MADRAS INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY

DEPARTMENT OF INFORMATION TECHNOLOGY

IT5601 EMBEDDED SYSTEMS AND INTERNET OF THINGS LABORATORY

RECORD

REGISTER NUMBER: 2019506076

NAME: ROZEN BERG D

SEMESTER: 6/8

DEPARTMENT OF INFORMATION TECHNOLOGY

ANNA UNIVERSITY, MIT CAMPUS

CHROMEPET, CHENNAI – 600 044

BONAFIDE CERTIFICATE

Certified that the bonafide record of the practical work done by Mr.______Rozen Berg D_______Register Number (2019506076) of **Sixth** Semester, **B.Tech Information Technology** in the **IT5601 EMBEDDED SYSTEMS AND INTERNET OF THINGS** during the academic period from **March 2022 to June 2022**

Submitted for Practical Examination held on ______________

Date: Course Instructor

D Bala Gayathri

Internal Examiner External Examiner

TABLE OF CONTENTS

EX. NO	DATE	TITLE	PAGE NO	SIGNATURE
01.	09/03/22	Study of 8051 Microcontroller and Instruction Set	01	
02.	16/03/22	8-bit Arithmetic Operations	13	
03.	30/03/22	16-bit Arithmetic Operations	17	
04.	13/04/22	Multibyte Arithmetic Operations	20	
05.	20/04/22	Sorting Program	24	
06.	27/04/22	Code Convertors	26	
07.	11/05/22	Addition of N-Numbers	29	
08.	01/06/22	Timer delay using Embedded C on Keil	30	
09.	01/06/22	Serial communications using Embedded C on Keil	33	
10.	08/06/22	Timer delay using ALP on Keil	36	
11.	08/06/22	Serial communications using ALP on Keil	39	
12.	07/05/22	Blinking a LED	42	
13.	07/05/22	Measuring Temperature using Temperature Sensor	44	
14.	21/05/22	Infrared Sensor using Node MCU	46	
15.	14/05/22	Light Sensor	48	
16.	14/05/22	Proximity Detection	50	
17.	14/05/22	Servo Motor	53	
18.	14/05/22	Potentiometer with Servo Motor	55	
19.	14/05/22	IoT with Cloud	57	
20.	19/05/22	Making a Node Server	59	
21.	19/05/22	Interfacing GPS	62	
22.	19/05/22	Interfacing Bluetooth	68	
23.	23/05/22	Interfacing GSM	70	
24.	23/05/22	Collecting and Processing Data from IoT Systems in the Cloud using ThingSpeak	72	
25.	23/05/22	Develop IoT Applications using Bluemix	75	

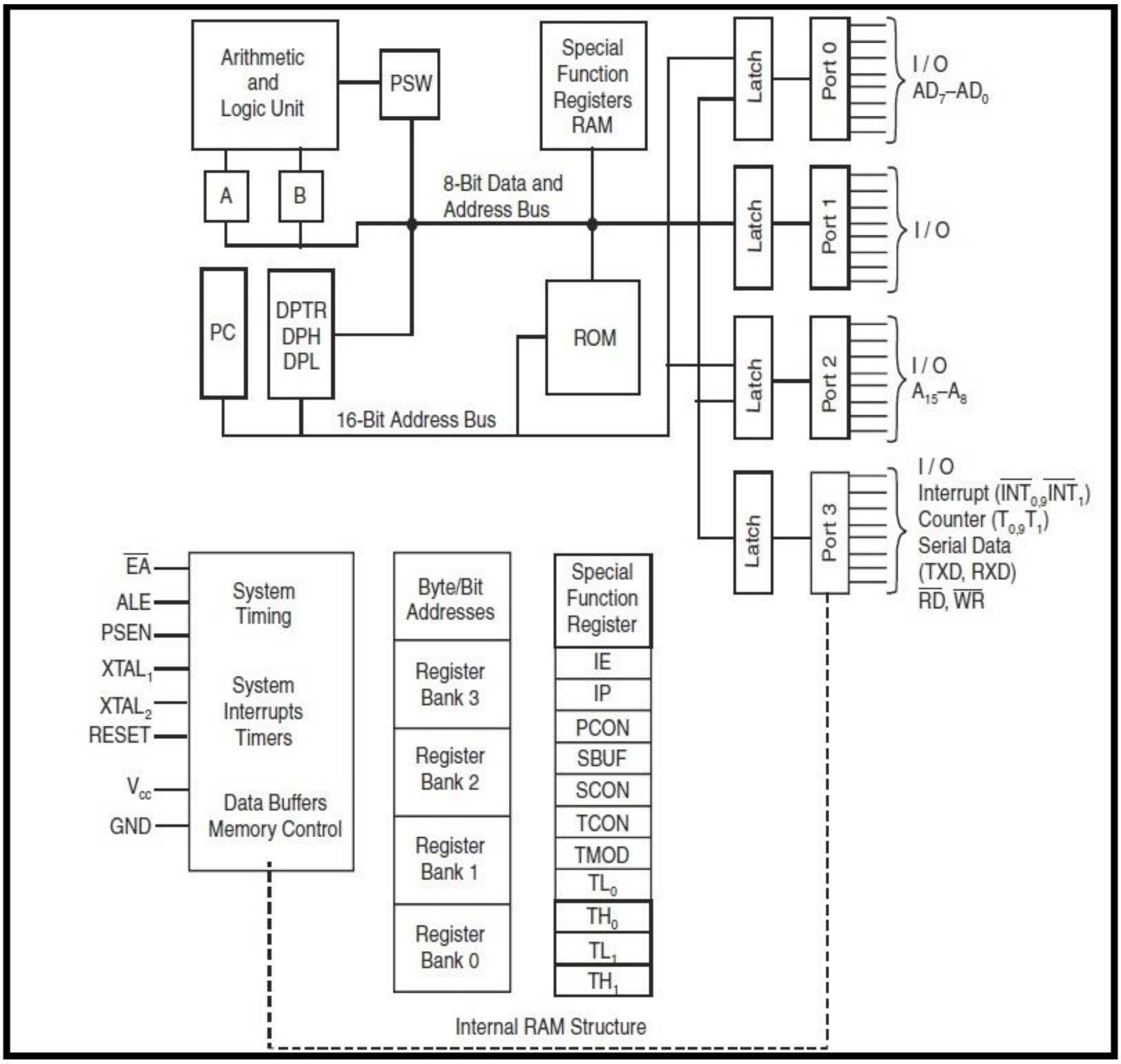
Exp.No: 1 **Date:** 09-03-22

Study of 8051 Embedded Systems Architecture and Instruction Set

Aim:

To perform a study of 8051 Embedded Systems Architecture and Instruction Set

Architecture:



INSTRUCTION SET

The 8051 Instruction Set is optimized for 8-bit content application. It offers possibilities in control area, serial I/O, arithmetic, byte and bit manipulation. It can be classified into

1. Data Transfer Instructions
2. Arithmetic Instructions
3. Logical Instructions
4. Boolean or Bit Manipulation Instructions
5. Program Branching Instructions

DATA TRANFER INSTRUCTIONS:

OPERATION	MNEMONICS	DESCRIPTION
Register to register	MOV A, Rn	[A]<-[Rn]
	MOV Rn, A	[Rn]<-[A]
	XCH A, Rn	[A]<-[Rn]
Memory to register	MOV A, @Rn	[A]<-[Address in register]
	MOV A, address	[A]<-[Address]
	MOV Rn, address	[Rn]<-[Address]
	MOVX A, @Rn	[A]<-[Address in External ROM]
	MOVC A, @A+DPTR	[A]<-[Address in Internal ROM]
	MOVC A, @A+PC	[A]<-[Address in Internal ROM]
	MOVX A, @DPTR	[A]<-[Address in External ROM]

	XCH A, @Rn	[A]<-[Address]
	XCHD A, @Rn	[A]<-[Address]
	XCH A, address	[A]<-[Address]
Register to memory	MOVX @Ri, A	[Address]<-[A]
	MOV a8, A	[Address]<-[A]
	MOV a8, Rn	[Address]<-[Rn]
	MOVX @DPTR, A	[Address]<-[A]
	MOV @Rn, A	[Address]<-[A]
Data to Register	MOV A, #data	[A]<-[Data]
	MOV Rn, #data	[Rn]<-[Data]
	MOV DPTR, #data	[DPTR]<-[Data]
Address to Address	MOV a8, @Rn	[Address]<-[Address]
	MOV address, address	[Address]<-[Address]
	MOV @Rn, address	[Address]<-[Address]

Data to address	MOV address, #data	[Address]<-[Data]
	MOV @Rn, #d8	[Address]<-[Data]
Stack	PUSH a8	Data added to stack
	POP a8	Data removed from the stack

ARITHMETIC INSTRUCTIONS:

OPERATION	OPERAND	DESCRIPTION
Addition	A, Rn	[A]<-[A]+[Rn]
	A, Address	[A]<-[A]+[Data at Address]
	A, @Rn	[A]<-[A]+[Data at Address pointed by Rn]
	A, #data	[A]<-[A]+[Data]
	A, Rn	[A]<-[A]+[Rn]+[Carry flag]
	A, Address	[A]<-[A]+[Data at Address]+[Carry flag]

	A, @Rn	[A]<-[A]+[Data at Address pointed by Rn]+[Carry flag]
	A, #data	[A]<-[A]+[Data]]+[Carry flag]
Subtraction	A, Rn	[A]<-[A]-[Rn]
	A, Address	[A]<-[A]-[Data at Address]
	A, @Rn	[A]<-[A]-[Data at Address pointed by Rn]
	A, #data	[A]<-[A]-[Data]
Increment	A	[A]<-[A+1]
	Rn	[Rn]<-[Rn+1]
	Address	[Data at Address]<-[Data at Address+1]
	@Rn	[Data at Address pointed by register]<-[Data at Address pointed by register+1]
	DPTR	[DPTR]<-[DPTR +1]

Decrement	A	[A]<-[A-1]
	Rn	[Rn]<-[Rn-1]
	Address	[Data at Address]<-[Data at Address-1]
	@Rn	[Data at Address pointed by register]<- [Data at Address pointed by register-1]
Multiplication	A,B	[A]<-[A]*[B]
Division	A,B	[A]<-[A]/[B]
Decimal adjust	A	Coverts binary addition to BCD

LOGICAL INSTRUCTIONS:

OPERATION	**MNEMONICS**	**DESCRIPTION**
AND	ANL A, Rn	[A]<-[A] AND [Rn]
	ANL A, Address	[A]<-[A] AND [Data at Address]
	ANL A, @Rn	[A]<-[A] AND [Data at Address in Rn]

	ANL A, #data	[A]<-[A] AND [Data]
	ANL Address, A	[Data at Address]<-[Data at Address] AND [A]
	ANL Address, #data	[Data at Address]<-[Data at Address] AND [Data]
OR	ORL A, Rn	[A]<-[A] OR [Rn]
	ORL A, Address	[A]<-[A] OR [Data at Address]
	ORL A, @Rn	[A]<-[A] OR [Data at Address in Rn]
	ORL A, #data	[A]<-[A] OR [Data]
	ORL Address, A	[Data at Address]<-[Data at Address] OR [A]
	ORL Address, #data	[Data at Address]<-[Data at Address] OR [Data]
XOR	XRL A, Rn	[A]<-[A] XOR [Rn]
	XRL A, Address	[A]<-[A] XOR [Data at Address]

	XRL A, @Rn	[A]<-[A] XOR [Data at Address in Rn]
	XRL A, #data	[A]<-[A] XOR [Data]
	XRL Address, A	[Data at Address]<-[Data at Address] XOR [A]
	XRL Address, #data	[Data at Address]<-[Data at Address] XOR [Data]
Clear	CLR A	[A]<-0
Complement	CPL A	[A]<-[A]’
Rotate left	RL A	Shifts the value in accumulator to the left
	RLC A	Shifts the value in accumulator to the left through the accumulator
Rotate right	RR A	Shifts the value in accumulator to the right
	RRC A	Shifts the value in accumulator to the right through the accumulator
Swap	SWAP A	Swaps the upper nibble of the accumulator with the lower nibble

BIT MANIPULATION INSTRUCTIONS:

OPERATION	MNEMONICS	DESCRIPTION
Clear	CLR C	[CY]<-0
	CLR Address	[Address]<-0
Complement	CPL C	[CY]<-[CY]’
	CPL Address	[Address]<-[Address]’
Setting value	SETB C	[CY]<-1
	SETB Address	[Address]<-1
AND	ANL C, Address	[CY]<-[CY] AND [Address]
	ANL C, /Address	[CY]<-[CY] AND [Address]’
OR	ORL C, Address	[CY]<-[CY] OR [Address]
	ORL C, /Address	[CY]<-[CY] OR [Address]’
Move	MOV C, Address	[CY]<-[Address]

	MOV Address,C	[Address]<-[CY]
Jump	JC Address	Jump to address if [C]=1
	JNC Address	Jump to address if [C]=0
	JNB Address, Address	Jump to destination address if source address =0
	JB Address, Address	Jump to destination address if source address =1
	JBC Address, Address	Jump to destination address if source address =1 and sets carry flag to 0

PROGRAM BRANCHING INSTRUCTIONS:

OPERATION	MNEMONICS	DESCRIPTION
Call	ACALL Address11	Calls a subroutine in the maximum address range of 2K bytes
	LCALL Address16	Calls a subroutine in the maximum address range of 64K bytes

Return	RET	Returns the control from subroutine
	RETI	Returns the control from an interrupt subroutine
Jump	AJMP Address11	Jumps to an address in a 2KB range
	LJMP Address16	Jumps to an address in a 64KB range
	SJMP Relative address	Jumps to an address in a 256-byte range (0 to 127 (0-7FH) range and -1 to -128 (FFH-80H).
	JMP @A+DPTR	[DPTR]<-[DPTR+A]
	JZ Relative address	Jumps to address when accumulator=0
	JNZ Relative address	Jumps to address when accumulator!=0
	CJNE A, Direct address, Relative address	Jumps to relative address when accumulator=data stored at a direct address
	CJNE A, #Data,Relative address	Jumps to relative address when accumulator=data given by the programmer

	CJNE @Rn, #Data,Relative address	Jumps to relative address when data at memory location stored in register=data given by the programmer
	DJNZ Rn, Relative address	Decrements value in Rn and jump to relative address till Rn!=0
	DJNZ Direct address, Relative address	Decrements value at memory location stored in a register and jump to relative address till memory location stored in register =0

Result:

Study of 8051 embedded systems architecture and instruction set has been done successfully.

Exp.No: 2 **Date:** 16-03-22

8-bit Arithmetic Operations

Aim:

To perform 8-bit addition , subtraction, multiplication and division with 8051 microcontroller.

Algorithm:

Addition:

1. Store the values to be added into two registers
2. Move first value to accumulator and add second value to accumulator
3. Accumulator stores the sum of the two values

Subtraction:

1. Store the values to be added into two registers
2. Move first value to accumulator and subtract second value from the accumulator
3. Accumulator stores the difference of the two values

Multiplication using repetitive addition:

1. Initialize by storing 00 in the accumulator
2. Move the value [n] that is to multiplied into R0
3. Add the value of multiplicand to A
4. Repeat the step 3 until R0 counter becomes zero
5. STOP

Multiplication:

1. Store the multiplicand and multiplier into A, F0 registers respectively
2. Multiply the two registers
3. Accumulator stores the product of two register values

Division:

1. Store the dividend and divisor into A, B registers respectively
2. Divide the contents of register A by register B.
3. Accumulator stores the quotient and register B stores the remainder

Code:

Addition:

ADDRESS	MNEMONICS
8500	MOV A,#30
8502	MOV R0,#10
8504	ADD A,R0
8505	LCALL 00BB

Output:

40

Subtraction:

A>B

ADDRESS	MNEMONICS
8500	MOV A,#027
8502	MOV R0,#7
8504	SUBB A,R0
8505	LCALL 00BB

Output:

20

A<B

ADDRESS	MNEMONICS
8500	MOV A,#7
8502	MOV R0,#27
8504	SUBB A,R0
8505	LCALL 00BB

Output:

EC

Multiplication using repetitive addition:

ADDRESS	MNEMONICS
8500	MOV A,#00
8502	MOV R0,#10
8504	ADD A,#2
8506	DJNZ R0,8504
8508	MOV R2,A
8509	LCALL 00BB

Output:

20

Multiplication:

Without registers

ADDRESS	MNEMONICS
8500	MOV A,#02
8502	MOV F0,#03
8505	MUL AB
8506	LCALL 00BB

Output:

06

With registers

ADDRESS	MNEMONICS
8500	MOV R1,#10
8502	MOV R2,#5
8504	MOV A,R1
8505	MOV F0,R2
8507	MUL AB
8508	MOV R3,A
8509	MOV R4,F0
850B	LCALL 00BB

Output:

50

With memory

ADDRESS	MNEMONICS
9000	MOV A,#03
9002	MOV F0,#04
9005	MUL AB
9006	MOV DPTR,#9000
9009	MOVX @DPTR,A
900A	INC DPTR
900B	MOV A,0B
900D	MOVX @DPTR,A
900E	LCALL 00BB

Output:

OC

Division:

Without registers

ADDRESS	MNEMONICS
8500	MOV A,#04
8502	MOV F0,#02
8505	DIV AB
8506	LCALL 00BB

Output:

A:2

B:0

With registers

ADDRESS	MNEMONICS
8500	MOV R1,#04
8502	MOV R2,#02
8504	MOV A,R1
8505	MOV F0,R2
8507	DIV AB
8508	MOV R3,A
8509	MOV R4,F0
850B	LCALL 00BB

Output:

R3:02

R4:00

Result:

8-bit arithmetic operations have been performed successfully and the results are verified.

Exp.No: 3 **Date:** 30-03-22

16-bit Arithmetic Operations

Aim:

To perform 16-bit addition, subtraction, multiplication and division with 8051 microcontrollers.

Algorithm:

16 Bit addition

1. Split the 16-bit numbers to be added into 8-bit numbers and store each in individual registers
2. Add the corresponding numbers and store the value in each register
3. The registers put together gives the result

16 Bit subtraction

1. Split the 16-bit numbers to be added into 8-bit numbers and store each in individual registers
2. Subtract the corresponding numbers and store the value in each register
3. The registers put together gives the result

16 Bit multiplication

1. Split the 16-bit numbers to be added into 8-bit numbers and store each in individual registers
2. Multiply the corresponding numbers and store the value in each register
3. The registers put together gives the result

Code:

16 Bit addition

ADDRESS	MNEMONICS
8500	MOV R0, #40
8502	MOV A, @R0
8503	INC R0
8504	ADD A, @R0
8505	MOV DPTR, #8500
8508	MOV @DPTR, A
8509	INC DPTR
850A	INC R0
850B	MOV A, @R0
850C	INC R0
850D	ADDC A, @R0
850E	MOV X @DPTR, A
850F	LCALL 00BB

Input:

40 → AA

41 → AA

Output:

DPTR → 0154H

16 BIT SUBTRACTION

ADDRESS	MNEMONICS
8700	MOV R1, #50
8702	MOV R2, #30
8704	MOV R3, #50
8706	MOV R4,#20
8708	MOV A, R1
8709	SUBB A, R2
870A	MOV R6, A
870B	MOV A, R3
870C	SUBB A, R4
870D	MOV R5, A
870E	LCALL 00BB

Output:

R6 → 20

R5 → 30

16 BIT MULTIPLICATION

ADDRESS	MNEMONICS
8600	MOV R1, #3F
8604	MOV R2, #23
8606	MOV R3, #11
8608	MOV A, R3
8609	MOV F0, R4
860B	MUL AB
860C	MOV R5, A
860D	MOV R6, F0
860F	MOV A, R4
8610	MOV F0, R1
8612	MUL AB
8613	MOV R7, F0
8615	ADDC A, R6
8616	MOV R6, A
8617	MOV A, R2
8618	MOV F0, R3
861A	MUL AB
861B	ADDC A, R6

861C	MOV R6, A
861D	MOV A, F0
861F	ADDC A, R7
8620	MOV R7, A
8621	MOV A, R1
8622	MOV F0, R2
8624	MUL AB
8625	ADDC A, R7
8626	MOV R7, A
8627	MOV A, 0B
8629	ADDC A, #00
862B	MOV R0, A
862C	LCALL 00BB

Output:

8DAF112

Result:

16-bit arithmetic operations have been performed successfully and the results are verified.

Exp.No: 4 **Date:** 13-04-22

Multibyte Arithmetic Operations

Aim:

To perform Multibyte arithmetic operation using 8051 microcontrollers.

Algorithm:

Addition:

1.START

2.Initialise Ro with 1st bit of 1st operand, R4 with 2nd operand address.

3. Initialize R2 with a.

4.Add 1st abuts and stare it in a location.

5.Increment Ro, RI

6.Perform jump operation count becomes 0.

7.STOP

Subtraction:

1.START

2.Initialise RO with 1st but of 1st operand address, R4 with 1st but of 2nd operand address and R2 as counter.

3.Subtract 1st 2 bits and increment pointer to point to next location.

4.Increment Ro and RI

5.Perform jump operation until counter becomes 0

6.STOP.

Multiplication:

1.START

2.Store the 1st 16-bit operand at address and the multiplier at another location.

3.Initialize R4 to 0.

4.Move the least significant bitt of Multiplier and multiplicand to A, B register and multiply them and store it in another location.

5.Increment R0 and R1 to point to next location.

6.Perform loop until counter becomes 0.

7.STOP

Code:

Multibyte addition:

ADDRESS	MNEMONICS
8500	MOV R0,#40
8502	MOV R1,#50
8504	MOV R3,#04
8506	MOV A,@RO
8507	ADDC A,@RO1
8508	MOV @R1,A
8509	INC RO
850A	INC R1
850B	DJNZ R3,8506
850D	LCALL 00BB

Input:

44 → 06	54 → 06
43 → 05	53 → 05
42 → 04	52 → 04
41 → 03	51 → 03

Output:

44 → 0C

43 → 0A

42 → 08

06 → 06

Multibyte subtraction:

ADDRESS	MNEMONICS
8700	MOV RO,#40
8702	MOV R1,#50
8704	MOV R2,#4
8706	CIR C
8707	MOV A,@RO
8708	SUBB A,@R1
8709	MOV @R1,A
870A	INC RO
870B	INC R1
870C	DJNZ R2,8707
870E	LCALL 00BB

Input:

44 → 0C	54 → 06
43 → 0A	53 → 05
42 → 08	52 → 04
41 → 06	51 → 03

Output:

44 → 06

43 → 05

42 → 04

06 → 03

Multibyte multiplication:

ADDRESS	MNEMONICS
8850	MOV R2,#02
8852	MOV R0,#40
8854	MOV R5,#50
8856	MOV R1,#60
8858	CLRC
8859	MOV R4,#00
885A	MOV A,@R0
885C	MOV F0,R5
885E	MUL AB
885F	ADDC A,R4
8860	MOV @R1,A
8861	MOV R4,F0
8863	INC R0
8864	INC R1
8865	DJNZ R2,885A
8867	MOV A,F0
8869	ADDC A,#00
886B	MOV @R1,A
886C	LCALL 00BB

Input:

40 → 02

41 → 53

42 → A2

60 → 49

Output:

51 → AB

52 → 49

53 → 2E

Result:

The multibyte arithmetic operations has been successfully executed and the results are verified

Exp.No: 5 **Date:** 20-04-22

SORTING

Aim:

To sort the given elements in ascending and descending order.

Algorithm:

1. Array of elements are stored in internal memory
2. Sort the given elements and store it in same memory by swapping adjacent numbers that are unordered for n-1 passes

Code:

Ascending order:

ADDRESS	MNEMONICS
8500	MOV R4,#05
8502	MOV R3,#05
8504	MOV R0,#40
8506	CLR C
8507	MOV A,@R0
8508	MOV R1,A
8509	INC R0
850A	MOV A,@R0
850B	SUBB A,R1
850C	JNC 8516
850E	MOV A,@R0
850F	DEC R0
8510	MOV @R0,A
8512	MOV A,R1
8513	INC R0
8514	MOV @R0,A
8516	DJNZ R3,8507
8518	DJNZ R4, 8502
8520	LCALL 00BB

Input:

40 → 13 41 → 1 42 → 7 43 → 3

Output:

40 → 1 41 → 3 42→7 43 →13

Descending order:

ADDRESS	MNEMONICS
8500	MOV R4,#05
8502	MOV R3,#05
8504	MOV R0,#40
8506	CLR C
8507	MOV A,@R0
8508	MOV R1,A
8509	INC R0
850A	MOV A,@R0
850B	SUBB A,R1
850C	JC 8516
850E	MOV A,@R0
850F	DEC R0
8510	MOV @R0,A
8512	MOV A,R1
8513	INC R0
8514	MOV @R0,A
8516	DJNZ R3,8507
8518	DJNZ R4, 8502

Input:

40 → 1 41 → 3 42 → 8 43 → 13

Output:

40 → 13 41 → 8 42→3 43 →1

Result:

The sorting programs are implemented successfully and the results are verified.

Exp.No: 6 **Date:** 27-04-22

CODE CONVERTORS

Aim:

To perform code conversions in assembly language using 8051 micro controllers.

ALGORITHM:

BCD-HEX:

1. Separate the first and last 4 digit[bit] by applying logical AND with 0F and F0
2. Swap the first and last 4 bits of port AND-ed with F0 to make it single digit
3. Multiply ten's place digit by 10 and add unit's place

HEX-BCD:

1. Divide the numerator by 0AThe quotient gives higher bits and remainder gives lower bits
2. Swap the quotient's bytes and lower bits and add with remainder

HEXADECIMAL-ASCII:

1. Store the data in a register and transfer to accumulator
2. Subtract 0A to see if it is from 0-9.If it is ,add 30 to convert to ASCII.
3. If it is from A-F add 37

BINARY-GRAYCODE:

1. Store the binary value in a register and transfer to accumulator
2. Right Rotate accumulator and XOR it with value in the register

GRAYCODE-BINARY:

1. Store the gray code in a register and move to accumulator
2. Initialize a counter at value 7
3. AND accumulator with 80H to retrieve MSB
4. Right Rotate Accumulator and AND with 7FH to retrieve next 7 bits. XOR the bits with 8 register.

Code:

BCD- HEX:

ADDRESS	MNEMONICS
8800	MOV R0,#99
8802	MOV A,R0
8803	ANL A,#0F
8805	MOV R1,A
8806	MOV A,R0
8807	ANL A,#F0
8809	SWAP A

880A	MOV F0,#0A
880D	MUL AB
880E	ADD A,R1
880F	MOV R1,A
8810	LCALL 00BB

Input:

R0 ← 99H (BCD)

Output:

$99_{(10)}$ ⇔ 63H

A=63H

HEX-BCD:

ADDRESS	MNEMONICS
8800	MOV R0,#62
8802	MOV A,R0
8805	MOV F0,#0A
8806	DIV AB
8807	SWAP A
8809	MOV R1,F0
880A	ADD A,R1
880B	MOV R1,A
880C	LCALL 00BB

Input:

A → 5FH (Hex)

Output:

$95_{(10)}$ ⇔ 1001 01011 (BCD)

40 → 95   41 → 00

HEXADECIMAL-ASCII:

ADDRESS	MNEMONICS
8750	MOV R0,#2A
8752	MOV A,R0
8753	CRL C
8754	SUBB A,#0A
8756	JC NUM
8758	ADD A,#37
875A	SJMP STORE
875C	MOV A,R2
875D	ADD A,#30H

875F	MOV R1,A
8760	LCALL 00BB

Output:

40 → A

50 → 41

BINARY-GRAY CODE:

ADDRESS	MNEMONICS
8550	CLR C
8551	MOV A,#0A
8553	MOV R0,A
8554	RRC A
8555	XRL A,R0
8556	LCALL 00BB

Output:

R7 = AA

GRAY CODE-BINARY:

ADDRESS	MNEMONICS
8650	MOV R0,#0F
8652	MOV A,R0
8653	MOV R3,#07
8655	MOV F0,A
8657	RRC A
8658	XRL A,B
865A	CLR C
865B	DJNZ R3,8657

Output:

R6 = CC

Result:

Code conversions are performed successfully and the results are verified.

Exp.No: 7 **Date:** 11-05-22

ADDITION OF N NUMBERS

Aim:

To perform addition of n numbers in assembly language using 8051 Micro Controller.

Algorithm:

1. START
2. Store the values of numbers in registers
3. Store the value of n in separate address location
4. Initialize A=0
5. Add value contained in indirect addressing mode of R0 to A
6. Increment R0
7. Repeat the steps until counter (R2) becomes zero
8. STOP

CODE:

ADDRESS	MNEMONICS
8300	MOV R0,#40
8302	MOV R1,#65
8304	MOV A,@R1
8305	MOV R2,A
8306	MOV A,#00
8308	ADDC A,@R0
8309	INC R0
830A	DJNZ R2,8308
830C	LCALL 00BB

Output:

65 → 3

40 → 3 41→ 5 42→ 2

A → A

Result:

Addition of n numbers has been executed successfully and the results are verified.

Exp.No: 8 **Date:** 01-06-22

Timer Delay using Embedded C

Aim:

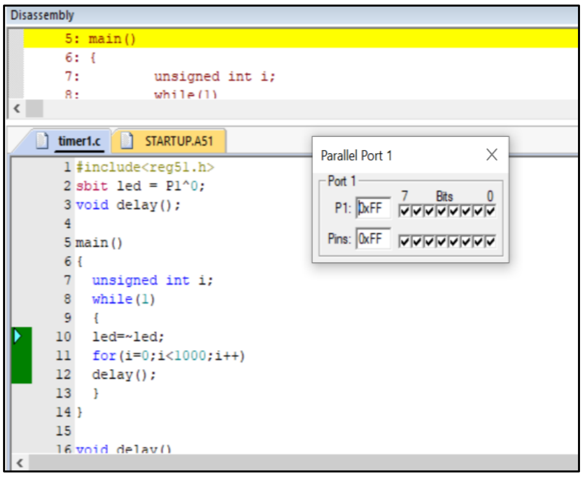
To perform timer delay using embedded C and Keil simulation.

Code:

Timer (Mode-1 with no interrupt)

```
#include<reg51.h>
sbit led = P1^0;
void delay();
main()
{
        unsigned int i;
        while(1)
        {
        led=~led;
        for(i=0;i<1000;i++)
        delay();
        }
}

void delay()
{
        TMOD = 0x01;
        TH0= 0xFC;
        TL0 = 0x66;
        TR0 = 1;
        while(TF0 == 0);
        TR0 = 0;
        TF0 = 0;
}
```

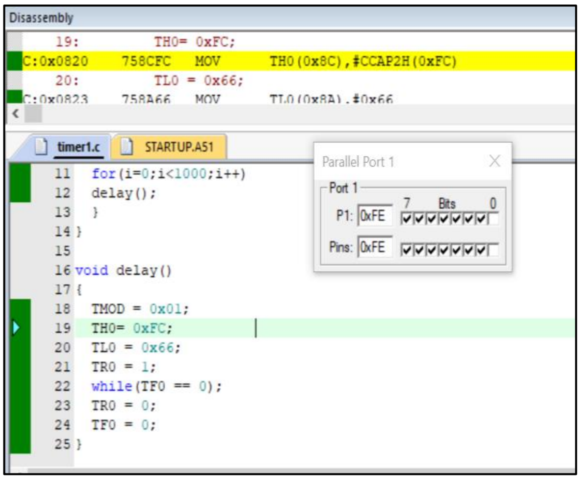
Output:



```
     5: main()
     6: {
     7:          unsigned int i;
     8:          while(1)
```

```
#include<reg51.h>
sbit led = P1^0;
void delay();

main()
{
  unsigned int i;
  while(1)
  {
  led=~led;
  for(i=0;i<1000;i++)
  delay();
  }
}

void delay()
```

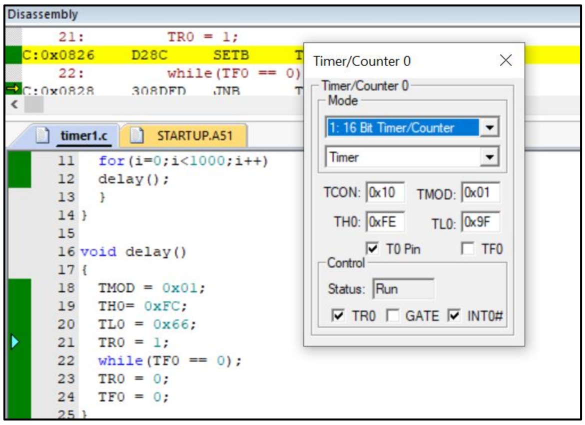
Port 1
P1: 0xFF Bits 7–0: ☑ ☑ ☑ ☑ ☑ ☑ ☑ ☑
Pins: 0xFF ☑ ☑ ☑ ☑ ☑ ☑ ☑ ☑



```
    19:          TH0= 0xFC;
C:0x0820    758CFC   MOV      TH0(0x8C),#CCAP2H(0xFC)
    20:          TL0 = 0x66;
C:0x0823    758A66   MOV      TL0(0x8A),#0x66
```

```
  for(i=0;i<1000;i++)
  delay();
  }
}

void delay()
{
  TMOD = 0x01;
  TH0= 0xFC;
  TL0 = 0x66;
  TR0 = 1;
  while(TF0 == 0);
  TR0 = 0;
  TF0 = 0;
}
```

Port 1
P1: 0xFE Bits 7–0: ☑ ☑ ☑ ☑ ☑ ☑ ☑ ☐
Pins: 0xFE ☑ ☑ ☑ ☑ ☑ ☑ ☑ ☐



```
    21:          TR0 = 1;
C:0x0826    D28C     SETB     T
    22:          while(TF0 == 0)
C:0x0828    308DFD   JNB      T
```

```
  for(i=0;i<1000;i++)
  delay();
  }
}

void delay()
{
  TMOD = 0x01;
  TH0= 0xFC;
  TL0 = 0x66;
  TR0 = 1;
  while(TF0 == 0);
  TR0 = 0;
  TF0 = 0;
}
```

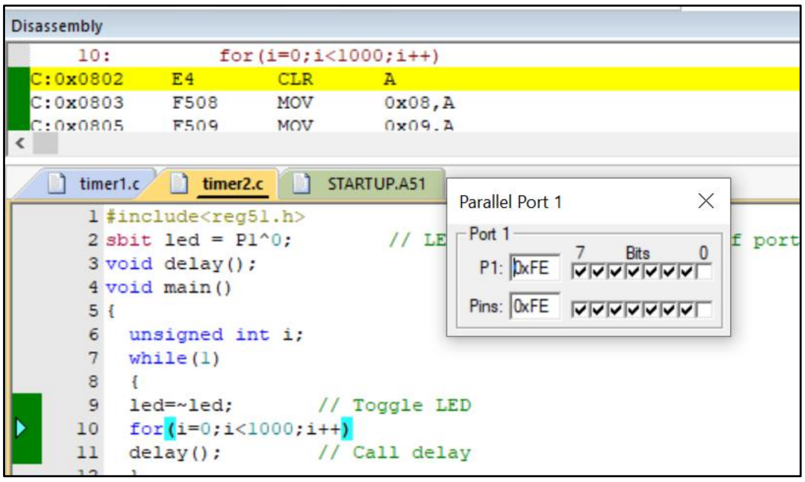
Timer/Counter 0
Mode: 1: 16 Bit Timer/Counter
Timer
TCON: 0x10 TMOD: 0x01
TH0: 0xFE TL0: 0x9F
☑ T0 Pin ☐ TF0
Control
Status: Run
☑ TR0 ☐ GATE ☑ INT0#

<u>Timer (Mode 2):</u>

```
#include<reg51.h>
sbit led = P1^0;                    // LED connected to 1st pin of port P1
void delay();
void main()
{
        unsigned int i;
        while(1)
        {
        led=~led;                   // Toggle LED
        for(i=0;i<1000;i++)
        delay();                    // Call delay
        }
}
```

```
void delay()
{
        TMOD = 0x02;                    // Mode2 of Timer0
        TH0= 0xA2;                      // Initial value loaded to Timer
        TR0 = 1;                        // Start Timer
        while(TF0 == 0);                // Polling for flag bit
        TR0 = 0;                        // Stop Timer
        TF0 = 0;                        // Clear flag
}
```

Output:



Disassembly

```
    10:           for(i=0;i<1000;i++)
C:0x0802    E4      CLR      A
C:0x0803    F508    MOV      0x08,A
C:0x0805    F509    MOV      0x09,A
```

timer1.c | timer2.c | STARTUP.A51

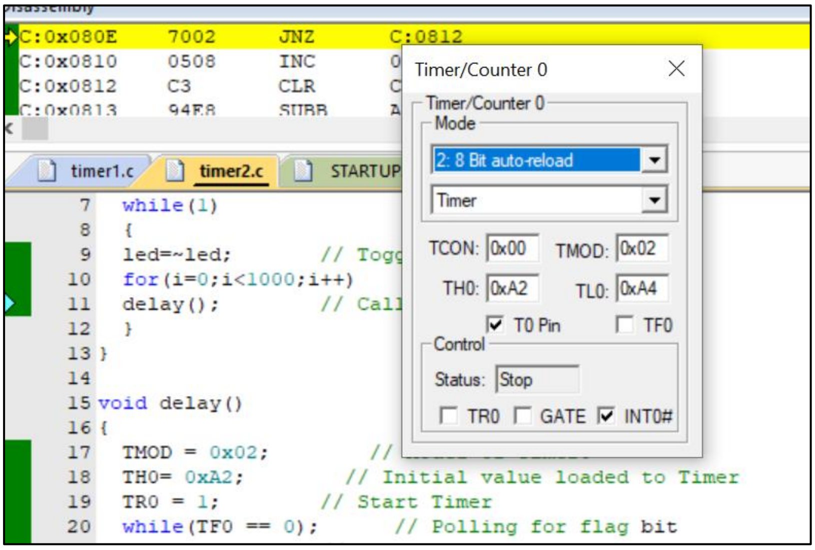
```
#include<reg51.h>
sbit led = P1^0;          // LE
void delay();
void main()
{
  unsigned int i;
  while(1)
  {
  led=~led;          // Toggle LED
  for(i=0;i<1000;i++)
  delay();           // Call delay
```

Parallel Port 1

Port 1

P1: 0xFE Bits 7 ☑☑☑☑☑☑☑□ 0

Pins: 0xFE ☑☑☑☑☑☑☑□



```
C:0x080E    7002    JNZ      C:0812
C:0x0810    0508    INC      0
C:0x0812    C3      CLR      C
C:0x0813    94E8    SUBB     A
```

timer1.c | timer2.c | STARTUP

```
  while(1)
  {
  led=~led;          // Togg
  for(i=0;i<1000;i++)
  delay();           // Call
  }
}

void delay()
{
  TMOD = 0x02;          //
  THO= 0xA2;          // Initial value loaded to Timer
  TR0 = 1;          // Start Timer
  while(TF0 == 0);          // Polling for flag bit
```

Timer/Counter 0

Timer/Counter 0

Mode: 2: 8 Bit auto-reload

Timer

TCON: 0x00 TMOD: 0x02

TH0: 0xA2 TL0: 0xA4

☑ T0 Pin □ TF0

Control

Status: Stop

□ TR0 □ GATE ☑ INT0#

Result:

Timer delay has been successfully done using embedded C and Keil simulation.

Exp.No: 9 **Date:** 01-06-22

Serial Communication using Embedded C

Aim:

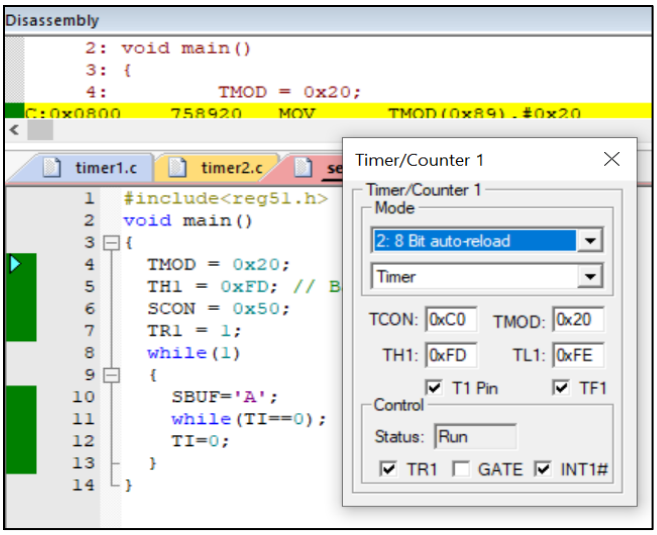
To perform serial communication using embedded C and Keil simulation.

Code:

<u>Serial (send):</u>

```
#include<reg51.h>
void main()
{
        TMOD = 0x20;
        TH1 = 0xFD; // Baud rate = 9600
        SCON = 0x50;
        TR1 = 1;
        while(1)
        {
                SBUF='A';
                while(TI==0);
                TI=0;
        }
}
```

Output:



Disassembly

```
     2: void main()
     3: {
     4:         TMOD = 0x20;
C:0x0800    758920    MOV        TMOD(0x89),#0x20
```

```
#include<reg51.h>
void main()
{
   TMOD = 0x20;
   TH1 = 0xFD; // B
   SCON = 0x50;
   TR1 = 1;
   while(1)
   {
     SBUF='A';
     while(TI==0);
     TI=0;
   }
}
```

Timer/Counter 1

Timer/Counter 1

Mode: 2: 8 Bit auto-reload

Timer

TCON: 0xC0 TMOD: 0x20

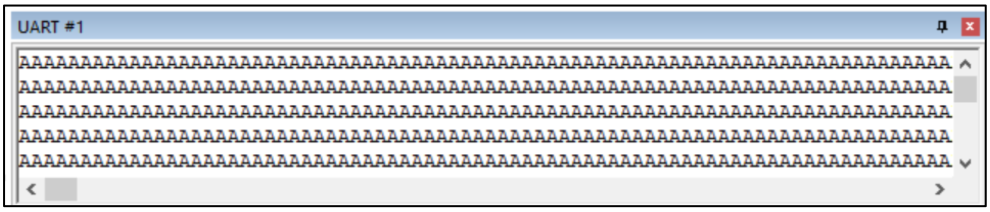
TH1: 0xFD TL1: 0xFE

☑ T1 Pin ☑ TF1

Control

Status: Run

☑ TR1 □ GATE ☑ INT1#

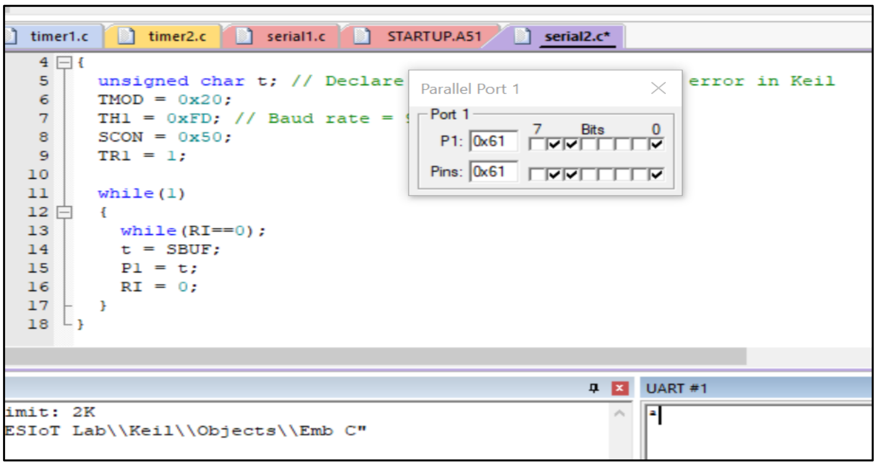


```
UART #1
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Serial (Receive):

```
#include<reg51.h>
void main()
{
        unsigned char t; // Declare here first else results error in Keil
        TMOD = 0x20;
        TH1 = 0xFD; // Baud rate = 9600
        SCON = 0x50;
        TR1 = 1;
        while(1)
        {
                while(RI==0);
                t = SBUF;
                P1 = t;
                RI = 0;
        }
}
```

Output:



```
{
  unsigned char t; // Declare                error in Keil
  TMOD = 0x20;
  TH1 = 0xFD; // Baud rate = 
  SCON = 0x50;
  TR1 = 1;

  while(1)
  {
    while(RI==0);
    t = SBUF;
    P1 = t;
    RI = 0;
  }
}
```

Parallel Port 1

Port 1

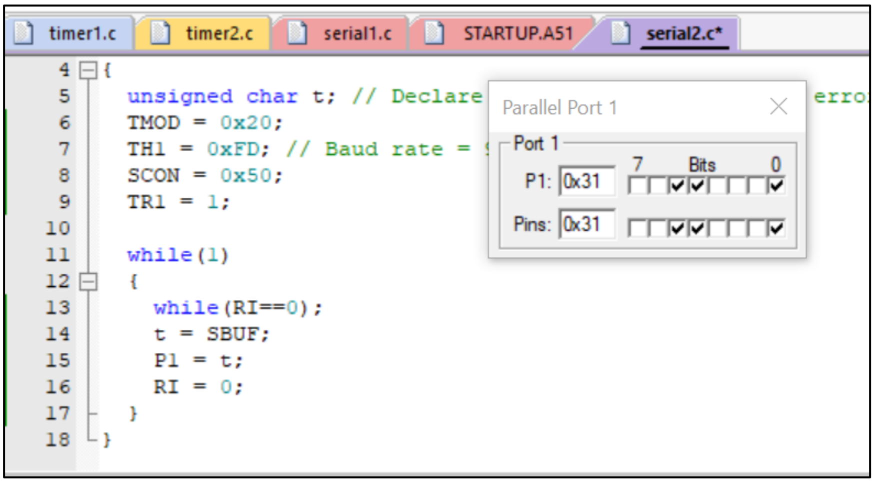
P1: 0x61 □☑☑□□□□☑

Pins: 0x61 □☑☑□□□□☑

```
imit: 2K
ESIoT Lab\\Keil\\Objects\\Emb C"
```

UART #1

```
a
```



```
{
  unsigned char t; // Declare                erro
  TMOD = 0x20;
  TH1 = 0xFD; // Baud rate =
  SCON = 0x50;
  TR1 = 1;

  while(1)
  {
    while(RI==0);
    t = SBUF;
    P1 = t;
    RI = 0;
  }
}
```

Parallel Port 1

Port 1

		7 Bits 0
P1:	0x31	□□☑☑□□□☑
Pins:	0x31	□□☑☑□□□☑

Result:

Serial Communication has been successfully done using embedded C and Keil simulation.

Exp.No: 10 **Date:** 08-06-22

Timer Delay using ALP

Aim:

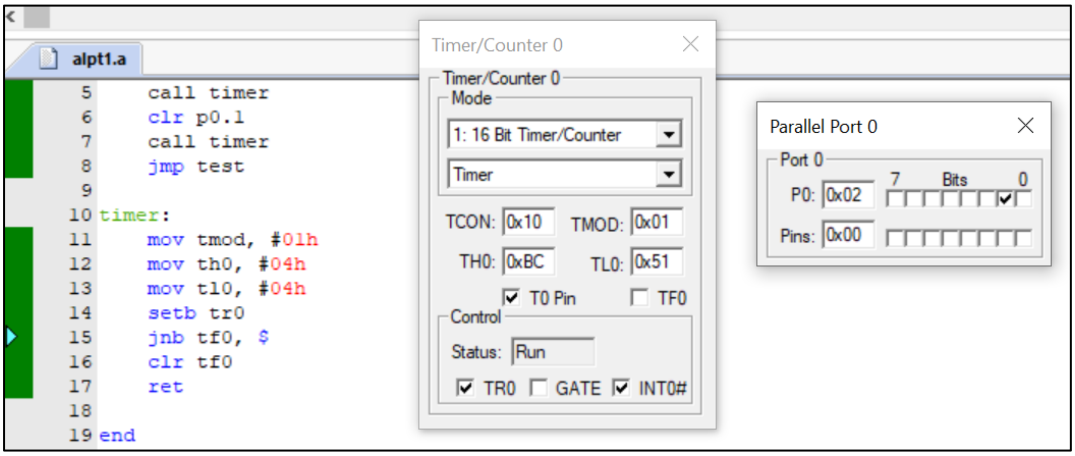
To perform timer delay using ALP and Keil simulation.

Code:

Timer (no interrupt):

```
org 0h
mov p0, #00h
test:
        setb p0.1
        call timer
        clr p0.1
        call timer
        jmp test

timer:
        mov tmod, #01h
        mov th0, #04h
        mov tl0, #04h
        setb tr0
        jnb tr0
        jnb tf0, $
        clr tf0
        ret
end
```

Output:



alpt1.a

```
    call timer
    clr p0.1
    call timer
    jmp test

timer:
    mov tmod, #01h
    mov th0, #04h
    mov tl0, #04h
    setb tr0
    jnb tf0, $
    clr tf0
    ret

end
```

Timer/Counter 0

Timer/Counter 0 Mode: 1: 16 Bit Timer/Counter; Timer

TCON: 0x10 TMOD: 0x01

TH0: 0xBC TL0: 0x51

☑ T0 Pin □ TF0

Control

Status: Run

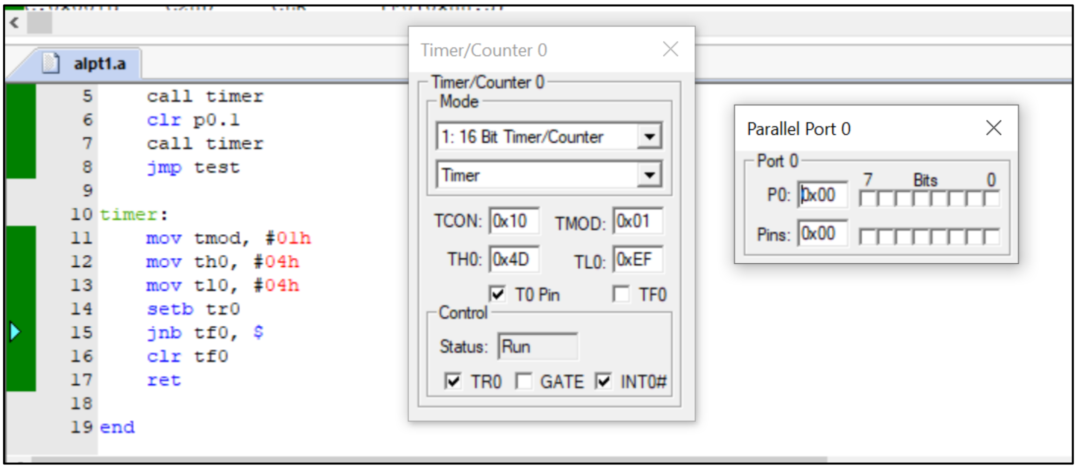
☑ TR0 □ GATE ☑ INT0#

Parallel Port 0

Port 0 — 7 Bits 0

P0: 0x02 □□□□□□☑□

Pins: 0x00 □□□□□□□□



alpt1.a

```
    call timer
    clr p0.1
    call timer
    jmp test

timer:
    mov tmod, #01h
    mov th0, #04h
    mov tl0, #04h
    setb tr0
    jnb tf0, $
    clr tf0
    ret

end
```

Timer/Counter 0

Timer/Counter 0 Mode: 1: 16 Bit Timer/Counter; Timer

TCON: 0x10 TMOD: 0x01

TH0: 0x4D TL0: 0xEF

☑ T0 Pin □ TF0

Control

Status: Run

☑ TR0 □ GATE ☑ INT0#

Parallel Port 0

Port 0 — 7 Bits 0

P0: 0x00 □□□□□□□□

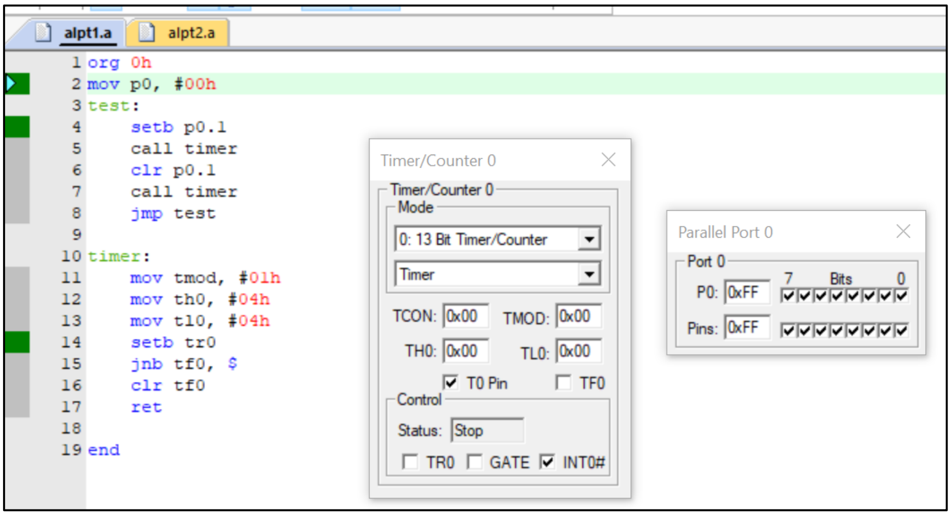
Pins: 0x00 □□□□□□□□

Timer (with interrupt):

```
org 0h
test:
        mov p1, #00h
        call inter


timer:
        mov tmod, #01h
        mov th0, #04h
        mov tl0, #04h
        setb tr0
        jnb tf0, $
        clr tf0
        ret
```

```
inter:
        jnb ie.0, $
        setb p1.0
        call timer
        clr p1.0
        jmp inter
end
```

Output:



```
org 0h
mov p0, #00h
test:
    setb p0.1
    call timer
    clr p0.1
    call timer
    jmp test

timer:
    mov tmod, #01h
    mov th0, #04h
    mov tl0, #04h
    setb tr0
    jnb tf0, $
    clr tf0
    ret

end
```

Timer/Counter 0

Timer/Counter 0

Mode: 0: 13 Bit Timer/Counter

Timer

TCON: 0x00 TMOD: 0x00

TH0: 0x00 TL0: 0x00

☑ T0 Pin ☐ TF0

Control

Status: Stop

☐ TR0 ☐ GATE ☑ INT0#

Parallel Port 0

Port 0

P0: 0xFF 7 Bits 0 ☑☑☑☑☑☑☑☑

Pins: 0xFF ☑☑☑☑☑☑☑☑

Result:

Timer Delay has been successfully done using ALP and Keil simulation.

Exp.No: 11 **Date:** 08-06-22

Serial Communication using ALP

Aim:

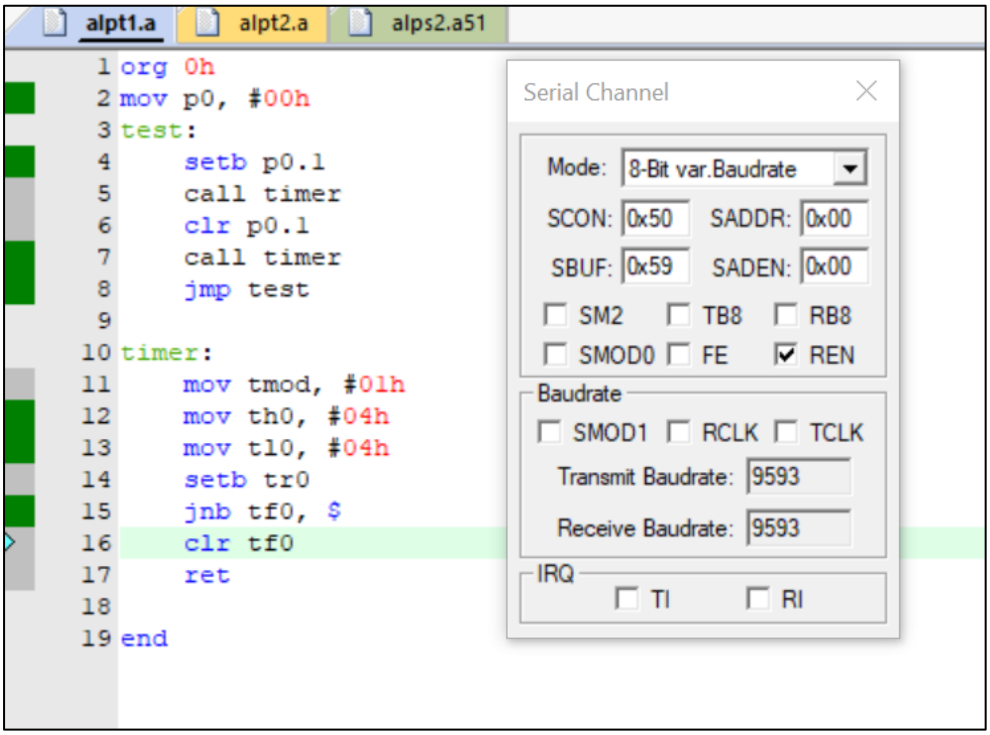
To perform serial communication using ALP and Keil simulation.

Code:

Serial (Send):

```
org 0h
mov scon, #50h
mov tmod, #20h
mov th1, #-3
setb tr1

repeat:
        mov sbuf, #"Y"
        acall tran
        mov sbuf, #"E"
        acall tran
        mov sbuf, #"S"
        acall tran
        sjmp repeat

tran:
        jnb ti, $
        clr ti
        ret

end
```

Output:

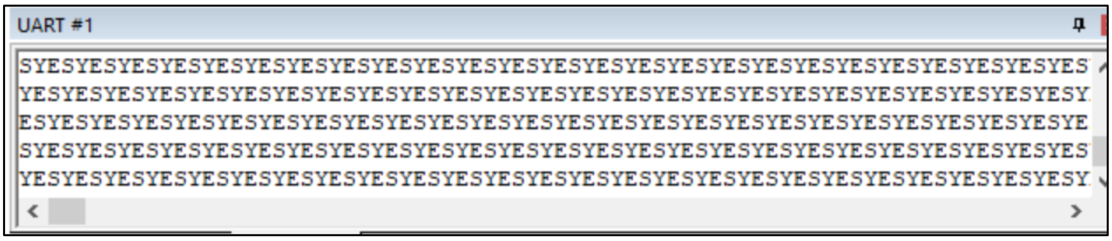


```
org 0h
mov p0, #00h
test:
     setb p0.1
     call timer
     clr p0.1
     call timer
     jmp test

timer:
     mov tmod, #01h
     mov th0, #04h
     mov tl0, #04h
     setb tr0
     jnb tf0, $
     clr tf0
     ret

end
```

Mode: 8-Bit var.Baudrate
SCON: 0x50 SADDR: 0x00
SBUF: 0x59 SADEN: 0x00
☐ SM2 ☐ TB8 ☐ RB8
☐ SMOD0 ☐ FE ☑ REN

Baudrate
☐ SMOD1 ☐ RCLK ☐ TCLK
Transmit Baudrate: 9593
Receive Baudrate: 9593

IRQ
☐ TI ☐ RI

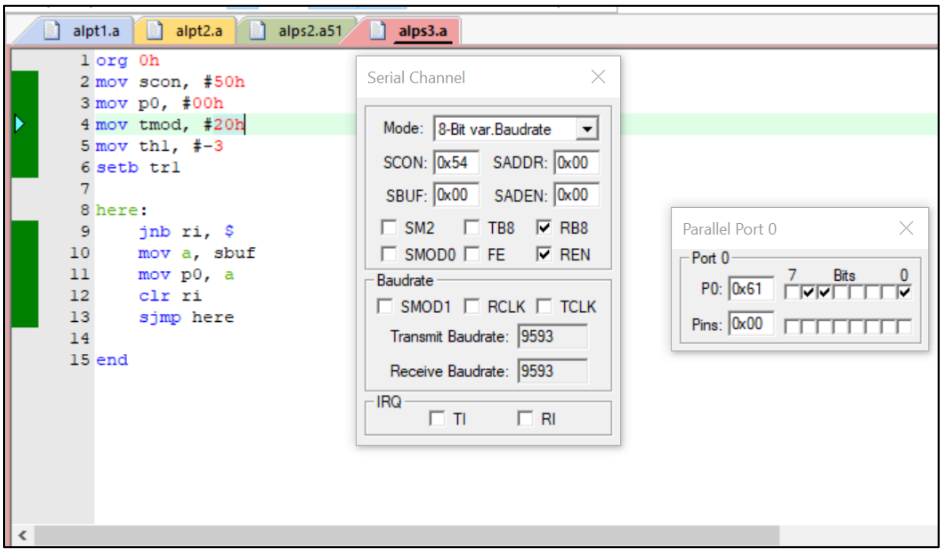


```
SYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYES
YESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESY
ESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYE
SYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYES
YESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESYESY
```

<u>Serial (receive):</u>

```
org 0h
mov scon, #50h
mov p0, #00h
mov tmod, #20h
mov th1, #-3
setb tr1


here:
        jnb ri, $
        mov a, sbuf
        mov p0, a
        clr ri
        sjmp here


end
```

Output:



```
org 0h
mov scon, #50h
mov p0, #00h
mov tmod, #20h
mov th1, #-3
setb tr1

here:
    jnb ri, $
    mov a, sbuf
    mov p0, a
    clr ri
    sjmp here

end
```

Serial Channel

Mode: 8-Bit var.Baudrate

SCON: 0x54 SADDR: 0x00

SBUF: 0x00 SADEN: 0x00

□ SM2 □ TB8 ☑ RB8

□ SMOD0 □ FE ☑ REN

Baudrate

□ SMOD1 □ RCLK □ TCLK

Transmit Baudrate: 9593

Receive Baudrate: 9593

IRQ

□ TI □ RI

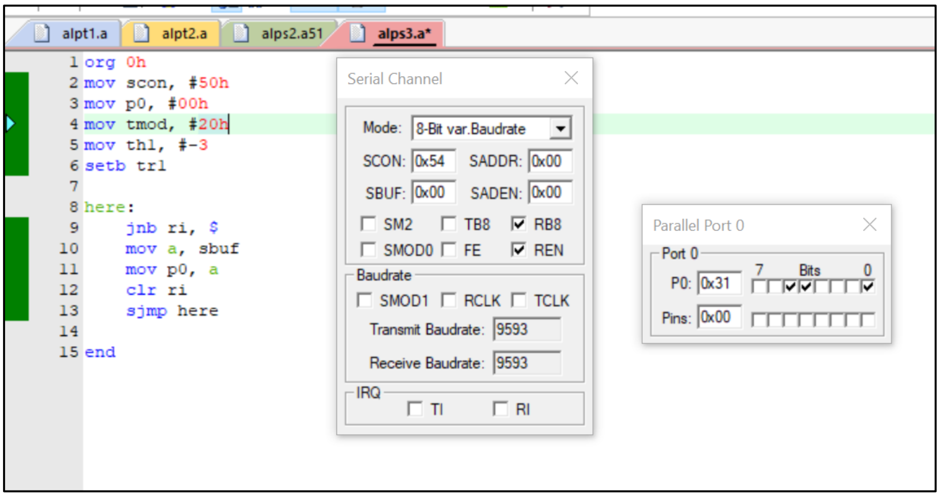
Parallel Port 0

Port 0

Bits 7 … 0

P0: 0x61 □☑☑□□□□☑

Pins: 0x00 □□□□□□□□



```
org 0h
mov scon, #50h
mov p0, #00h
mov tmod, #20h
mov th1, #-3
setb tr1

here:
    jnb ri, $
    mov a, sbuf
    mov p0, a
    clr ri
    sjmp here

end
```

Serial Channel

Mode: 8-Bit var.Baudrate

SCON: 0x54 SADDR: 0x00

SBUF: 0x00 SADEN: 0x00

□ SM2 □ TB8 ☑ RB8

□ SMOD0 □ FE ☑ REN

Baudrate

□ SMOD1 □ RCLK □ TCLK

Transmit Baudrate: 9593

Receive Baudrate: 9593

IRQ

□ TI □ RI

Parallel Port 0

Port 0

Bits 7 … 0

P0: 0x31 □□☑☑□□□☑

Pins: 0x00 □□□□□□□□

Result:

Serial Communication has been successfully done using ALP and Keil simulation.

Exp.No: 12 **Date:** 07-05-22

Blinking a LED

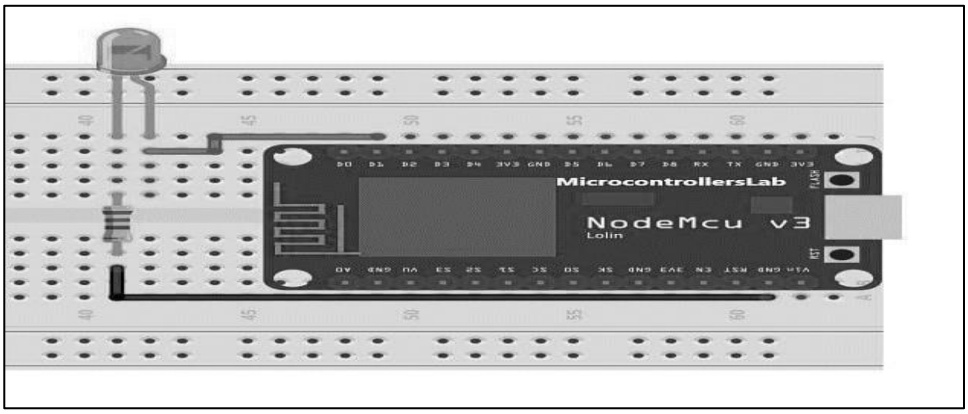
Aim:

To blink a LED using Node MCU

Hardware Requirements:

- NodeMCU x 1
- LED x 1
- BreadBoard
- 200 ohm – 1K ohm resistor x 1
- Micro USB cable x 1
- PC x 1
- Software Arduino IDE(version 1.6.4+)
- Jumper Wires (Male – Female & Male – Male)

Circuit:


MicrocontrollersLab
NodeMcu v3
Lolin
FLASH
RST
D0 D1 D2 D3 D4 3V3 GND D5 D6 D7 D8 RX TX GND 3V3
A0 GND VU S3 S2 S1 SC S0 SK GND 3V3 EN RST GND VIN
40
45
50
55
60

Code:

```
int LED = 5; // Assign LED pin i.e: D1 on NodeMCU
void setup() {
// initialize GPIO 5 as an output
pinMode(LED, OUTPUT);
}
// the loop function runs over and over again forever
void loop() {
digitalWrite(LED, HIGH); // turn the LED on
delay(1000); // wait for a second
digitalWrite(LED, LOW); // turn the LED off
delay(1000); // wait for a second
}
```

Output:

LED is ON when input is HIGH

LED is OFF when input is LOW

Result:

Blinking of LED has been done successfully using Node MCU and the results are verified.

Exp.No: 13 **Date:** 07-05-22

Measuring Temperature using temperature sensor

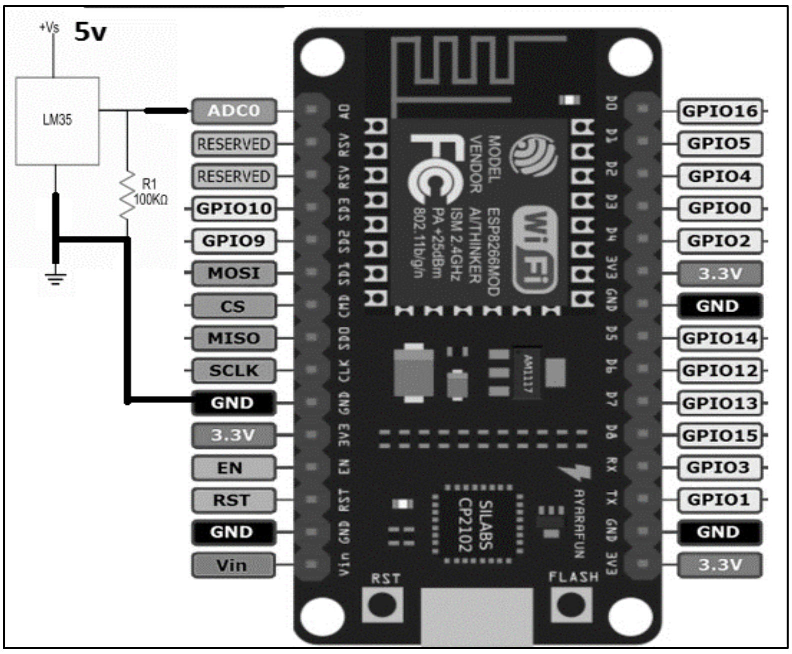
Aim:

To measure the temperature of the room / surrounding using temperature sensor & Node MCU

Hardware Requirements:

- NodeMCU
- LM35 Temperature Sensor
- Bread Board
- Jumper Wires
- Micro USB Cable
- Arduino IDE
- 200 ohm – 1K ohm resistor x 1
- PC

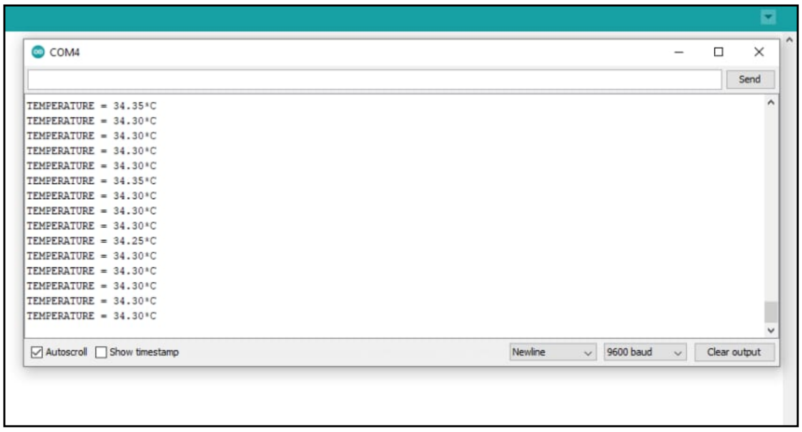
Circuit Diagram:



Code:

```
// initializes or defines the output pin of the LM35 temperature sensor
int outputpin= A0;
//this sets the ground pin to LOW and the input voltage pin to high
void setup() {
Serial.begin(9600);
}
void loop()       //main loop
{
int analogValue = analogRead(outputpin);
float millivolts = (analogValue/1024.0) * 3300; //3300 is the voltage provided by NodeMCU
float celsius = millivolts/10;
Serial.print("in DegreeC=   ");
Serial.println(celsius);
//----------Calculation for Fahrenheit ----------//
float fahrenheit = ((celsius * 9)/5 + 32);
Serial.print(" in Farenheit=   ");
Serial.println(fahrenheit);
delay(1000);
}
```

Output:



```
TEMPERATURE = 34.35°C
TEMPERATURE = 34.30°C
TEMPERATURE = 34.30°C
TEMPERATURE = 34.30°C
TEMPERATURE = 34.30°C
TEMPERATURE = 34.35°C
TEMPERATURE = 34.30°C
TEMPERATURE = 34.30°C
TEMPERATURE = 34.30°C
TEMPERATURE = 34.25°C
TEMPERATURE = 34.30°C
TEMPERATURE = 34.30°C
TEMPERATURE = 34.30°C
TEMPERATURE = 34.30°C
TEMPERATURE = 34.30°C
```

☑ Autoscroll □ Show timestamp Newline 9600 baud

Result:

Temperature has been measured successfully using Temperature sensor and the results are verified.

Exp.No: 14 **Date:** 07-05-22

InfraRed Sensor using Node MCU

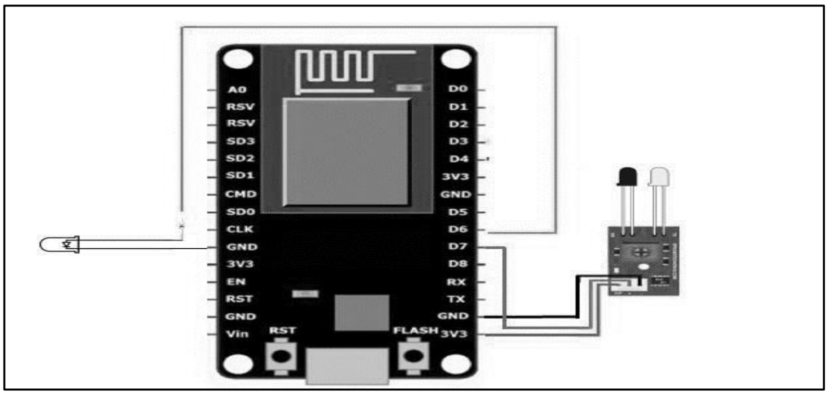
Aim:

To detect any motion in the room / surrounding using temperature sensor & Node MCU

Hardware Requirements:

- NodeMCU
- IR Sensor
- Bread Board
- Jumper Wires
- Micro USB Cable
- Arduino IDE
- 200 ohm – 1K ohm resistor x 1
- PC
- LED

Circuit Diagram:



Code:

```
int ledPin = 12; // choose pin for the LED

int inputPin = 13; // choose input pin (for Infrared sensor)

int val = 0; // variable for reading the pin status


void setup()

{

  pinMode(ledPin, OUTPUT); // declare LED as output

  pinMode(inputPin, INPUT); // declare Infrared sensor as input
}


void loop()
{
  val = digitalRead(inputPin); // read input value
  if (val == HIGH)
  { // check if the input is HIGH
    digitalWrite(ledPin, LOW); // turn LED OFF
  }
  else
  {
    digitalWrite(ledPin, HIGH); // turn LED ON
  }
}
```

Output:

IR sensor identifies object nearby and turns LED ON

If no object is present in front then LED is OFF.

Result:

IR sensor has been used successfully using Node MCU and the results are verified.

Exp.No: 15 **Date:** 14-05-22

Light Sensor

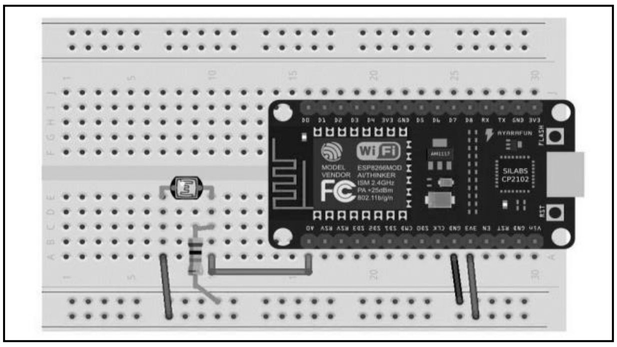
Aim:

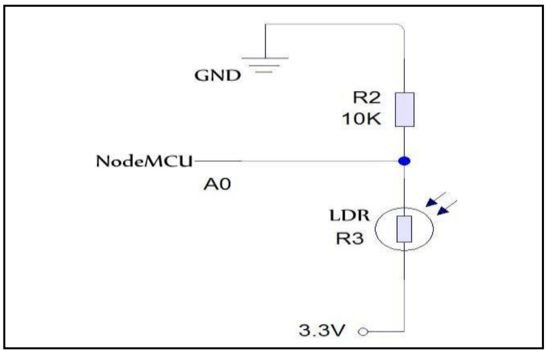
To use Light Sensor using Node MCU

Hardware Requirements:

- NodeMCU
- LDR/PhotoResistor
- Bread Board
- Jumper Wires
- Micro USB Cable
- Arduino IDE (with ESP8266 Library installed)
- 200 ohm – 1K ohm resistor x 1
- PC

Circuit Diagram:



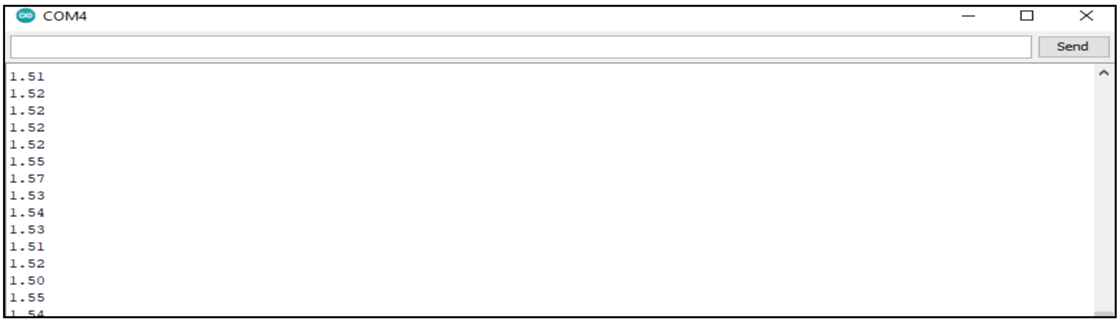


Code:

```
void setup() {
        Serial.begin(9600);   // initialize serial communication at 9600 BPS
}
void loop() {
        int sensorValue = analogRead(A0);   // read the input on analog pin 0
        float voltage = sensorValue * (5.0 / 1023.0);   // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V)
        Serial.println(voltage);   // print out the value you read
}
```

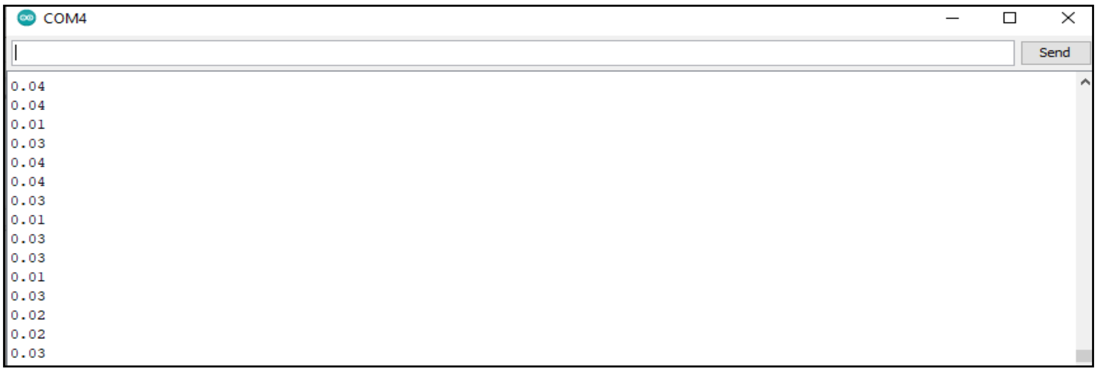
Output:

with light



```
1.51
1.52
1.52
1.52
1.52
1.55
1.57
1.53
1.54
1.53
1.51
1.52
1.50
1.55
1.54
```

without light



```
0.04
0.04
0.01
0.03
0.04
0.04
0.03
0.01
0.03
0.03
0.01
0.03
0.02
0.02
0.03
```

Result:

Light Sensor has been used successfully using Node MCU and the results are verified.

Exp.No: 16 **Date:** 14-05-22

Proximity Detection

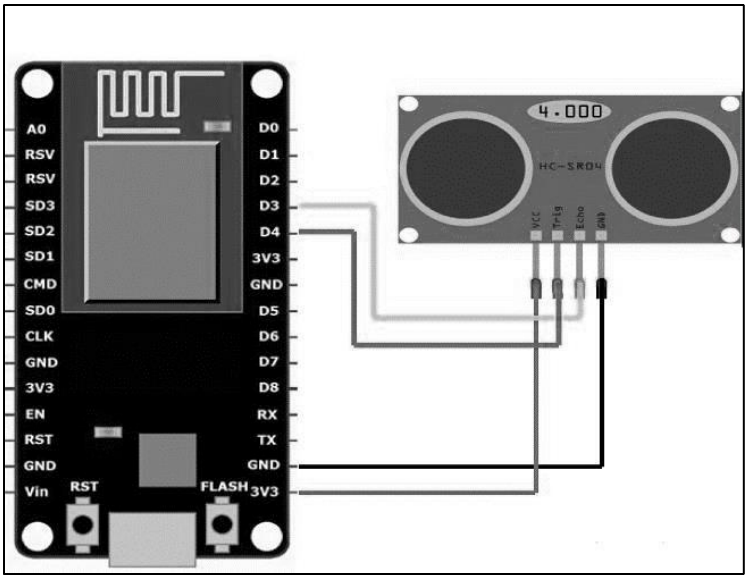
Aim:

To detect the location using ultrasonic sensor

Hardware Required:

- NodeMCU
- HC-SR04 (Ultra-sonic Sensor)
- Bread Board
- Jumper Wires
- Micro USB Cable

Circuit Diagram:



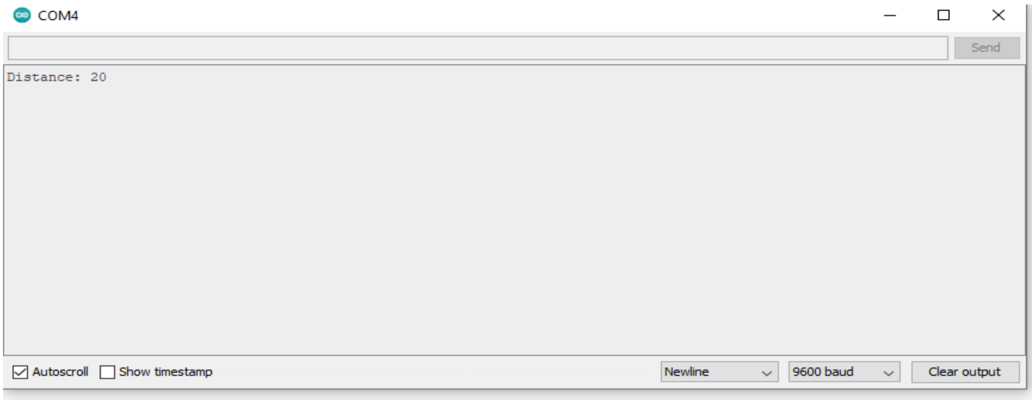
Code: (1)

```
void setup() {

  // ultrosonic  HCSR -04 - proximity detection

  pinMode(D6,OUTPUT); //trigger pin

  pinMode(D7,INPUT);//Echopin //rcr

}


void loop() {

  // put your main code here, to run repeatedly:

//trigger sonic waves
digitalWrite(D6,LOW);
delayMicroseconds(2);
digitalWrite(D6,HIGH);
delayMicroseconds(10);
// receive ECHO and find DISTANCE
long duration= pulseIn(D7,HIGH);
float dist= duration * 0.034/2;
Serial.println("Distance is...");
Serial.println(dist);
delay(2000);
}
```

Code:(2)

```
// defines pins numbers
const int trigPin = 2;  //D4
const int echoPin = 0;  //D3

// 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: ");
Serial.println(distance);
delay(2000);
}
```

Output:



```
Distance: 20
```

Result:

Proximity Detection using Ultra Sonic Sensor has been used successfully using Node MCU and the results are verified.

Exp.No: 17 **Date:** 14-05-22

Servo Motor

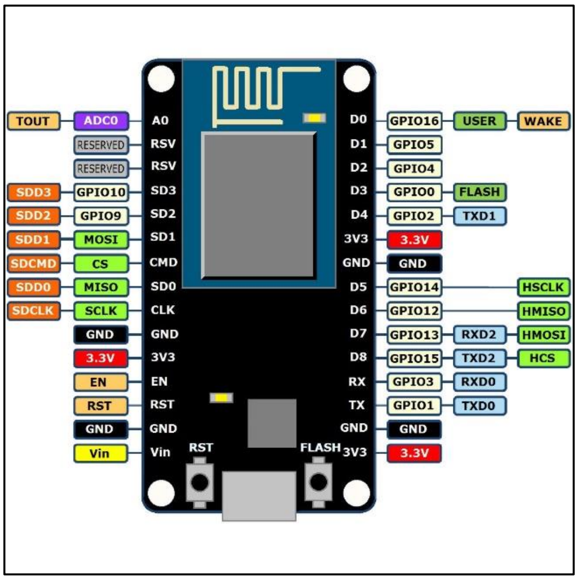
Aim:

To push or rotate an object with great precision at specific angles or distance using servo motor

Hardware Required:

- NodeMCU
- Servo Motor
- Bread Board
- Jumper Wires
- Micro USB Cable

Circuit Diagram:



If your servo has Orange - Red - Brown wires, then connect it as follows

- Orange wire connects to Digital pin D4.
- Brown wire connects to GND pin
- Red wire connects to 3V3 pin

Code:

```
#include <Servo.h>

Servo newservo1;//define a name for servo

void setup() {

  // put your setup code here, to run once:
```

```
  newservo1.attach(D6);
 }
 void loop() {
  // put your main code here, to run repeatedly:
  newservo1.write(180);
  delay(1000);
  newservo1.write(0);
  delay(1000);
}
```

Output:

Servo Motor rotates 180 degree and comes back with 1 second delay

Result:

Servo motor has been used successfully rotated using Node MCU and the results are verified

Exp.No: 18 **Date:** 14-05-22

Potentiometer with servo motor

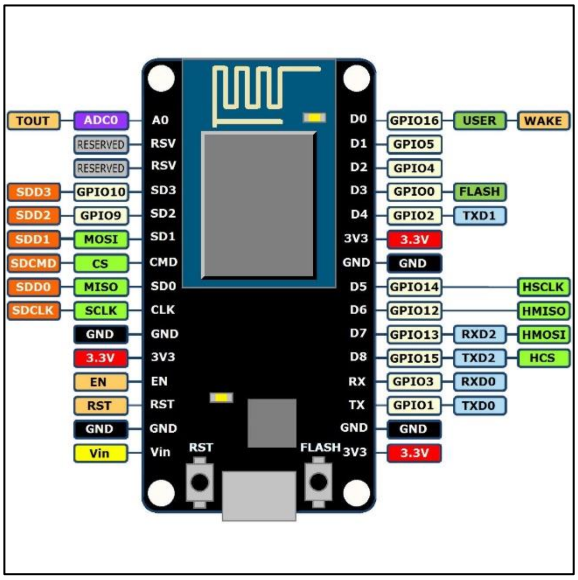
Aim:

To use a potentiometer with servo motor

Hardware Required:

- NodeMCU
- Servo Motor
- Potentiometer
- Bread Board
- Jumper Wires
- Micro USB Cable

Circuit Diagram:



If your servo has Orange - Red - Brown wires, then connect it as follows

- Orange wire connects to Digital pin D4.
- Brown wire connects to GND pin
- Red wire connects to 3V3 pin

Code:

```
# include<Servo.h>

Servo mew 1;

Setup ()

 New1.attach(D6);
```

Serial.begin(9600);

pinMode(A0, INPUT);

loop()

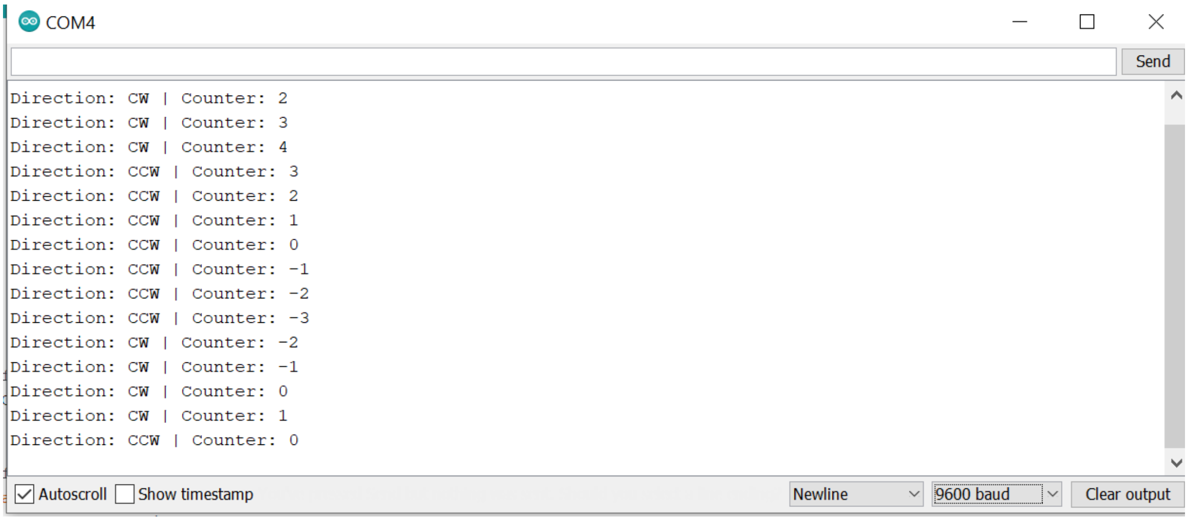
float x=analogRead(A0);

x=map(x,0,1023,0,180);

new1.write (x);

delay(15);}

Output:



```
Direction: CW | Counter: 2
Direction: CW | Counter: 3
Direction: CW | Counter: 4
Direction: CCW | Counter: 3
Direction: CCW | Counter: 2
Direction: CCW | Counter: 1
Direction: CCW | Counter: 0
Direction: CCW | Counter: -1
Direction: CCW | Counter: -2
Direction: CCW | Counter: -3
Direction: CW | Counter: -2
Direction: CW | Counter: -1
Direction: CW | Counter: 0
Direction: CW | Counter: 1
Direction: CCW | Counter: 0
```

Servo Motor rotates 180 degree once counter is above 4 and comes back to same position when counter is less than 4

Result:

Servo motor with potentiometer has been used successfully rotated using Node MCU and the results are verified

Exp.No: 19 **Date:** 14-05-22

IoT with cloud

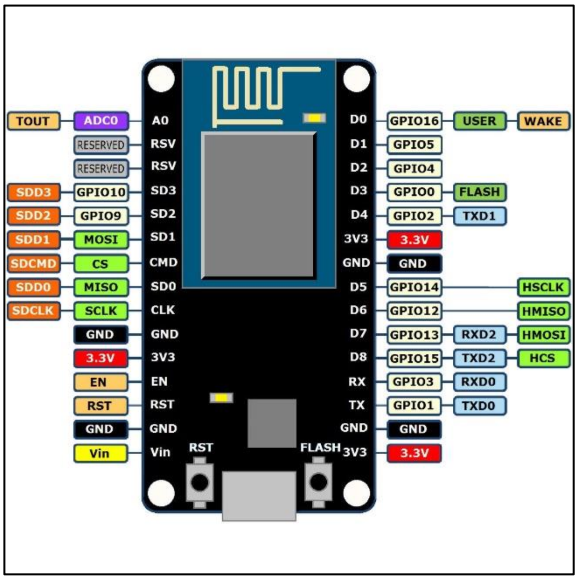
Aim:

To connect with Internet through WIfi Acces point using Node MCU

Hardware Required:

- NodeMCU
- Micro USB Cable

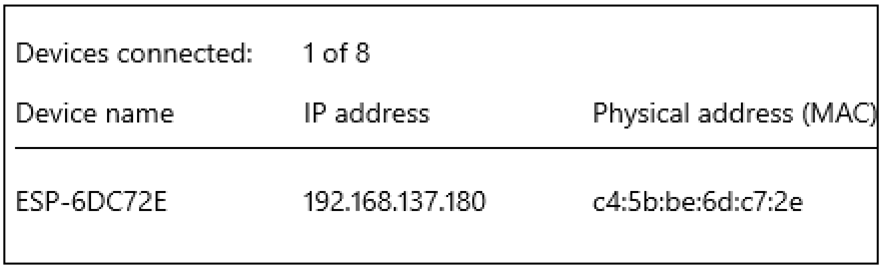
Circuit Diagram:



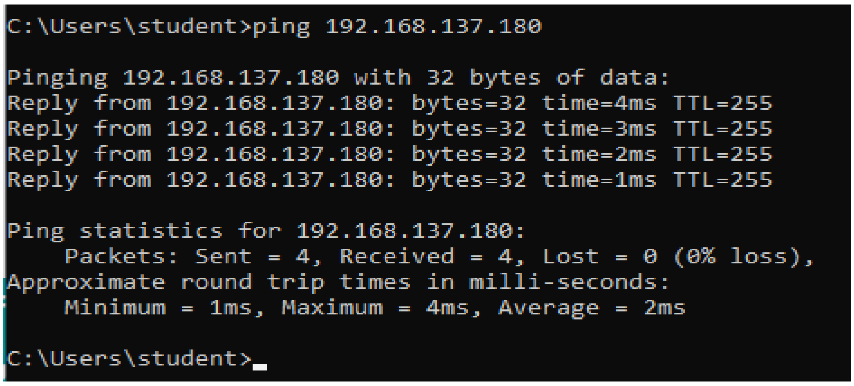
Code:

```
#include<ESP8266WIFI>

Char* SSid =”rajesh”;

Char*  Password =”Koushan”

Void setup()

WiFi.begin(SSid, Password);

Serial.begin(115200);

Serial.print(“Connecting: ”);

While (WiFi.status()!= WL_CONNECTED)

{
 Serial.print(“Waiting to connect”)

While(WiFi.Status()!=WL_CONNECTED)

{

Serial. Print(“Waiting to connect”);

delay(1000);

}

Serial.println(‘\n’);

Serial.println (“Connection established”);

Serial.println (“IP Address\t”);

Serial.println(WiFi.LocalIP());

}
```

Output:



Devices connected:	1 of 8	
Device name	IP address	Physical address (MAC)
ESP-6DC72E	192.168.137.180	c4:5b:be:6d:c7:2e



```
C:\Users\student>ping 192.168.137.180

Pinging 192.168.137.180 with 32 bytes of data:
Reply from 192.168.137.180: bytes=32 time=4ms TTL=255
Reply from 192.168.137.180: bytes=32 time=3ms TTL=255
Reply from 192.168.137.180: bytes=32 time=2ms TTL=255
Reply from 192.168.137.180: bytes=32 time=1ms TTL=255

Ping statistics for 192.168.137.180:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 4ms, Average = 2ms

C:\Users\student>_
```

Result:

Connecting with internet has been done successfully using Node MCU and the results are verified

Exp.No: 20 **Date:** 14-05-22

Making a node as Server

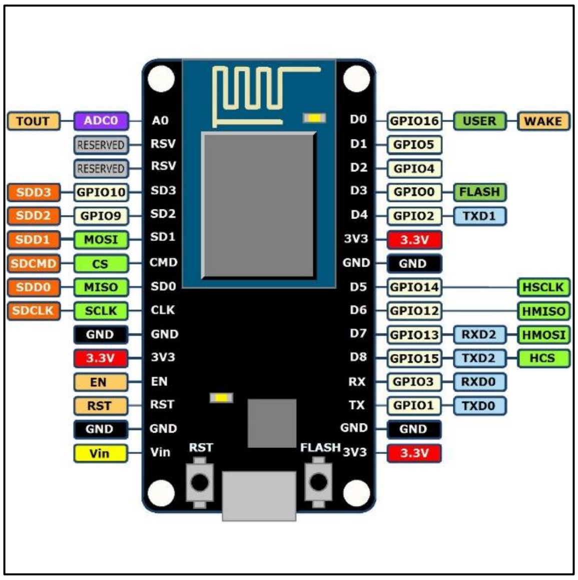
Aim:

To make a node as a server using Node MCU

Hardware Required:

- NodeMCU
- Micro USB Cable

Circuit Diagram:

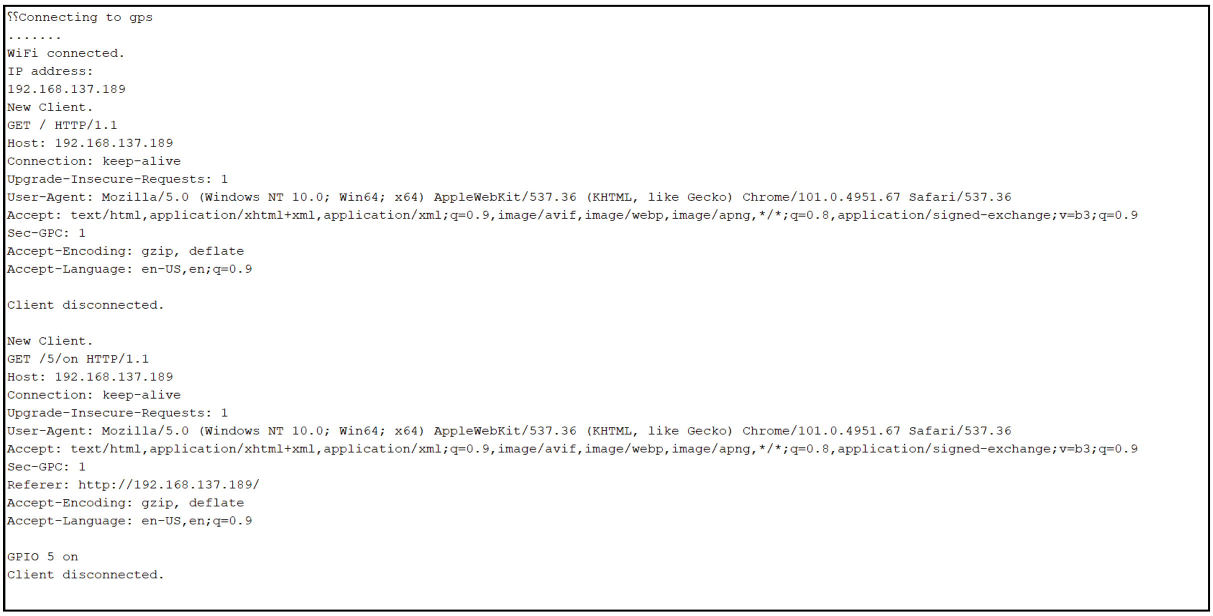


Code:

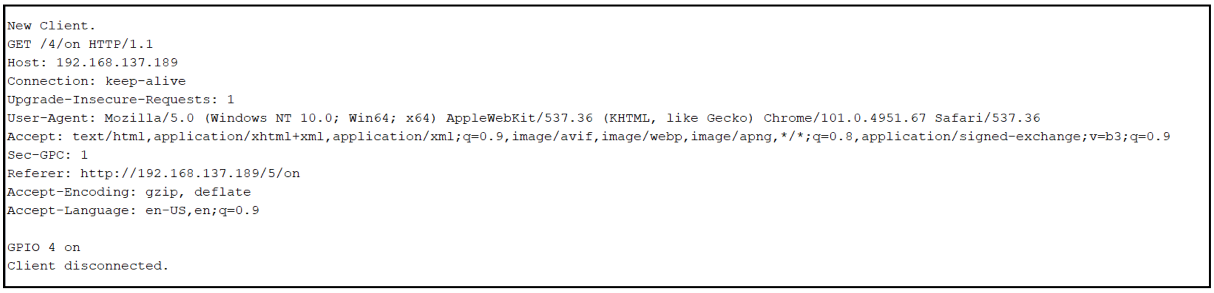
```
#include<ESP8266WiFi>

#include<ESP8266Webserver.h>

ESP8266Webserver server(80)

Char* ssid, Char pass;

Void setup()

WiFi.begin(SSid, Password);

Serial.begin(115200);

Serial.print(“Connecting: ”);

While (WiFi.status()!= WL_CONNECTED)
```

```
{
 Serial.print(“Waiting to connect”)
While(WiFi.Status()!=WL_CONNECTED)
{
Serial. Print(“Waiting to connect”);
delay(1000);
}
Serial.println(‘\n’);
Serial.println (“Connection established”);
Serial.println (“IP Address\t”);
Serial.println(WiFi.LocalIP());
}
Server.on(%[]() { Server.send()
}
```

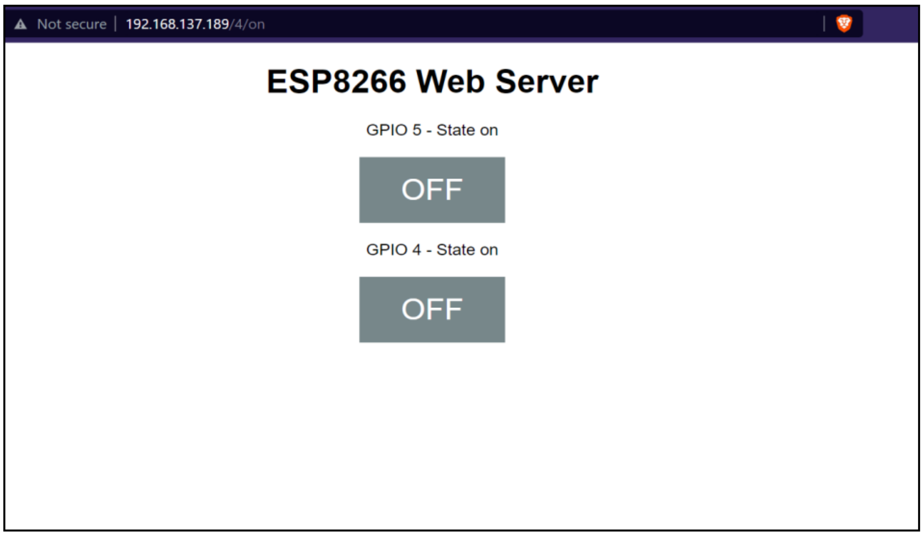
Output:



```
⸮⸮Connecting to gps
.......
WiFi connected.
IP address:
192.168.137.189
New Client.
GET / HTTP/1.1
Host: 192.168.137.189
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.67 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-GPC: 1
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

Client disconnected.

New Client.
GET /5/on HTTP/1.1
Host: 192.168.137.189
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.67 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-GPC: 1
Referer: http://192.168.137.189/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

GPIO 5 on
Client disconnected.
```



```
New Client.
GET /4/on HTTP/1.1
Host: 192.168.137.189
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.67 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-GPC: 1
Referer: http://192.168.137.189/5/on
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

GPIO 4 on
Client disconnected.
```



ESP8266 Web Server

GPIO 5 - State on

GPIO 4 - State on

Result:

Making a node has been done successfully using Node MCU and the results are verified

Exp.No: 21 **Date:** 19-05-22

Interfacing a GPS

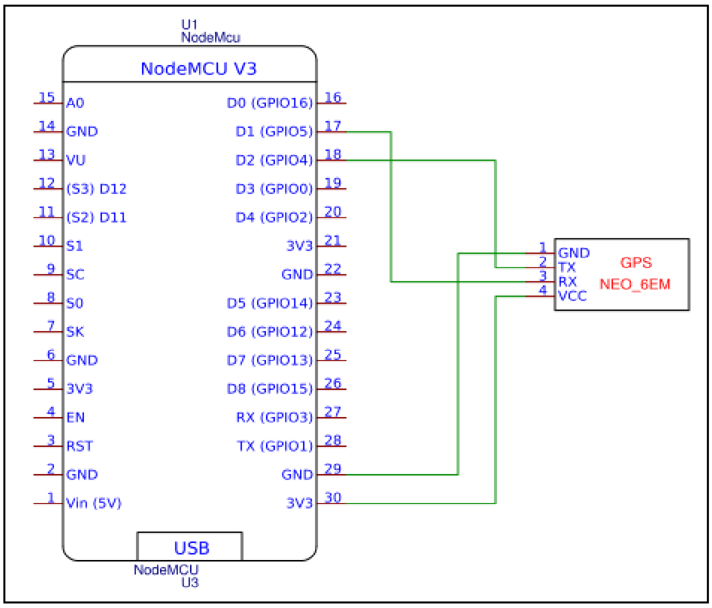
Aim:

To find the present location using GPS module

Hardware Required:

- NodeMCU ESP8266
- GPS module
- Bread Board
- Jumper wires

Circuit Diagram:



Code:

```
#include <TinyGPS++.h> // library for GPS module

#include <SoftwareSerial.h>

#include <ESP8266WiFi.h>

TinyGPSPlus gps;  // The TinyGPS++ object

SoftwareSerial ss(4, 5); // The serial connection to the GPS device

const char* ssid = "Onlilo_SP"; //ssid of your wifi

const char* password = "ArduinoUno"; //password of your wifi

float latitude , longitude;
```

```
int year , month , date, hour , minute , second;
String date_str , time_str , lat_str , lng_str;
int pm;
WiFiServer server(80);
void setup()
{
  Serial.begin(115200);
  ss.begin(9600);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password); //connecting to wifi
  while (WiFi.status() != WL_CONNECTED)// while wifi not connected
  {
    delay(500);
    Serial.print("."); //print "...."
  }
  Serial.println("");
  Serial.println("WiFi connected");
  server.begin();
  Serial.println("Server started");
  Serial.println(WiFi.localIP());  // Print the IP address
}
void loop()
{
  while (ss.available() > 0) //while data is available
    if (gps.encode(ss.read())) //read gps data
    {
      if (gps.location.isValid()) //check whether gps location is valid
      {
```

```
  latitude = gps.location.lat();
  lat_str = String(latitude , 6); // latitude location is stored in a string
  longitude = gps.location.lng();
  lng_str = String(longitude , 6); //longitude location is stored in a string
}
if (gps.date.isValid()) //check whether gps date is valid
{
  date_str = "";
  date = gps.date.day();
  month = gps.date.month();
  year = gps.date.year();
  if (date < 10)
    date_str = '0';
  date_str += String(date);// values of date,month and year are stored in a string
  date_str += " / ";

  if (month < 10)
    date_str += '0';
  date_str += String(month); // values of date,month and year are stored in a string
  date_str += " / ";
  if (year < 10)
    date_str += '0';
  date_str += String(year); // values of date,month and year are stored in a string
}
if (gps.time.isValid())  //check whether gps time is valid
{
  time_str = "";
  hour = gps.time.hour();
  minute = gps.time.minute();
  second = gps.time.second();
  minute = (minute + 30); // converting to IST
  if (minute > 59)
  {
    minute = minute - 60;
    hour = hour + 1;
  }
  hour = (hour + 5) ;
  if (hour > 23)
    hour = hour - 24;   // converting to IST
  if (hour >= 12)  // checking whether AM or PM
    pm = 1;
  else
    pm = 0;
  hour = hour % 12;
  if (hour < 10)
    time_str = '0';
  time_str += String(hour); //values of hour,minute and time are stored in a string
  time_str += " : ";
  if (minute < 10)
    time_str += '0';
  time_str += String(minute); //values of hour,minute and time are stored in a string
  time_str += " : ";
  if (second < 10)
    time_str += '0';
  time_str += String(second); //values of hour,minute and time are stored in a string
  if (pm == 1)
    time_str += " PM ";
  else
    time_str += " AM ";
}
```

```
    }


 WiFiClient client = server.available(); // Check if a client has connected
  if (!client)
  {
    return;
  }
  // Prepare the response
  String s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n <!DOCTYPE html>
<html> <head> <title>GPS DATA</title> <style>";
  s += "a:link {background-color: YELLOW;text-decoration: none;}";
  s += "table, th, td </style> </head> <body> <h1  style=";
  s += "font-size:300%;";
  s += " ALIGN=CENTER> GPS DATA</h1>";
  s += "<p ALIGN=CENTER style=""font-size:150%;""";
  s += "> <b>Location Details</b></p> <table ALIGN=CENTER style=";
  s += "width:50%";
  s += "> <tr> <th>Latitude</th>";
  s += "<td ALIGN=CENTER >";
  s += lat_str;
  s += "</td> </tr> <tr> <th>Longitude</th> <td ALIGN=CENTER >";
  s += lng_str;
  s += "</td> </tr> <tr>  <th>Date</th> <td ALIGN=CENTER >";
  s += date_str;
  s += "</td></tr> <tr> <th>Time</th> <td ALIGN=CENTER >";
  s += time_str;
  s += "</td>  </tr> </table> ";


  s += "</body> </html>"
  client.print(s); // all the values are send to the webpage
  delay(100);
```

```
}
```

Output:

Webpage displays date, time , longitude and latitude of the location.

Result:

Location has been successfully identified using GPS and Node MCU and the results are verified.

Exp.No: 22 **Date:** 19-05-22

Interfacing Bluetooth

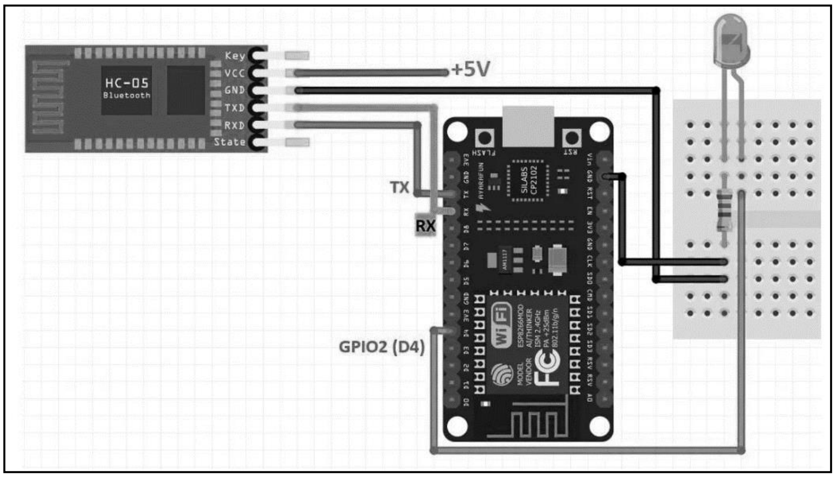
Aim:

To interface Bluetooth module using Node MCU

Hardware Required:

- ESP8266 Node MCU
- Bluetooth Module
- LED
- 1K-ohm resistor
- Connecting Wires
- Breadboard

Circuit Diagram:



Code:

```
int led_pin = 2;
void setup() {
 pinMode(led_pin, OUTPUT);
 Serial.begin(9600);
}
void loop() {


 if (Serial.available())
```

```
  {
    char data_received;
    data_received = Serial.read();
    if (data_received == 'O')
    {
     digitalWrite(led_pin, HIGH);
     Serial.write("LED is now ON!\n");
    }
    else if (data_received == 'X')
    {
     digitalWrite(led_pin, LOW);
     Serial.write("LED is now OFF!\n");
    }
    else
    {
     Serial.write("Specify correct option\n");
    }
  }
}
```

Output:

LED is ON when O is sent

LED is OFF when X is sent

Result:

Bluetooth module has been successfully interfaced with Node MCU and the results are verified.

Exp.No: 23 **Date:** 19-05-22

Interfacing GSM

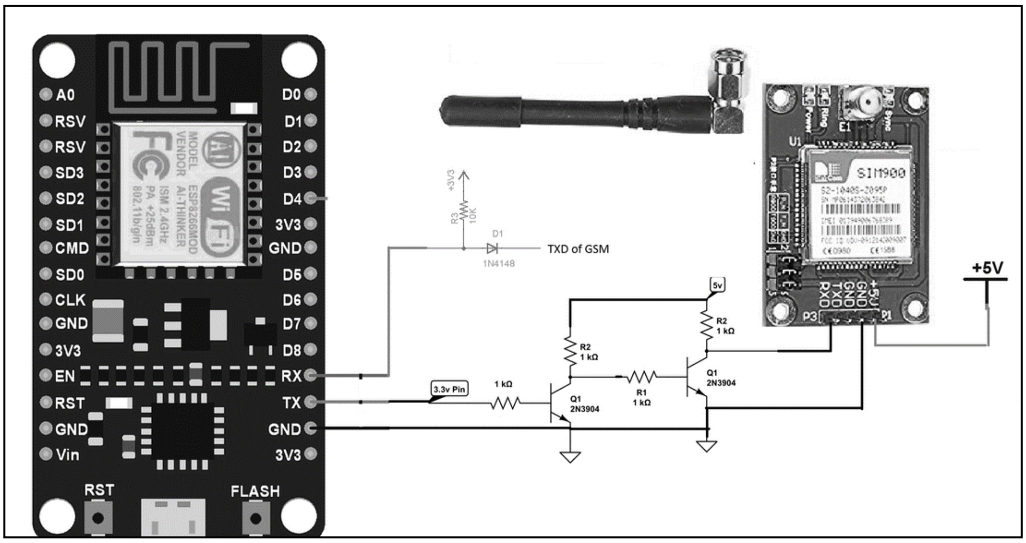
Aim:

To interface with GSM and send message using Node MCU

Hardware Required:

- GSM SIM900A
- Node MCU
- Jumper Wire
- Power adapter 5V
- SIM card
- Breadboard

Circuit Diagram :



Code:

```
void setup()
    {
      //Begin nodemcu serial-0 channel
      Serial.begin(9600);
    }
  void loop()
    {
    Serial.print("AT");  //Start Configuring GSM Module
    delay(1000);        //One second delay
    Serial.println();
    Serial.println("AT+CMGF=1");  // Set GSM in text mode
    delay(1000);                  // One second delay
    Serial.println();
    Serial.print("AT+CMGS=");     // Enter the receiver number
    Serial.print("\"+91XXXXXXXXXX\"");
    Serial.println();
    delay(1000);
    Serial.print("IOT LAB"); // SMS body - Sms Text
    delay(1000);
    Serial.println();
    Serial.write(26);              //CTRL+Z Command to send text and end session
    while(1);                      //Just send the text ones and halt
    }
```

Output:

SMS is received at the given mobile number

Result:

Interface with GSM and sending a SMS using Node MCU has been done successfully and the results are verified.

Exp.No: 24 **Date:** 23-05-22

IoT Cloud Applications

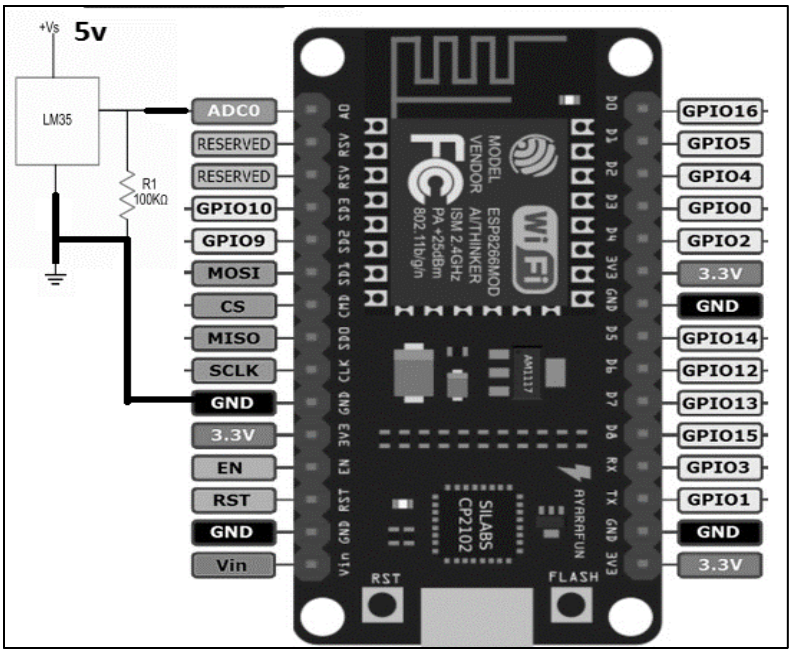
1)**Aim:**

To collecting and processing data from IoT systems in the cloud using Thing Speak.

Hardware Requirements:

- NodeMCU
- LM35 Temperature Sensor
- Bread Board
- Jumper Wires
- Micro USB Cable
- Arduino IDE
- 200 ohm – 1K ohm resistor x 1
- PC

Circuit Diagram:

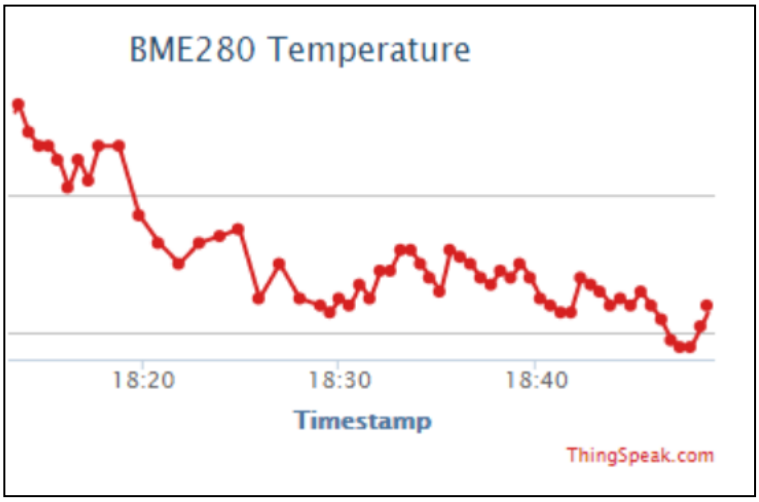


Code:

```
#include <ESP8266WiFi.h>

#include "ThingSpeak.h"

char msg[50];

const char* ssid = "boolean";   // your network SSID (name)

const char* password = "meowmeow";   // your network password

WiFiClient  client;

unsigned long myChannelNumber = 0000000;  // Your channel number
```

```
const char * myWriteAPIKey = "UGGJHGJHJHJ";  // Your WriteAPI Key
// Timer variables
unsigned long lastTime = 0;
unsigned long timerDelay = 30000;
// Variable to hold temperature readings
float temperatureC;
int outputpin = A0;
void setup() {
  Serial.begin(115200);  //Initialize serial
  WiFi.mode(WIFI_STA);
  ThingSpeak.begin(client);  // Initialize ThingSpeak
  if (WiFi.status() != WL_CONNECTED) {
      Serial.print("Attempting to connect");
      while (WiFi.status() != WL_CONNECTED) {
        WiFi.begin(ssid, password);
        delay(5000);
      }
      Serial.println("\nConnected.");
    }
}
void loop() {
  if ((millis() - lastTime) > timerDelay || 1) {
    int analogValue = analogRead(outputpin);
    float millivolts = (analogValue / 1024.0) * 3300;
    temperatureC = millivolts / 80 + random(10);
    Serial.print("Temperature (ºC): ");
    Serial.println(temperatureC);
    int x = ThingSpeak.writeField(myChannelNumber, 1, temperatureC, myWriteAPIKey);
    if (x == 200) {
      Serial.println("Channel update successful.");
    }
    else {
      Serial.println("Problem updating channel. HTTP error code " + String(x));
    }
    lastTime = millis();
  }
}
```

Output:



Result:

Data has been successfully connected and processed using ThingSpeak.

Exp.No: 25 **Date:** 23-05-22

Develop IoT Applications using Bluemix

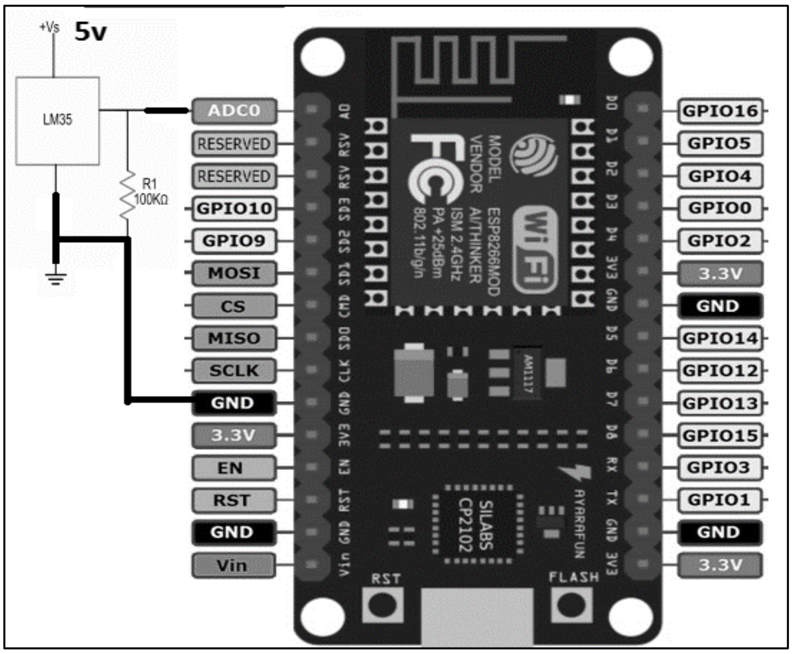
Aim:

To develop IoT applications using Bluemix platform.

Hardware Requirements:

- NodeMCU
- LM35 Temperature Sensor
- Bread Board
- Jumper Wires
- Micro USB Cable
- Arduino IDE
- 200 ohm – 1K ohm resistor x 1
- PC

Circuit Diagram:

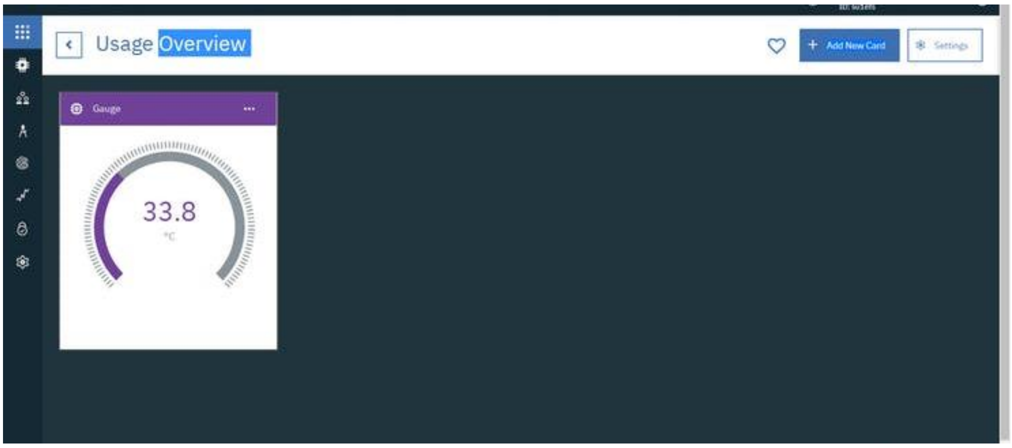


Code:

```
#include <ESP8266WiFi.h>

#include <ArduinoJson.h>

#include <PubSubClient.h>

// Watson IoT connection details

#define MQTT_HOST "vzrica.messaging.internetofthings.ibmcloud.com"

//change  su1efs

#define MQTT_PORT 1883
```

```
#define MQTT_DEVICEID "d:vzrica:ESP8266:dev01"
//change  su1efs
#define MQTT_USER "use-token-auth"
#define MQTT_TOKEN "Manoj_Selvam" // change your auth_id :
#define MQTT_TOPIC "iot-2/evt/status/fmt/json"
#define MQTT_TOPIC_DISPLAY "iot-2/cmd/display/fmt/json"
// Add WiFi connection information
char ssid[] = "galaxy123";     //  your network SSID (name)
char pass[] = "tebv7047";  // your network password
// MQTT objects
void callback(char* topic, byte* payload, unsigned int length);
WiFiClient wifiClient;
PubSubClient mqtt(MQTT_HOST, MQTT_PORT, callback, wifiClient);
// variables to hold data
StaticJsonDocument<100> jsonDoc;
JsonObject payload = jsonDoc.to<JsonObject>();
JsonObject status = payload.createNestedObject("d");
static char msg[50];
float t = 0.0;
void callback(char* topic, byte* payload, unsigned int length) {
  // handle message arrived
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] : ");
  payload[length] = 0; // ensure valid content is zero terminated so can treat as c-string
  Serial.println((char *)payload);
}
void setup() {
 // Start serial console
  Serial.begin(115200);
  Serial.setTimeout(2000);
  while (!Serial) { }
  Serial.println();
  Serial.println("ESP8266 Sensor Application");
  // Start WiFi connection
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi Connected");
  // Connect to MQTT - IBM Watson IoT Platform
  if (mqtt.connect(MQTT_DEVICEID, MQTT_USER, MQTT_TOKEN)) {
    Serial.println("MQTT Connected");
    mqtt.subscribe(MQTT_TOPIC_DISPLAY);
  } else {
    Serial.println("MQTT Failed to connect!");
    ESP.reset();
  }
}
void loop() {
  mqtt.loop();
  while (!mqtt.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (mqtt.connect(MQTT_DEVICEID, MQTT_USER, MQTT_TOKEN)) {
      Serial.println("MQTT Connected");
      mqtt.subscribe(MQTT_TOPIC_DISPLAY);
      mqtt.loop();
    } else {
      Serial.println("MQTT Failed to connect!");
      delay(5000);
    }
  }



  // get t from temp sensor
  int analogValue = analogRead(A0);
  float millivolts = (analogValue/1024.0) * 3300;
  t = millivolts/40;
  // Check if any reads failed and exit early (to try again).
  if ( isnan(t) )  Serial.println("Failed to read from DHT sensor!");
  else {
    // Send data to Watson IoT Platform
    status["temp"] = t+random(10);
    serializeJson(jsonDoc, msg, 50);
    Serial.println(msg);
    if (!mqtt.publish(MQTT_TOPIC, msg)) {
      Serial.println("MQTT Publish failed");
    }
  }
  // Pause - but keep polling MQTT for incoming messages
  for (int i = 0; i < 10; i++) {
    mqtt.loop();
    delay(1000);}
}
}
```

Output:



Result:

IoT applications using Bluemix platform has been successfully developed and the results are verified.