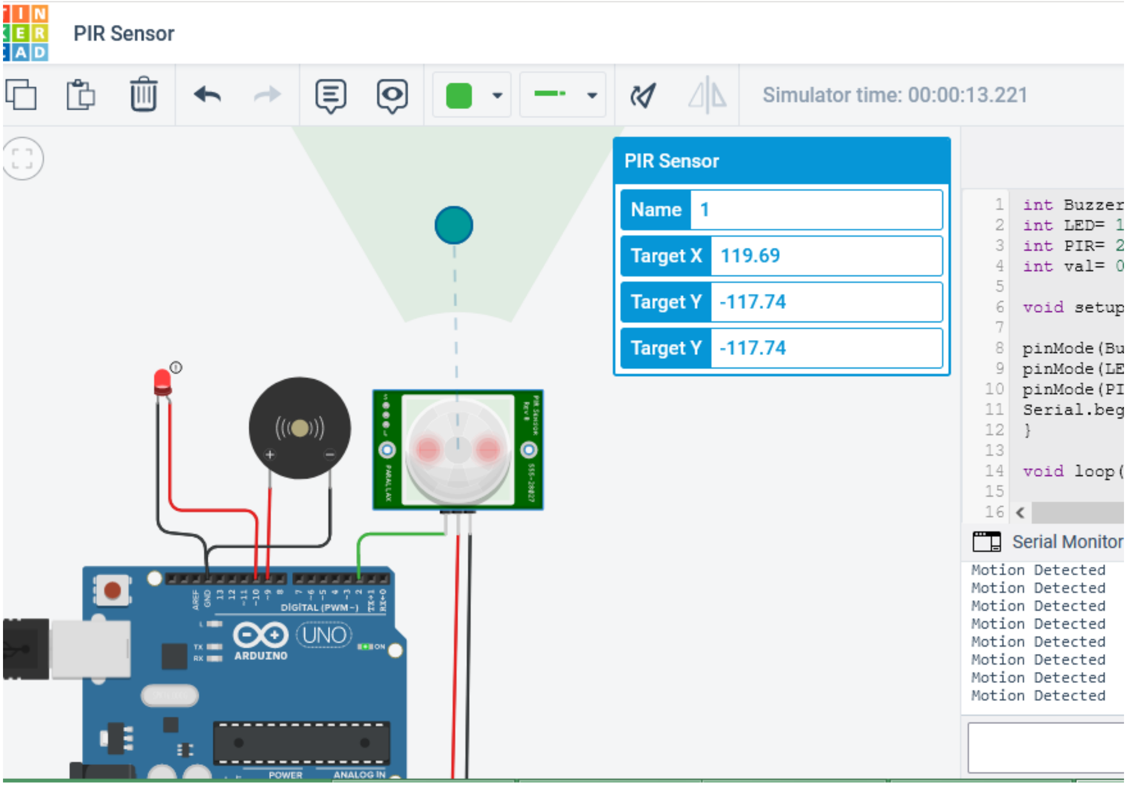
SCHEMATIC DIAGRAM:



```
int pinsensor = 2;
int pinled = 10;
int pinbuzzer = 9;
int pirsensor=0;
void setup()
{
    pinMode(pinsensor, INPUT);
    pinMode(pinled, OUTPUT);
    pinMode(pinbuzzer, OUTPUT);
    Serial.begin(9600);
}
void loop()
```

```
    {
        pirsensor = digitalRead(pinsensor);
        if (pirsensor == HIGH)
        {
            digitalWrite(pinled, HIGH);
            tone(pinbuzzer,1000,500);
            Serial.begin(‘‘motion detected’’)
        }
        else
        {
            digitalWrite(pinled, LOW);
            noTone(pinbuzzer,9);
        }
        delay(10);
    }
```

OUTPUT :- When Motion is detected.



Name	1
Target X	119.69
Target Y	-117.74
Target Y	-117.74

```
int Buzzer
int LED= 1
int PIR= 2
int val= 0

void setup

pinMode(Bu
pinMode(LE
pinMode(PI
Serial.beg
}

void loop(
```

```
Motion Detected
Motion Detected
Motion Detected
Motion Detected
Motion Detected
Motion Detected
Motion Detected
Motion Detected
```

CONCLUSION:

____

____

____

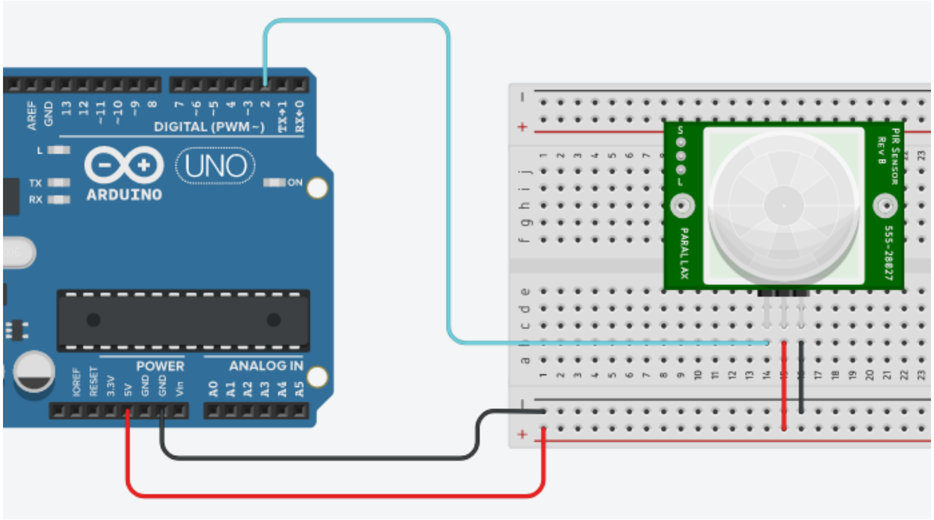
Click to see video

2.1 PIR sensor on Human detection

PIR SENSOR INTERFACING Roll No.:-

AIM: Interface PIR sensor with Arduino UNO and write a code to display status of person detection on the hyperterminal.

SCHEMATIC DIAGRAM:



```
void setup()
{
    Serial.begin(9600);
    pinMode(2, INPUT);
}

void loop()
{

    if(digitalRead(2)==1)
    {
        Serial.println("Living␣being␣detected");
        delay(100);
    }
}
```

CONCLUSION:

3 Ultrasonic Sensor interfacing

ULTRASONIC SENSOR INTERFACING Roll No.:-

AIM: Interface ultrasonic sensor with Arduino UNO board & write the program to measure and display the distance on serial monitor.

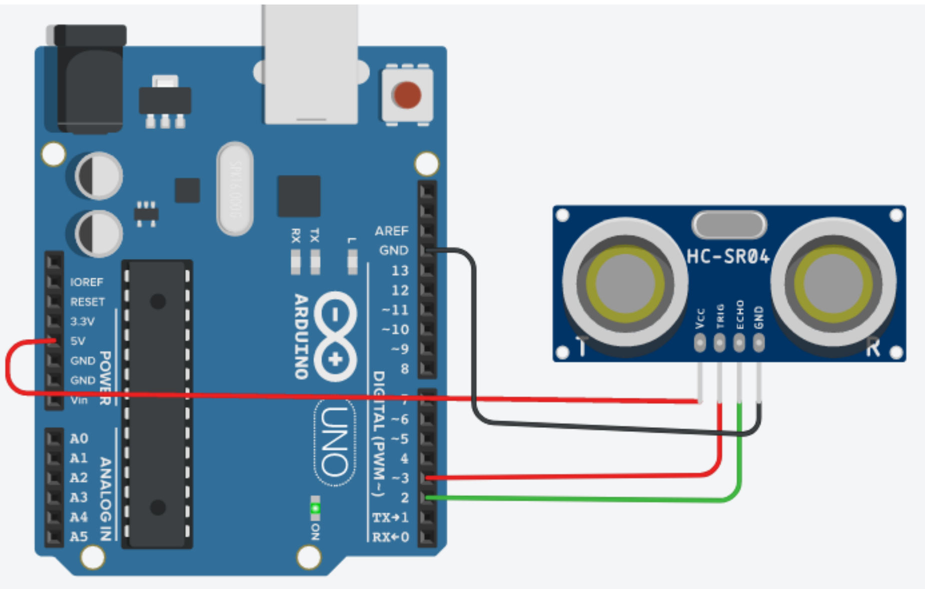
PREREQUISITES:

1. Knowledge of Arduino Uno Board interfaces.
2. Tool- Arduino IDE.

OBJECTIVES:

1. To connect & operate Ultrasonic sensor connected to pins of an Arduino.
2. To understand concept of Interfacing with microcontroller.
3. To understand requirement of microcontroller for interfacing any external devices.

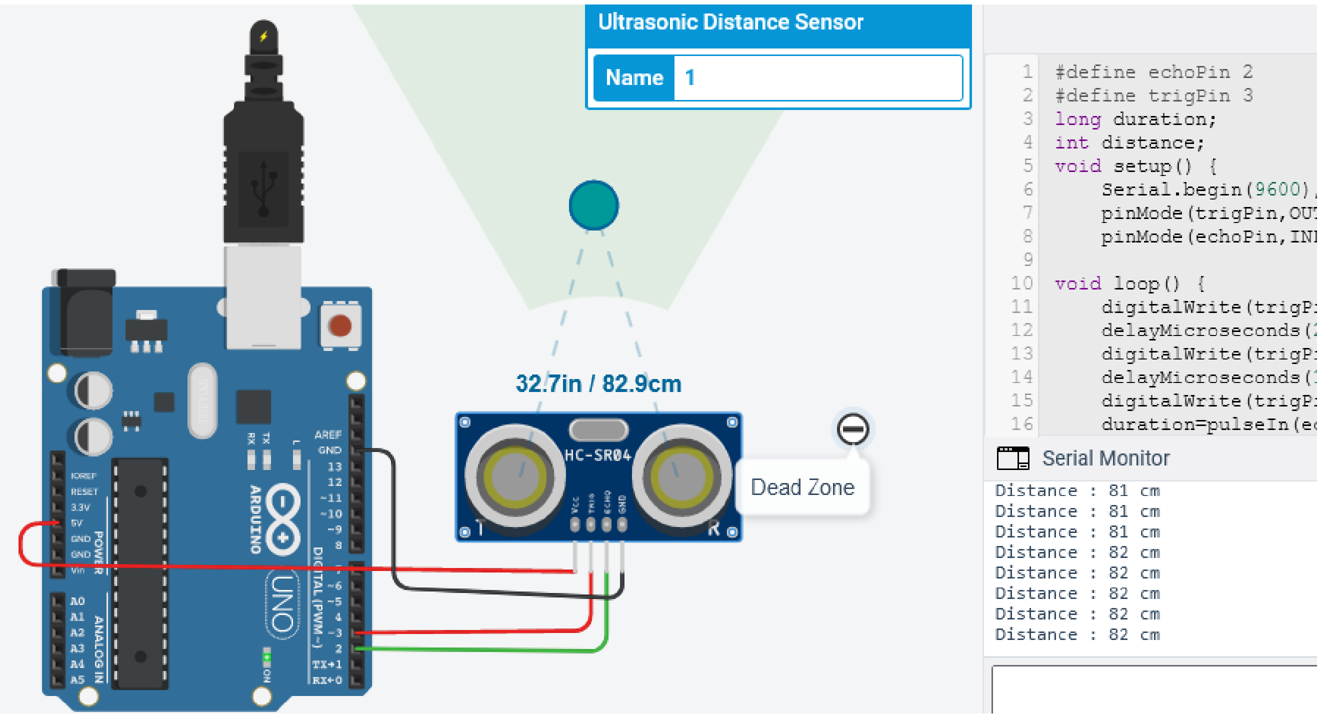
SCHEMATIC DIAGRAM:



```
// defines pins numbers
const int trigPin = 3;
const int echoPin = 2;
// defines variables
long duration;
int distance;
void setup() {
    pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
    pinMode(echoPin, INPUT); // Sets the echoPin as an Input
    Serial.begin(9600); // Starts the serial communication
}
void loop() {
    // Clears the trigPin
    digitalWrite(trigPin, LOW);
```

```
        delayMicroseconds(2);
        // Sets the trigPin on HIGH state for 10 micro seconds
        digitalWrite(trigPin, HIGH);
        delayMicroseconds(10);
        digitalWrite(trigPin, LOW);
        // Reads the echoPin, returns the sound wave travel time in microseconds
        duration = pulseIn(echoPin, HIGH);
        // Calculating the distance
        distance = duration * 0.034 / 2;
        // Prints the distance on the Serial Monitor
        Serial.print("Distance␣in␣cm:␣");
        Serial.println(distance);
    }
```

OUTPUT :-



```
#define echoPin 2
#define trigPin 3
long duration;
int distance;
void setup() {
    Serial.begin(9600),
    pinMode(trigPin,OU
    pinMode(echoPin,IN

void loop() {
    digitalWrite(trigP
    delayMicroseconds(
    digitalWrite(trigP
    delayMicroseconds(
    digitalWrite(trigP
    duration=pulseIn(e
```

```
Distance : 81 cm
Distance : 81 cm
Distance : 81 cm
Distance : 82 cm
Distance : 82 cm
Distance : 82 cm
Distance : 82 cm
Distance : 82 cm
```

CONCLUSION:

____

____

Click to see video

3.1 Ultrasonic sensor LED ON/OFF

ULTRASONIC SENSOR INTERFACING Roll No.:-

AIM: Interface ultrasonic sensor with Arduino UNO board and write the program to Turn ON the LED if distance is more than 10 cm else turn OFF the LED

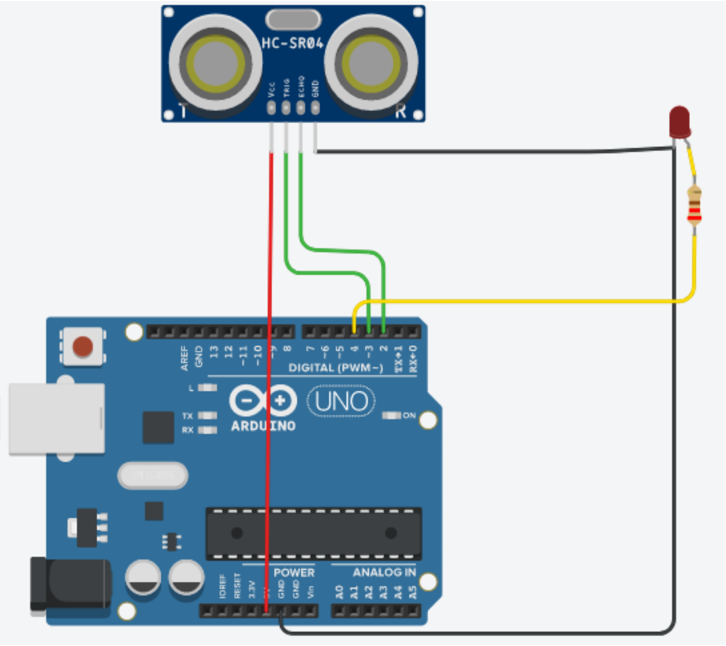
PREREQUISITES:

1. Knowledge of Arduino Uno Board interfaces.
2. Tool- Arduino IDE.

OBJECTIVES:

1. To connect & operate Ultrasonic sensor connected to pins of an Arduino.
2. To understand concept of Interfacing with microcontroller.
3. To understand requirement of microcontroller for interfacing any external devices.

SCHEMATIC DIAGRAM:



```
int trigPin = 3;
int echoPin = 2;
int led1 =4 ;
long duration;
int distance;
void setup(){

    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    pinMode(led1, OUTPUT);
}
void loop()
{
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    duration = pulseIn(echoPin, HIGH);
    distance= (duration/2) * 0.034;


    if (distance > 10){
        digitalWrite(led1, HIGH);
    }
    else if (distance <=10){
        {
            digitalWrite(led1, LOW);
        }
    }
}
```

CONCLUSION:

____

____

____

3.2 Ultrasonic Sensor Obstacle detection

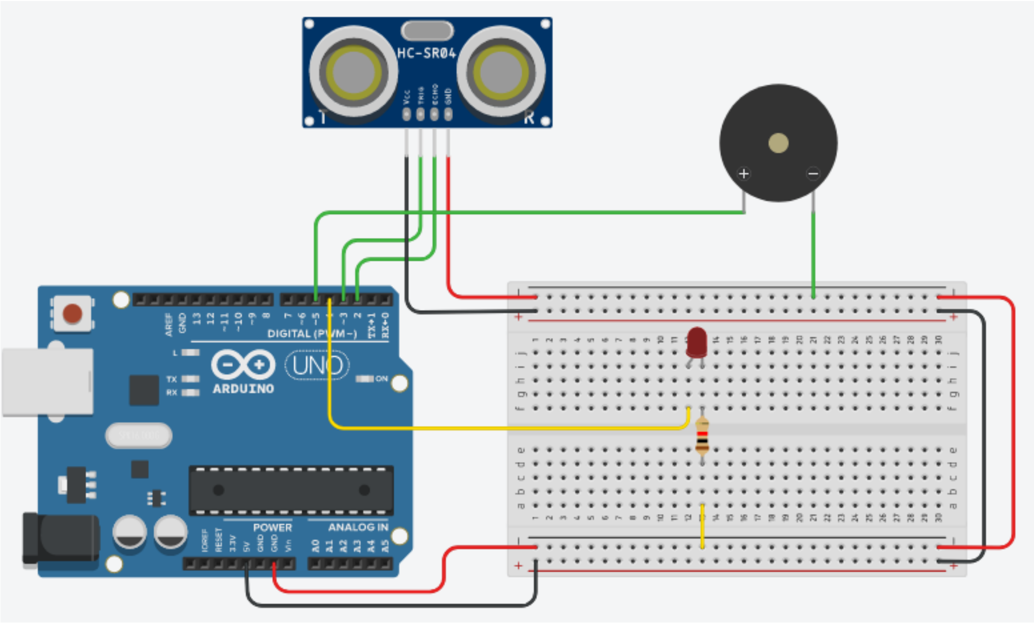
ULTRASONIC SENSOR INTERFACING Roll No.:-

AIM: Interface ultrasonic sensor with Arduino UNO board and write the program if distance is less than 10 cm else turn ON the buzzer else turn ON the LED

PREREQUISITES:

1. Knowledge of Arduino Uno Board interfaces.
2. Tool- Arduino IDE.

SCHEMATIC DIAGRAM:



```
int trigPin = 3;
int echoPin = 2;
int led1 =4 ;
int buzzer =5;
long duration;
int distance;
void setup(){
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    pinMode(led1, OUTPUT);
    pinMode(buzzer, OUTPUT);
}
void loop()
{
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    duration = pulseIn(echoPin, HIGH);
```

```
        distance= (duration/2) * 0.034;
        if (distance > 10){
            digitalWrite(led1, HIGH);
          digitalWrite(buzzer, LOW);
        }
        else if (distance <=10){
            {
                digitalWrite(led1, HIGH);
                digitalWrite(buzzer, HIGH);
            }
        }
    }
```

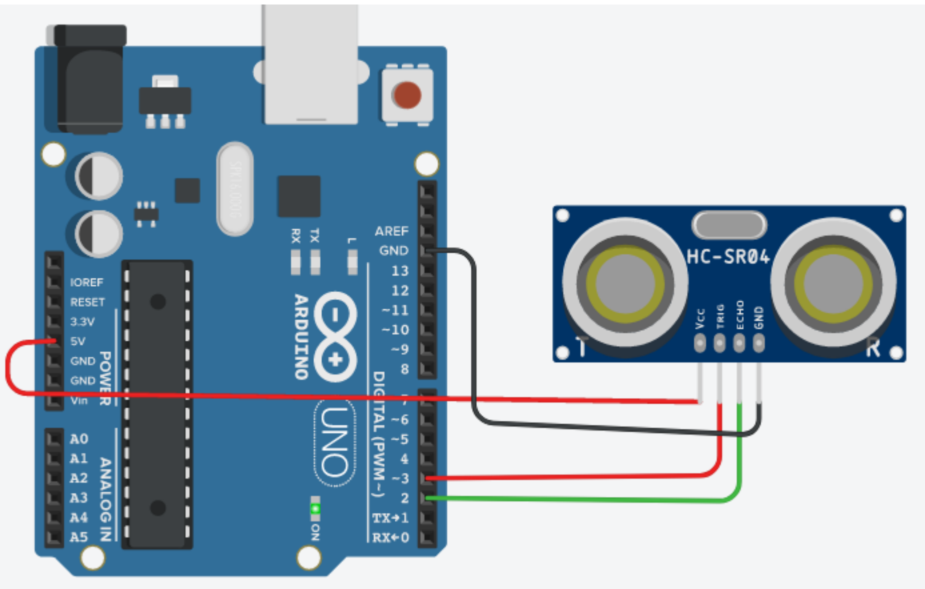
CONCLUSION:

ULTRASONIC SENSOR INTERFACING

Roll No.:-

AIM: Interface ultrasonic sensor with Arduino UNO board & write the program to detect the object if it is at a distance ≤= 15cm and display message 'Object Detected'.

SCHEMATIC DIAGRAM:



```
// defines pins numbers
const int trigPin = 3;
const int echoPin = 2;
// defines variables
long duration;
int distance;
void setup() {
    pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
    pinMode(echoPin, INPUT); // Sets the echoPin as an Input
    Serial.begin(9600); // Starts the serial communication
}
void loop() {
    // Clears the trigPin
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    // Sets the trigPin on HIGH state for 10 micro seconds
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    // Reads the echoPin, returns the sound wave travel time in microseconds
    duration = pulseIn(echoPin, HIGH);
    // Calculating the distance
    distance = duration * 0.034 / 2;  // Prints the distance on the Serial Monitor
    if(distance <= 15)
    {
         Serial.print("Object␣Detected");
        Serial.println(distance);
```

```
                }

        }
```

OUTPUT :-

CONCLUSION:

4 LM35 temperature Sensor

TEMPERATURE SENSOR INTERFACING. Roll No.:-

AIM: Interface Temperature sensor with Arduino UNO board & write the program to measure temperature and display it on serial monitor.

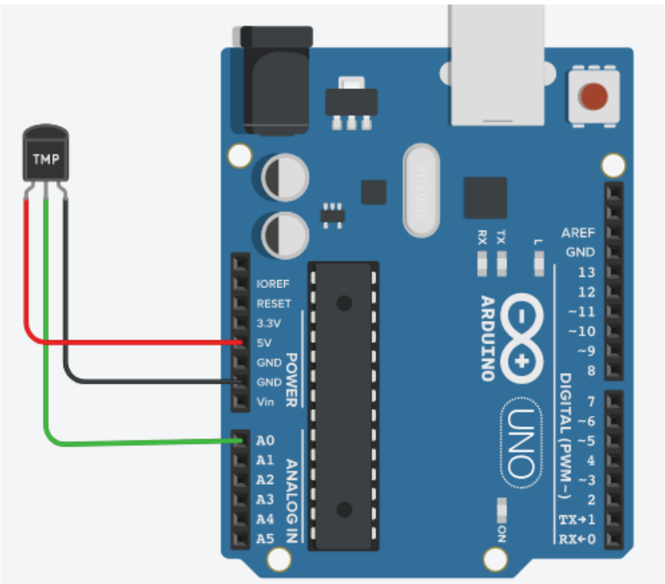
PREREQUISITES:

1. Knowledge of Arduino Uno Board interfaces.
2. Tool- Arduino IDE.

OBJECTIVES:

1. To connect & operate temperature sensor connected to pins of an Arduino.
2. To understand concept of Interfacing with microcontroller.
3. To understand requirement of microcontroller for interfacing any external devices.

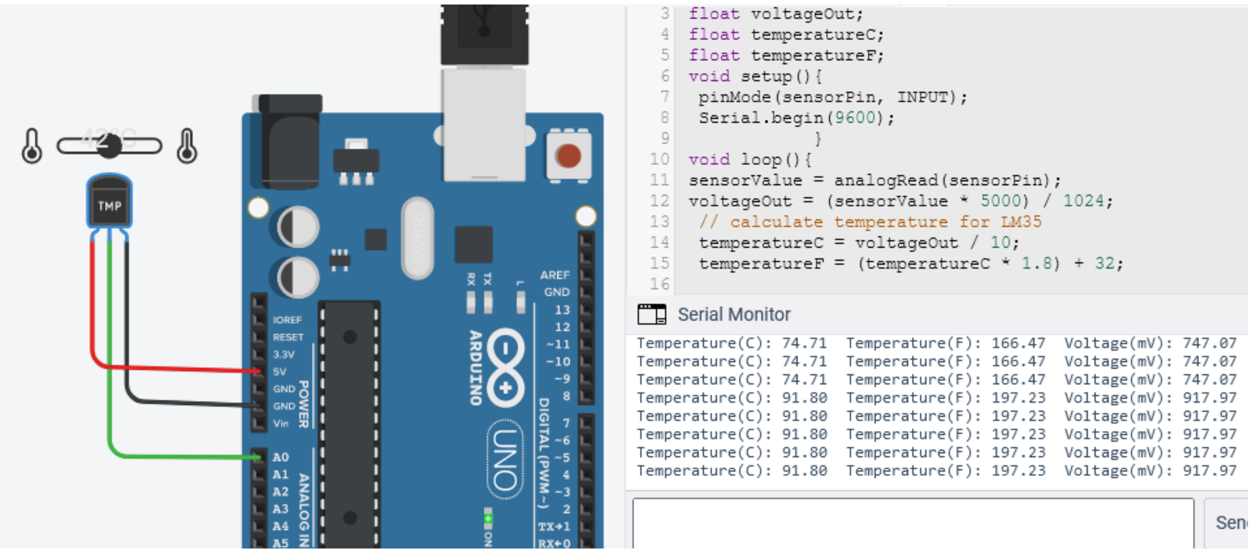
SCHEMATIC DIAGRAM:



```
const int sensorPin = A0;
float sensorValue;
float voltageOut;

float temperatureC;
float temperatureF;

void setup() {
    pinMode(sensorPin, INPUT);
    Serial.begin(9600);
}

void loop() {
    sensorValue = analogRead(sensorPin);
```

```
        voltageOut = (sensorValue * 5000) / 1024;

        // calculate temperature for LM35
        temperatureC = (voltageOut / 10)-50; //-50 for tinkercad sensor calibration
        temperatureF = (temperatureC * 1.8) + 32;

        Serial.print("Temperature(C):␣");
        Serial.print(temperatureC);
        Serial.print("Temperature(F):␣");
        Serial.print(temperatureF);
        Serial.print("␣Voltage(mV):␣");
        Serial.println(voltageOut);
        delay(1000);
    }
```

OUTPUT :-



```
float voltageOut;
float temperatureC;
float temperatureF;
void setup(){
 pinMode(sensorPin, INPUT);
 Serial.begin(9600);
             }
void loop(){
sensorValue = analogRead(sensorPin);
voltageOut = (sensorValue * 5000) / 1024;
 // calculate temperature for LM35
 temperatureC = voltageOut / 10;
 temperatureF = (temperatureC * 1.8) + 32;

```

```
Temperature(C): 74.71  Temperature(F): 166.47  Voltage(mV): 747.07
Temperature(C): 74.71  Temperature(F): 166.47  Voltage(mV): 747.07
Temperature(C): 74.71  Temperature(F): 166.47  Voltage(mV): 747.07
Temperature(C): 91.80  Temperature(F): 197.23  Voltage(mV): 917.97
Temperature(C): 91.80  Temperature(F): 197.23  Voltage(mV): 917.97
Temperature(C): 91.80  Temperature(F): 197.23  Voltage(mV): 917.97
Temperature(C): 91.80  Temperature(F): 197.23  Voltage(mV): 917.97
Temperature(C): 91.80  Temperature(F): 197.23  Voltage(mV): 917.97
```

CONCLUSION:

____

____

Click to see video

4.1 Temperature sensor with Fan

TEMPERATURE SENSOR INTERFACING. Roll No.:-

AIM: To interface a temperature sensor to Arduino UNO board & write the program to switch on a relay to operate a fan if temperature exceeds given threshold. Also display the temperature on serial monitor.

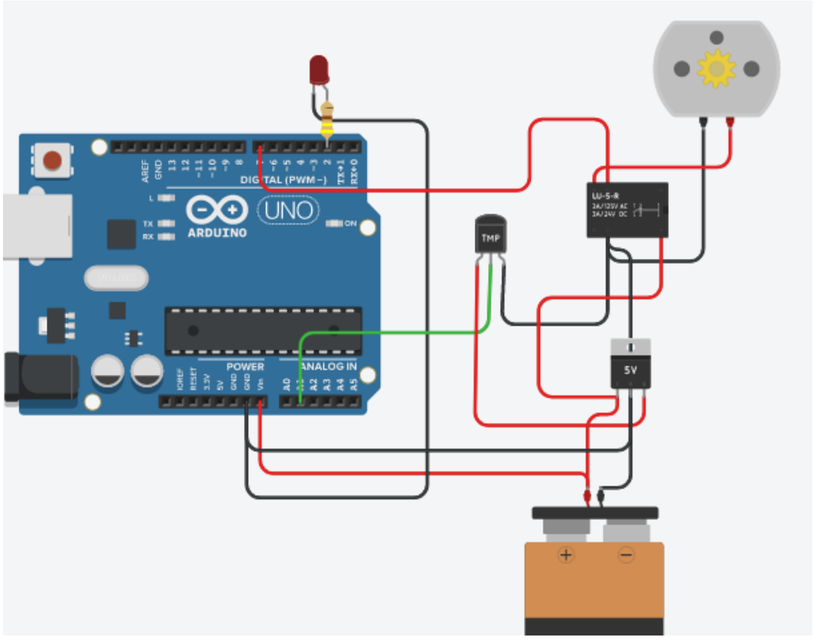
PREREQUISITES:

1. Knowledge of Arduino Uno Board interfaces.
2. Tool- Arduino IDE.

OBJECTIVES:

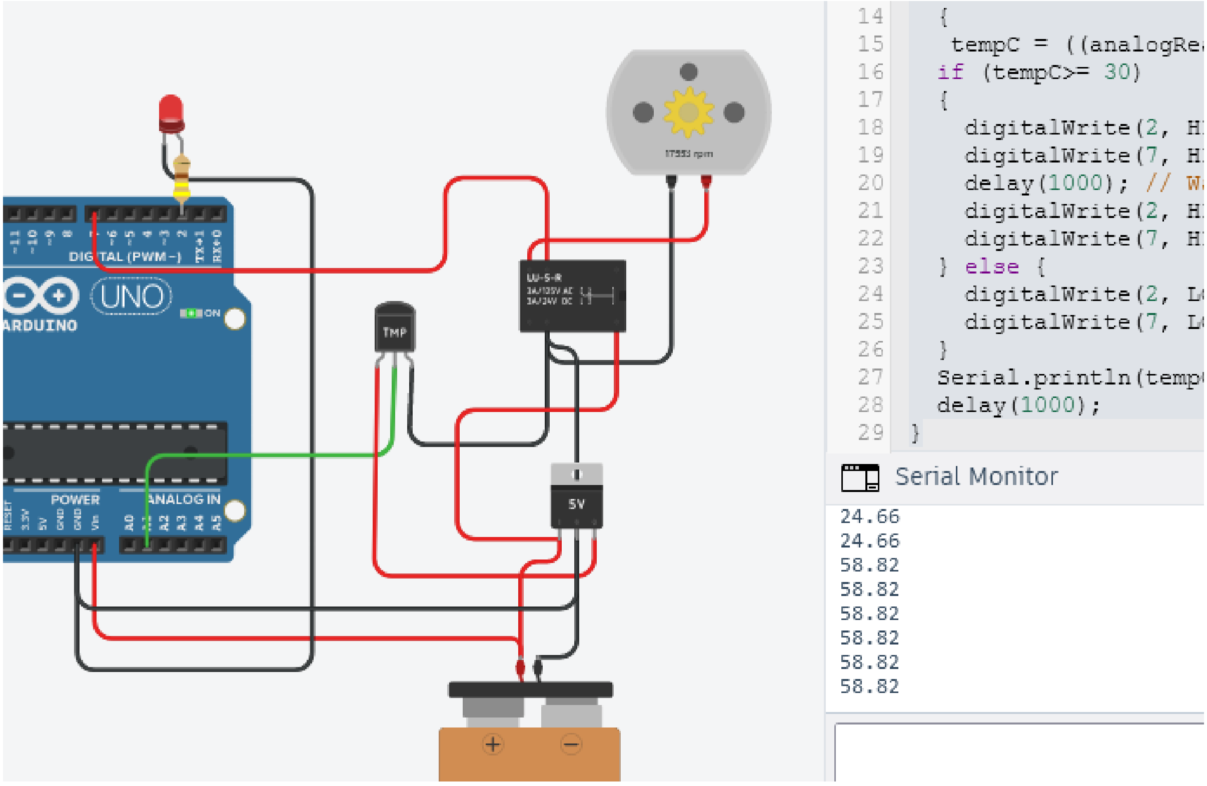
1. To connect & operate temperature sensor connected to pins of an Arduino.
2. To understand concept of Interfacing with microcontroller.
3. To understand requirement of microcontroller for interfacing any external devices.

SCHEMATIC DIAGRAM:



```
float tempC=0;
void setup()
{

    pinMode(2, OUTPUT);
    pinMode(7, OUTPUT);
    pinMode(A1, INPUT);
    Serial.begin(9600);
}

void loop()
{
    tempC = ((analogRead(A1)*4.88)/10 - 50);
    if (tempC>= 30)
    {
        digitalWrite(2, HIGH);
        digitalWrite(7, HIGH);
        delay(1000); // Wait for 1000 millisecond(s)
        digitalWrite(2, HIGH);
        digitalWrite(7, HIGH);
    } else {
        digitalWrite(2, LOW);
        digitalWrite(7, LOW);
    }
    Serial.println(tempC);
    delay(1000);
}
```

OUTPUT :-



```
  {
   tempC = ((analogRe
  if (tempC>= 30)
  {
    digitalWrite(2, HI
    digitalWrite(7, HI
    delay(1000); // W
    digitalWrite(2, HI
    digitalWrite(7, HI
  } else {
    digitalWrite(2, LO
    digitalWrite(7, LO
  }
  Serial.println(temp
  delay(1000);
}
```

Serial Monitor

```
24.66
24.66
58.82
58.82
58.82
58.82
58.82
58.82
```

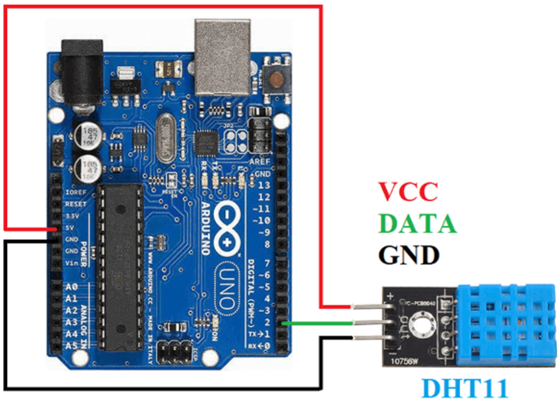
CONCLUSION:

5 Temperature/Humidity Sensor interfacing with Arduino

DHT11/22-Temperature/Humidity Sensor interfacing with Arduino Roll No.:-

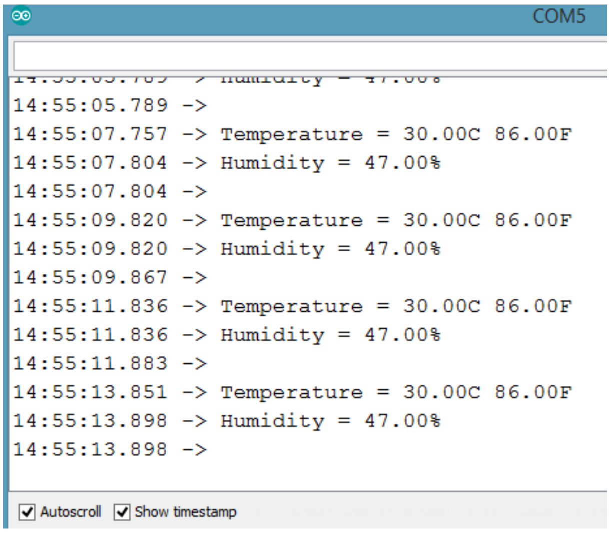
AIM:- To interface a temperature sensor DHT11 to Arduino UNO board & write the program to display the temperature on serial monitor.

Installation of dht library :- In order to establish communication with DHT11 sensor, its library needs to be installed. To install the library, navigate to Sketch - Include Library - Manage Libraries. Wait for the Library Manager to download the libraries index and update the list of installed libraries. Search for **dhtlib**. Install the library before interfacing the sensor to arduino. **SCHEMATIC DIAGRAM:**



```
#include <dht.h>          // Include library
#define outPin 2          // Defines pin number to which the sensor is connected
dht DHT;                  // Creates a DHT object
void setup() {
    Serial.begin(9600);
}

void loop() {
    int readData = DHT.read11(outPin);

    float t = DHT.temperature;         // Read temperature
    float h = DHT.humidity;            // Read humidity

    Serial.print("Temperature␣=␣");
    Serial.print(t);
    Serial.print("C␣");
    Serial.print((t*9.0)/5.0+32.0);        // Convert celsius to fahrenheit
    Serial.println("F␣");
    Serial.print("Humidity␣=␣");
    Serial.print(h);
    Serial.println("%␣");
    Serial.println("");

    delay(2000); // wait two seconds
  }


}
```

OUTPUT :-



```
14:55:05.789 -> Humidity = 47.00%
14:55:05.789 ->
14:55:07.757 -> Temperature = 30.00C 86.00F
14:55:07.804 -> Humidity = 47.00%
14:55:07.804 ->
14:55:09.820 -> Temperature = 30.00C 86.00F
14:55:09.820 -> Humidity = 47.00%
14:55:09.867 ->
14:55:11.836 -> Temperature = 30.00C 86.00F
14:55:11.836 -> Humidity = 47.00%
14:55:11.883 ->
14:55:13.851 -> Temperature = 30.00C 86.00F
14:55:13.898 -> Humidity = 47.00%
14:55:13.898 ->
```

☑ Autoscroll ☑ Show timestamp

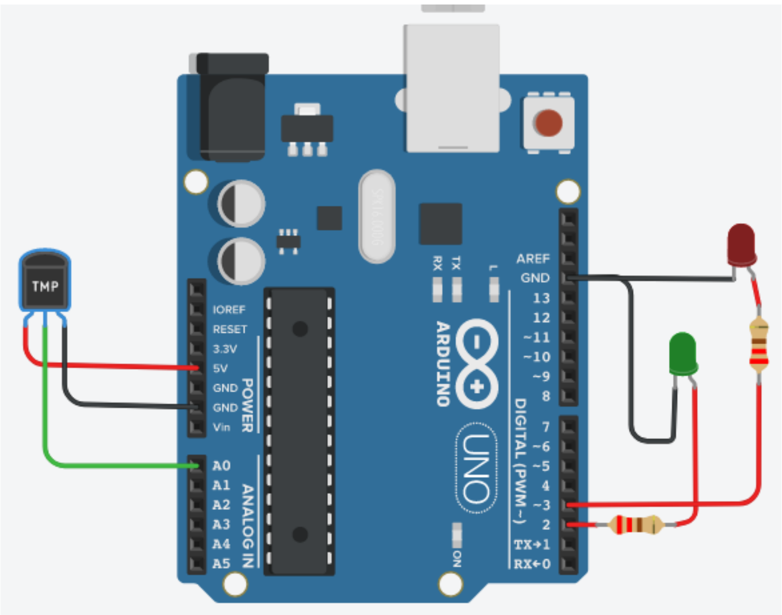
CONCLUSION:

5.1 Temperature controlled switching with Arduino

Temperature controlled switching with Arduino Roll No.:-

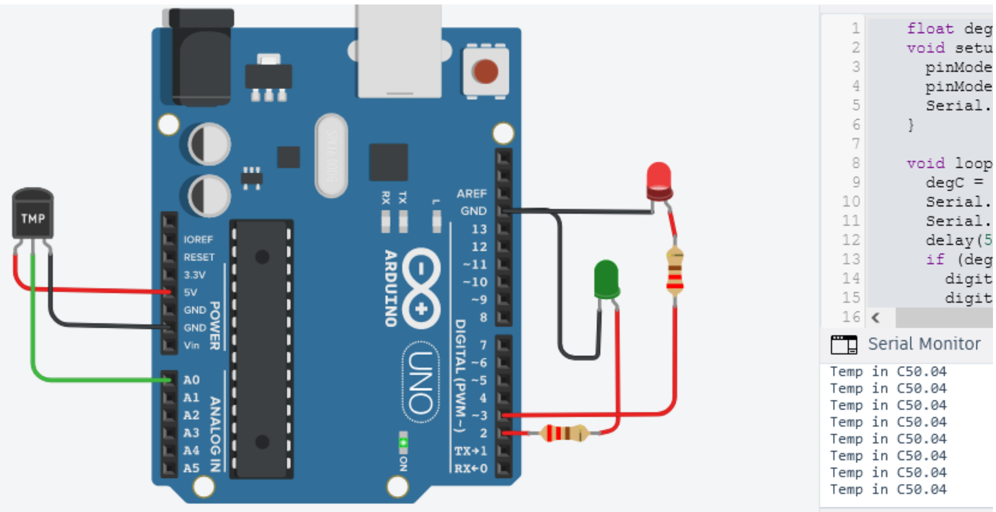
AIM:- Interface a temperature sensor to Arduino UNO board & write the program to switch ON Green LED if temperature is between 25 degree to 40 degree else turn ON the Red LED.

SCHEMATIC DIAGRAM:



```
float degC;
void setup() {
    pinMode(2, OUTPUT);
    pinMode(3, OUTPUT);
    Serial.begin(9600);
}

void loop() {
    degC = ((analogRead(A0)*4.88)/10 - 50);
    Serial.print("Temp␣in␣C");
    Serial.println(degC);
    delay(500);
    if (degC < 25){
        digitalWrite(2, LOW);
        digitalWrite(3, LOW);
    }
    if (degC>= 25 && degC< 40) {
        digitalWrite(2, HIGH);
        digitalWrite(3, LOW);
    }
    if (degC >= 40 ) {
        digitalWrite(2, LOW);
        digitalWrite(3, HIGH);
    }
}
```

OUTPUT :-



```
    float deg
    void setu
      pinMode
      pinMode
      Serial.
    }

    void loop
      degC =
      Serial.
      Serial.
      delay(5
      if (deg
        digit
        digit
```

```
Temp in C50.04
Temp in C50.04
Temp in C50.04
Temp in C50.04
Temp in C50.04
Temp in C50.04
Temp in C50.04
Temp in C50.04
```

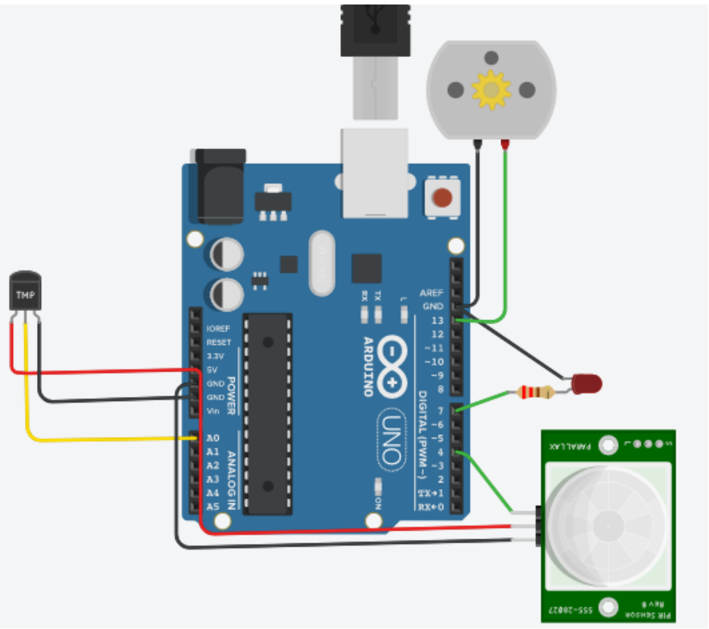
CONCLUSION:

5.2 PIR,Temperature FAN interfacing with Arduino

Temperature controlled switching with Arduino Roll No.:-

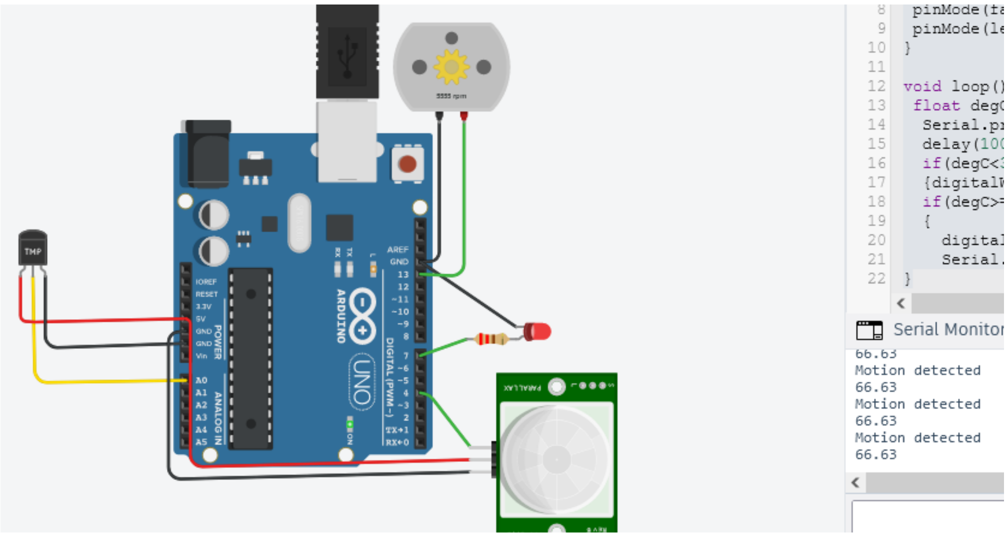
AIM:- Interface a temperature sensor, PIR Sensor to Arduino UNO board & write the program to switch on a fan if temperature exceeds 30 degrees and person is detected. Also display the temperature on serial monitor

SCHEMATIC DIAGRAM:



```
int pir=4;
int fan=13;
int led1=7;

void setup() {
    Serial.begin(9600);
    pinMode(pir,INPUT);
    pinMode(fan,OUTPUT);
    pinMode(led1,OUTPUT);
}

void loop() {
    float degC= ((analogRead(A0)*4.88)/10 - 50);;//stores the value read by sensor
    Serial.println(degC);
    delay(1000);
    if(degC<30)
    {digitalWrite(fan,LOW);digitalWrite(7,LOW);}
    if(degC>=30 && digitalRead(pir==HIGH))
    {
        digitalWrite(fan,HIGH);
        Serial.println("Motion␣detected");digitalWrite(7,HIGH);}
}
```

OUTPUT :-



```
 pinMode(fa
 pinMode(le
}

void loop()
 float degC
  Serial.pr
  delay(100
  if(degC<3
  {digitalW
  if(degC>=
  {
    digital
    Serial.
}
```

```
66.63
Motion detected
66.63
Motion detected
66.63
Motion detected
66.63
```

CONCLUSION:

Chapter V

Appendix

Copyright Disclaimer