

ST. JOSEPH COLLEGE OF ENGINEERING

SRIPERUMBUDUR



DEPARTMENT OF INFORMATION TECHNOLOGY

LAB RECORD

CS3691 – EMBEDDED SYSTEMS AND IOT

(AS PER ANNA UNIVERSITY REGULATIONS 2021)

B.Tech - IT-V SEMESTER

2023-2024

**ST.JOSEPH COLLEGE OF ENGINEERING
SRIPERUMBUDUR**

CS3691- EMBEDDED SYSTEMS AND IOT



NAME OF THE STUDENT :

DEPARTMENT :

REGISTER NO :

ROLL NO :

YEAR/SEM :

INDEX

S.No	Title of the Experiment	Page No.
1.		
2.		
3.		
4.		
5.		
6.		
7.		
8.		
9.		
10.		
11.		

Ex.No: 1

Date :

Addition and Subtraction of two Hexadecimal Numbers using 8051 Assembly Language Programming

Aim :

To perform addition and subtraction of two hexa decimal numbers using 8051 assembly Language Programming

Software Required :

1. Keil μ Vision Software

Program for Addition :

Algorithm:

- (i) Load the address of the first operand into R0
- (ii) Load the address of the second operand into R1
- (iii) Load the value at the address in R0 into accumulator A
- (iv) Copy the value to register R5
- (v) Initialize a counter in R4 to 0
- (vi) Load the value at the address in R1 into accumulator A
- (vii) Add the value in accumulator A to the value in R5
- (viii) Jump to SAVE if there is no carry (no overflow)
- (ix) If there is a carry, increment the counter in R4
- (x) Move the value in R4 to the memory location pointed to by R1
- (xi) Increment the address in R1 to point to the next location
- (xii) Save the result (accumulator A) at the address in R1

Program:

ORG 00H

SJMP START

ORG 30H

START:

MOV R0,#20H

MOV R1,#30H

MOV A,@R0

MOV R5,A

MOV R4,#00H

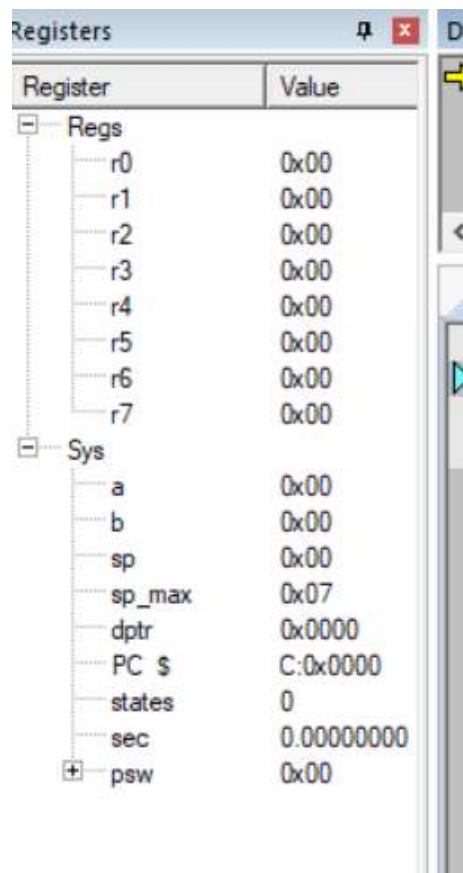
```

INC R0
MOV A,@R1
ADD A,R5
JNC SAVE
INC R4
MOV B,R4
MOV @R1,B
INC R1
SAVE: MOV @R1,A
HALT: SJMP HALT
END

```

Execution :



Before Execution:



Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x00
sp_max	0x07
dptr	0x0000
PC \$	C:0x0000
states	0
sec	0.00000000
psw	0x00

After Execution :

Register	Value
[-] Regs	
r0	0x21
r1	0x30
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
[-] Sys	
a	0x00
b	0x00
sp	0x07
sp_max	0x07
dptr	0x0000
PC \$	C:0x0044
states	4798749
sec	5.20697591
+ psw	0x00

 Project  Registers

Program for Subtraction:

Algorithm:

- (i) Load the address of the first operand into R0
- (ii) Load the address of the second operand into R1
- (iii) Load the value at the address in R0 into accumulator A
- (iv) Copy the value to register R5
- (v) Initialize a counter in R4 to 0
- (vi) Load the value at the address in R1 into accumulator A
- (vii) Subtract the value in accumulator A to the value in R5
- (viii) Jump to SAVE if there is no borrow (no overflow)
- (ix) If there is a carry, increment the counter in R4
- (x) Move the value in R4 to the memory location pointed to by R1
- (xi) Increment the address in R1 to point to the next location
- (xii) Save the result (accumulator A) at the address in R1

Program:

ORG 00H

SJMP START

ORG 30H

START:

MOV R0, #20H

MOV R1, #30H

MOV A, @R0

MOV R5, A

MOV R4, #00H

INC R0

MOV A, @R1

SUB A, R5

JNC SAVE

INC R4

MOV B, R4

MOV @R1, B

INC R1

SAVE: MOV @R1, A



HALT: SJMP HALT

END

Execution :

Before Execution:

Register	Value
<input type="checkbox"/> Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
<input type="checkbox"/> Sys	
a	0x00
b	0x00
sp	0x00
sp_max	0x07
dptr	0x0000
PC \$	C:0x0000
states	0
sec	0.00000000
<input checked="" type="checkbox"/> psw	0x00

 Project  Registers

After Execution :

Register	Value
[-] Regs	
r0	0x74
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
[-] Sys	
a	0x00
b	0x00
sp	0x07
sp_max	0x07
dptr	0x0000
PC \$	C:0x0803
states	4310546
sec	4.46265100
[+] psw	0x00

Project Registers

Results:

Thus the addition and subtraction of hexadecimal numbers is performed using simulator software Keil μ Vision.

Ex.No: 2

Date :

Multiplication and Division of two Hexadecimal Numbers using 8051 Assembly Language Programming

Aim :

To perform multiplication and division. of two hexa decimal numbers using 8051 assembly Language Programming

Algorithm:

- (i) Initialize Address of the first hexadecimal number
- (ii) Initialize Address of the second hexadecimal number
- (iii) Initialize Address to store the result
- (iv) Clear accumulator A
- (v) Clear register B
- (vi) Initialize Outer loop for the first hexadecimal number
- (vii) Initialize Counter for the inner loop
- (viii) Load a byte from the first hexadecimal number
- (ix) Copy to carry for rotation
- (x) Load a byte from the second hexadecimal number
- (xi) Multiply A and B (B holds the first number)
- (xii) Add the previous value in @R2 to the result
- (xiii) Store the result back in memory
- (xiv) Move to the next memory location
- (xv) Move to the next byte of the second number
- (xvi) Decrement inner loop counter and repeat if not zero
- (xvii) Move to the next byte of the first number
- (xviii) Check if the outer loop counter is zero, if not, repeat
- (xix) End the program with an infinite loop

Program:

```
MOV R0, #20H
MOV R1, #30H
MOV R2, #40H
MOV A, #00H
MOV B, #00H
OUTER_LOOP:
    MOV R3, #04H
```

INNER_LOOP:

MOV A, @R0

MOV C, A

MOV A, @R1

MUL AB

ADD A, @R2

MOV @R2, A

INC R2

INC R1

DJNZ R3, INNER_LOOP

DEC R0

JNZ OUTER_LOOP

END

Before Execution:

Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x07
sp_max	0x07
dptr	0x0000
PC	0x0000
states	0
sec	0.00000000
psw	0x00

After Execution:

Register	Value
Regs	
r0	0x35
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x07
sp_max	0x07
dptr	0x0000
PC \$	C:0x0803
states	5767151
sec	5.97065480
psw	0x00

Division of two numbers

Algorithm:

- (i) Set the origin address
- (ii) Initialize data pointer to the data section
- (iii) Load the first byte of the dividend from memory location 0x0030
- (iv) Store it in R0
- (v) Move to the next byte in the data section
- (vi) Load the second byte of the dividend from memory location 0x0031
- (vii) Move to the memory location 0x0032 (divisor)
- (viii) Load the divisor from memory location 0x0032
- (ix) Store it in R1
- (x) Divide the 16-bit number in R0 and R1 by the divisor in R1

- (xi) Quotient is stored in R2
- (xii) Remainder is stored in R3
- (xiii) our result is now in R2 (quotient) and R3 (remainder)

Program

ORG 0x0000

MOV DPTR, #0x0030

MOVX A, @DPTR

MOV R0, A

INC DPTR

MOVX A, @DPTR

INC DPTR

MOVX A, @DPTR

MOV R1, A

DIV AB

MOV R2, A

MOV R3, B

END

Register	Value
[-] Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
[-] Sys	
a	0x00
b	0x00
sp	0x07
sp_max	0x07
dptr	0x0000
PC \$	C:0x0000
states	0
sec	0.000000...
[+] psw	0x00

Memory 1	
Address:	0x0030
C:0x0030:	5E 1F 00
C:0x004B:	00 00
C:0x0066:	00 00
C:0x0081:	00 00
C:0x009C:	00 00
C:0x00B7:	00 00
C:0x00D2:	00 00
C:0x00ED:	00 00
Simulation	
t1: 5.29918989 sec	
L:136 C:1	
CAP NUM SCRL OVR R/W	
31°C Mostly cloudy	
12:26 PM	
11/17/2023	

After Execution :

Register	Value
Regs	
r0	0x04
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x07
sp_max	0x07
dptr	0x0000
PC \$	C:0x0804
states	5277548
sec	5.29918989
psw	0x00

Result:

Thus the Multiplication and subtraction of hexadecimal numbers is performed using simulator software Keil μ Vision.

Ex.No: 3

Date :

Test the data transfer between Data and Memory using 8051 Assembly Language Programming

Aim :

To test the data transfer between data and memory using 8051 assembly Language Programming

Algorithm:

- (i) Set the origin address
- (ii) Load a byte of data into register A
- (iii) Set the destination memory location (replace with your desired memory location)
- (iv) Store the data in the specified memory location
- (v) Set the source memory location
- (vi) Load the data from the specified memory location into register A
- (vii) Register A contains the data that was transferred from memory

Program

ORG 0x0000

MOV A, #0x55

MOV DPTR, #0x0030

MOVX @DPTR, A

MOV DPTR, #0x0030

MOVX B, @DPTR

END

Before Execution:

Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x00
sp_max	0x07
dptr	0x0000
PC \$	C:0x0000
states	0
sec	0.00000000
psw	0x00

After Execution:

Register	Value
Regs	
r0	0x2f
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x07
sp_max	0x07
dptr	0x0000
PC \$	C:0x0804
states	8364523
sec	8.65967952
psw	0x00

Result:

Thus the transfer of data is performed using Keil μ Vision simulator software.

Ex.No: 4

Date :

Program on performing AND, OR and XOR operations on Hexa Decimal Numbers

Aim :

To test the data transfer between data and memory using 8051 assembly Language Programming

Program to perform AND operation on HexaDecimal Numbers

Algorithm:

- (i) Initialize Starting address of the program
- (ii) Load the first 8-bit number (37H in hexadecimal) into register R1
- (iii) Load the second 8-bit number (15H in hexadecimal) into register R2
- (iv) Move the content of register R1 to the accumulator A
- (v) Perform bitwise AND operation with the content of Register R2
- (vi) Store the result in REGISTER
- (vii) End of the program

Program:

```
ORG 0H
MOV R1, #37H
MOV R2, #15H
MOV A, R1
ANL A, R2
MOV R0, A
END
```

Before Execution:

Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x00
sp_max	0x07
dptr	0x0000
PC \$	C:0x0000
states	0
sec	0.00000000
psw	0x00

After Execution:

Register	Value
Regs	
r0	0x12
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x07
sp_max	0x07
dptr	0x0000
PC \$	C:0x0803
states	4966694
sec	5.14195230
psw	0x00

Program to perform OR operation on HexaDecimal Numbers

Algorithm:

- (i) Initialize Starting address of the program
- (ii) Load the first 8-bit number (37H in hexadecimal) into register R1
- (iii) Load the second 8-bit number (15H in hexadecimal) into register R2
- (iv) Move the content of register R1 to the accumulator A
- (v) Perform bitwise OR operation with the content of Register R2
- (vi) Store the result in REGISTER
- (vii) End of the program

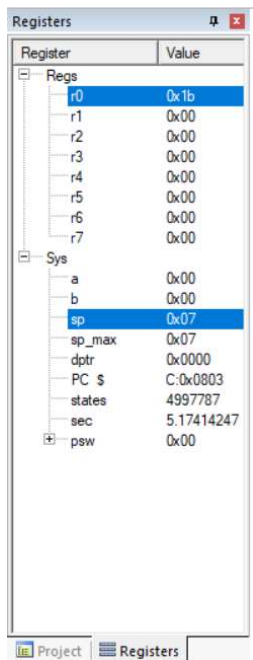
Program:

```
ORG 0H  
MOV R1, #37H  
MOV R2, #15H  
MOV A, R1  
ORL A, R2  
MOV R0, A  
END
```

Before Execution:

Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x00
sp_max	0x07
dptr	0x0000
PC \$	C:0x0000
states	0
sec	0.00000000
psw	0x00

After Execution:



Register	Value
r0	0x1b
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
a	0x00
b	0x00
sp	0x07
sp_max	0x07
dptr	0x0000
PC	0x0803
states	4997787
sec	5.17414247
psw	0x00

Program to perform XOR operation on HexaDecimal Numbers

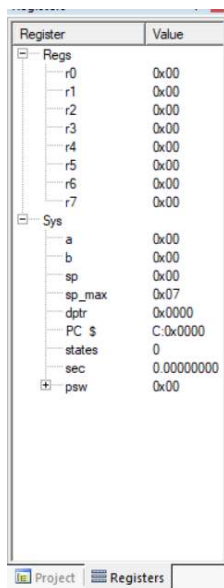
Algorithm:

- (i) Initialize Starting address of the program
- (ii) Load the first 8-bit number (37H in hexadecimal) into register R1
- (iii) Load the second 8-bit number (15H in hexadecimal) into register R2
- (iv) Move the content of register R1 to the accumulator A
- (v) Perform bitwise XOR operation with the content of Register R2
- (vi) Store the result in REGISTER R0
- (vii) End of the program

Program:

```
ORG 0H
MOV R1, #37H
MOV R2, #15H
MOV A, R1
XRL A, R2
MOV R0, A
END
```

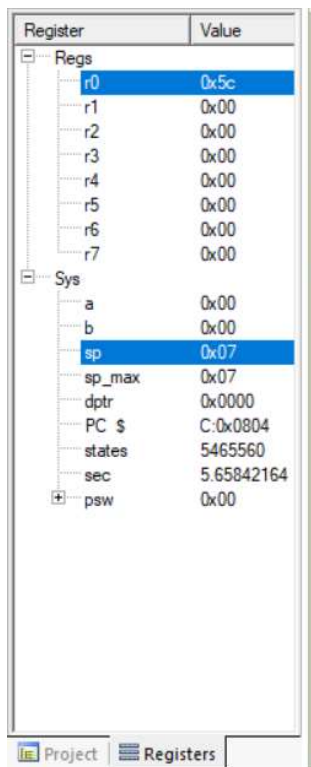
Before Execution:



The image shows the 'Registers' window in Keil uVision before execution. It displays a list of registers and their current values. The 'Regs' section contains r0 through r7, all with a value of 0x00. The 'Sys' section contains a, b, sp, sp_max, dptr, PC \$, states, sec, and psw. The PC \$ register shows the address C:0x0000. The 'Project' and 'Registers' tabs are visible at the bottom.

Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x00
sp_max	0x07
dptr	0x0000
PC \$	C:0x0000
states	0
sec	0.00000000
psw	0x00

After Execution:



The image shows the 'Registers' window in Keil uVision after execution. The values of several registers have changed. r0 is now 0x5c. The stack pointer (sp) is 0x07. The program counter (PC \$) is C:0x0804. The 'states' register is 5465560. The 'sec' register is 5.65842164. The 'Project' and 'Registers' tabs are visible at the bottom.

Register	Value
Regs	
r0	0x5c
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x07
sp_max	0x07
dptr	0x0000
PC \$	C:0x0804
states	5465560
sec	5.65842164
psw	0x00

Result:

Thus the Bit wise Boolean operations are performed on the Hexadecimal data using Keil μ Vision Software.

Ex.No: 5

Date :

Program on Addition , Subtraction and Multiplication of two hexadecimal numbers using Embedded C

Aim :

To write a program to add , subtract and multiply two hexadecimal numbers using Embedded C.

Program:

```
#INCLUDE <reg51.h>
void main()
(
    unsigned int i,j;
    i=0x89;
    j=0x23;
    P0=i+j;
    P1=i-j;
    P2=i*j;

}
```

Before Execution:

Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x00
sp_max	0x07
dptr	0x0000
PC s	C:0x0000
states	0
sec	0.00000000
psw	0x00

Parallel Port 0

Port 0

P0: 0xFF

Pins: 0xFF

Parallel Port 1

Port 1

P1: 0xFF

Pins: 0xFF

Parallel Port 2

Port 2

P2: 0xFF

Pins: 0xFF

After Execution:

Parallel Port 0

Port 0

P0: 0xAC

Pins: 0xAC

7	6	5	4	3	2	1	0
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Parallel Port 1

Port 1

P1: 0x66

Pins: 0x66

7	6	5	4	3	2	1	0
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Parallel Port 2

Port 2

P2: 0xBB

Pins: 0xBB

7	6	5	4	3	2	1	0
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Result:

Thus the program is written to add, subtract and multiply two hexadecimal numbers using embedded C.

Ex.No: 6

Date :

Program to find sum of first ten numbers using Embedded C

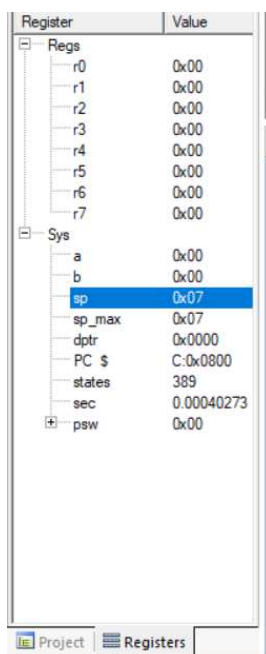
Aim :

To write a program to find the sum of first 10 numbers using Embedded C.

Program:

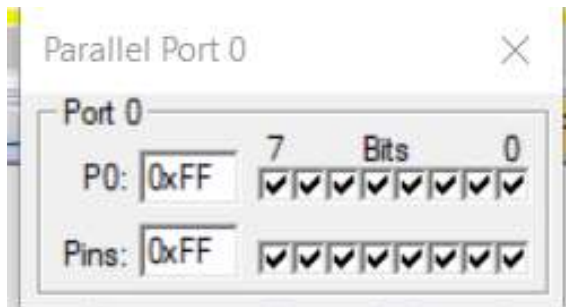
```
void main() {  
    unsigned int i;  
    unsigned int sum = 0;  
    for (i = 1; i <= 10; i++) {  
        sum += i;  
    }  
    P0 = sum;  
    while (1) {  
    }  
}
```

Before Execution:



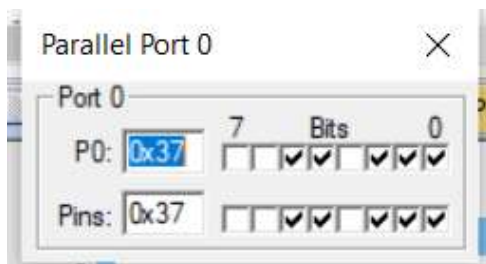
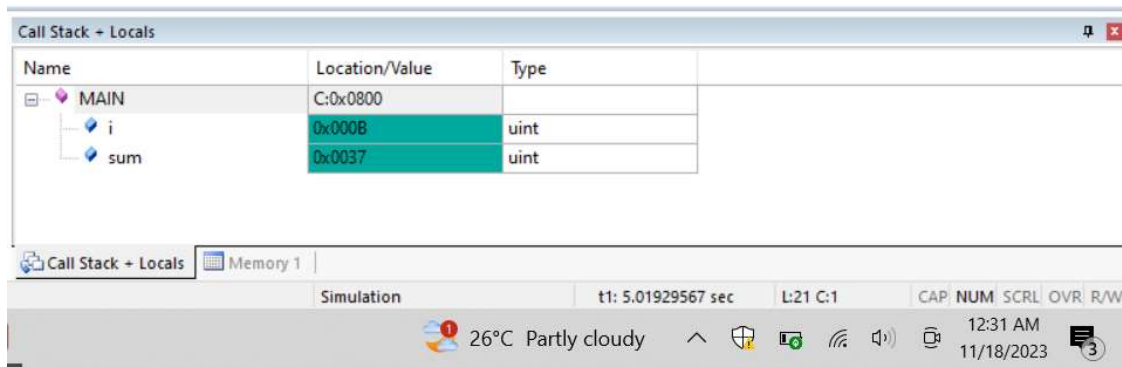
Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x07
sp_max	0x07
dptr	0x0000
PC \$	C:0x0800
states	389
sec	0.00040273
psw	0x00

Call Stack + Locals		
Name	Location/Value	Type
MAIN	C:0x0800	
i	0x0000	uint
sum	0x0000	uint



After Execution:

Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x0b
r6	0x00
r7	0x37
Sys	
a	0x00
b	0x00
sp	0x07
sp_max	0x07
dptra	0x0000
PC \$	C:0x0819
states	4848218
sec	5.019295...
psw	0x00



Result :

Thus the sum of first 10 numbers has been calculated using embedded C.

Ex.No: 7

Date :

Smallest and Largest Number in an array

Aim :

To find the smallest and largest number in an array.

Largest Number in an array:

```
#include <reg51.h>
```

```
#define ARRAY_SIZE 6
```

```
void main()
```

```
{
```

```
    unsigned char array[ARRAY_SIZE] = {0x25, 0x55, 0x13, 0x62, 0x0A, 0xFF};
```

```
    unsigned char largest = array[0];
```

```
    unsigned char smallest = array[0];
```

```
    unsigned char i;
```

```
    for (i = 1; i < ARRAY_SIZE; i++)
```

```
    {
```

```
        if (array[i] > largest)
```

```
        {
```

```
            largest = array[i];
```

```
        }
```

```
        if (array[i] < smallest)
```

```
        {
```

```
            smallest = array[i];
```

```
        }
```

```
}
```

```
P0 = largest;
```

```
P1 = smallest;
```

```
while (1)
```

```
{
```

```
}
```

```
}
```

Before Execution:

Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x0d
sp_max	0x0d
dptr	0x0000
PC \$	C:0x08F6
states	389
sec	0.00040273
psw	0x00

Parallel Port 0

Port 0

P0: 0xFF 7 Bits 0

Pins: 0xFF

Parallel Port 1

Port 1

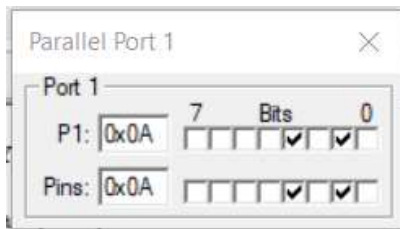
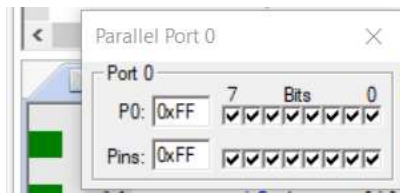
P1: 0xFF 7 Bits 0

Pins: 0xFF

Call Stack + Locals		
Name	Location/Value	Type
MAIN	C:0x08F6	
array	D:0x08 ""	array[6] of uchar
largest	0x00	uchar
smallest	0x00	uchar
i	0x00	uchar

After Execution:

Register	Value
Regs	
r0	0x0d
r1	0x08
r2	0x00
r3	0x00
r4	0xff
r5	0x06
r6	0x0a
r7	0xff
Sys	
a	0xf5
b	0x08
sp	0x0d
sp_max	0x0f
dp_ptr	0x093e
PC \$	C:0x092A
states	5885019
sec	6.09268197
psw	0x00



Call Stack + Locals		
Name	Location/Value	Type
MAIN	C:0x08F6	
array	D:0x08 "%U0b\ny"	array[6] of uchar
largest	0xFF 'y'	uchar
smallest	0x0A	uchar
i	0x06	uchar

Result :

Thus the largest and smallest number of an array is found using Keil μ Vision Software.

Ex.No: 8

Date :

LED Control using Arduino

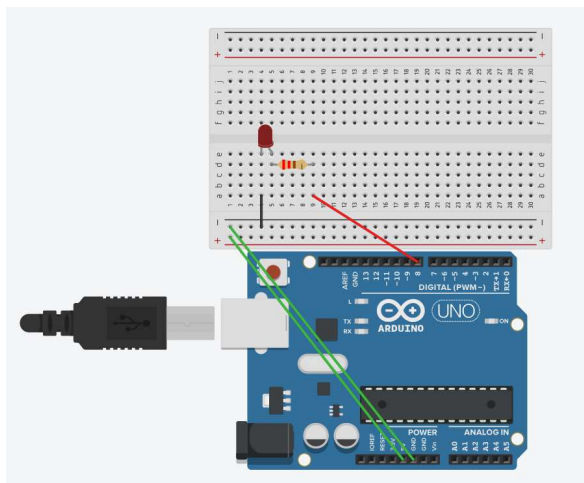
Aim :

To control LED using Arduino board.

Apparatus Required:

S.No	Apparatus	Quantity
1.	Arduino Board	1
2.	Bread Board	1
3.	LED	1
4.	Power Jack	1
5.	USB Cable	1
6.	Jumper wires	As Required

Circuit Connection:



Hardware Procedure:

- (i) LED pin is Connected to Arduino Uno pin of 2.
- (ii) Power jack is connected to the Arduino Uno.
- (iii) USB connector is connected to Arduino Uno to monitor.
- (iv) Connect the 12V power supply to development board.
- (v) Check the output from the development board.

Software Procedure:

- (i) Click on Arduino IDE
- (ii) Click on file
- (iii) Click on New

- (iv) Write a Program as per circuit Pin connections
- (v) Click on Save
- (vi) Click on Verify
- (vii) Click on Upload the code into Arduino Uno by using USB cable.

Program:

```
int ledPin=8; //definition digital 8 pins as pin to control the LED

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

void loop()
{
    digitalWrite(ledPin,HIGH);    //HIGH is set to about 5V PIN8
    delay(1000);                  //Set the delay time, 1000 = 1S
    digitalWrite(ledPin,LOW);     //LOW is set to about 5V PIN8
    delay(1000);                  //Set the delay time, 1000 = 1S
}
```

Result :

LED is successfully controlled using Arduino microcontroller board

Ex.No: 8

Date :

Interfacing Buzzer Alarm with Arduino

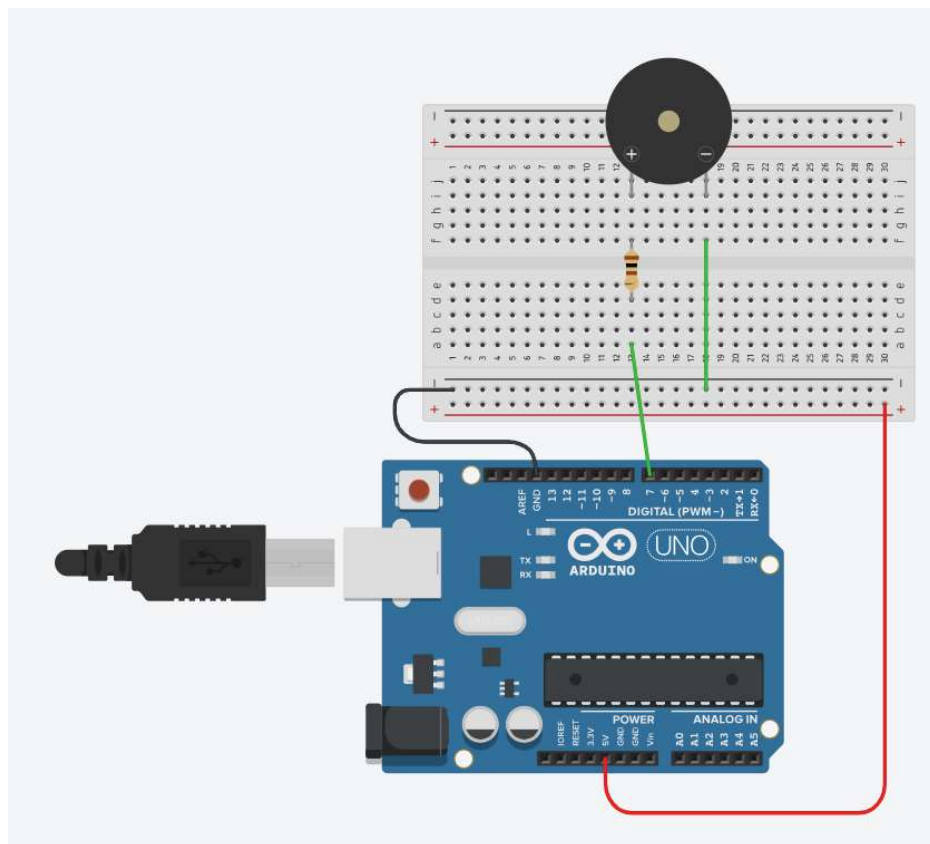
Aim :

To interface a buzzer with Arduino board

Apparatus Required:

S.No	Apparatus	Quantity
1.	Arduino Board	1
2.	Bread Board	1
3.	LED	1
4.	Power Jack	1
5.	USB Cable	1
6.	Resistor (100 Ω)	1
7.	Jumper wires	As Required

Connection Setup for interfacing Arduino with buzzer:



Hardware Procedure:

- (i) Positive pin of Buzzer is Connected to Arduino Uno pin of 12. Negative pin is connected to ground.
- (ii) Power jack is connected to the Arduino Uno.
- (iii) USB connector is connected to Arduino Uno to monitor.
- (iv) Connect the 12V power supply to development board.
- (v) Check the output from the development board.

Software Procedure:

- (i) Click on Arduino IDE
- (ii) Click on file
- (iii) Click on New
- (iv) Write a Program as per circuit Pin connections
- (v) Click on Save
- (vi) Click on Verify
- (vii) Click on Upload the code into Arduino Uno by using USB cable.

Program:

```
void setup() {  
    pinMode(7, OUTPUT);  
}  
  
void loop()  
{  
    tone(7, 220, 100);  
    delay(200);  
}
```

Result :

Thus the buzzer sound is produced using Buzzer device and Arduino.

Hardware Procedure:

- (i) Positive pin of Buzzer is Connected to Arduino Uno pin of 12. Negative pin is connected to ground.
- (ii) Power jack is connected to the Arduino Uno.
- (iii) USB connector is connected to Arduino Uno to monitor.
- (iv) Connect the 12V power supply to development board.
- (v) Check the output from the development board.

Software Procedure:

- (i) Click on Arduino IDE
- (ii) Click on file
- (iii) Click on New
- (iv) Write a Program as per circuit Pin connections
- (v) Click on Save
- (vi) Click on Verify
- (vii) Click on Upload the code into Arduino Uno by using USB cable.

Program:

```
void setup() {  
    pinMode(7, OUTPUT);  
}  
  
void loop()  
{  
    tone(7, 220, 100);  
    delay(200);  
}
```

Result :

Thus the buzzer sound is produced using Buzzer device and Arduino.

Ex.No: 10

Date :

Interfacing Bluetooth HC-05 with Arduino

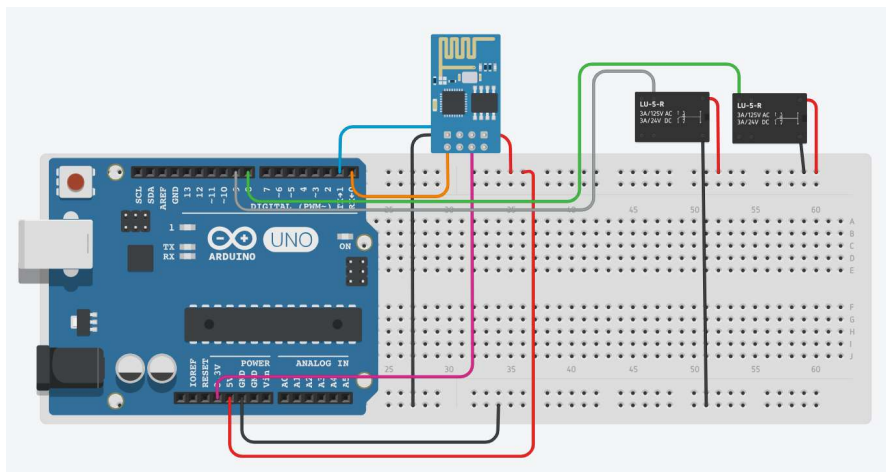
Aim :

To interface a Bluetooth with Arduino board

Apparatus Required:

S.No	Apparatus	Quantity
1.	Arduino Board	1
2.	Bread Board	1
3.	LED	1
4.	Power Jack	1
5.	USB Cable	1
6.	Resistor (100 Ω)	1
7.	Bluetooth HC - 05	1
7.	Jumper wires	As Required

Circuit Connections:



Hardware Procedure:

- (i) Connections are given as per the circuit above
- (ii) Power jack is connected to the Arduino Uno.
- (iii) USB connector is connected to Arduino Uno to monitor.
- (iv) Connect the 12V power supply to development board.
- (v) Check the output from the development board.

Software Procedure:

- (i) Click on Arduino IDE

- (ii) Click on file
- (iii) Click on New
- (iv) Write a Program as per circuit Pin connections
- (v) Click on Save
- (vi) Click on Verify
- (vii) Click on Upload the code into Arduino Uno by using USB cable.
- (viii) Download Bluetooth serial controller APP from Google Play
- (ix) Send Data as “A” or “B” from the App

Program :

```

int led=13;

int data;

void setup()
{
  serial. begin (9600);
  pinmode(13, OUTPUT);
}

Void loop()
{
  While ( serial.available () > 0)
  {
    Data=serial. read()
    Serial. println(data)
    If (data == 'A')
    {
      digitalWrite(13, HIGH);
    }
    If (data == 'B')
    {
      Digitalwrite (13, LOW);
    }
  }
}

```

}

Result:

Thus the message from mobile is transferred to Bluetooth through Arduino.

Ex.No: 10

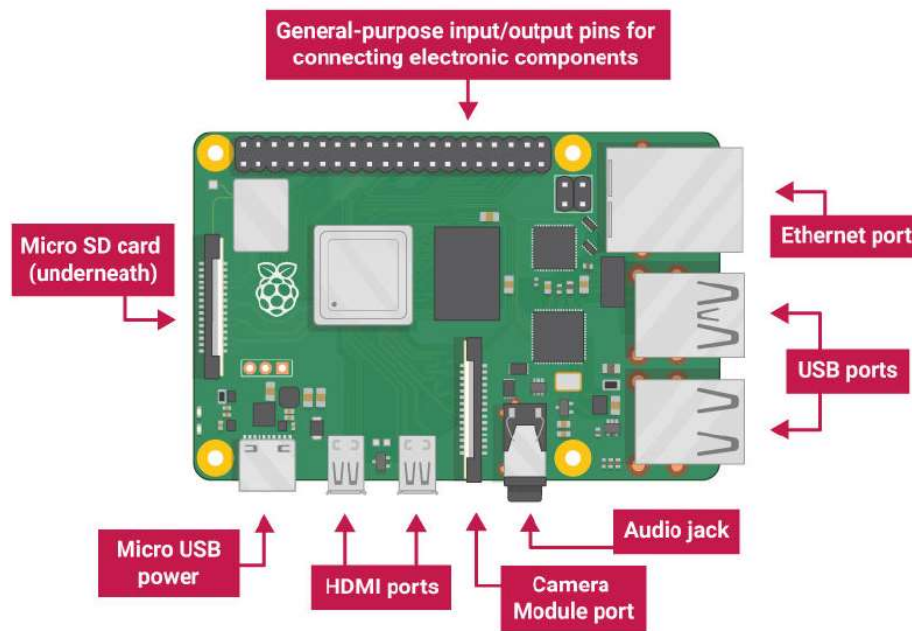
Date :

Introduction to Raspberry Pi Platform and Python Programming

Aim :

To introduce about the Raspberry Pi board and programming Raspberry Pi in python

Introduction to Raspberry Pi:



USB Port – Used to connect a mouse and a keyboard

SD Card Slot – SD Card can be slotted here

Ethernet Port – Used to connect Raspberry Pi with the network through the cable

Audio Jack – Can connect headphones and speakers

HDMI Port – the monitor (or projector) can be connected to display the output from Raspberry Pi.

Micro USB Power Connector – Power Supply is connected here

GPIO Ports – Allows us to connect electronic components such as LEDs and buttons to Raspberry Pi.

Setting up our SD Card:

- The Raspberry Pi imager is an easiest way to install the Raspberry Pi OS.
- The Raspberry Pi imager software is available in [**Raspberry Pi Downloads Page**](#)
- Insert the SD Card into the computer or SD Card Slot.
- Install the Raspberry Pi Imager software in the SD Card.

Connecting our Raspberry Pi:

- The slot for SD Card is available in the underside of the Raspberry Pi.
- Connect the mouse and keyboard through the USB Slots.

Starting Up Raspberry Pi

- Plug the power supply into the socket and connect it to Raspberry Pi Power Port.
- A red LED Light lights up in Raspberry Pi indicating that Raspberry Pi is connected to power.
- After few seconds the Raspberry Pi OS Desktop will appear.

Finishing the set up:

- When we start the Raspberry Pi for the first time, the Welcome to Raspberry Pi application will pop – up and will guide through the entire set up.
- Click Next to start the setup.
- Set your Country, Language and Time zone, then click next again.
- Enter the new password for your Raspberry Pi and click next.
- Connect to your WiFi Network by selecting its name, entering the password and clicking next.
- Click next for the wizard check for updates to Raspbian and install them .
- Click done or reboot to finish the setup.

Introduction to Python

Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems. This versatility, along with its beginner-friendliness, has made it one of the most-used programming languages today.

Python is commonly used for developing websites and software, task automation, data analysis, and data visualization. Since it's relatively easy to learn, Python has been adopted by many non-programmers such as accountants and scientists, for a variety of everyday tasks, like organizing finances.

Python finds its applications in the fields of :

- Data analysis and machine learning

- Web development
- Automation or scripting
- Software testing and prototyping
- Everyday tasks

Reason for Popularity of python :

Python is popular for a number of reasons. Here's a deeper look at what makes it so versatile and easy to use for coders.

- It has a simple syntax that mimics natural language, so it's easier to read and understand. This makes it quicker to build projects, and faster to improve on them.
- It's versatile. Python can be used for many different tasks, from web development to machine learning.
- It's beginner friendly, making it popular for entry-level coders.
- It's open source, which means it's free to use and distribute, even for commercial purposes.
- Python's archive of modules and libraries—bundles of code that third-party users have created to expand Python's capabilities—is vast and growing.
- Python has a large and active community that contributes to Python's pool of modules and libraries, and acts as a helpful resource for other programmers. The vast support community means that if coders run into a stumbling block, finding a solution is relatively easy; somebody is bound to have encountered the same problem before.

Result :

Thus a brief introduction to Raspberry Pi Platform and python programming is seen.

Ex.No: 11

Date :

Interfacing light sensor with Raspberry Pi

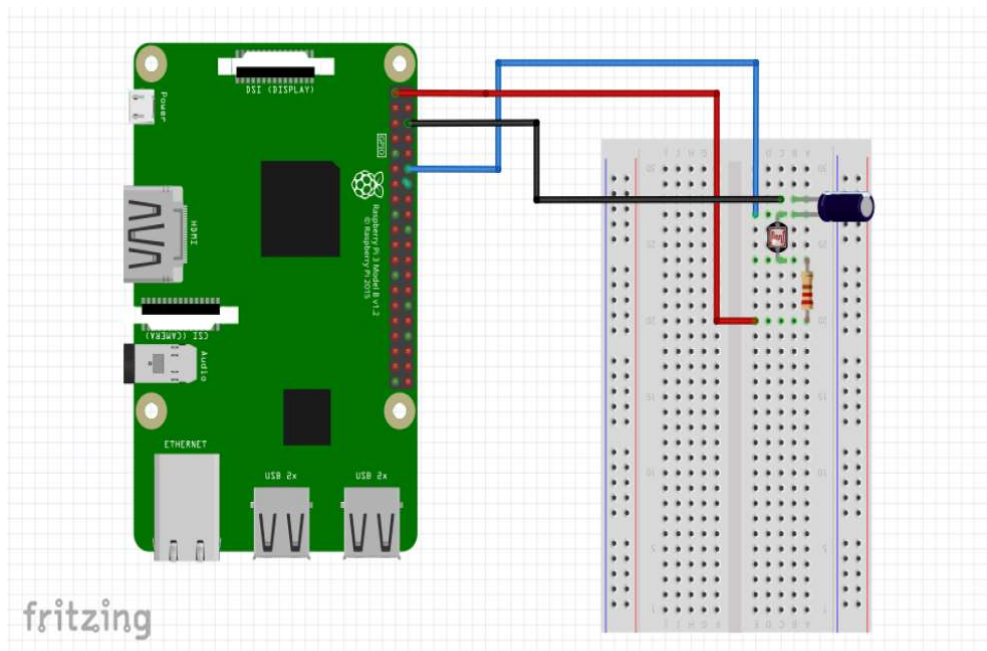
Aim :

To monitor the surrounding light intensity using light sensors and Raspberry Pi

Apparatus Required:

S.No	Apparatus	Quantity
1.	Raspberry Pi Board	1
2.	Bread Board	1
3.	Photocell	1
4.	Resistor (1 k Ω)	1
5.	Capacitor (1 μ F)	1

Circuit Connection:

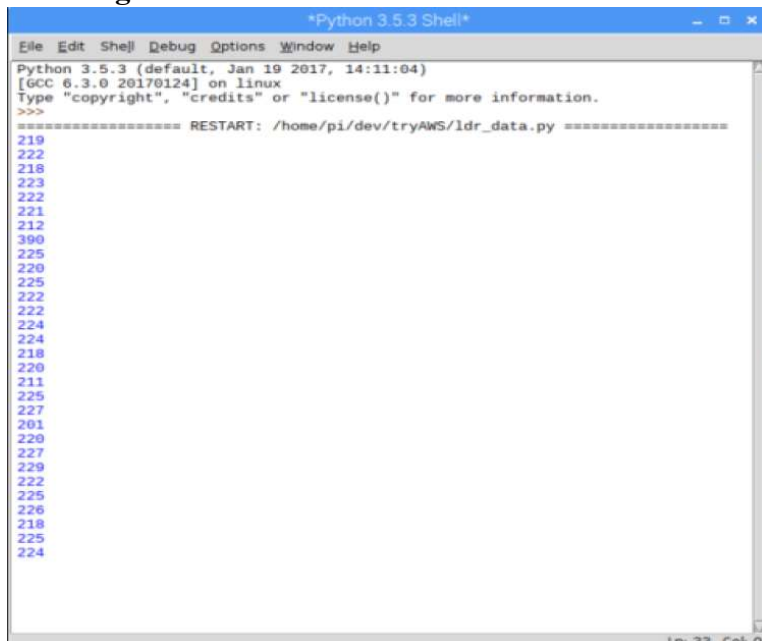


Hardware Connections:

- Insert the photocell in the breadboard.
- Connect the GPIO pin 1 (3.3 V) to the resistor which is connected serial to the photocell.
- Connect the other end of the photocell to the GPIO pin 12 and the capacitor as shown in the diagram above.
- GPIO pin 6 (ground) is connected to the other end of the capacitor (short end).

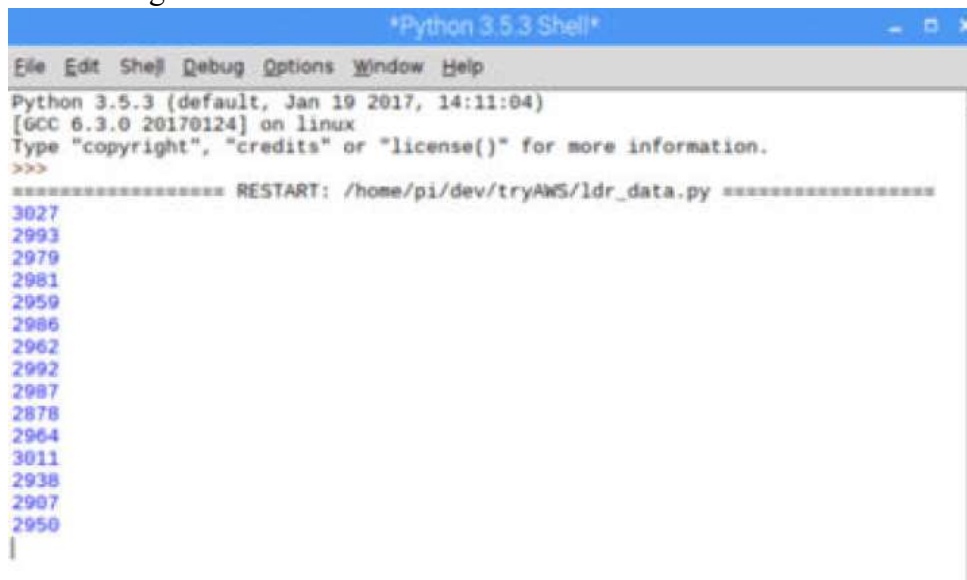
Code:

```
Import RPi.GPIO as GPIO.tie.os
DEBUG = 1;
GPIO.setmode (GPIO.BOARD)
GPIO.setwarnings(False)
def RCTime(RCpin):
reading = 0
GPIO.setup(RCpin.GPIO.OUT)
GPIO.output(RCpin.GPIO.LOW)
Time.sleep(2)
GPIO.setup(RCpin.GPIO.IN)
While(GPIO.input(RCpin)==GPIO.LOW);
Reading+=1
Return reading
While true:
Print RCTime(12)
```

Output:**1. With Light:**

```
*Python 3.5.3 Shell*
File Edit Shell Debug Options Window Help
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/pi/dev/tryAWS/ldr_data.py =====
219
222
218
223
222
221
212
390
225
220
225
222
222
224
224
218
220
211
225
227
201
220
227
220
222
225
226
218
225
224
Ln: 33 Col: 0
```

2. Without Light



```
*Python 3.5.3 Shell*
File Edit Shell Debug Options Window Help
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/pi/dev/tryAWS/ldr_data.py =====
3027
2993
2979
2981
2959
2986
2962
2992
2987
2878
2964
3011
2938
2907
2950
|
```

Result :

Thus the experiment on the sensing of light with the help of Raspberry Pi is done.

Ex.No: 12

Date :

Soil Moisture Sensing using Arduino and Soil Moisture Sensor

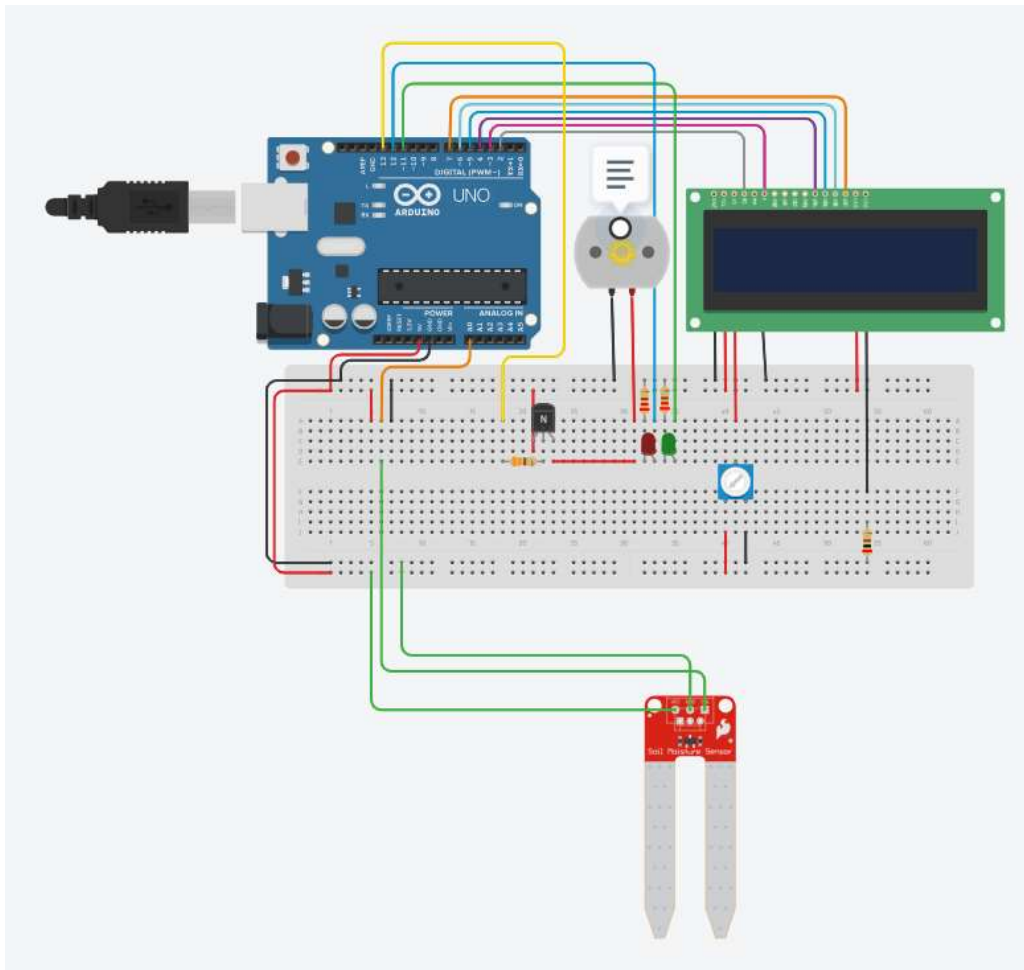
Aim :

To develop a mini project on soil moisture sensing with soil moisture sensor and Arduino.

Apparatus Required:

S.No	Apparatus	Quantity
1.	Arduino Board	1
2.	Soil moisture sensor(EC1258)	1
3.	LCD Display	1
4.	Resistor (1 k Ω)	1
5.	Capacitor (1 μ F)	1

Circuit Connection:



Code:

```
#include <LiquidCrystal.h>
```

```
const int LM35 = A0;
```

```
const int motor = 13;
```

```
const int LedRed = 12;
```

```
const int LedGreen = 11;
```

```
int percentValue = 0;
```

```
LiquidCrystal lcd(2, 3, 4, 5, 6, 7);
```

```
void setup() {
```

```
    Serial.begin(9600);
```

```
    lcd.begin(16, 2);
```

```
    lcd.print("Automated Crop");
```

```
    lcd.setCursor(0,1);
```

```
    lcd.print("Watering System!");
```

```
    pinMode(motor, OUTPUT);
```

```
    pinMode(LedRed, OUTPUT);
```

```
    pinMode(LedGreen, OUTPUT);
```

```
    delay(2000);
```

```
    lcd.clear();
```

```
    lcd.print("SoilM = ");
```

```
    lcd.setCursor(0,1);
```

```
    lcd.print("WaterPump= ");
```

```
}
```

```

void loop() {

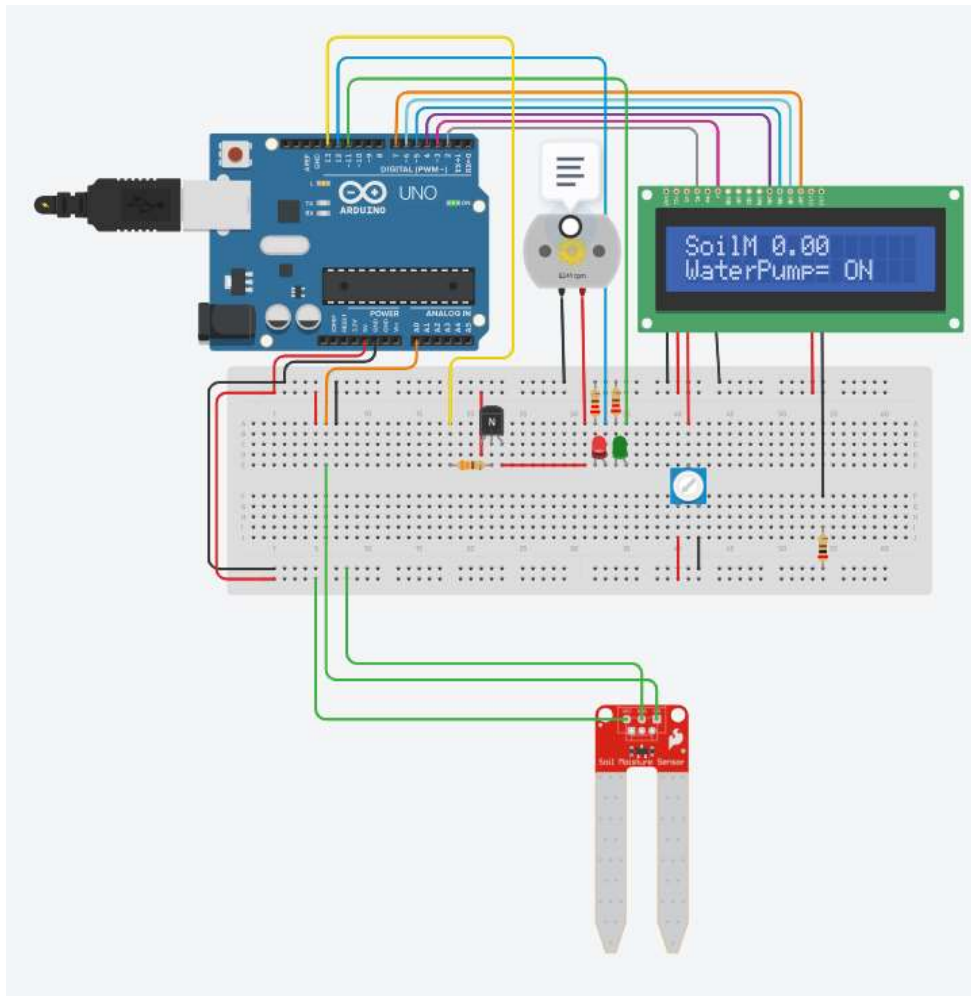
    int value = analogRead(LM35);
    float Moisture = value * 500.0 / 1023.0;
    lcd.setCursor(6,0);
    lcd.print(Moisture);
    lcd.setCursor(11,1);

    if (Moisture < 310){
        digitalWrite(motor, HIGH);
        digitalWrite(LedRed, HIGH);
        digitalWrite(LedGreen, LOW);
        lcd.print("ON ");
    }
    else {
        digitalWrite(motor, LOW);
        digitalWrite(LedRed, LOW);
        digitalWrite(LedGreen, HIGH);
        lcd.print("OFF");
    }

    Serial.print ("Moisture");
    Serial.println (Moisture);
}

```

Output:



Result :

Thus this mini project of soil moisture sensing is done with the help of soil moisture sensor and Arduino microcontroller.