MAHA BARATHI ENGINEERING COLLEGE

NH-79, SALEM-CHENNAI HIGHWAY, A.VASUDEVANUR, CHINNASALEM (TK), KALLAKURICHI (DT) 606 201.
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai
Accredited by NAAC and Recognized under section 2(f) & 12(B) status of UGC, New Delhi

www.mbec.ac.in | Ph: 04151-256333, 257333 | E-mail: mbec123@gmail.com

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

ET3491- EMBEDDED SYSTEMS AND IOT DESIGN

III Year/ VI Semester B.E ECE

Regulation 2021
(As Per Anna University, Chennai syllabus)

MAHA BARATHI ENGINEERING COLLEGE

NH-79, SALEM-CHENNAI HIGHWAY, A.VASUDEVANUR, CHINNASALEM (TK), KALLAKURICHI (DT) 606 201.
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai
Accredited by NAAC and Recognized under section 2(f) & 12(B) status of UGC, New Delhi

www.mbec.ac.in | Ph: 04151-256333, 257333 | E-mail: mbec123@gmail.com

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

BONAFIDE CERTIFICATE

NAME: **COURSE: B.E-ECE**

REG.NO.: **SEMESTER:VI**

This is to certify that the bonafide record of work done by the student in the

ET3491- EMBEDDED SYSTEMS AND IOT DESIGN in the Department of Electronics and Communication Engineering of Maha Barathi Engineering College during the Academic year 2023-24.

Faculty In-Charge **Head of the Department**
Date:

Submitted for the Practical Examination held on…………….

INTERNAL EXAMINER **EXTERNAL EXAMINER**

INDEX

ARITHMETIC OPERATIONS USING 8051

Experiment No.1 (a) **Date:**

AIM:

To perform 8 bit arithmetic operations of two numbers using 8051MicroController (MC) kit.

APPARATUS REQUIRED:

- 8051 Microcontroller kit, Keyboard & Power cable

PROGRAM: 8 BIT ADDITION

MEMORY ADDRESS	OBJECT CODES	MNEMONICS
4100	C3	CLR C
4101	74	MOV A,#DATA1
4102	20	
4103	94	ADD A,#DATA2
4104	10	
4105	90	MOV DPTR,#4500
4106	45	
4107	00	
4108	F0	MOVX @DPTR,A
4109	80	HERE: SJMP HERE
410A	FE	

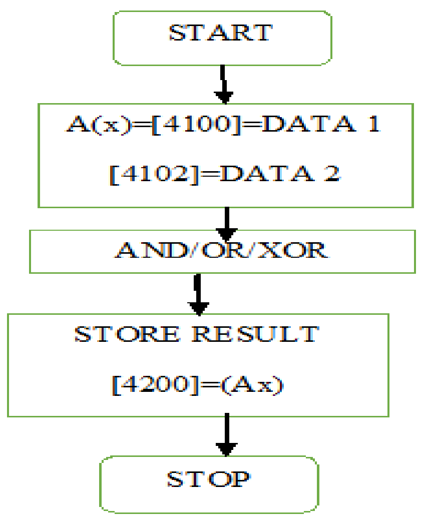
PROCEDURE:

1. Enter the above opcodes from 4100.

2. Execute the program; see that the result is stored correctly.

3.Change data and check if the results are correct each time

FLOW CHART:



PROGRAM: 8 BIT SUBTRACTION

MEMORY ADDRESS	OBJECT CODES	MNEMONICS
4100	74	MOV A,#DATA1
4101	20	
4102	94	SUBB A,#DATA2
4103	10	
4104	90	MOV DPTR,#4500
4105	45	
4106	00	
4107	F0	MOVX @DPTR,A
4108	80	HERE: SJMP HERE

<u>PROGRAM :8 BIT MULTIPLICATION</u>

MEMORY ADDRESS	OBJECT CODES	MNEMONICS
4100	74	MOV A,#DATA1
4101	0A	
4102	75	MOV B,#DATA2
4103	F0	
4104	88	
4105	A4	MUL AB
4106	90	MOV DPTR,#4500
4107	45	
4108	00	
4109	F0	MOVX @DPTR,A
410A	A3	INC DPTR
410B	E5	MOV A,B
410C	F0	
410D	F0	MOVX @DPTR,A
410E	80	HERE: SJMP HERE

PROGRAM :8 BIT DIVISION

MEMORY ADDRESS	OBJECT CODES	MNEMONICS
4100	74	MOV A,#DATA1
4101	0A	
4102	75	MOV B,#DATA2
4103	F0	
4104	84	DIV A,B
4105	90	MOV DPTR,#4500
4106	45	
4107	00	
4108	F0	MOVX @DPTR,A
4109	A3	INC DPTR
410A	E5	MOV A,B
410B	F0	
410C	F0	MOVX @DPTR,A
410D	80	HERE: SJMP HERE

RESULT:

Thus the arithmetic operations of 8051Microcontroller have been executed and output was verified successfully.

LOGICAL OPERATIONS USING 8051

Experiment No.1 (b) **Date:**

AIM:

To perform 8 bit logical operations of two numbers using 8051MicroController (MC) kit.

APPARATUS REQUIRED:

- 8051 Microcontroller kit, Keyboard & Power cable

PROGRAM: 8 BIT LOGIC OPERATION

MEMORY ADDRESS	OBJECT CODES	MNEMONICS
4100	74	MOV A,#DATA1
4101	20	
4102	94	ANL A,#DATA2 ORL A,#DATA2 XRL, #DATA2
4103	10	
4104	90	MOV DPTR,#4500
4105	45	
4106	00	
4107	F0	MOVX @DPTR,A
4108	80	HERE: SJMP HERE

PROCEDURE:

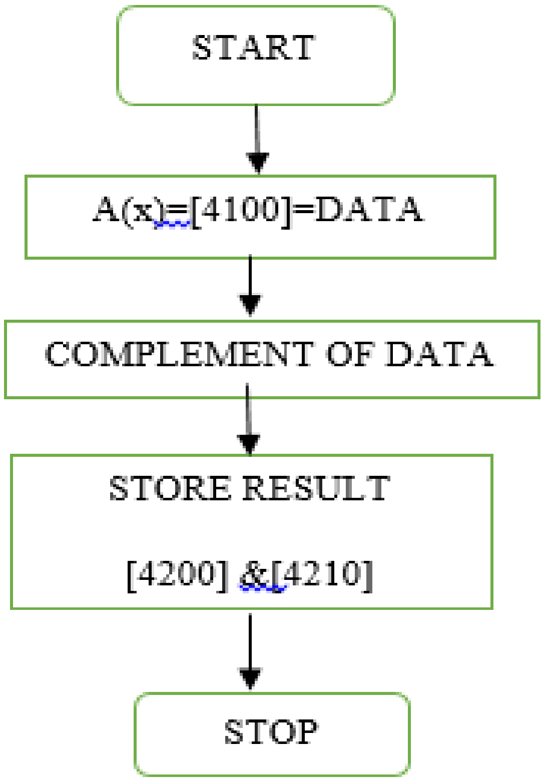
1. Enter the opcodes and the data in the trainer.
2. Execute the program and check for results.
3. Change data and check for the corresponding results.

ONE'S AND TWO'S COMPLEMENT OF NUMBER

PROGRAM:

MEMORY ADDRESS	OBJECT CODES	MNEMONICS
4100	74	MOV A,#DATA
4101	CC	
4102	F4	CPL A
4103	90	MOV DPTR,#4200
4104	42	
4105	00	
4106	F0	MOVX @DPTR,A
4107	04	INC A
4108	A3	INC DPTR
4109	F0	MOVX @DPTR,A
410A	80	HERE: SJMP HERE
410B	FE	

FLOW CHART:



RESULT:

Thus the logical operations of 8051Microcontroller have been executed and output was verified successfully.

GENERATION OF SQUARE WAVEFORM

Experiment No.2 **Date:**

AIM:

To interface the DAC with 8051 microcontroller and generate the square wave, saw tooth wave and triangular wave.

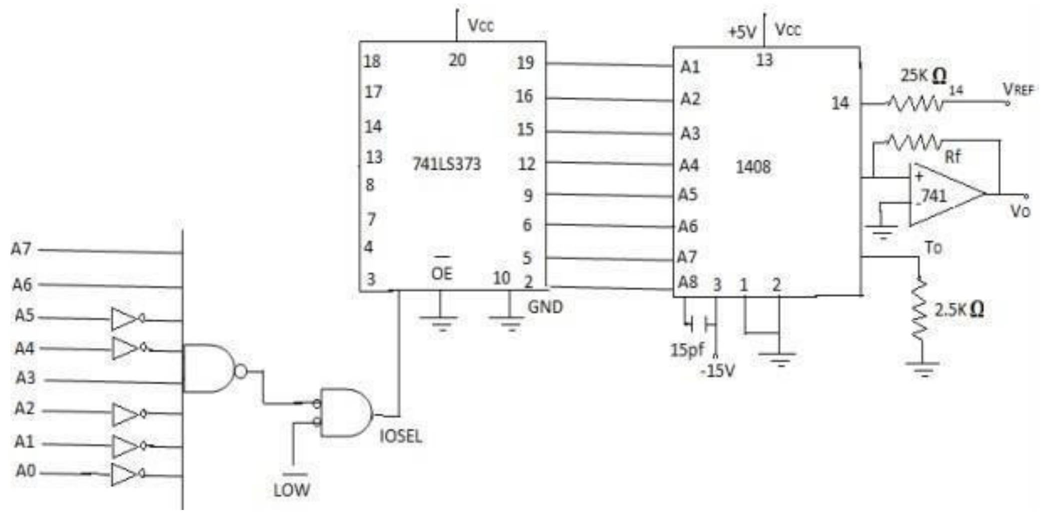
APPARATUS REQUIRED:

- 8051 Microcontroller kit
- Keyboard
- Power cable
- DAC interfacing board and CRO

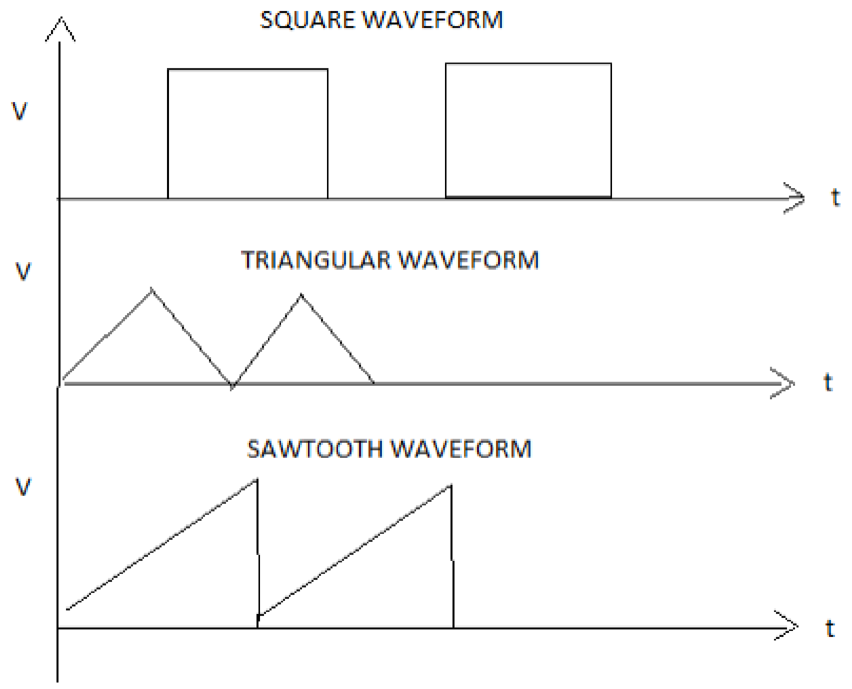
PROGRAM: SQUARE WAVEFORM

ADDRESS	LABEL	MNEMONICS	OPCODE	COMMENTS
4100		MOV DPTR, # E0C0	90,FF,C8	Move the immediateData EOCO Initialize the AccumulatorTo zero Long call the delay Move the content of Accumulator to FF Long call delay Long jump to start Move the 05 data To R register Decrement Jump NON zeroReturn to main program Short jump to start
4103	ST ART	MOV A, # 00	74,00	
4105		MOVX @ DPTR, A	F0	
4106		LCALL DELAY	12,41,12	
4109		MOV A, # FF	74,FF	
410B		MOVX @ DPTR, A	F0	
410C		LCALL DELAY	12,41,12	
410F		LJMP START	02,41,03	
4112	DELAY	MOV R1, # 05	79,05	
4114	LOOP	MOV R2, # FF	74,FF	
4116	HE RE	DJNZ R2, HERE	DA,FE	
4118		DJNZ R1, LOOP	D9,FA	
411A		RET	22	
411C		SZMP START	80,E3	

CIRCUIT DIAGRAM:



WAVEFORMS:



SQUARE WAVE

OBSERVATION:

AMPLITUDE	TIME PERIOD

RESULT:

Thus the generation of square waveform using 8051Microcontroller has been executed and the output was verified successfully.

PROGRAMMING USING ON CHIP PORTS IN 8051

Experiment No.3 **Date:**

AIM:

To create an assembly language program for on chip ports in 8051 Microcontroller.

APPARATUS REQUIRED:

- ❖ 8051 Microcontroller kit
- ❖ Keyboard
- ❖ Power cable

MEMORY ADDRESS	OBJECT CODES	MNEMONICS
4100	74	MOV P1,#0xFF
4101	FF	
4102	75	CLRP1.0
4103	01	
4104	D2	LOOP:SETB P1.0
4105	02	ACALL DELAY
4106	XX	
4107	C2	CLRP1.0
4108	02	ACALL DELAY
4109	XX	
410A	80	SJMP LOOP
410B	FA	
410C	75	DELAY:MOV R2,#0Xff
410D	FF	
410E	75	OUT LOOP:MOV R1,#0xFF
410F	FF	
4110	DE	IN LOOP:DJNZ R1, IN LOOP
4111	01	
4112	DE	DJNZ R2, OUT LOOP
4113	02	
4114	22	RET

RESULT:

Thus the programming on chip port of 8051Microcontroller has been executed and the output was verified successfully.

PROGRAMMING USING SERIAL PORTS IN 8051

Experiment No.4 **Date:**

AIM:

To create an assembly language program for serial ports in 8051 Microcontroller.

APPARATUS REQUIRED:

- 8051 Microcontroller kit
- Keyboard
- Power cable

PROGRAM:

```
MOV TMOD, #20H
MOV TH1, #-3 H
MOV SCON, #50H
SET TR1
MOV  SBUF  ,#"Y"
JNB TI, HERE
CLR TI
MOV SBUF, #"N"
SJMP AGAIN
```

RESULT:

Thus the programming serial port of 8051Microcontroller has been executed and the output was verified successfully.

DIGITAL CLOCK

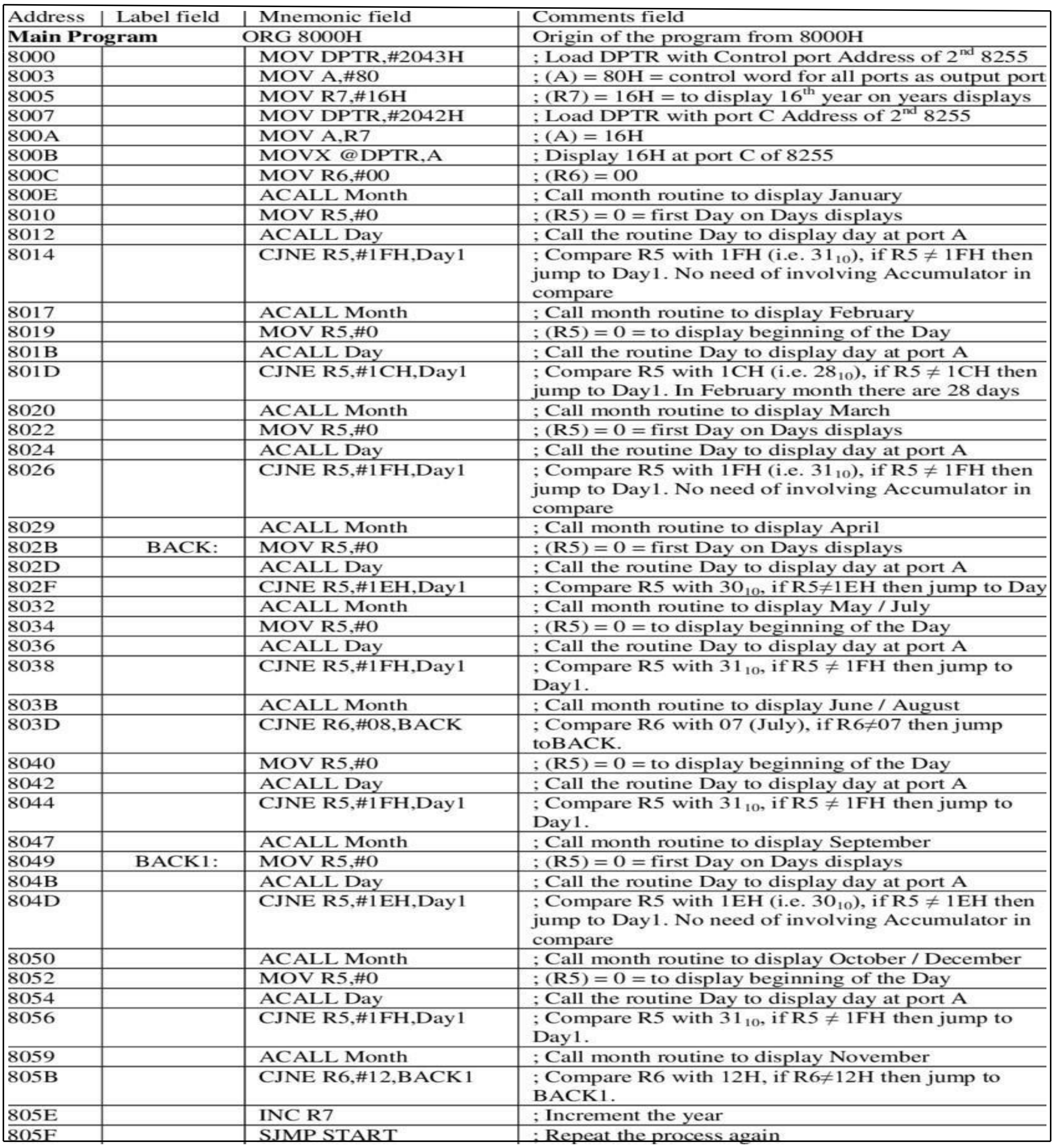
Experiment No. 5 **Date:**

D
I
G
I

AIM:

To display the digital clock specifically by displaying the hours, minutes and seconds using 8051 kits

PROGRAM:



Address	Label field	Mnemonic field	Comments field
Main Program		ORG 8000H	Origin of the program from 8000H
8000		MOV DPTR,#2043H	; Load DPTR with Control port Address of 2nd 8255
8003		MOV A,#80	; (A) = 80H = control word for all ports as output port
8005		MOV R7,#16H	; (R7) = 16H = to display 16th year on years displays
8007		MOV DPTR,#2042H	; Load DPTR with port C Address of 2nd 8255
800A		MOV A,R7	; (A) = 16H
800B		MOVX @DPTR,A	; Display 16H at port C of 8255
800C		MOV R6,#00	; (R6) = 00
800E		ACALL Month	; Call month routine to display January
8010		MOV R5,#0	; (R5) = 0 = first Day on Days displays
8012		ACALL Day	; Call the routine Day to display day at port A
8014		CJNE R5,#1FH,Day1	; Compare R5 with 1FH (i.e. 31_{10}), if R5 ≠ 1FH then jump to Day1. No need of involving Accumulator in compare
8017		ACALL Month	; Call month routine to display February
8019		MOV R5,#0	; (R5) = 0 = to display beginning of the Day
801B		ACALL Day	; Call the routine Day to display day at port A
801D		CJNE R5,#1CH,Day1	; Compare R5 with 1CH (i.e. 28_{10}), if R5 ≠ 1CH then jump to Day1. In February month there are 28 days
8020		ACALL Month	; Call month routine to display March
8022		MOV R5,#0	; (R5) = 0 = first Day on Days displays
8024		ACALL Day	; Call the routine Day to display day at port A
8026		CJNE R5,#1FH,Day1	; Compare R5 with 1FH (i.e. 31_{10}), if R5 ≠ 1FH then jump to Day1. No need of involving Accumulator in compare
8029		ACALL Month	; Call month routine to display April
802B	BACK:	MOV R5,#0	; (R5) = 0 = first Day on Days displays
802D		ACALL Day	; Call the routine Day to display day at port A
802F		CJNE R5,#1EH,Day1	; Compare R5 with 30_{10}, if R5≠1EH then jump to Day
8032		ACALL Month	; Call month routine to display May / July
8034		MOV R5,#0	; (R5) = 0 = to display beginning of the Day
8036		ACALL Day	; Call the routine Day to display day at port A
8038		CJNE R5,#1FH,Day1	; Compare R5 with 31_{10}, if R5 ≠ 1FH then jump to Day1.
803B		ACALL Month	; Call month routine to display June / August
803D		CJNE R6,#08,BACK	; Compare R6 with 07 (July), if R6≠07 then jump toBACK.
8040		MOV R5,#0	; (R5) = 0 = to display beginning of the Day
8042		ACALL Day	; Call the routine Day to display day at port A
8044		CJNE R5,#1FH,Day1	; Compare R5 with 31_{10}, if R5 ≠ 1FH then jump to Day1.
8047		ACALL Month	; Call month routine to display September
8049	BACK1:	MOV R5,#0	; (R5) = 0 = first Day on Days displays
804B		ACALL Day	; Call the routine Day to display day at port A
804D		CJNE R5,#1EH,Day1	; Compare R5 with 1EH (i.e. 30_{10}), if R5 ≠ 1EH then jump to Day1. No need of involving Accumulator in compare
8050		ACALL Month	; Call month routine to display October / December
8052		MOV R5,#0	; (R5) = 0 = to display beginning of the Day
8054		ACALL Day	; Call the routine Day to display day at port A
8056		CJNE R5,#1FH,Day1	; Compare R5 with 31_{10}, if R5 ≠ 1FH then jump to Day1.
8059		ACALL Month	; Call month routine to display November
805B		CJNE R6,#12,BACK1	; Compare R6 with 12H, if R6≠12H then jump to BACK1.
805E		INC R7	; Increment the year
805F		SJMP START	; Repeat the process again

Observation:

Input

1200	00
1201	00
1202	00
1203	00
1204	00

Output:

Time is displayed in the RTC board as

	! Hour l·		Minutes	l· seconds l·	
X	0	0	0	5	9

X	0	0	1	0	0

RESULT:

Thus the design of digital clock has been executed in 8051 Microcontroller and output was verified successfully.

INTERFACING ADC WITH ARM PROCESSOR

Experiment No.1(a) **Date:**

AIM:

To interface and Convert Analog Signal in Digital form using ARM processor.

APPARTAUS REQUIRED:

1. PC with keil µversion 5and flash magic workbench software
2. LPC2148 processor kit
3. USB Cable

THEORY:

In electronics, an analog-to-digital converter (ADC, A/D, A–D, or A-to-D) is a system that converts an analog signal into a digital signal.

An ADC may also provide an isolated measurement such as an electronic device that converts an input analog voltage or current to a digital number proportional to the magnitude of the voltage or current. Typically the digital output will be a two's complement binary number that is proportional to the input, but there are other possibilities.

PROCEDURE:

Keil Compiler

Step 1

Open keil µversion project → menu → project → new µ version project → Select location → open new folder → give file name → save

Step 2

Menu → file → new

Step3

Type a program → save a program with .c or .hextensions

Step 4

Right corner → target → source group → right click → add existing files to group "source group1" → add "c" and "h" files

Step5

Top of main window → target1 → options for target → small window open → output (create hex file) enable → listing(assembler listing) disable → Asm(enable ARM thump inter working) disable → linker (use memory layout from target dialog) enable → OK

Step 6 Build a program $\rightarrow$ check errors and warnings

Flash magic

Open Flash Magic s/w

Step 1: Select a device $\rightarrow$ LPC2148

Select comport $\rightarrow$ USB comport

Baud rate $\rightarrow$ 9600

Oscillator freq $\rightarrow$ 12MHZ

Step 2: Erase blocks used by Hex files (enable)

Step 3:Browse.hex file

Step 4: Select verifying after programming

Step5: Click start

PROGRAM CODE:

MAIN ADC TEST

```
#include<LPC214x.H>
#include"ADC_Driver.c"

#include "lcd.c"
#include <stdio.h>
int main (void)
 {
  unsigned int adc_val;
  unsigned int temp;
  unsigned char buf[4] = {0,0,0,0};
  ADCInit();
  lcdinit();
    //wait();
            clrscr(10);
    printstr("ADC Test",0,0);
            wait();
  while(1)
   {
            adc_val = ADC_ReadChannel();
            temp = (unsigned int)((3*adc_val*100)/1024);
            sprintf(buf,"%d",temp);
            printstr(buf,0,1);
   }
 }
```

LCD.C

```
#include <LPC214x.h>
#defineRS      0x00000400
#defineCE      0x00001800
void clrscr(charch);
void lcdinit(void);
void lcdcmd(char);
void lcddat(char);
void gotoxy(char,char);
void printstr(unsigned char *,char,char);
void wait (void);
void split_numbers(unsigned int number);
#define SET1
#define OFF0
unsigned int thousands,hundreds,tens,ones;
void wait (void){
  int d;
  for (d = 0; d <100000;d++);
void lcdinit()
{
    IODIR0 |= 0xFFFFFFFF;
            IOCLR0 |= 0X00000FFF;
            lcdcmd(0x28);
    lcdcmd(0x28);
    lcdcmd(0x0c);
    lcdcmd(0x06);
    lcdcmd(0x01);
    lcdcmd(0x0f);
    wait();
}
void gotoxy(char x, char y)
{
    if(y == 0)
        lcdcmd(0x80+x);
    else
        lcdcmd(0xc0+x);
}
void printstr(unsigned char *str, char x, char y)
{
    char i;
    gotoxy(x,y);
    wait();//(500);
    for(i=0;str[i]!='\0';i++)
        lcddat(str[i]);
}
void lcdcmd(char cmd)
{
unsigned char LCDDAT;
             LCDDAT = (cmd&0xf0);
             IOSET0 = LCDDAT;
     IOCLR0 = RS;
             IOSET0 = CE;
     wait();//(100);

     IOCLR0 =CE;
             IOCLR0 = 0X00000FFF;
 LCDDAT = ((cmd<<0x04)&0xf0);
             IOSET0 = LCDDAT;
     IOCLR0 = RS;
             IOSET0 = CE;
     wait();//(100);

     IOCLR0 =CE;
```

```
            IOCLR0 = 0X00000FFF;
}
void lcddat(char cmd)
{
  unsigned char LCDDAT;
            LCDDAT = (cmd&0xf0);
            IOSET0 = LCDDAT;
    IOSET0 = RS;
            IOSET0 = CE;
    wait();//(100);

    IOCLR0 =CE;
            IOCLR0 = 0X00000FFF;
    LCDDAT = ((cmd<<0x04) & 0xf0);
            IOSET0 = LCDDAT;
    IOSET0 = RS;
            IOSET0 = CE;
    wait();//(100);

    IOCLR0 =CE;
            IOCLR0 = 0X00000FFF;
 }
void clrscr(char ch)
{
   if(ch==0)
   {
      printstr("                ",0,0);
      gotoxy(0,0);
   }
   else if(ch == 1)
   {
      printstr("                ",0,1);
      gotoxy(0,1);
   }
   else
   {
        lcdcmd(0x01);
       // delay(100);
    }
}
void split_numbers(unsigned int number)
{
  thousands = (number /1000);
  number %= 1000;
  hundreds = (number / 100);
  number %= 100;
  tens = (number / 10);
  number %= 10;
  ones = number ;
}
void Wait_Msg(void)
{
 lcdcmd(0x01);
 printstr(" Please Wait ", 0, 0);
}
void Welcome_Msg(void)
{
 lcdcmd(0x01);
 printstr(" Welcome to        ", 0,0);
 printstr("    MAHABARATHI     ", 0,1);
}
```

ADC_ DRIVER.C

```
#include<LPC214x.H>
Void ADC Init(void)
 {
  PINSEL1 |= 0x04000000;
  IODIR0 |= ~(0x04000000);
  AD0CR |= 0x00200204;
  AD0GDR
  ;
 }
void ADC_Start Conversion(void)
 {
   AD0CR |= (1<<24);
 }
void ADC_Stop Conversion(void)
 {
   AD0CR &= (~ (1<<24));
 }
unsigned int ADC_Read Channel(void)
 {
// unsigned int i;
   unsigned long ADC_Val, t;
   ADC_StartConversion();
   while((AD0DR2&0x80000000)==0);
   if(AD0STAT & 0x00000400)
               {
                //printstr("OVR",0,1);
                return(0);
               }
   t = AD0DR2;
   ADC_Val = ((t>>6) & 0x000003FF);//(AD0DR2 & 0x000003FF); //((AD0CR>>6) &
0x000003FF);
   //ADC_StopConversion();
   return(ADC_Val);
```

Testing:

- Connect Multimeter **BLACK**pin to the **Gnd** pin of LPC2148.
- Connect Multimeter **RED** pin to the **ADC** pin ofLPC2148.
- When working with the ADC, keep the switch position as givenbelow.

RESULT:

Thus the interfacing of analog to digital converter (ADC) has been executed in LPC2148 kit and output was verified successfully.

INTERFACING DAC WITH ARM PROCESSOR

Experiment No.1(b) **Date:**

AIM:

To interface and convert Digital signal in Analog form using ARM Processor.

APPARTAUS REQUIRED:

1. PC with keil µversion 5and flash magic workbench software
2. LPC2148 processor kit
3. Multimeter
4. USB Cable

THEORY:

Digital-to-analog conversion is a process in which signals having a few (usually two)defined levels or states (digital) are converted into signals having a theoretically infinite number of states (analog). A common example is the processing, by a modem, of computer data into audio-frequency (AF) tones that can be transmitted over a twisted pair telephone line. The circuit that performs this function is a *digital-to-analog converter (DAC)*.

Binary digital impulses, all by themselves, appear as long strings of ones and zeros, and have no apparent meaning to a human observer. But when a DAC is used to decode the binary digital signals, meaningful output appears. This might be a voice, a picture, a musical tune, or mechanical motion.

PROCEDURE:

Keil Compiler

Step 1

Open keil µversion project → menu → project → new µ version project → Select location → open new folder → give file name → save

Step 2

Menu → file → new

Step3

Type a program → save a program with .c or .h extensions

Step 4

Right corner → target → source group → right click → add existing files to group "source group1" → add "c" and "h" files

working) disable → linker (use memory layout from target dialog) enable → OK

Step 6

Build a program → check errors and warnings

Flash magic

Open Flash Magic s/w

Step 1: Select a device $\rightarrow$ LPC2148

Select comport $\rightarrow$ USB comport

Baud rate $\rightarrow$ 9600

Oscillator freq $\rightarrow$ 12MHZ

Step 2: Erase blocks used by Hex files (enable)

Step 3:Browse.hex file

Step 4: Select verifying after programming

Step5: Click start

PROGRAMCODE:

DAC.C

```
#include<LPC214.H>

void wait_long(void)
{
intd;
  for (d = 0; d <1000000;d++);
}
int main()
 {
      wait_long ();
      wait_long ();
      IODIR0  =  0X00000FFF;
      IODIR1  =  0XFFFF0000;
      IOSET0                    =
      0XFFFFFFFF;   IOCLR1
      = 0XFFFF0000;
      PINSEL1                  |=
      0x00080000;

        //DACR= 0X00017FC0;

          While (1);
        }
```

Testing with DAC:

- Connect Multimeter **BLACK** pin to the **Gnd** pin ofLPC2148.
- Connect Multimeter **RED** pin to the **DAC** pin ofLPC2148.

Output:

Measuring the voltage –

RESULT:

Thus the interfacing of Digital To Analog converter (DAC) have been executed in LPC2148 kit and output was verified successfully.

FLASHING OF LEDS

Experiment No.2(a) **Date:**

AIM:

To interface LED with ARM processor.

APPARATUS REQUIRED :

1. PC with keil µversion 5and flash magic workbench software
2. LPC2148 processor kit
3. USB Cable

THEORY

LED

Light Emitting Diodes (LED) is the most commonly used output components, usually for displaying pins digital states. Typical uses of LEDs include alarm devices, timers and confirmation of user input such as a mouse click or keystroke.

PROCEDURE:

Keil Compiler

Step 1

Open keil µversion project → menu → project → new µ version project → Select location → open new folder → give filename → save

Step 2 Menu → file → new

Step3

Type a program → save a program with .c or.h extensions

Step 4

Right corner → target → source group → right click → add existing files to group "source group1" → add "c" and "h" files

Step5

output(create hex file) enable → listing (assembler listing) disable → Asm (enable ARM thump inter working) disable → linker (use memory layout from target dialog) enable → OK

Step 6

Build a program → check errors and warnings.

PROGRAM CODE:

lcd.c

```
#include <LPC214x.h>
void wait (void)
{
 int d; for (d = 0; d <100000;d++);
}
int main(void)
{
    IODIR0 = 0x80002000;
    While(1)
{
    IOCLR0= 0x80002000;
  Wait();
    IOSET0= 0x80002000;
Wait();
}
}
```

Testing the LED with LPC2148

Give +3.3V power supply to LPC2148 Primer Board; the LED is connected with LPC2148 Primer Board. When the program is downloading into LPC2148 in Primer Board, the LED output is working that the LED is ON.

Output:

The **LPC2148 Kit** has 16 nos., of Point LEDs, connected with port pins (P1.16 to P1.31), to make port pins high LED will glow.

RESULT:

Thus the interfacing of LED with ARM processor is done successfully.

INTERFACE LCD WITH ARM PROCESSOR

Experiment No.2(b) **Date:**

AIM:

To interface LCD with ARM processor

APPARTAUS REQUIRED:

1. PC with keil µversion 5and flash magic workbench software
2. LPC2148 processor kit
3. USB Cable

THEORY:

LCD (Liquid Crystal Display)

Liquid Crystal Display also called as LCD is very helpful in providing user interface as well as for debugging purpose. A liquid crystal display (LCD) is a flat panel display that uses the light modulating properties of liquid crystals (LCs). LCD Modules can present textual information to user.

PROCEDURE:

Keil ompiler

Step 1

Open keil µversion project → menu → project → new µ version project → Select location → open new folder → give file name → save

Step 2

Menu → file → new

Step3

Type a program → save a program with .c or .h extensions

Step 4

Right corner → target → source group → right click → add existing files to group "source group1" → add "c" and "h" files

working) disable → linker (use memory layout from target dialog) enable → OK

Step 6

Build a program → check errors and warnings

Flash magic

Open Flash Magic s/w

Step 1: Select a device $\rightarrow$ LPC2148

Select comport $\rightarrow$ USB comport

Baud rate $\rightarrow$ 9600

Oscillator freq $\rightarrow$ 12MHZ

Step 2: Erase blocks used by Hex files (enable)

Step 3:Browse.hex file

Step 4: Select verifying after programming

Step5: Click start

PROGRAM CODE:

lcd.h

```
void clrscr(char ch);
void lcdinit(void);
void lcdcmd(char);
void lcddat(char);
void gotoxy(char,char); //x,y ; x-char position(0 - 16) y-line number 0 or
1 void printstr(char *,char,char); //string,column(x),line(y) void wait
(void);
void split_numbers(unsigned int number);
void Wait_Msg(void);
void Welcome_Msg(void);
```

lcd.c

```
#include <LPC214x.h>
#define RS 0x00000400     /* P0.10 */
#define CE 0x00001800     /* P1.11 */
void clrscr(char ch);
void lcdinit(void);
void lcdcmd(char);
void lcddat(char);
void gotoxy(char,char); //x,y ; x-char position(0 - 16) y-line number 0 or 1
```

```
voidprintstr(char*,char,char);      //string,column(x),line(y)
void wait(void);
void split_numbers(unsigned int number);
#define SET1
#define OFF0
unsigned int thousands,hundreds,tens,ones;
void wait (void){                          /* wait function*/
  int d;
  for (d = 0; d <100000;d++);              /* only to delay for LED flashes*/
}
void lcdinit()
{
    IODIR0 |= 0x0000FFFF;
            IOCLR0 |= 0X00000FFF;
            lcdcmd(0x28);
    lcdcmd(0x28);
    lcdcmd(0x0c);
    lcdcmd(0x06);
    lcdcmd(0x01);
    lcdcmd(0x0f);
    wait();
}
void gotoxy(char x, char y)
{
    if(y == 0)
        lcdcmd(0x80+x);
    else
        lcdcmd(0xc0+x);
}
void printstr(char *str, char x, char y)
{
   char i;
    gotoxy(x,y);
    wait();//(500);
    for(i=0;str[i]!='\0';i++)
        lcddat(str[i]);
}
void lcdcmd(char cmd)
{
unsigned char LCDDAT;
            LCDDAT = (cmd&0xf0);        //highernibble
            IOSET0 = LCDDAT;
            IOCLR0 = RS;
            IOSET0 =CE;
     wait();//(100);                         //enable lcd
            IOCLR0 =CE;
            IOCLR0 = 0X00000FFF;
     LCDDAT = ((cmd<<0x04) & 0xf0);           //lower nibble
            IOSET0 = LCDDAT;
     IOCLR0 = RS;
            IOSET0 = CE;
     wait();//(100);                         //enable lcd
            IOCLR0 =CE;
            IOCLR0 = 0X00000FFF;
}
void lcddat(char cmd)
{
   unsigned char LCDDAT;
            LCDDAT = (cmd&0xf0);        //highernibble
    IOSET0 = LCDDAT;
     IOSET0 = RS;
     IOSET0 = CE;
     wait();//(100);                         //enable lcd
            IOCLR0 =CE;
            IOCLR0 = 0X00000FFF;
     LCDDAT = ((cmd<<0x04)&0xf0);             //lowernibble
            IOSET0 = LCDDAT;
     IOSET0 = RS;
```

```
            IOSET0 = CE;
      wait();//(100);                         //enable lcd
            IOCLR0 =CE;
            IOCLR0 = 0X00000FFF;
 }
void clrscr(char ch)
{
   if(ch==0)
   {
      printstr("                 ",0,0);
      gotoxy(0,0);
   }
   else if(ch ==1)
   {
      printstr("                 ",0,1);
      gotoxy(0,1);
   }
   else
   {
      lcdcmd(0x01);
     // delay(100);
   }
}
void split_numbers(unsigned int number)
{
  thousands = (number /1000);
  number %= 1000;
  hundreds = (number / 100);
  number %= 100;
  tens = (number / 10);
  number %= 10;
  ones = number ;
}
void Wait_Msg(void)
```

```
{
 lcdcmd(0x01);
 printstr(" WELCOME TO ", 0, 0);
 printstr("MAHABARATHI ENGG COLLEGE", 0, 1);
}
void Welcome_Msg(void)
{
 lcdcmd(0x01);
 printstr(" ARM-7 LPC2148 ", 0, 0);
 printstr("32-Bitcontroller", 0, 1);
}
```

main_LCD_Test.c

```
#include <LPC214x.H>/* LPC214x definitions */
#include "lcd.h"     /* includes lcd driver funtions*/
int main(void)
 {
  lcdinit();      /*Initializelcd*/
  Wait_Msg();    /*Display message - "Please Wait"*/
  Welcome_Msg(); /*Display message - "Welcome to MAHABARATHI ENGG COLLEGE"*/
  while(1)                     /*LoopForever*/
    {
          }
}
```

Testing the LCD Module with LPC2148

Give +3.3V power supply to LPC2148 Primer Board; the LCD is connected with microcontroller LPC2148 Primer Board. When the program is downloading into LPC2148 in Primer Board, the screen should show the text messages.

OUTPUT:

```
"WELCOMETO"

"MAHA BARATHI ENGG COLLEGE
```

RESULT:

Thus the interfacing LCD display with ARM processor is done and text is displayed in LCD

INTERFACING OF KEYBOARD WITH ARM PROCESSOR

Experiment No.3(a) **Date:**

AIM:

To interface Keyboard with ARM processor.

APPARATUS REQUIRED :

1. PC with keil µversion 5and flash magic workbench software
2. LPC2148 processor kit
3. USB Cable

KEYPAD:

A **keypad** is a set of buttons arranged in a block or "pad" which usually bear digits, symbols and usually a complete set of alphabetical letters. If it mostly contains numbers then it can also be called a **numeric keypad**. Keypads are found on manyalphanumeric keyboardsandon other devices such as calculators, push-button telephones, combination locks, and digital door locks, which require mainly numeric input. Here LPC2148 using 4 X 4 matrix keypad.

PROCEDURE:

Keil ompiler

Step 1

Open keil µversion project → menu → project → new µ version project → Select location → open new folder → give file name → save

Step 2

Menu → file → new

Step3

Type a program → save a program with .c or .h extensions

Step 4

Right corner → target → source group → right click → add existing files to group "source group1" → add "c" and "h" files

working) disable → linker (use memory layout from target dialog) enable → OK

Step 6

Build a program → check errors and warnings

Flash magic

Open Flash Magic s/w

Step 1: Select a device $\rightarrow$ LPC2148

Select comport $\rightarrow$ USB comport

Baud rate $\rightarrow$ 9600

Oscillator freq $\rightarrow$ 12MHZ

Step 2: Erase blocks used by Hex files (enable)

Step 3:Browse.hex file

Step 4: Select verifying after programming

Step5: Click start

PROGRAM CODE:

defs.h

```
/***********local defenitions ************************************/
  #define ROW1 0x00010000;
  #define ROW2 0x00020000;
  #define ROW3 0x00040000;
  #define ROW4 0x00080000;
  #define SW1 0x00000001;
  #define SW2 0x00000002;
  #define SW3 0x00000004;
  #define SW4 0x00000008;
  #define ERR0x00000000;
  #define ROW_MASK 0x000F0000;
  #define S7SEG_ENB 0x00B80000;
  #define DIGI1_ENB 0x00080000;
  #define DIGI2_ENB 0x00100000;
  #define DIGI3_ENB 0x00200000;
  #define DIGI4_ENB 0x00800000;
  #define S7SEG_LED0xff000000;
  #define ZERO 0x3F000000;
  #define ONE 0x06000000;
  #define TWO 0x5B000000;
  #define THREE0x4F000000;
```

```
   #define FOUR 0x66000000;
   #define FIVE 0x6D000000;
   #define SIX 0x7D000000;
   #define SEVEN 0x07000000;
   #define EIGHT 0x7F000000;
   #define NINE 0x6F000000;
   #define AAA 0x77000000;
   #define bbb 0x7C000000;
   #define ccc 0x39000000;
   #define ddd 0x5E000000;
   #define eee 0x79000000;
   #define fff0x71000000;
```

lcd.h

```
void clrscr(char ch);
void lcdinit(void);
void lcdcmd(char);
void lcddat(char);
void gotoxy(char,char); //x,y ; x-char position(0 - 16) y-line number 0 or
1 void printstr(char *,char,char); //string,column(x),line(y) void wait
(void);
void split_numbers(unsigned int number);
void Wait_Msg(void);
void Welcome_Msg(void);
```

mat_7seg.h

```
/* Function prototypes */
          void init_Matrix_7seg(void);
          unsigned long scan_row(unsigned int);
          unsigned int catch_key(void);
    void clearall_7seg(void);
          void clearDigit_7seg(int);
    void Digit_Dispay(int, unsigned int);
          void split_numbers(unsigned int);
```

```
     void Display_Number(unsigned int);
     void Alpha_Dispay(int, unsigned int);
```

main.c

```
#include <LPC214x.h>
#include "mat_7seg.h"
#include "lcd.h"
int main()
{
 unsigned int key, last_key, Disp_key;
 init_Matrix_7seg(); // Initialize matrix keyboard and 7segment dispaly
 clearall_7seg();           // clear 7 segment display
 last_key=0;                // Initialize this variable
 tozero while(1)
    {
              key =                     // scan for a valid key press
              catch_key();                        // zero means no key is pressed
              if(key != 0)
              {
               if(key != last_key) // check whether the same key is pressed again(assume this as
STEP1)
                {
                        Disp_key = key; // valid new key is stored in another variable
                last_key = key; // this variable's value is used for STEP1 }


              }
             Alpha_Dispay(4,Disp_key         /*this function is used to display number in hex format
             );
(single digit only)*/
             }
}
```

matrix_7seg_driver.c#inclu

```
de <LPC214x.h> #include
"defs.h"
void init_Matrix_7seg(void)
```

```
IODIR1 |= 0xff0f0000; // set 7seg LEDs as output ports and matrix's MSB as inputs and LSB
as outputs
IODIR0 |=                    // set P0.19 to P0.22 as outputs to drive 7seg enable pins
S7SEG_ENB;                    // since we are using active low 7 seg display, the enable
IOPIN0 |= S7SEG_ENB;            // should be initially set to HIGH.
signals
 }
   unsigned long scan_row(unsigned int row_num)
 {
  //unsigned int row,i;
 unsigned long val;
 IOSET1= ROW_MASK;          //clear the previous scan row output ie make all row ops high
switch(row_num)
 {
case 1: IOCLR1 = ROW1;break; // make P1.16 low
case 2: IOCLR1 = ROW2;break; // make P1.17 low
case 3: IOCLR1 = ROW3;break; // make P1.18 low
case 4: IOCLR1 = ROW4;break; // make P1.19 low
//default: row = ERR;
  }
//          for(i=0;i<=65000;i++);
            val=IOPIN1;          // read the matrixinputs
                 val = ((val >> 20) & 0x0000000F)^0x0000000F; // shift the colum value so that
                                                    it comes to LSB
                              // XORing is done to take 1's complement of shifted value.
 return(val);
}
unsigned int catch_key(void)
{
unsigned long v;
v = scan_row(1);
switch(v)
{
case 1:return(13);
case 2:return(14);
```

```
 case 4: return(15);
case 8: return(16);
}
v = scan_row(2);
switch(v)
{
case 1: return(9);
 case 2:return(10);
 case 4:return(11);
case 8: return(12);
 }
 v = scan_row(3);
 switch(v)
{
case 1: return(5);
 case 2: return(6);
case 4:return(7);
case 8:return(8);
 }
v = scan_row(4);
switch(v)
{
case 1:return(1);
case 2:return(2);
case 4:return(3);
case 8: return(4);
default: return(0);
}
}
void clearall_7seg(void)
 {
 IOPIN1 &= ~S7SEG_LED; // make all the 7seg led pins to LOW
IOPIN0 |=S7SEG_ENB           // Disable all the 7 segdisplay
                                                                  }
```

```
   void clearDigit_7seg(int digit_num)
 {
 IOPIN0 |= S7SEG_ENB; // clear enables first
 switch(digit_num)
{
case 1: {
 IOPIN0 =~DIGI1_ENB;        // now enable only thedigit1
break;
 }
case 2: {
IOPIN0 =~DIGI2_ENB;         // now enable only thedigit2
 break;
}
 case3:
{
  IOPIN0 =~DIGI3_ENB;       // now enable only thedigit3
 break;
 }
case4
 {
IOPIN0 =~DIGI4_ENB;         // now enable only thedigit4
break;
 }
  }
IOPIN1 &= ~S7SEG_LED; // make all the 7seg LED pins LOW
}
   void Digit_Dispay(int digit_num, unsigned int value)
 {
clearDigit_7seg(digit_num);
switch(value)
{
case 0: IOPIN1 |= ZERO;break;
case 1: IOPIN1 |= ONE; break;
case 2: IOPIN1 |= TWO; break;
case 3: IOPIN1 |= THREE;break;
case 4: IOPIN1 |= FOUR; break;
case 5: IOPIN1 |= FIVE; break;
case 6: IOPIN1 |= SIX; break;
case 7: IOPIN1 |= SEVEN; break;
case 8: IOPIN1 |= EIGHT; break;
case 9: IOPIN1 |= NINE;break;
}
 }
   void Alpha_Dispay(int digit_num, unsigned int value)
     {
            clearDigit_7seg(digit_num);
            switch(value)
             {
                    case 1: IOPIN1 |= ZERO;break;
                    case 2: IOPIN1 |= ONE; break;
                    case 3: IOPIN1 |= TWO; break;
                    case 4: IOPIN1 |= THREE;break;
                    case 5: IOPIN1 |= FOUR; break;
                    case 6: IOPIN1 |= FIVE; break;
                    case 7: IOPIN1 |= SIX; break;
                    case 8: IOPIN1 |= SEVEN; break;
                    case 9: IOPIN1 |= EIGHT; break;
                    case 10: IOPIN1 |= NINE; break;
                    case 11: IOPIN1 |= AAA; break;
                    case 12: IOPIN1 |= bbb; break;
                    case 13: IOPIN1 |= ccc; break;
                    case 14: IOPIN1 |= ddd; break;
                    case 15: IOPIN1 |= eee; break;
                    case 16: IOPIN1 |= fff;break;
                   }
          }
   void split_numbers(unsigned int number)
    {
     thousands = (number /1000);
     number %= 1000;
     hundreds = (number / 100);
     number %= 100;
     tens = (number / 10);
     number %= 10;
     ones = number ;
    }
/***************************************************************************/
   void Display_Number(unsigned int num)
    {
            unsigned int i;
            if(num <= 9999)
            {
             clearall_7seg();
             split_numbers((unsignedint)num);
             Digit_Dispay(4, ones);
             for(i=0;i<10000;i++);
             Digit_Dispay(3, tens);
             for(i=0;i<10000;i++);
             Digit_Dispay(2,hundreds);
             for(i=0;i<10000;i++);
             Digit_Dispay(1, thousands);
             for(i=0;i<10000;i++);
            }
           }
```

<u>lcd.c</u>

```
#include <LPC214x.h>
#define  RS  0x00000400  /*  P0.10  */
#define  CE  0x00001800  /*  P1.11  */
void clrscr(char ch);
void   lcdinit(void);
void lcdcmd(char);
void lcddat(char);
void gotoxy(char,char); //x,y ; x-char position(0 - 16) y-line number 0 or 1
voidprintstr(char*,char,char);      //string,column(x),line(y)
void wait (void);
void split_numbers(unsigned int number);
#define SET1
#define OFF0
unsigned int thousands,hundreds,tens,ones;
void wait (void){                        /* wait function*/
int d;
for (d = 0; d <100000;d++);            /* only to delay for LED flashes */
}
void lcdinit()
{
 IODIR0 |= 0x0000FFFF;
IOCLR0 |= 0X00000FFF;
lcdcmd(0x28);
lcdcmd(0x28);
lcdcmd(0x0c);
lcdcmd(0x06);
lcdcmd(0x01);
lcdcmd(0x0f);
wait();
}
void gotoxy(char x, char y)
{
 if(y == 0)
lcdcmd(0x80+x);
else
lcdcmd(0xc0+x);
}
void printstr(char *str, char x, char y)
{
char i;
```

```
gotoxy(x,y);
wait();//(500);
for(i=0;str[i]!='\0';i++)
lcddat(str[i]);
}
void lcdcmd(char cmd)
{
unsigned char LCDDAT;
LCDDAT = (cmd&0xf0);         //highernibble
 IOSET0 = LCDDAT;
IOCLR0 = RS;
IOSET0 =CE;
wait();//(100);                         //enable lcd
IOCLR0 =CE;
IOCLR0 = 0X00000FFF;
LCDDAT = ((cmd<<0x04)&0xf0);             //lowernibble
IOSET0 = LCDDAT;
IOCLR0 = RS;
IOSET0 =CE;
wait();//(100);                         //enable lcd
IOCLR0 =CE;
 IOCLR0 = 0X00000FFF;
}
void lcddat(char cmd)
{
 unsigned char LCDDAT;
 LCDDAT = (cmd&0xf0);         //highernibble
 IOSET0 = LCDDAT;
IOSET0 = RS;
IOSET0 =CE;
wait();//(100);                         //enable lcd
IOCLR0 =CE;
 IOCLR0 = 0X00000FFF;
LCDDAT = ((cmd<<0x04)&0xf0);             //lowernibble
```

```
 IOSET0 = LCDDAT;
IOSET0 = RS;
IOSET0 =CE;
 wait();//(100);                          //enable lcd
IOCLR0 =CE;
IOCLR0 = 0X00000FFF;
 }
void clrscr(char ch)
{ if(ch==0)
 {
 printstr("",0,0);
gotoxy(0,0);
}
 else if(ch == 1)
 {
printstr("                  ",0,1);
  gotoxy(0,1);
}
else
{
lcdcmd(0x01);
  //delay(100);
 }
}
void split_numbers(unsigned int number)
{
thousands = (number /1000);
 number %= 1000;
  hundreds = (number / 100);
 number %= 100;
tens = (number / 10);
 number %=10;
ones = number;
}
```

```
void Wait_Msg(void)
{
 lcdcmd(0x01);
 printstr("PLEASEWAIT      ", 0, 0);
}
void Welcome_Msg(void)
{
 lcdcmd(0x01);
 printstr(" WELCOME TO ", 0, 0);
 printstr("SM MICRRO SYSTEM", 0, 1);
}
```

Testing with Keyboard:

Each press of a key corresponding character is displayed on the Seven Segment Display.

Output:

Sw1 –

RESULT:

Thus the keyboard interfacing with ARM processor is done and pressed key is verified successfully.

INTERFACING OF STEPPER MOTOR USING ARMPROCESSOR

Experiment No.3(b) **Date:**

AIM:

To interface stepper motor with ARM processor.

APPARTAUS REQUIRED:

1. PC with keil µversion 5and flash magic workbench software
2. LPC2148 processor kit
3. USB Cable
4. Stepper motor with driver module

THEORY:

A stepper motor is a brushless, synchronous electric motor that converts digital pulses into mechanical shaft rotation. Every revolution of the stepper motor is divided into a discrete number of steps, and the motor must be sent a separate pulse for each step.

PROCEDURE:

Keil ompiler

Step 1

Open keil µversion project → menu → project → new µ version project → Select location → open new folder → give file name → save

Step 2

Menu → file → new

Step3

Type a program → save a program with .c or .h extensions

Step 4

Right corner → target → source group → right click → add existing files to group "source group1" → add "c" and "h" files

working) disable → linker (use memory layout from target dialog) enable → OK

Step 6

Build a program → check errors and warnings

Flash magic

Open Flash Magic s/w

Step 1: Select a device → LPC2148

Select comport → USB comport

Baud rate → 9600

Oscillator freq → 12MHZ

Step 2: Erase blocks used by Hex files (enable)

Step 3:Browse.hex file

Step 4: Select verifying after programming

Step5: Click start

PROGRAM CODE:

stepper.c

```
#include<LPC214x.H>      /* LPC214x definitions*/
#define  step1  0x00010000  /*  P1.16  */
#define  step2  0x00020000  /*  P1.17  */
void wait (void)
{                          /*waitfunction*/
  intd;
  for (d = 0; d <10000;d++);             /* only to delay for LED flashes*/
}
void call_stepper_forw()
{
 IOCLR1 = 0X00FF0000;
   IOSET1 = 0X00040000;
 wait();
 wait();
  IOCLR1 = 0X00FF0000;
   IOSET1 = 0X00060000;
 wait();
 wait();
  IOCLR1 = 0X00FF0000;
   IOSET1 = 0X00070000;
 wait();
 wait();
IOCLR1 = 0X00FF0000;
 IOSET1 = 0X00050000;
 wait();
 wait();
}
/*voidcall_reverse(void)
{
 IOCLR1 = 0X00FF0000;
 IOSET1 = 0X00050000;
 wait();
 wait();
IOCLR1 = 0X00FF0000;
 IOSET1 = 0X00070000;
 wait();
IOCLR1 = 0X00FF0000;
 IOSET1 = 0X00060000;
 wait();
IOCLR1 = 0X00FF0000;
 IOSET1 = 0X00040000;
 wait();
} */
int main (void)
 {
  IODIR1 |= 0xFFFFFFFF;
IOCLR1 |= 0X00FF0000;
wait();
  while(1)                    /*LoopForever*/
    {
            call_stepper_forw();
    //call_reverse();
            wait();
            IOCLR1 = 0X00FF0000;
          }
 }
```

Testing the Stepper Motor with LPC2148:

Give +3.3V power supply to LPC2148 Primer Board; the Stepper Motor is connected with LPC2148 Primer Board. When the program is downloading into LPC2148 in Primer Board, the LED output is working that the LED is ON some time period and the LED is OFF some other time period for a particular frequency. Now, the stepper motor is rotating.

Output:

The stepper motor was rotated in clockwise direction.

RESULT:

Thus the interfacing of stepper motor with ARM processor is done successfully.

Mini projects for IOT:

1. Garbage Segregator and Bin Level Indicator
2. Colour based Product Sorting
3. Image Processing based Fire Detection
4. Vehicle Number Plate Detection
5. Smart Lock System

Objective:

To apply the knowledge, they gained in doing the experiments.

Team constitution:

A team size may be from 3 to 4 students.

Guidelines:

1. Students shall from a group and can do their mini project.

2. Student must buy their own hardware setup for doing mini projects.

3. If they are utilizing the college resource, they should get approval from HoD.

4. At the end, a report along with hardware must be submitted to college.

5. If required students need to present their work as presentation.

1. Garbage Segregator and Bin Level Indicator

IDEA:

With progress in human technology we have seen a substantial progress in the amount of waste generated. Recycling is the only way to manage this huge amount of waste. But recycling requires garbage to be segregated. Without segregation garbage cannot be recycled because different type of garbage requires different recycling processes.

Also it is important to educate users and instruct them every time they come near the dustbin about instructions about throwing the trash. For this purpose we design a garbage disposal system that uses multiple dustbins with a voice based system that speaks to the user each time he she stands before the dustbin.

The system makes use of a camera to detect presence if any person in front of the dustbin. If a person is detected, the system issues voice instructions to the user about throwing right garbage inthe right bin. In case the dustbin is full it instructs the user to find another dustbin to throw garbage in.

To develop this system we make use of a raspberry Pi controller. The controller is interfaced witha camera and a voice speaker for detection and communication. The controller gets dustbin level input using ultrasonic level sensors each having LED indicators interfaced to it. The level sensors are used to constantly feed the raspberry pi with bin levels.

The Arduino Uno microcontroller is also interfaced with a Wifi module to transmit the level data over the internet. The Level sensor panels are made to be easily mounted over any dustbin. This allows the system to be easily screwed over any dustbin for instant installation.

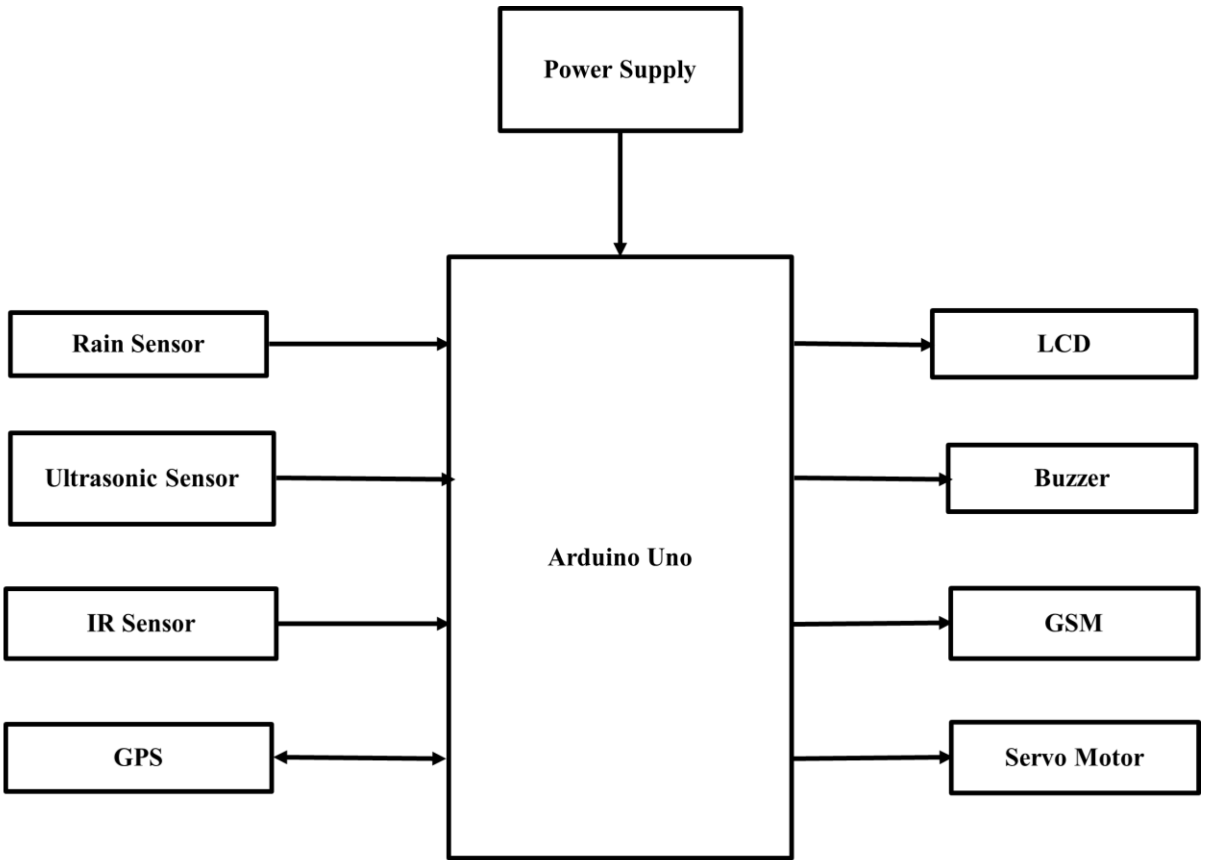
The data is transmitted over IOT to IOT gecko platform which displays the bin level data over internet. This indication can be used to alert the authorities that the garbage bins need to be emptied. Thus the system automates garbage segregation and level monitoring to help counter the garbage crisis using IOT.

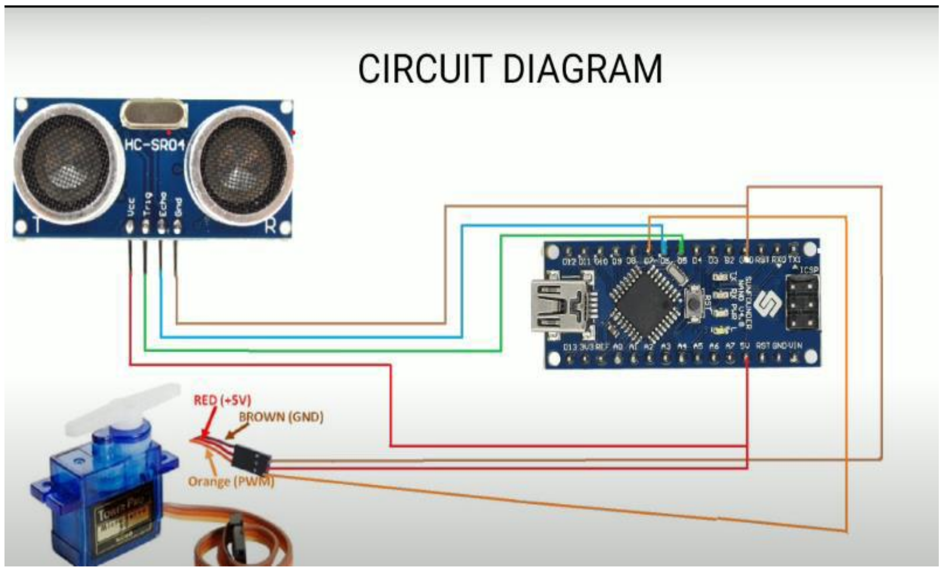
Components

Arduino Uno microcontroller, Ultrasonic level sensors,

Ultrasonic Level Sensors,LED Indicators, IR Sensors, LCD,

Buzzer, GSM, GPS, Servo motor and IR sensors


Power Supply
Rain Sensor
Ultrasonic Sensor
IR Sensor
GPS
Arduino Uno
LCD
Buzzer
GSM
Servo Motor

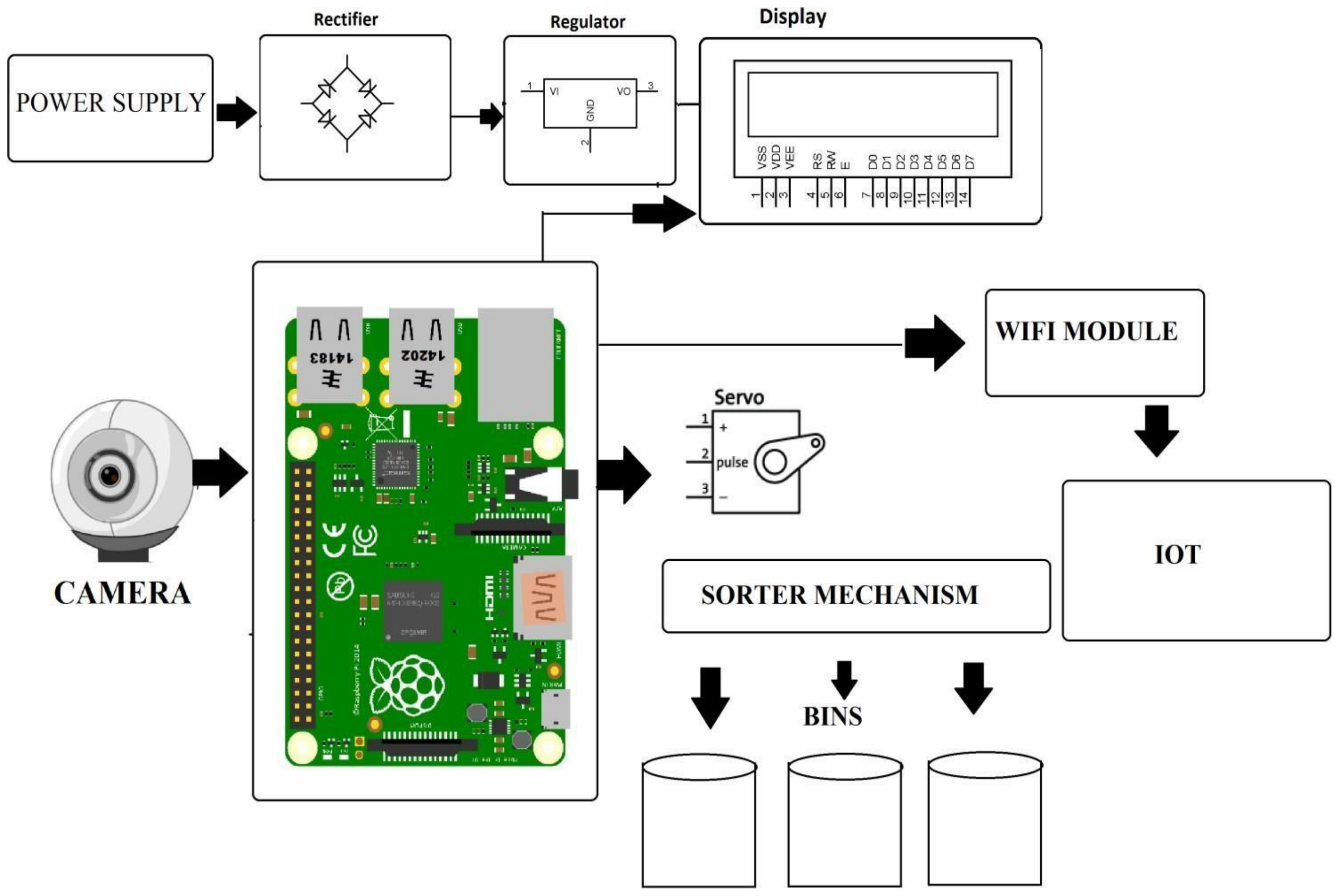

CIRCUIT DIAGRAM
HC-SR04
Vcc
Trig
Echo
Gnd
RED (+5V)
BROWN (GND)
Orange (PWM)

2. Colour based Product Sorting

IDEA:

The proposed methodology is designed to efficiently sort items based on colour using machine learning with the help of Raspberry Pi, camera modules, sensors, servo motors, and the IBM Watson visual recognition model. The methodology includes several crucial steps to achieve accurate sorting:

Firstly, a comprehensive colour database is compiled, containing various product images captured under various lighting conditions and angles. These images serve as the foundation for training the machine learning model. Next, the learning model is constructed using Python and libraries for implementation. The model undergoes preprocessing to enhance image quality and reduce noise, ensuring optimal performance during classification. Finally, the robotic arm selectively picks up and sorts the products into designated bins based on their predicted colours. This automated sorting process ensures efficient and accurate handling of large volumes of diverse products. In summary, the proposed methodology combines machine learning, image processing, and robotic automation to create a robust system for colour-based product sorting. This approach offers a scalable solution for various industrial and commercial applications by leveraging Python programming and advanced technologies.



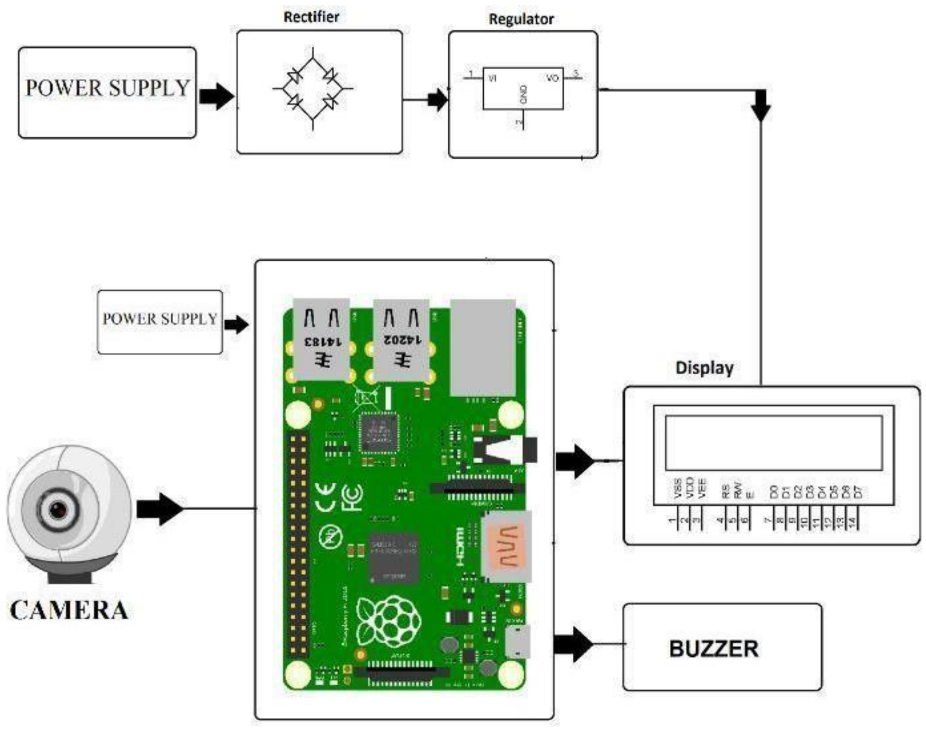
3. Image Processing based Fire Detection

IDEA:

The main advantage of Image Processing Based Fire Detection System is the early warning benefit. This system can be installed just about anywhere in a commercial building, malls and at many more public places for fire detection. This system uses camera for detecting fires. So we do not need any other sensors to detect fire. System processes the camera input and then processor processes it to detect fires. The heat signatures and fire illumination patterns are detected in images to determine if it is a fire and take action accordingly. On detecting fire system goes into emergency mode and sounds an alarm. Also displays the status on the LCD display informing about the system.

Hardware Specifications

BLOCK DIAGRAM:



4. Vehicle Number Plate Detection

IDEA:

OBJECTIVE

Project Code :TEMBMA3101

The main objective of this project is to design an efficient automatic authorized vehicle identification system by using the vehicle number plate.

ABSTRACT

The basic idea of this project is to build a **number plate recognition system using python.** Real-Time license plate detection and recognition can be very useful for automating toll booths, finding out traffic rule breakers, and for addressing other vehicle-related security and safety issues.

The system uses a camera interfaced to a Raspberry Pi. The system constantly processes incoming camera footage to detect any trace of number plates. On sensing a number plate in front of the camera, it processes the camera input, extracts the number plate part from the image. Processes the extracted image using OCR and extracts the number plate number from it. The system then displays the extracted number on the monitor.

Keywords: Raspberry Pi, Ultrasonic sensor, web camera, power supply

SPECIFICATIONS

Hardware components:

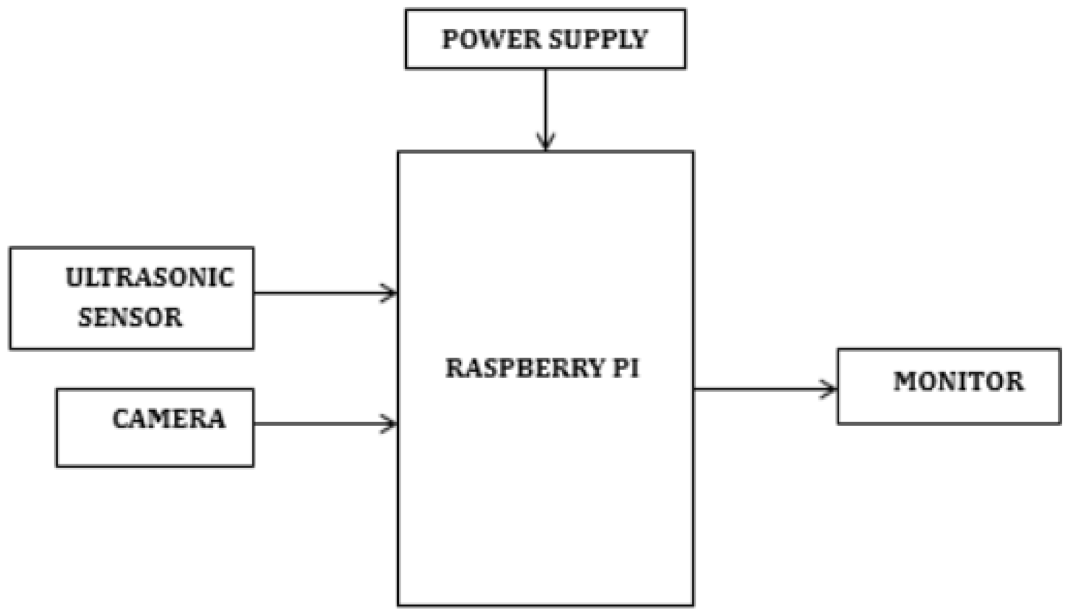
- Raspberry pi
- Memory Card
- Ultrasonic sensor
- Web camera
- Monitor
- 5V 2A Adapter

Software requirements:

- Python3 IDLE
- NOOBS OS
- VNC Viewer
- Advanced IP Scanner

BLOCK DIAGRAM

Project Code :TEMBMA3101



5. SMART LOCK SYSTEM

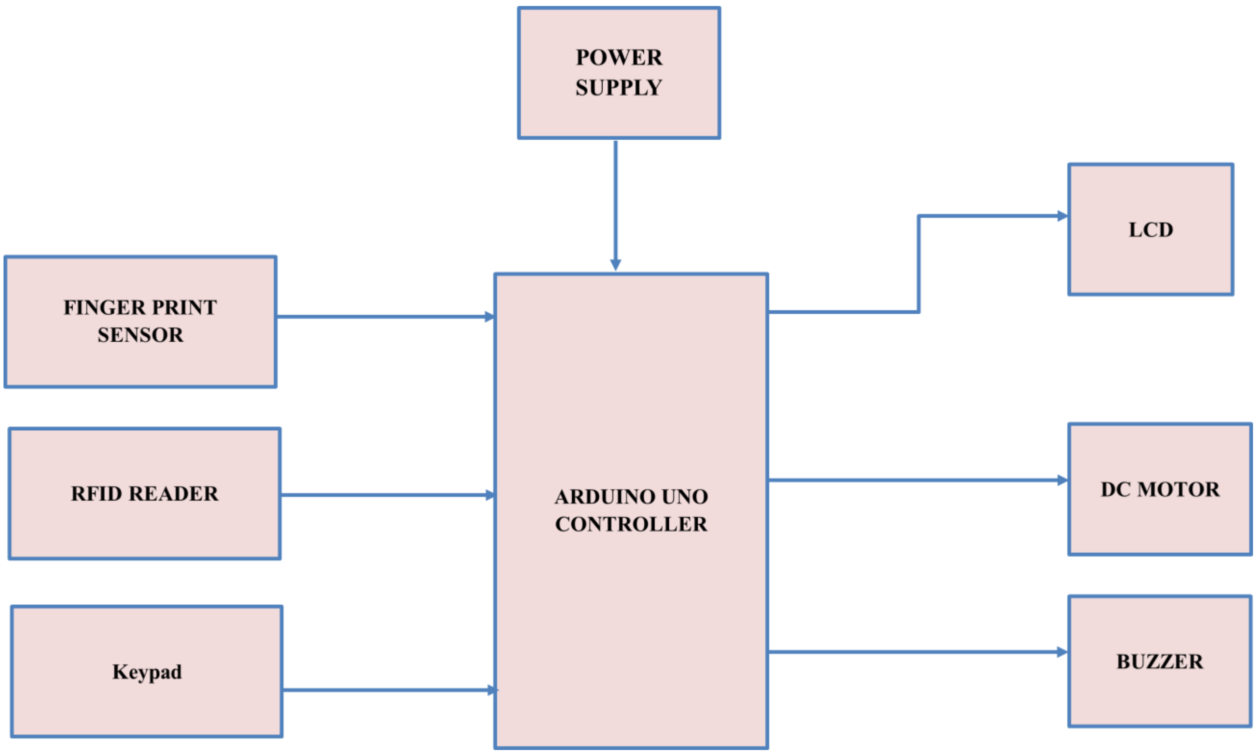
IDEA:

The Smart Lock System with Face Recognition is a revolutionary innovation in home and office security, combining advanced technologies like face recognition, keypad entry, and fingerprint scanning. This innovative solution redefines access control, offering a seamless and user-friendly experience, replacing traditional lock and key mechanisms. It goes beyond traditional methods, providing a safer and more convenient solution for occupants.

The smart lock uses advanced face recognition technology, utilizing advanced algorithms and artificial intelligence, to accurately identify and authenticate individuals based on facial features. This technology eliminates the need for physical keys or cards, providing a secure, touch less access. The system can even distinguish between live faces and photographs, ensuring robust security.

The keypad entry system complements the face recognition feature, providing users with a unique PIN (Personal Identification Number) code for access. This adds security and serves as a backup in case of low lighting or temporary obstructions, ensuring occupants can always gain access, even in challenging situations.

The three-layer security system incorporates fingerprint scanning technology, a unique biometric identifier for each individual. This method adds personalization and security, providing quick and convenient access for authorized users. Fingerprint recognition is highly accurate and highly accurate, making it an ideal biometric identifier.


POWER SUPPLY
FINGER PRINT SENSOR
RFID READER
Keypad
ARDUINO UNO CONTROLLER
LCD
DC MOTOR
BUZZER