ECA14 – Embedded System

List of Experiments

S. No	Name of the Experiment	Remarks
1	Study Of Keil And Proteus micro Vision	
2	Blinking Of Led Using 8051 Microcontroller Using Keil And Proteus	
3	Generation Of Square Wave Using Keil And Proteus	
4	Fade In Fade Out Of Led Using 8051 Using Keil And Proteus	
5	Stepper Motor Using 8051 Using Keil And Proteus	
6	Interfacing Of Relay Using 8051 Using Keil And Proteus	
7	Led Toggle Using 8051 Using Keil And Proteus	
8	7 Segment Display Using 8051 Using Keil And Proteus	
9	Led Chaser Using 8051 Using Keil And Proteus	
10	DC MOTOR INTERFACING WITH 8051 MICROCONTROLLER USING KEIL	
11	Study of ARM Processor	
12	Write and execute C program to blink LEDs using software delay routine in LPC2148 kit	
13	Write and execute C program to read the switch and display in the LEDs using LPC2148 kit	
14	Write and execute C program to display a number in seven segment LED in LPC2148 kit	
15	Write and execute C program for serial transmission and reception using on-chip UART in LPC2148 kit.	
16	Write and execute C program for accessing an internal ADC and display the binary output in LEDS in LPC2148 kit.	

17	**Study of Arduino**	
18	**BLINKING OF AN LED WITH AN ARDUINO USING PROTEUS**	
19	**FADING OF AN LED WITH AN ARDUINO USING PROTEUS**	
20	**TURNING AN LED ON AND OFF USING A PUSHBUTTON WITH AN ARDUINO USING PROTEUS**	
21	**INTERFACING A WATER-LEVEL SENSOR WITH AN ARDUINO USING PROTEUS**	
22	**INTERFACING AN ULTRASONIC SENSOR WITH AN ARDUINO USING PROTEUS**	
23	**INTERFACE BUZZER (OR PIEZO SPEAKER) WITH ARDUINO USING PROTEUS**	
24	**MQ-6 GAS SENSOR INTERFACING WITH ARDUINO USING PROTEUS**	
25	**BLINKING OF AN LED USING ARDUINO TRAINER KIT**	
26	**INTERFACING A WATER-LEVEL SENSOR WITH AN ARDUINO TRAINER KIT**	
27	**MQ-6 GAS SENSOR INTERFACING WITH ARDUINO TRAINER KIT**	

Experiment no -01

Study of Keil and ProteusMicro Vision

Keil MicroVision is a free software which solves many of the main points for an embedded program developer. This software is an integrated development environment (IDE), which integrated a text editor to write programs, a compiler and it will convert your source code to hex files too. µVision4 introduces a flexible window management system, enabling us to drag and drop individual windows anywhere on the visual surface including support for Multiple Monitors.

KEIL PROCEDURE:

1. Open the software, Click on project and open new version project.
2. Create a new project file
3. Enter AT89C51
4. Click NO
5. Click [Ctrl +N] and Type the code
6. Open project and click Build target
7. Open Build target and open source file and ADD, CLOSE
8. Click build target
9. Next debug start and stop
10. Open peripherals and select port 2
11. Now run the program in Debug
12. Open project and click optional properties and in that give output as hex file.
13. Create hex file.

PROTEUS PROCEDURE:

- Open proteus by clicking run as administrator.
- Open new project and enter the file name.
- Click next, next, next and finish.
- Click P symbol and search keyword and place the required components
- Now connect the components as required
- Give input to AT89C51 as HEX file.
- Start the simulation process

Result: Thus the studying of keil and proteus working and functionality is successfully completed.

Experiment 02

BLINKING OF LED USING 8051 MICROCONTROLLER USING KEIL AND PROTEUS

AIM:

To Write an assembly language program to LED blink using 8051

SOFTWARES REQUIRED:

- Keil software

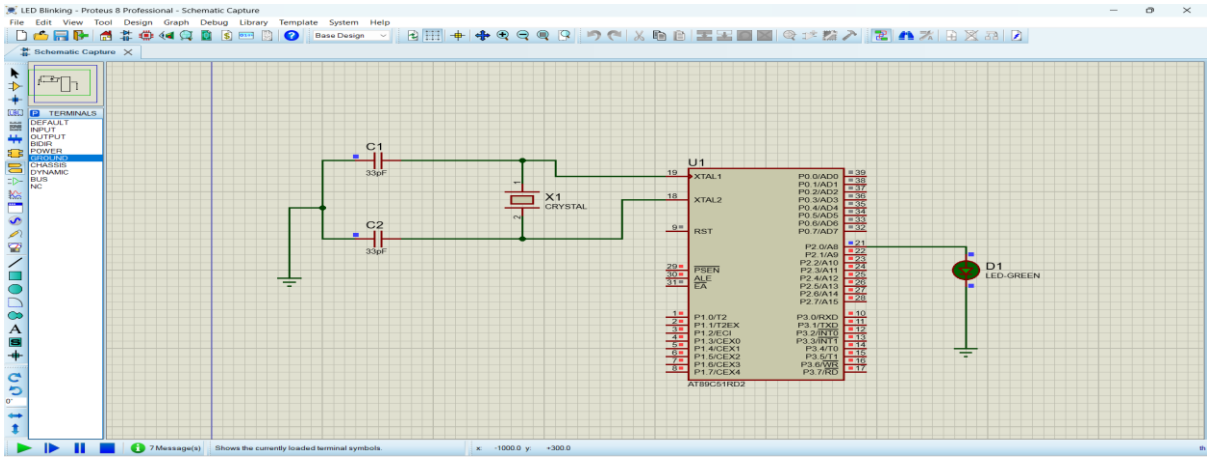
PROCEDURE:

- Open the Proteus software and create new project of 8051 microcontroller.
- Pick the required components and connect the circuit based on the circuit diagram.
- Open the keil software and create a hex code
- Copy the the file path and paste into the microcontroller of proteus software.
- Run the simulation in the proteus software and observer the output.

PROGRAM

```
        ORG 0000H
UP: SETB P2.0
        ACALL DELAY
        CLR P2.0
        ACALL DELAY
        SJMP UP
DELAY: MOV R4,#35
H1:MOV R3,#255
H2:DJNZ R3,H2
        DJNZ R4,H1
        RET
        END
```

CIRCUIT DIAGRAM:



RESULT

Thus the program has been successfully verified and executed.

Experiment 3

GENERATION OF SQUARE WAVE USING KEIL AND PROTEUS

AIM:

Write an assembly language program to Generate square wave using 8051.

SOFTWARE REQUIRED:

- Keil software.
- Proteus 8 software.

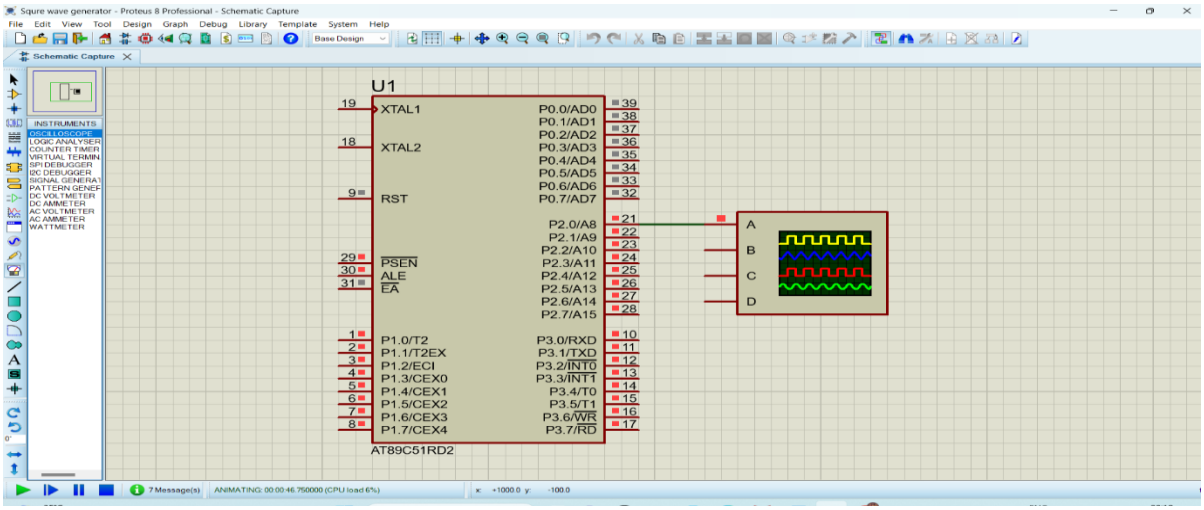
PROCEDURE:

- Open the Proteus software and create new project of 8051 microcontroller.
- Pick the required components and connect the circuit based on the circuit diagram.
- Open the keil software and create a hex code
- Copy the the file path and paste into the microcontroller of proteus software.
- Run the simulation in the proteus software and observer the output.

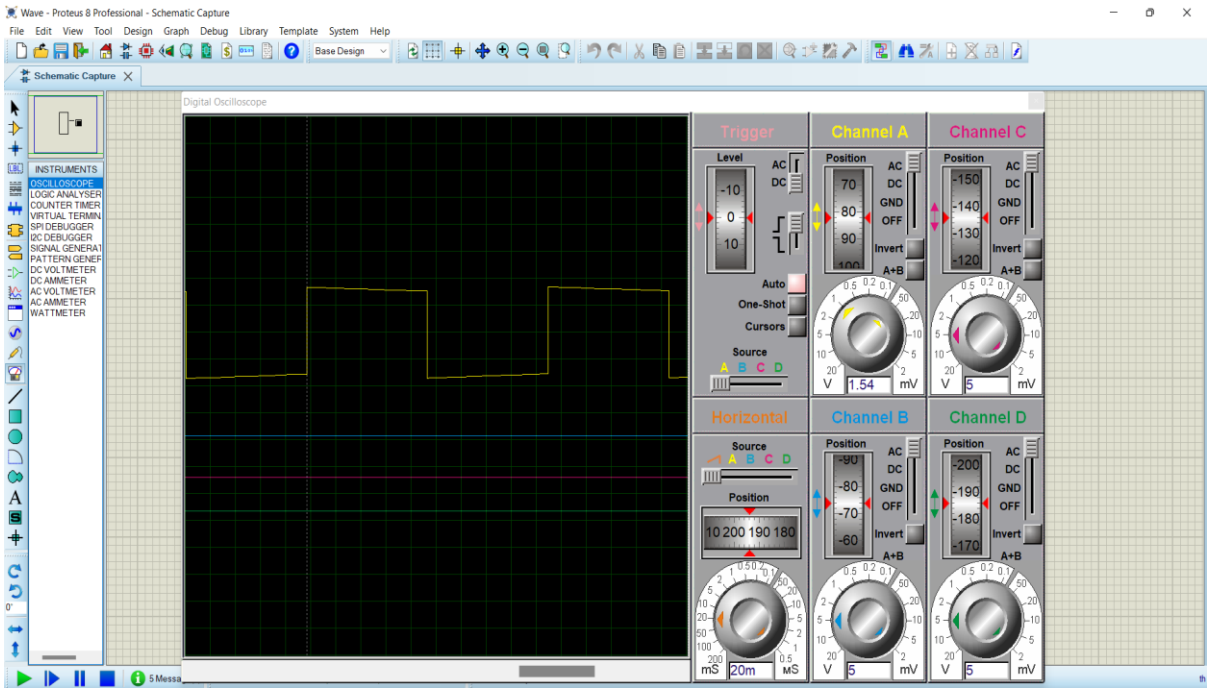
PROGRAM

```
        ORG 0000H
    UP: SETB P2.0
        ACALL DELAY
        CLR P2.0
        ACALL DELAY
        SJMP UP
DELAY: MOV R4,#35
    H1:MOV R3,#255
    H2:DJNZ R3,H2
       DJNZ R4,H1
       RET
       END
```

CIRCUIT DIAGRAM:



OUTPUT:



RESULT:

Thus the program has been successfully verified and executed.

Experiment 4

FADE IN FADE OUT OF LED USING 8051 USING KEIL AND PROTEUS

AIM:

Write an assembly language program forFade in Fade out ofLED Using 8051 using Keil and Proteus

SOFTWARE REQUIRED:

- Keil software 5.
- Proteus 8 software.

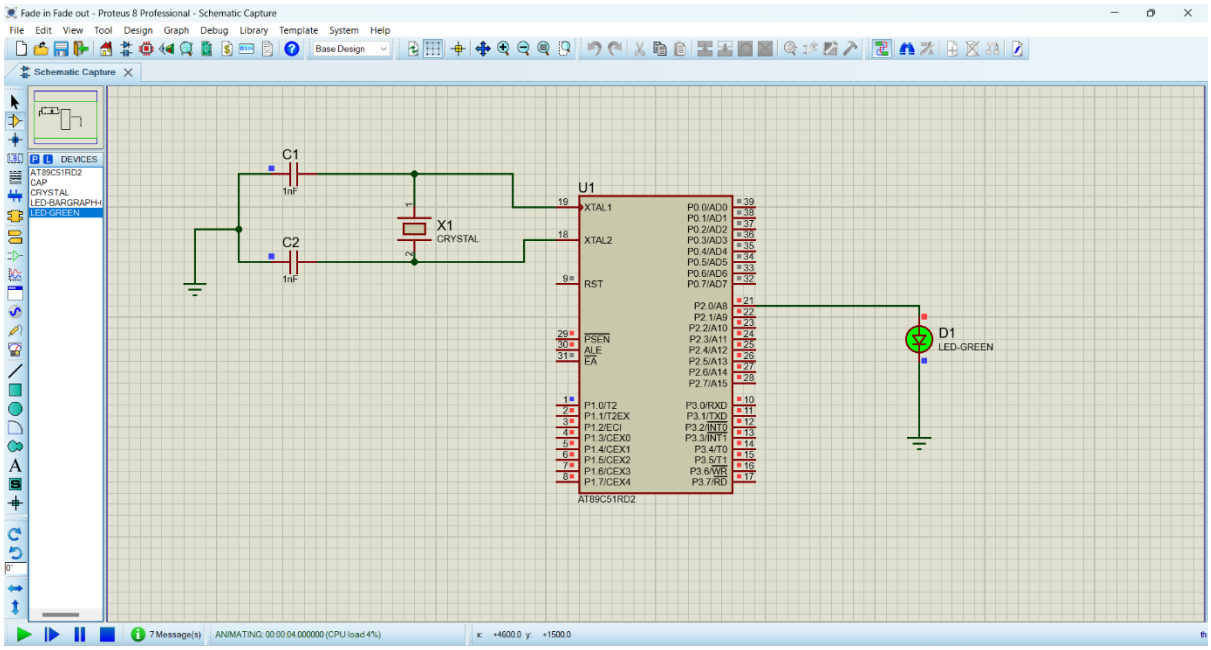
PROCEDURE:

- Open the Proteus software and create new project of 8051 microcontroller.
- Pick the required components and connect the circuit based on the circuit diagram.
- Open the keil software and create a hex code
- Copy the the file path and paste into the microcontroller of proteus software.
- Run the simulation in the proteus software and observer the output.

PROGRAM:

```
#include <REGX52.h>
delay(unsigned int y)
{
   unsigned int i,j;
for(i=0;i<y;i++)
{
   for(j=0;j<1275;j++){}
}
}
main()
{
while(1)
{
delay(100);
      P1_0 = 0;
delay(100);
      P1_0 = 1;
   }
}
```

CIRCUIT DIAGRAM:



RESULT:

Thus the program has been successfully verified and executed.

Experiment 5

STEPPER MOTOR USING 8051 USING KEIL AND PROTEUS

AIM:

Write an assembly language program for Stepper Motor Using 8051 using Keil and Proteus

SOFTWARE REQUIRED:

- Keil software 5.
- Proteus 8 software.

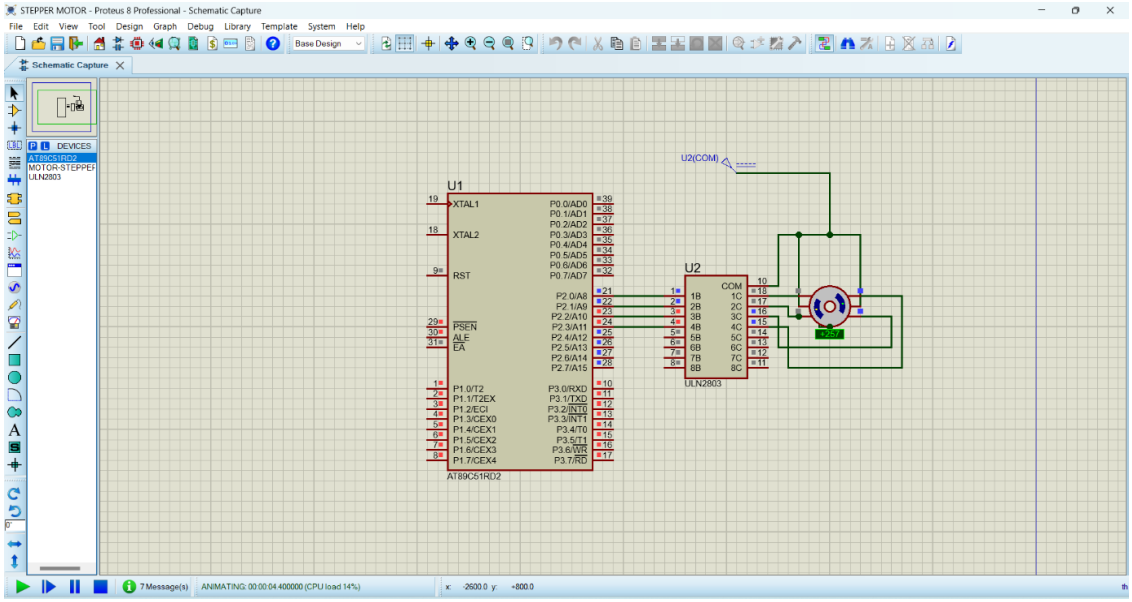
PROCEDURE:

- Open the Proteus software and create new project of 8051 microcontroller.
- Pick the required components and connect the circuit based on the circuit diagram.
- Open the keil software and create a hex code
- Copy the the file path and paste into the microcontroller of proteus software.
- Run the simulation in the proteus software and observer the output.

PROGRAM:

```
ORG 0000H
UP: MOV P2,#09H
ACALL DELAY
MOV P2,#0CH
ACALL DELAY
MOV P2,#06H
ACALL DELAY
MOV P2,#03H
ACALL DELAY
SJMP UP
DELAY:MOV R4,#18
H1:MOV R3,#255
H2:DJNZ R3,H2
DJNZ R4,H1
RET
END
```

Circuit Diagram:



RESULT:

Thus the program has been successfully verified and executed.

Experiment 6

INTERFACING OF RELAY USING 8051 USING KEIL AND PROTEUS

AIM:

Write an assembly language program for Interfacing of Relay Using 8051 using Keil and Proteus

SOFTWARE REQUIRED:

- Keil software 5.
- Proteus 8 software.

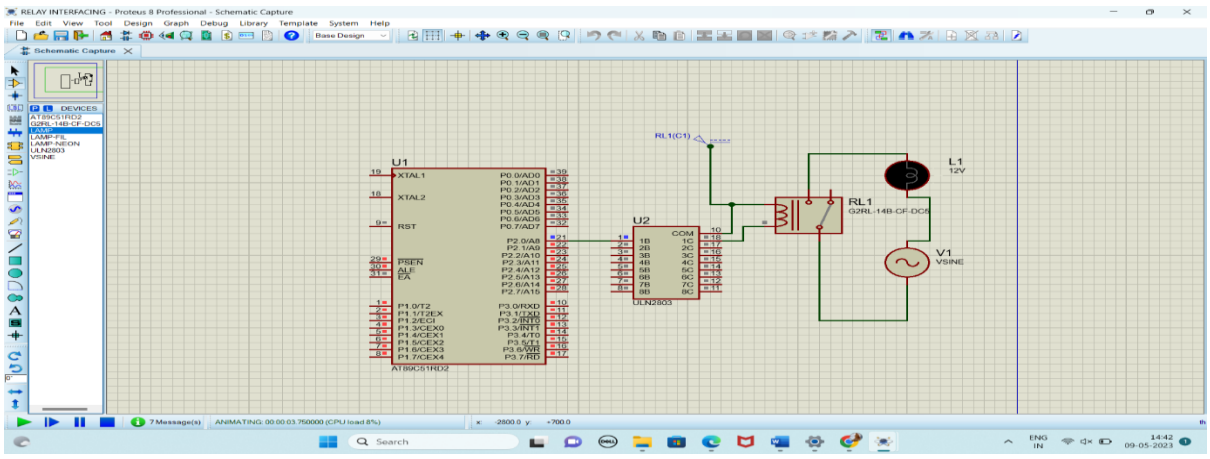
PROCEDURE:

- Open the Proteus software and create new project of 8051 microcontroller.
- Pick the required components and connect the circuit based on the circuit diagram.
- Open the keil software and create a hex code
- Copy the the file path and paste into the microcontroller of proteus software.
- Run the simulation in the proteus software and observer the output.

PROGRAM:

```
ORG 0000H
UP:SETB P2.0
ACALL DELAY
CLR P2.0
ACALL DELAY
SJMP UP
DELAY:MOV R4,#18
H1:MOV R3,#255
H2:DJNZ R3,H2
DJNZ R4,H1
RET
END
```

CIRCUIT DIAGRAM:



RESULT:

Thus the program has been successfully verified and executed.

Experiment 7

LED TOGGLE USING 8051 USING KEIL AND PROTEUS

AIM:

Write an assembly language program for LED Toggle Using 8051 using Keil and Proteus

SOFTWARE REQUIRED:

- Keil software 5.
- Proteus 8 software.

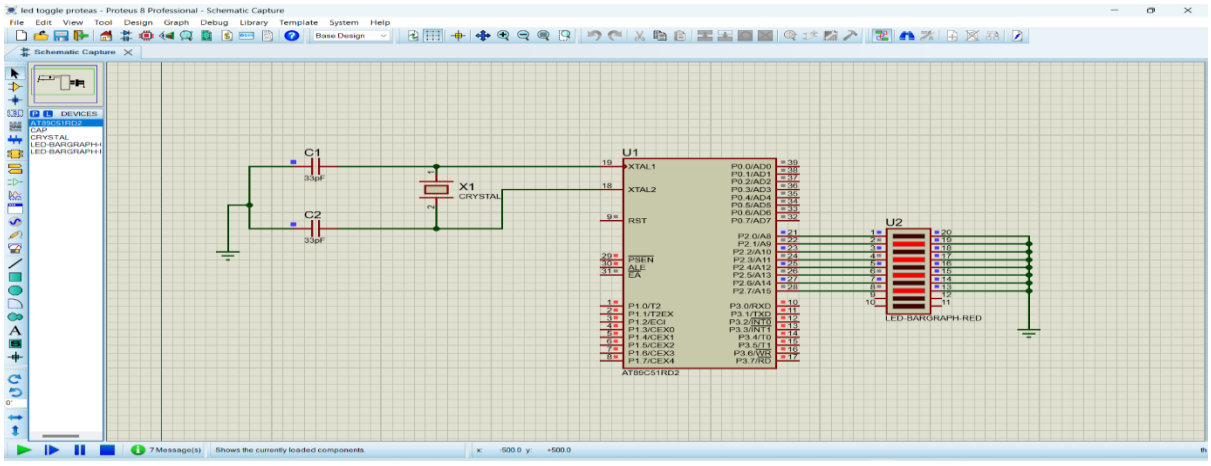
PROCEDURE:

- Open the Proteus software and create new project of 8051 microcontroller.
- Pick the required components and connect the circuit based on the circuit diagram.
- Open the keil software and create a hex code
- Copy the the file path and paste into the microcontroller of proteus software.
- Run the simulation in the proteus software and observer the output.

PROGRAM:

```
ORG 0000H
UP: MOV P2,#55H
ACALL DELAY
MOV P2,#0AAH
ACALL DELAY
SJMP UP
DELAY:MOV R4,#10
H1:MOV R3,#255
H2:DJNZ R3,H2
DJNZ R4,H1
RET
END
```

CIRCUIT DIAGRAM:



RESULT:

Thus the program has been successfully verified and executed.

Experiment 8

7 SEGMENT DISPLAY USING 8051 USING KEIL AND PROTEUS

AIM:

Write an assembly language program for 7 Segment Display Using 8051 using Keil and Proteus

SOFTWARE REQUIRED:

- Keil software 5.
- Proteus 8 software.

PROCEDURE:

- Open the Proteus software and create new project of 8051 microcontroller.
- Pick the required components and connect the circuit based on the circuit diagram.
- Open the keil software and create a hex code
- Copy the the file path and paste into the microcontroller of proteus software.
- Run the simulation in the proteus software and observer the output.

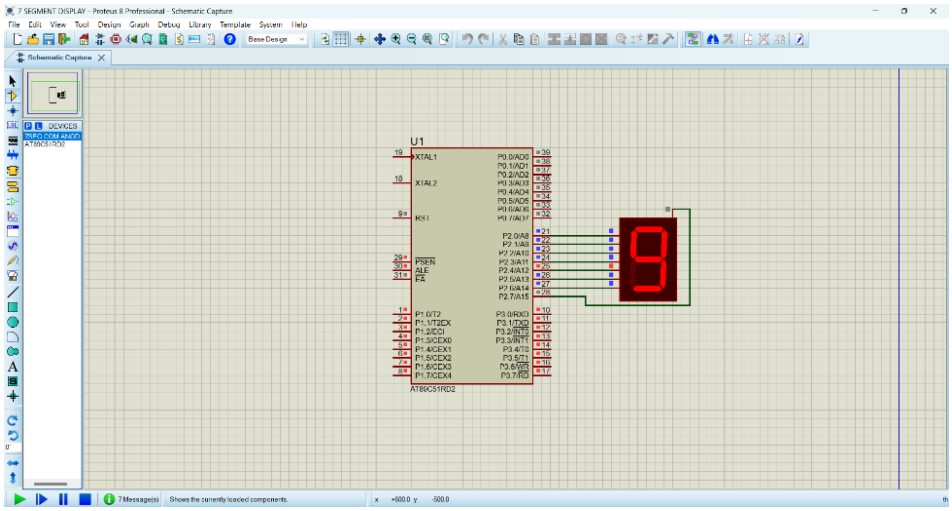
PROGRAM:

```
ORG 000H
UP:MOV P2,#0C0H
ACALL DELAY
MOV P2,#0F9H
ACALL DELAY
MOV P2,#0A4H
ACALL DELAY
MOV P2,#0B0H
ACALL DELAY
MOV P2,#99H
ACALL DELAY
MOV P2,#92H
ACALL DELAY
MOV P2,#82H
ACALL DELAY
MOV P2,#0F8H
ACALL DELAY
MOV P2, #80H
ACALL DELAY
MOV P2,#90H
ACALL DELAY

DELAY: MOV R5,#10
H1:MOV R4,#180
```

H2:MOV R3,#255
H3:DJNZ R3,H3
DJNZ R4,H2
DJNZ R5,H1
RET
END

CIRCUIT DIAGRAM:



RESULT:
Thus the program has been successfully verified and executed.

Experiment 9

LED CHASER USING 8051 USING KEIL AND PROTEUS

AIM:

Write an assembly language program for LED Chaser Using 8051 using Keil and Proteus

SOFTWARE REQUIRED:

- Keil software 5.
- Proteus 8 software.

PROCEDURE:

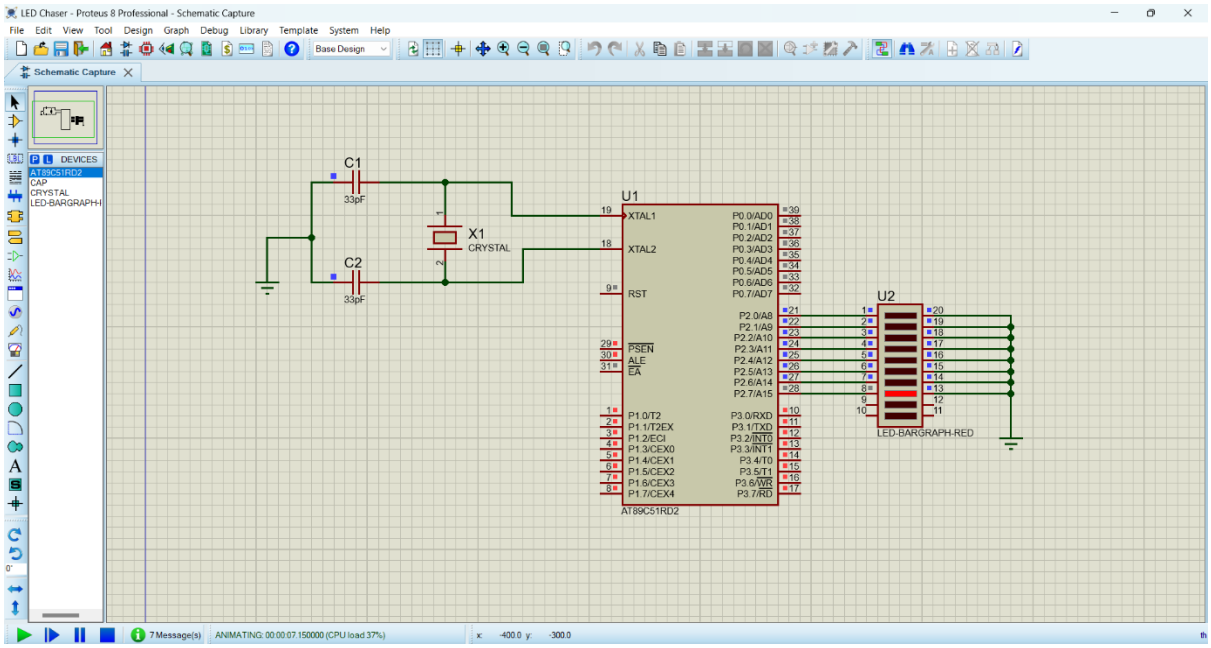
- Open the Proteus software and create new project of 8051 microcontroller.
- Pick the required components and connect the circuit based on the circuit diagram.
- Open the keil software and create a hex code
- Copy the the file path and paste into the microcontroller of proteus software.
- Run the simulation in the proteus software and observer the output.

PROGRAM:

```
ORG 0000H
UP: MOV P2,#01H
ACALL DELAY
MOV P2,#02H
ACALL DELAY
MOV P2,#04H
ACALL DELAY
MOV P2,#08H
ACALL DELAY
MOV P2,#10H
ACALL DELAY
MOV P2,#20H
ACALL DELAY
MOV P2,#40H
ACALL DELAY
MOV P2,#80H
ACALL DELAY
SJMP UP

DELAY: MOV R4,#255
H1: DJNZ R4,H1
RET
END
```

CIRCUIT DIAGRAM:



RESULT:

Thus the program has been successfully verified and executed.

Exp. No.: 10 **SERVO (DC) MOTOR INTERFACING USING 8051 MICROCONTROLLER AND KEIL C- AT89C51**

Aim:

To write a program to perform interfacing of DC motor using 8051 microcontroller and Keil C-AT89C51.

Components required:

- 8051 Microcontroller
- Keil uVision4

Theory:

Servo motors are used in **robotics, embedded systems** and industries because they are very precise and reliable. They are used to operate remote control toy cars, airplanes or robots. Their motion can be controlled by rotating them in particular angle. Servo motor can be controlled by **PWM signal**.

A DC servo motor consists of:

- DC motor
- Potentiometer (variable resistor)
- Gear assembly
- Control circuit

It consists of a closed loop system. Positive feedback is given to control the motion and position of the shaft. Feedback signal is generated by comparing output signal and reference inputsignal. Now, this feedback signal will act as input signal to control device. This signal will remain present as long as there remains a difference between reference input signal and output signal. So we have to maintain output of this system at desired value in presence of noises.

Code of Servo Motor Interfacing with 8051 Microcontroller:

```
#include <REGX51.H>
void Delay_servo(unsigned int);
sbit control_pin=P2^0;
sbit B1=P0^0;
sbit B2=P0^1;
sbit B3=P0^2;
void main()
{
P0=0x07;                                  // input port
control_pin=0;                                    // output pin
do
{
if(B1==1)
        {                                  //Turn to 90 degree
        control_pin=1;
        Delay_servo(1218);
```

```
        control_pin=0;
        }
else if(B2==1)

        {                               //Turn to 180 degree
        control_pin=1;
        Delay_servo(1470);
        control_pin=0;
        }
else if(B3==1)
        {                               //Turn to 270 degree
        control_pin=1;
        Delay_servo(1720);
        control_pin=0;
        }
}
while(1);
}
void Delay_servo(unsigned int d)
  {
  TMOD &=0xF0;                  // Clear 4bit field for Timer0
  TMOD|=0x01;                           // Set timer0 in mode1, 16bit
TH0=0xFF - (d>>8)&0xFF;                 // Load delay vales Timer 0 + Bitwise right shift[c][d]a
>> b
  TL0=0xFF- d&0xFF;
  ET0=1;                                //Enable timer0 interrupts
  EA=0;                                 // Global Interrupt
  TR0=1;                                // Start timer 0
  while(TF0==0);                        // Wait for overflow
 TR0=0;                               //Stop timer0
  TF0=0;                                // Clear Flag
  }
```

Result:

Thus using 8051 microcontroller with Keil, interfacing of servo motor is performed.

Exp. 11 **STUDY OF ARM PROCESSOR**

Date:

Aim:

To study the architecture and working of an ARM processor.

Component Required:

- ARM LPC2148

Theory:

ARM is a family of instruction set architectures for computer processors based on a reduced instruction set computing (RISC) architecture developed by British company ARM Holdings. A RISC-based computer design approach means ARM processors require significantly fewer transistors than typical processors in average computers. This approach reduces costs, heat and power use. These are desirable traits for light, portable, battery-powered devices—including smartphones, laptops, tablet and notepad computers), and other embedded systems. A simpler design facilitates more efficient multi-core CPUs and higher core counts at lower cost, providing higher processing power and improved energy efficiency for servers and supercomputers.

Features of LPC214x Series Controllers:

- 8 to 40 kB of on-chip static RAM and 32 to 512 kB of on-chip flash program memory.128 bit wide interface/accelerator enables high speed 60 MHz operation.
- In-System/In-Application Programming (ISP/IAP) via on-chip boot-loader software. Single flash sector or full chip erase in 400 ms and programming of 256 bytes in 1ms.
- Embedded ICE RT and Embedded Trace interfaces offer real-time debugging with the on-chip Real Monitor software and high speed tracing of instruction execution.
- USB 2.0 Full Speed compliant Device Controller with 2 kB of endpoint RAM. In addition, the LPC2146/8 provides 8 kB of on-chip RAM accessible to USB by DMA.
- One or two (LPC2141/2 vs. LPC2144/6/8) 10-bit A/D converters provide a total of 6/14analog inputs, with conversion times as low as 2.44 us per channel.
- Single 10-bit D/A converter provides variable analog output.
- Two 32-bit timers/external event counters (with four capture and four compare channels each), PWM unit (six outputs) and watchdog.
- Low power real-time clock with independent power and dedicated 32 kHz clock input.
- Multiple serial interfaces including two UARTs (16C550), two Fast I2C-bus (400 kbit/s), SPI and SSP with buffering and variable data length capabilities.
- Vectored interrupt controller with configurable priorities and vector addresses.
- Up to 45 of 5 V tolerant fast general purpose I/O pins in a tiny LQFP64 package.
- Up to nine edge or level sensitive external interrupt pins available.
- On-chip integrated oscillator operates with an external crystal in range from 1 MHz to30 MHz and with an external oscillator up to 50 MHz.
- Power saving modes include Idle and Power-down.
- Individual enable/disable of peripheral functions as well as peripheral clock scaling for additional power optimization.
- Processor wake-up from Power-down mode via external interrupt, USB, Brown-Out Detect (BOD) or Real-Time Clock (RTC).
- Single power supply chip with Power-On Reset (POR) and BOD circuits – CPU operating voltage range of 3.0 V to 3.6 V (3.3 V ± 10 %) with 5 V tolerant I/O pads.

LPC2148 needs the following hardware to work properly:

- Power Supply
- Crystal Oscillator
- Reset Circuit
- RTC crystal oscillator
- UART

Power Supply

LPC2148 works on 3.3 V power supply. LM 117 can be used for generating 3.3 V supply. However, basic peripherals like LCD, ULN 2003 (Motor Driver IC) etc. works on 5V. So AC mains supply is converted into 5V using below mentioned circuit and after that LM 117 is used to convert 5V into 3.3V.

Reset Circuit

Reset button is essential in a system to avoid programming pitfalls and sometimes to manually bring back the system to the initialization mode. MCP 130T is a special IC used for providing stable RESET signal to LPC 2148.

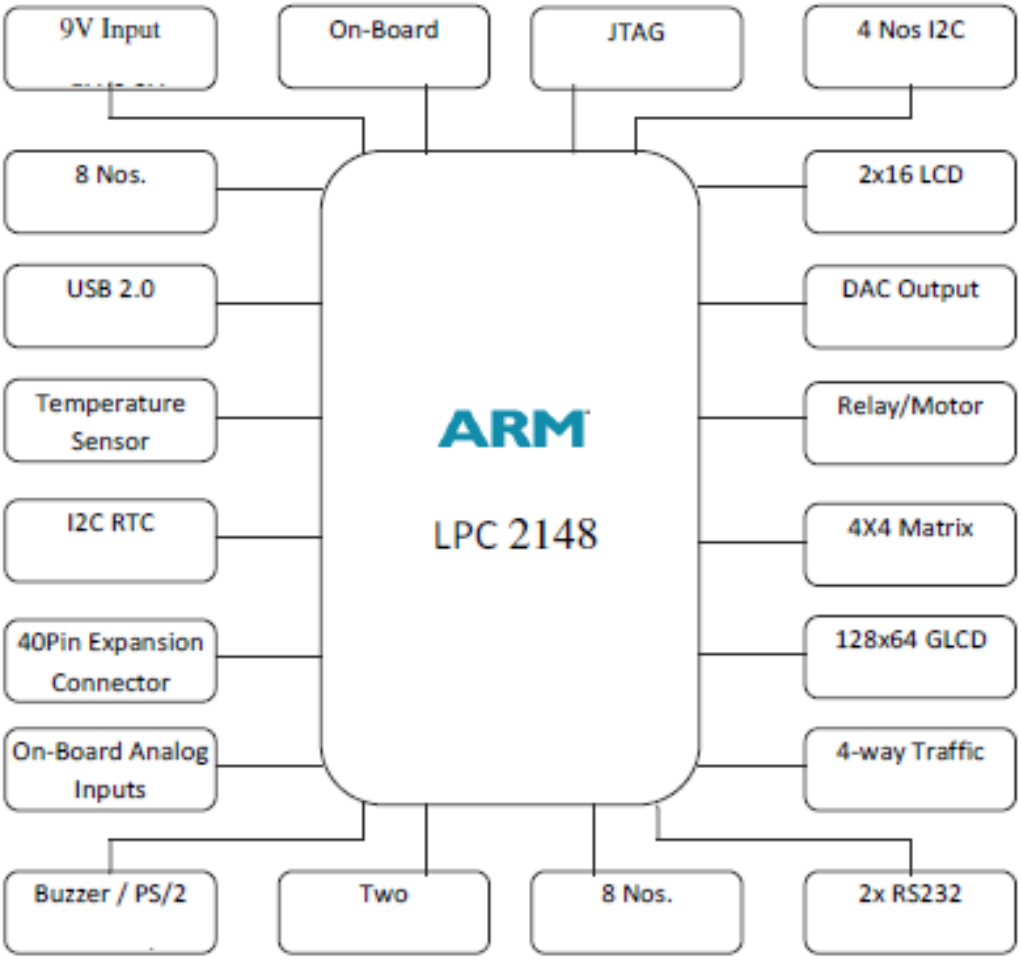


Figure - 3.1: ARM Processor

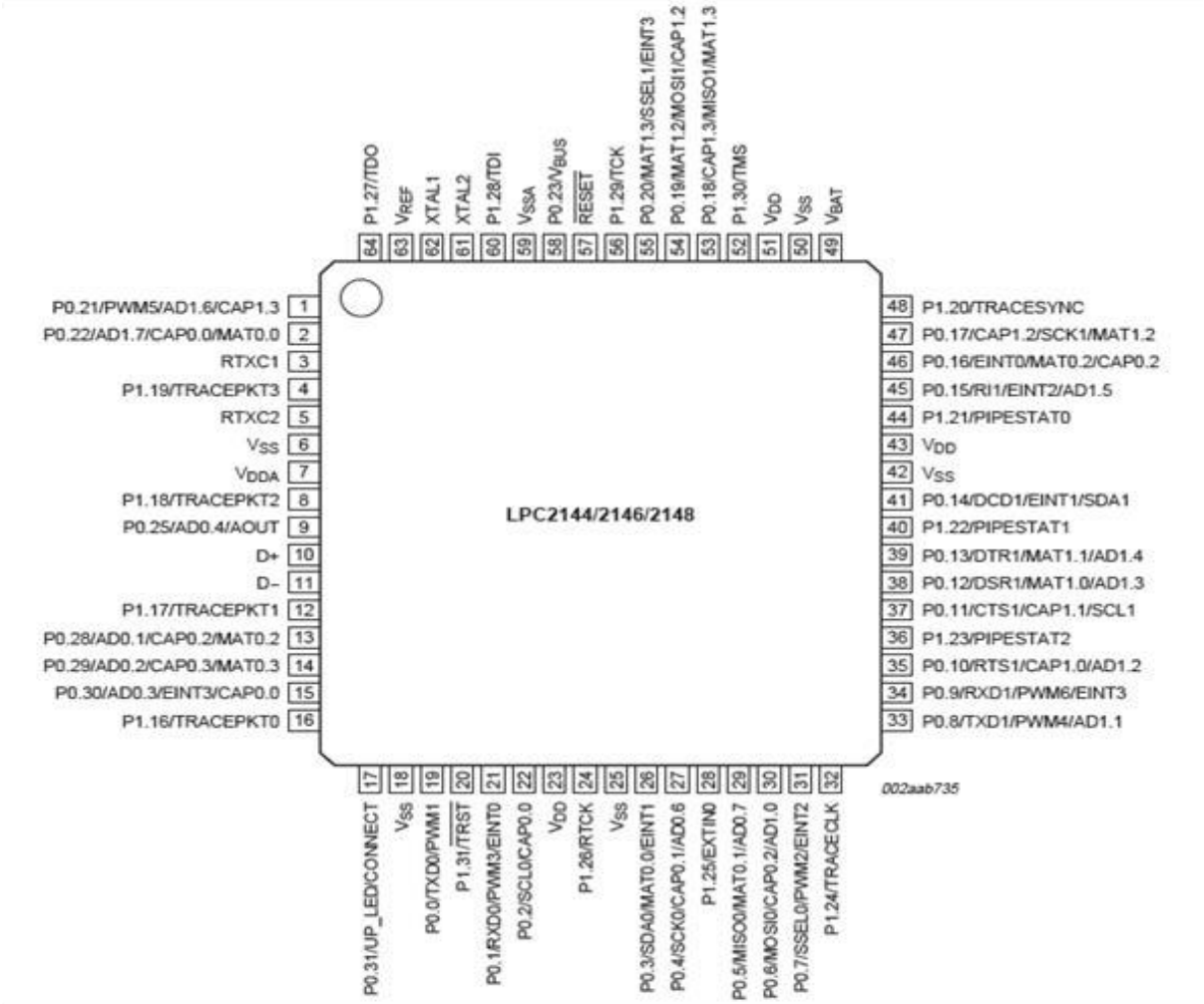
Flash Programming Utility

NXP Semiconductors produce a range of Microcontrollers that feature both on-chip. Flash memory and the ability to be reprogrammed using In-System Programming technology.

On-board Peripherals

• 8-Nos. of Point LED's (Digital Outputs)
• 8-Nos. of Digital Inputs (Slide Switch)
• 2 Lines X 16 Character LCD Display
• I2C Enabled 4 Digit Seven-Segment Display
• 128x64 Graphical LCD Display
• 4 X 4 Matrix keypad
• Stepper Motor Interface
• 2 Nos. Relay Interface
• Two UART for Serial Port Communication through PC
• Serial EEPROM
• On-chip Real Time Clock with Battery Backup
• PS/2 Keyboard Interface (Optional)
• Temperature Sensor
• Buzzer (Alarm Interface)
• Traffic Light Module (Optional)

Pin Configuration



Result:

The ARM processor has been studied successfully.

Exp. 12

Write and execute C program to blink LEDs using software delay routine in LPC2148 kit

Aim: To write and execute a C program to blink LEDs using software delay routine in LPC 2148 kit

Apparatus Required:

Keil uVision5 Software

Philips Flah Programmer

LPC 2148 kit

Procedure:

- Create new project on Keil software then add and compile the code.
- Create the hex code in the keil software and copy the path of the file location.
- Now interface the LPC2148 lit with the laptop/desktop to run the specific function.
- Open the flash magic software and paste the hex code path in the flash magic software.
- Select the respected COM Port where we interface the LPC2148 with laptop/desktop.
- Run the program in the trainer kit and observe the output .

Program:

```
#include "lpc214x.h"
void delay (unsigned int k);
void main(void)
{
IODIR0 = 0xFFFFFFFF; //Configure Port0 as output Port
PINSEL0 = 0;                    //Configure Port0 as General Purose IO
while(1)
{
IOSET0 = 0x0000ff00;  //Set P0.15-P0.8 to '1'
delay(1000);          //1 sec Delay
IOCLR0 = 0x0000ff00;  //Set P0.15-P0.8 to '0'
delay(1000);          //1 Sec Delay
}
}
```

```
//Delay Program
//Input - delay value in milli seconds
void delay(unsigned int k)
{
        unsigned int i,j;
        for (j=0; j<k; j++)
                for(i = 0; i<=800; i++);
```

Output: LEDs P0.15-P0.8 are blinking

Result:

Thus the C program to blink LEDs using software delay routine was written and executed in LPC 2148 kit

Write and execute C program to read the switch and display in the LEDs using LPC2148 kit

Exp: 13

Aim: To write and execute C program to read the switch and display in the LEDs using LPC2148 kit

Apparatus Required:

Keil uVision5 Software

Philips Flah Programmer

LPC 2148 kit

Procedure:

- Create new project on Keil software then add and compile the code.
- Create the hex code in the keil software and copy the path of the file location.
- Now interface the LPC2148 lit with the laptop/desktop to run the specific function.
- Open the flash magic software and paste the hex code path in the flash magic software.
- Select the respected COM Port where we interface the LPC2148 with laptop/desktop.
- Run the program in the trainer kit and observe the output .

Program:

```
#include "lpc214x.h"
int main(void)
{
 unsigned int sw_sts;

 IODIR0 = 0x0000ff00; //Configure Port0
        PINSEL0 = 0;                    //Configure Port0 as General Purose IO
while(1)
{
 sw_sts = IOPIN0;
 IOSET0 = 0x0000ff00;   //Set P0.15-P0.8 to '1'
 IOCLR0 = sw_sts >> 8;  //Set P0.15-P0.8 to '0'
}
```

```
}
```

Output: LEDs P0.15-P0.08 displayed the bits entered in the switches

Result:

Thus C program was written read the switch and display in the LEDs using LPC2148 kit\

Write and execute C program to display a number in seven segment LED in LPC2148 kit

Exp : 14

Aim: To write and execute C program to display a number in seven segment LED in LPC2148 kit

Apparatus Required:

Keil uVision5 Software

Philips Flah Programmer

LPC 2148 kit

Procedure:

- Create new project on Keil software then add and compile the code.
- Create the hex code in the keil software and copy the path of the file location.
- Now interface the LPC2148 lit with the laptop/desktop to run the specific function.
- Open the flash magic software and paste the hex code path in the flash magic software.
- Select the respected COM Port where we interface the LPC2148 with laptop/desktop.
- Run the program in the trainer kit and observe the output .

Program:

```
//SEVEN SEGMENT LED DISPLAY INTERFACE IN C
/* Program to Count 0-9 and Display it in 7 segment Display (MUX) DS4
 * Display Select DS3 ==> "P0.13"  Enable --> '0',  Disable --> '1'
 * Display Select DS4 ==> "P0.12"  Enable --> '0',  Disable --> '1'
 */
/* Segment Connection Display 1 & 2            Enable --> '1',  Disable --> '0'
 *-------------------------------------------------------------------
 * MSB                                        LSB
 *   Dp     G      F      E      D      C      B      A
 * P0.23  P0.22  P0.21  P0.20   P0.19  P0.18  P0.17  P0.16
 *   0      0      0      0      0      1      1      0   --> 6 => '1'
 *-------------------------------------------------------------------*/

#include <LPC214X.H>
```

```
#define DS3   1<<13             // P0.13
#define DS4   1<<12             // P0.12
#define SEG_CODE 0xFF<<16     // Segment Data from P0.16 to P0.23

unsigned char const seg_dat[]={0x3F, 0x6, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x7, 0x7F,
0x67};

void delayms(int n)
{
int i,j;
for(i=0;i<n;i++)
 {for(j=0;j<5035;j++)   //5035 for 60Mhz ** 1007 for 12Mhz
   {;}
 }
}

int main (void)
{
  unsigned char count;
PINSEL0 = 0; // Configure Port0 as General Purpose IO => P0.0  to P0.15
PINSEL1 = 0; // Configure Port0 as General Purpose IO => P0.16 to P0.31
IODIR0 = SEG_CODE | DS3 | DS4; //Configure Segement data & Select signal as output
IOSET0 = SEG_CODE | DS3 ;  //Disable DS3 display
IOCLR0 = DS4;  //Enable DS4 Display
count  = 0; //Initialize Count
//Display Count value
IOCLR0 = SEG_CODE;
IOSET0 = seg_dat[count]<<16;
while(1)
 {
        delayms(1000); //1 sec delay
   count++;             //Increment count
   if(count>9) count=0;  //Limit 0-9
```

```
    //Display Count value
    IOCLR0 = SEG_CODE;
    IOSET0 = seg_dat[count]<<16;
   }
}
```

Output: 7-Segment display counting from 0 to 9

Result:

Thus C program, was written and executed to display a number in seven segment LED in LPC2148 kit

Exp. 15

Write and execute C program for serial transmission and reception using on-chip UART in LPC2148 kit.

Date:

Aim: To write and execute C program for serial transmission and reception using on-chip UART in LPC2148 kit.

Apparatus Required:

Keil uVision5 Software

Philips Flah Programmer

LPC 2148 kit

Procedure:

- Create new project on Keil software then add and compile the code.
- Create the hex code in the keil software and copy the path of the file location.
- Now interface the LPC2148 lit with the laptop/desktop to run the specific function.
- Open the flash magic software and paste the hex code path in the flash magic software.
- Select the respected COM Port where we interface the LPC2148 with laptop/desktop.
- Run the program in the trainer kit and observe the output .

Program:

```
#include <lpc214x.h>
void UART0_Init(void)
{
 PLL0CON = 0;
 PLL0FEED=0xAA;
 PLL0FEED=0x55;
 VPBDIV = 1;
 // Fpclk = 12.000.000 MHz
 // DLM,DLH = Fpclk / (19200*16) = 39 = 0x27
 PINSEL0 |= 0x5; // Select UART0 RXD/TXD
 U0FCR = 0; // Disable FIFO's
 U0LCR = 0x83; // 8N1, enable Divisor latch bit
 U0DLL = 0x27; // baud rate fixed to 19200 @ PCLK = 12 Mhz
```

```
 U0DLM = 0;
 U0LCR = 3; // Disable Divisor latch bit
}
/*---------------------------------------------------------------*/
/* Function to send one char. to Serial Port */
void sout(unsigned char dat1)
{
 while(!(U0LSR & 0x20));//Wait for Tx Buffer Empty
 U0THR = dat1;   //Send to UART1
}
/*-------------------------------------------------------*/

int main (void)
{  int dat;
   UART0_Init();
 do
  {
   if(U0LSR & 1)  /* Check for RDR (Receiver Data Ready)command */
    {
      dat = U0RBR;// Receive Data from Srial Port
      sout(dat);  // Send Data to Srial Port
          }
   }while(1);
}
```

Output: Data was serially transmitted

Result:

Thus a C program was Written and executed for serial transmission and reception using on-chip UART in LPC2148 kit

Exp. 16

Write and execute C program for accessing an internal ADC and display the binary output in LEDS in LPC2148 kit

Date:

Aim: To write and execute C program for accessing an internal ADC and display the binary output in LEDS in LPC2148 kit.

Apparatus Required:

Keil uVision5 Software

Philips Flah Programmer

LPC 2148 kit

Procedure:

- Create new project on Keil software then add and compile the code.
- Create the hex code in the keil software and copy the path of the file location.
- Now interface the LPC2148 lit with the laptop/desktop to run the specific function.
- Open the flash magic software and paste the hex code path in the flash magic software.
- Select the respected COM Port where we interface the LPC2148 with laptop/desktop.
- Run the program in the trainer kit and observe the output .

Program:

```
#include <LPC214X.H>

#define LEDS  0xFF<<8   //LED => P0.8 to P0.15

//////////////////////////////////////////
/*--- ADC Signal Declaration */
//////////////////////////////////////////
#define AD0_1 1<< 24
#define CLK_DIV 1<<8
#define PDN 1<<21
#define SOC 1<<24
#define BURST 1<<16
#define DONE 1<<31
```

```
/*-------------------------------------------------------------*/
//Delay Program
//Input - delay value in milli seconds
void delay(unsigned int k)
{
        unsigned int i,j;
        for (j=0; j<k; j++)
                for(i = 0; i<=800; i++);
}
/*-------------------------------------------------------------*/
void adc_init()
{
unsigned long int ADC_CH;

         ADC_CH = 0 | 1 << 1; //Channel AD0.1
   AD0CR = SOC | PDN | CLK_DIV | ADC_CH | BURST ;
}
/*-------------------------------------------------------------*/
unsigned int adc_read( unsigned char channel)
{
          unsigned int aval;
          unsigned long int val;

   if (channel == 1) val = AD0DR1;
   else if (channel == 2) val = AD0DR2;
   else if (channel == 3) val = AD0DR3;

   val = val >> 6;
   val = val & 0x3FF;
   aval = val;
   return (aval);
}
/*-------------------------------------------------------------*/
```

```
//////////////////////////
/*----Main Program------*/
//////////////////////////
int main(void)
{
unsigned int tp1;

  IODIR0   = LEDS; //Configure Port0 as output Port
  PINSEL0 = 0; //Configure Port0 as General Purpose IO
  PINSEL1 = 0 | AD0_1; // Enable AD0.1

    adc_init(); //Initialise on-chip ADC

 do
  {  tp1 = adc_read(1); // Channel AD0 0.1
                    tp1 = tp1 >> 2;  // ADC 10 bit But LED 8bit, Truncate lsb 2 bits
     IOSET0 =  LEDS;  //Switch OFF all LEDS
     IOCLR0 = tp1 << 8;  //Set VAlue
     delay(1000);

  }while(1);

}
```

Output: The Potentiometer knob was adjusted to generate Analog input and Digital display is observed

Result:

Thus C program was Written and executed for accessing an internal ADC and display the binary output in LEDS in LPC2148 kit.

Exp. No : 17 **STUDY OF ARDUINO**
Date:

Aim:

To study the features, working, and architecture of an Arduino.

Component required:

- Arduino UNO

Theory:

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

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

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The software, too, is open-source, and it is growing through the contributions of users worldwide.

Why Arduino?

Thanks to its simple and accessible user experience, Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux. Teachers and students use it to build low cost scientific instruments, to prove chemistry and physics principles, or to get started with programming and robotics. Designers and architects build interactive prototypes, musicians and artists use it for installations and to experiment with new musical instruments. Makers, of course, use it to build many of the projects exhibited at the Maker Faire, for example. Arduino is a key tool to learn new things. Anyone - children, hobbyists, artists, programmers - can start tinkering just following the step by step instructions of a kit, or sharing ideas online with other members of the Arduino community.

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

- Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50
- Cross-platform - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.
- Simple, clear programming environment - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.
- Open source and extensible software - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.
- Open source and extensible hardware - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

Arduino UNO

The Arduino UNO is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 Digital pins, 6 Analog pins, and programmable with the Arduino IDE (Integrated Development Environment) via a type B USB cable. It can be powered by a USB cable or by an external 9 volt battery, though it accepts voltages between 7 and 20 volts.

"Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform. The ATmega328 on the Arduino Uno comes pre-programmed with a bootloader that allows uploading new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol. The Uno also differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it uses the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

Technical specifications:

- Microcontroller: Microchip ATmega328P
- Operating Voltage: 5 Volt
- Input Voltage: 7 to 20 Volts
- Digital I/O Pins: 14 (of which 6 provide PWM output)
- Analog Input Pins: 6
- DC Current per I/O Pin: 20 mA
- DC Current for 3.3V Pin: 50 mA
- Flash Memory: 32 KB of which 0.5 KB used by bootloader
- SRAM: 2 KB
- EEPROM: 1 KB
- Clock Speed: 16 MHz
- Length: 68.6 mm
- Width: 53.4 mm
- Weight: 25 g

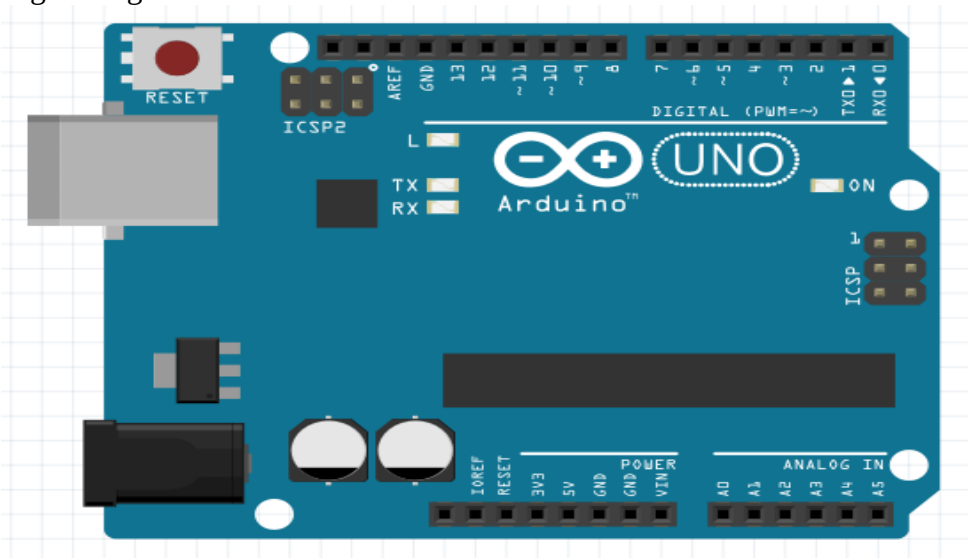


Figure - 2.1: Arduino UNO

Pin Details:

General pin functions

- LED: There is a built-in LED driven by digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- VIN: The input voltage to the Arduino/Genuino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- 5V: This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 20V), the USB connector (5V), or the VIN pin of the board (7-20V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage the board.
- 3.3V: A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- GND: Ground pins.

- IOREF: This pin on the Arduino/Genuino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs to work with the 5V or 3.3V.
- Reset: Typically used to add a reset button to shields which block the one on the board.[7]

Special Pin Functions

Each of the 14 digital pins and 6 Analog pins on the Uno can be used as an input or output, using pin Mode (), digital Write (), and digital Read () functions. They operate at 5 volts. Each pin can provide or receive 20 mA as recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50k ohm. A maximum of 40mA is the value that must not be exceeded on any I/O pin to avoid permanent damage to the microcontroller.The Uno has 6 analog inputs, labelled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function.

In addition, some pins have specialized functions:

- Serial: pins 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- External Interrupts: pins 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
- PWM(Pulse Width Modulation) 3, 5, 6, 9, 10, and 11 Can provide 8-bit PWM output with the analogWrite() function.
- SPI(Serial Peripheral Interface): 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.
- TWI(Two Wire Interface): A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.
- AREF(Analog REFerence): Reference voltage for the analog inputs.

Communication

The Arduino/Genuino Uno has a number of facilities for communicating with a computer, another Arduino/Genuino board, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The 16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino Software (IDE) includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1). A SoftwareSerial library allows serial communication on any of the Uno's digital pins.

Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino/Genuino Uno board is designed in a way that allows it to be reset by software

running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip.

This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the boot loader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened.

Result:

The features, working, and architecture of an Arduino have been studied.

Exp. No.: 18 **BLINKING OF AN LED USING ARDUINO**

Date:

Aim:

To interface an LED with Arduino and to write a program to make the LED blink at defined intervals.

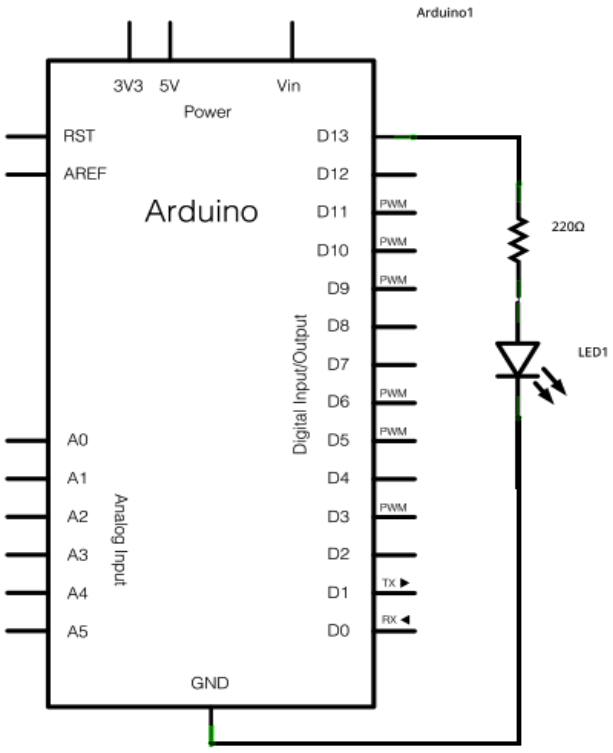
Components required:

- PC or laptop
- Arduino UNO
- 220 Ω resistor
- LED
- Breadboard
- Connecting wires

Software required:

- Arduino IDE

Circuit Diagram



.

Procedure:

1. Connect the components on a breadboard as mentioned above and connect the Arduino to the PC/laptop.
2. Open Aurdino IDE and open a new editor.
3. Enter the required code save it as a "ino" file.
4. In the "Tools" menu go to the "Port" tab and select the appropriate port.
5. Upload the program into the Arduino and verify the output.

Program:

To make the built-in LED blink:

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

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

To make the externally connected LED blink:

```
int led = 13;

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

// the loop function runs over and over again forever
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

Result:

A program has been written to make the built-in and externally connected LED blink and the output verified.

Exp. No.: 19 **FADING OF AN LED USING ARDUINO**

Date:

Aim:

To interface an LED with Arduino and to write a program to make the LED fade.

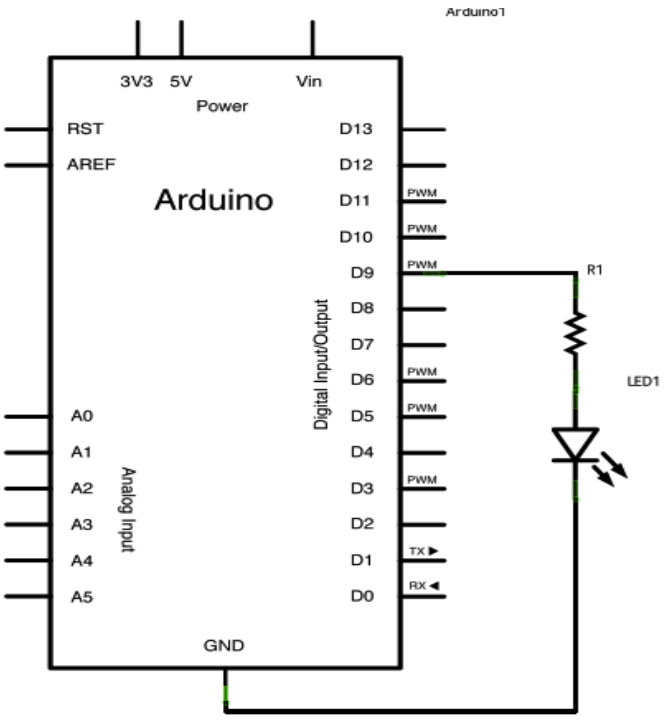
Components required:

- PC or laptop
- Arduino UNO
- 220 Ω resistor
- LED
- Breadboard
- Connecting wires

Software required:

- Arduino IDE

Circuit Diagram:



Procedure:

1. Connect the components on a breadboard as mentioned above and connect the Arduino to the PC/laptop.
2. Open Arduino IDE and open a new editor.
3. Enter the required code save it as a "ino" file.
4. In the "Tools" menu go to the "Port" tab and select the appropriate port.

5. Upload the program into the Arduino and verify the output.

Program:

```
int led = 9;          // the PWM pin the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by
// the setup routine runs once when you press reset:
void setup() {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
}
// the loop routine runs over and over again forever:
void loop() {
  // set the brightness of pin 9:
  analogWrite(led, brightness);
  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;
  // reverse the direction of the fading at the ends of the fade:
  if (brightness <= 0 || brightness >= 255) {
    fadeAmount = -fadeAmount;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

Result:

A program has been written to make the LED connected to an Arduino fade and the output verified.

Exp. No.: 20 **TURNING OF AN LED ON AND OFF USING A PUSH BUTTON**

Date:

To interface a pushbutton with an Arduino and to write a program to turn the built-in LED on and off using the pushbutton.

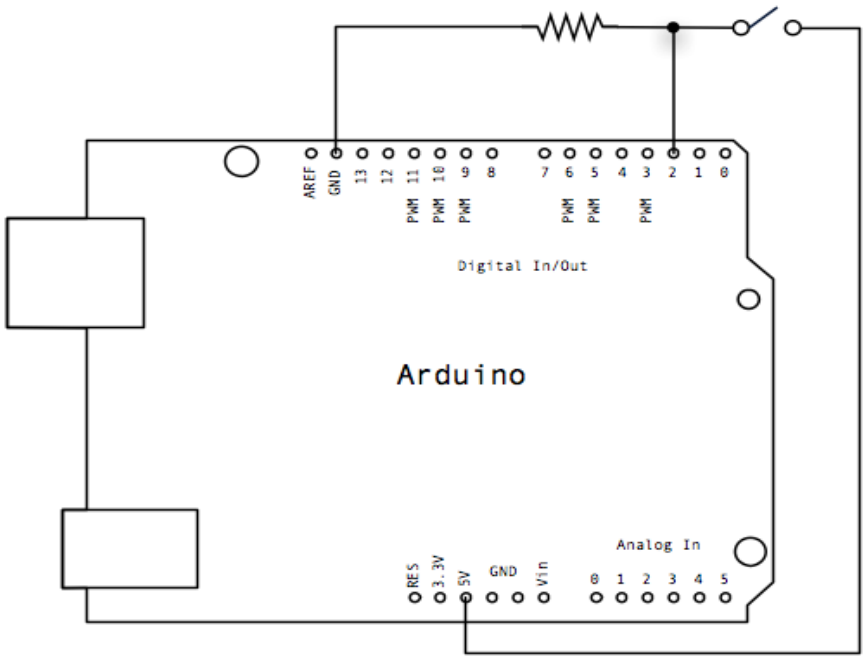
Components required:

- PC or laptop
- Arduino UNO
- 10 kΩ resistor
- Pushbutton
- Breadboard and connecting wires

Software required:

- Arduino IDE

Circuit Diagram:



Procedure:

1. Connect the components on a breadboard as mentioned above and connect the Arduino to the PC/laptop.
2. Open Arduino IDE and open a new editor.
3. Enter the required code save it as an "ino" file.
4. In the "Tools" menu go to the "Port" tab and select the appropriate port.
5. Upload the program into the Arduino and verify the output.

Program:

```
// constants won't change. They're used here to set pin numbers:

const int buttonPin = 2;     // the number of the pushbutton pin

const int ledPin =  13;      // the number of the LED pin

// variables will change:

int buttonState = 0;         // variable for reading the pushbutton status

void setup() {

 // initialize the LED pin as an output:

 pinMode(ledPin, OUTPUT);

 // initialize the pushbutton pin as an input:

 pinMode(buttonPin, INPUT);

}

void loop() {

 // read the state of the pushbutton value:

 buttonState = digitalRead(buttonPin);

 // check if the pushbutton is pressed. If it is, the buttonState is HIGH:

 if (buttonState == HIGH) {

   // turn LED on:

   digitalWrite(ledPin, HIGH);

 } else {

   // turn LED off:

   digitalWrite(ledPin, LOW);

 } }
```

Result:

A pushbutton has been interfaced with an Arduino and a program written, to turn thebuilt-in LED on and off, has been verified

Exp. No.: 21 **INTERFACING A WATER-LEVEL SENSOR WITH AN ARDUINO**

Date:

Aim:

To interface a water-level sensor with an Arduino and to write a program to measure the water level.

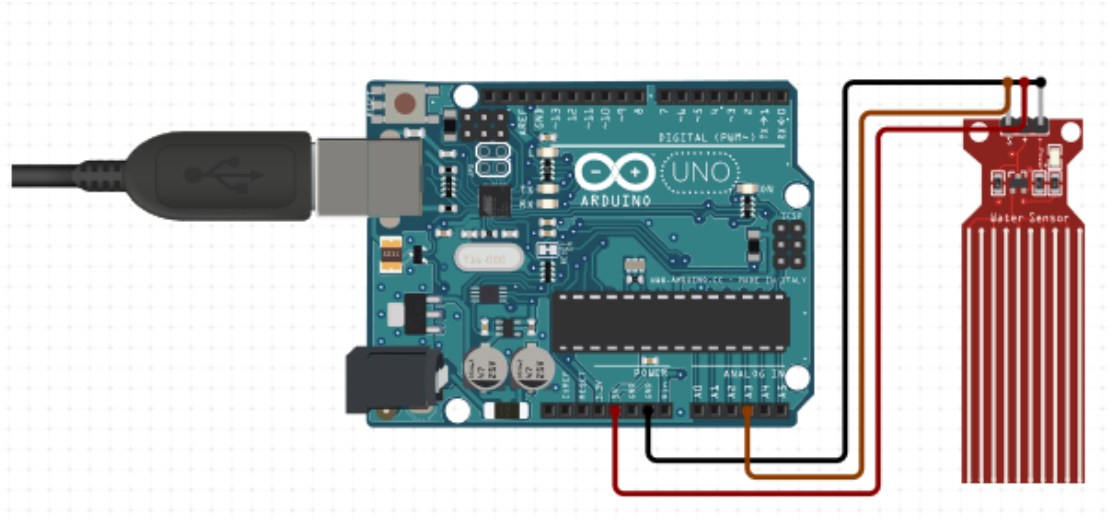
Components required:

- PC or laptop
- Arduino UNO
- Water-level sensor
- Breadboard
- Connecting wires

Software required:

- Arduino IDE

Circuit Diagram:



Procedure:

1. Connect the components on a breadboard as mentioned above and connect the Arduino to the PC/laptop.
2. Open Arduino IDE and open a new editor.
3. Enter the required code save it as an “ino” file.
4. In the “Tools” menu go to the “Port” tab and select the appropriate port.
5. Upload the program into the Arduino and verify the output on the serial monitor in Arduino IDE.

Program:

```
int sensorPin = A3;
int sensorValue = 0;
int value;
void setup() {
// put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(sensorPin, INPUT);
}
void loop() {
  // put your main code here, to run repeatedly:
  //sensorValue = analogRead(sensorPin);
  value = analogRead(sensorPin);
 if (value<=480){
    Serial.println("Water level: 0mm - Empty!");
  }
else if (value>480 && value<=530){
    Serial.println("Water level: 0mm to 5mm");
  }
  else if (value>530 && value<=615){
    Serial.println("Water level: 5mm to 10mm");
  }
  else if (value>615 && value<=660){
    Serial.println("Water level: 10mm to 15mm");
  }
else if (value>660 && value<=680){
    Serial.println("Water level: 15mm to 20mm");
  }
else if (value>680 && value<=690){
    Serial.println("Water level: 20mm to 25mm");
  }
```

```
else if (value>690 && value<=700){
   Serial.println("Water level: 25mm to 30mm");
  }
else if (value>700 && value<=705){
   Serial.println("Water level: 30mm to 35mm");
  }
else if (value>705){
   Serial.println("Water level: 35mm to 40mm");
  }
  delay(2000);
}
```

Result:

A water-level sensor was interfaced with an Arduino and a program to measure the water level was written and executed.

Exp. No.: 22 **INTERFACING AN ULTRSONIC SENSOR WITH AN ARDUINO**

Date:

Aim:

To interface an ultrasonic sensor with an Arduino and to write a program to measure the distance of an object from the sensor.

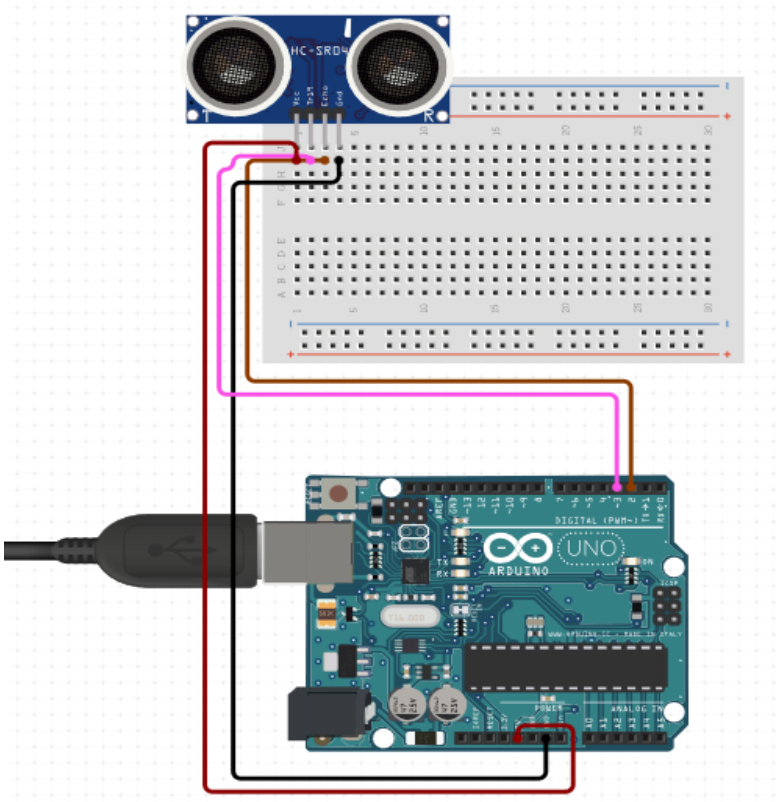
Components required:

- PC or laptop
- Arduino UNO
- Ultrasonic sensor HCSR04
- Breadboard
- Connecting wires

Software required:

- Arduino IDE

Circuit diagram:



.

Procedure:

1. Connect the components on a breadboard as mentioned above and connect the Arduino to the PC/laptop.
2. Open Arduino IDE and open a new editor.
3. Enter the required code save it as an "ino" file.
4. In the "Tools" menu go to the "Port" tab and select the appropriate port.
5. Upload the program into the Arduino and verify the output on the serial monitor in Arduino IDE.

Program:

```
// Include Libraries

#include "Arduino.h"

#include "NewPing.h"

// Pin Definitions

#define HCSR04_PIN_TRIG     3

#define HCSR04_PIN_ECHO 2

// Global variables and defines

// object initialization

NewPing hcsr04(HCSR04_PIN_TRIG,HCSR04_PIN_ECHO);

// define vars for testing menu

const int timeout = 10000;       //define timeout of 10 sec

char menuOption = 0;

long time0;

// Setup the essentials for your circuit to work. It runs first every time your circuit is powered with electricity.

void setup()

{

  // Setup Serial which is useful for debugging

   // Use the Serial Monitor to view printed messages

   Serial.begin(9600);

   while (!Serial) ; // wait for serial port to connect. Needed for native USB

   Serial.println("start");

   menuOption = menu();
```

```
// Main logic of your circuit. It defines the interaction between the components you selected. After setup, it runs over and over again, in an eternal loop.
void loop()
{
  if(menuOption == '1') {
  // Ultrasonic Sensor - HC-SR04 - Test Code
  // Read distance measurment from UltraSonic sensor
  int hcsr04Dist = hcsr04.ping_cm();
  delay(1001);
  Serial.print(F("Distance: ")); Serial.print(hcsr04Dist); Serial.println(F("[cm]"));
  }
  if (millis() - time0 > timeout)
  {
    menuOption = menu();
  }
}
// Menu function for selecting the components to be tested
// Follow serial monitor for instructions
char menu()
{
  Serial.println(F("\nWhich component would you like to test?"));
  Serial.println(F("(1) Ultrasonic Sensor - HC-SR04"));
  Serial.println(F("(menu) send anything else or press on board reset button\n"));

  while (!Serial.available());

  // Read data from serial monitor if received
  while (Serial.available())
  {
    char c = Serial.read();
    if (isAlphaNumeric(c))
```

```
    {
        if(c == '1')
            Serial.println(F("Now Testing Ultrasonic Sensor - HC-SR04"));
        else
        {
          Serial.println(F("illegal input!"));
          return 0;
        }
        time0 = millis();
        return c;
     }
  }
}
```

Result:

An ultrasonic sensor has been interfaced with the Arduino and the distance of an object from the sensor has been measured.

Exp. No.: 23 TO USE A BUZZER (OR PIEZO SPEAKER) WITH ARDUINO

Date:

Aim:

To interface an buzzer sensor with an Arduino.

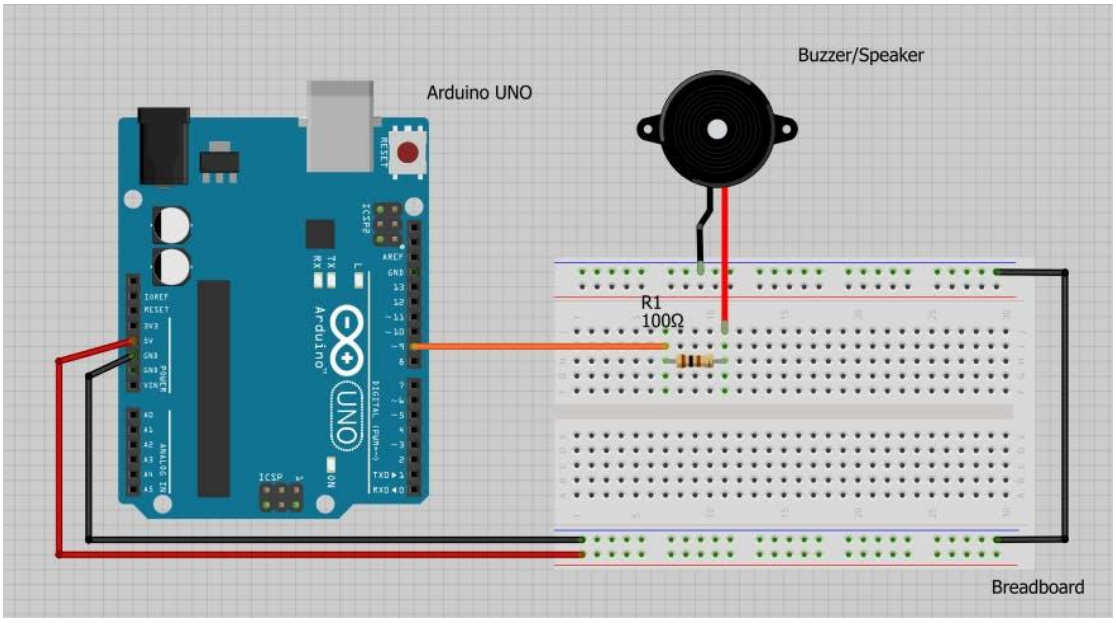
Components required:

- PC or laptop
- Arduino UNO
- Buzzer
- Breadboard
- Connecting wires

Software required:

- Arduino IDE

Connections:



Procedure:

6. Connect the components on a breadboard as mentioned above and connect the Arduino to the PC/laptop.
7. Open Arduino IDE and open a new editor.
8. Enter the required code save it as an “ino” file.
9. In the “Tools” menu go to the “Port” tab and select the appropriate port.
10. Upload the program into the Arduino and verify the output on the serial monitor in Arduino IDE.

Program:

```
const int buzzer = 9; //buzzer to arduino pin 9
void setup(){
   pinMode(buzzer, OUTPUT); // Set buzzer - pin 9 as an output
}
void loop(){
  tone(buzzer, 1000); // Send 1KHz sound signal...
  delay(1000);        // ...for 1 sec
 noTone(buzzer);     // Stop sound...
  delay(1000);        // ...for 1sec

}
```

Result:

Exp. No.: 24 **MQ-6 GAS SENSOR INTERFACING WITH ARDUINO**

Date:

Aim:

To interface a MQ-6 Gas sensor with an Arduino.

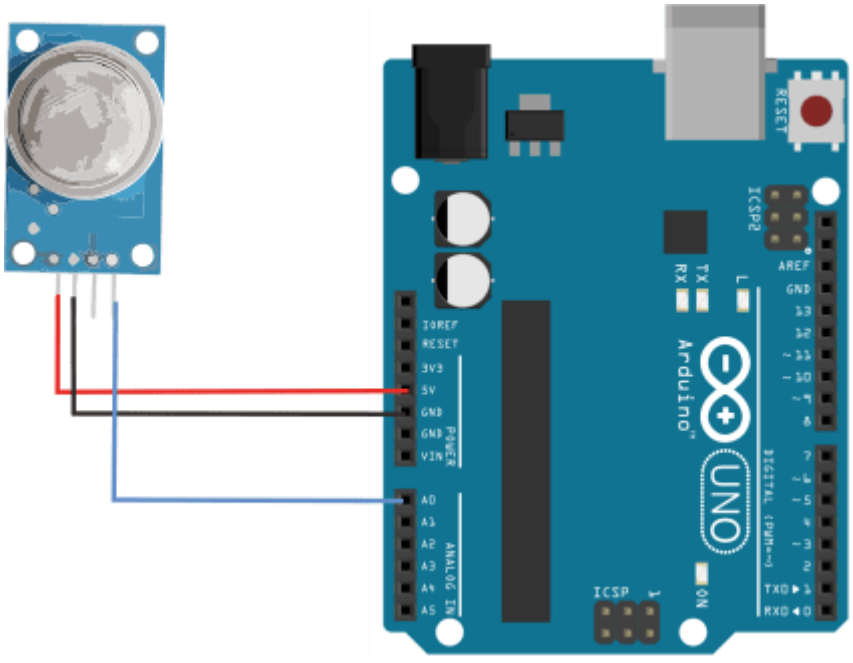
Components required:

- PC or laptop
- Arduino UNO
- MQ-6 GAS SENSOR
- Breadboard
- Connecting wires

Software required:

- Arduino IDE

Connections:



VCC - 5V

GND - GND

S - Analog pin0

Procedure:

- Connect the components on a breadboard as mentioned above and connect the Arduino to the PC/laptop.

- Open Arduino IDE and open a new editor.
- Enter the required code save it as an “ino” file.
- In the “Tools” menu go to the “Port” tab and select the appropriate port.
- Upload the program into the Arduino and verify the output on the serial monitor in Arduino IDE.

Program:

```
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1000);
}
```

Result:

Exp. No.: 25 **BLINKING OF AN LED USING ARDUINO**

Date:

Aim:

To interface an LED with Arduino and to write a program to make the LED blink at defined intervals.

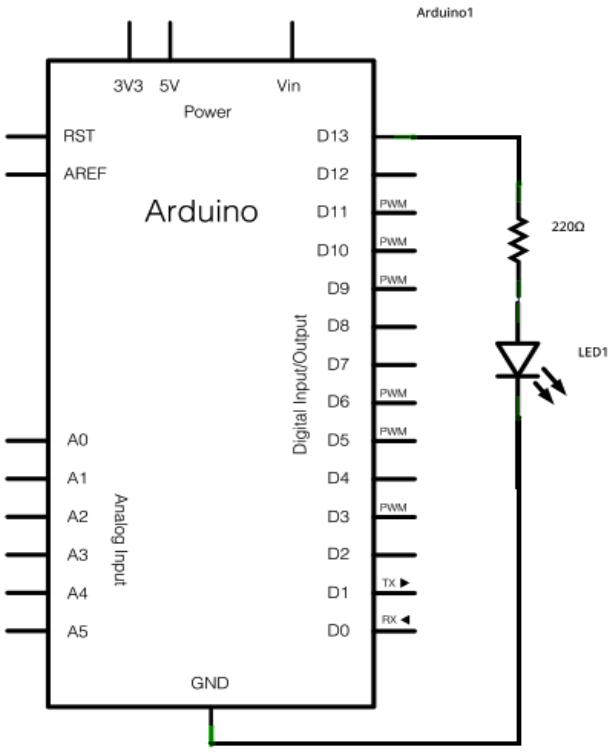
Components required:

- PC or laptop
- Arduino UNO
- 220 Ω resistor
- LED
- Breadboard
- Connecting wires

Software required:

- Arduino IDE

Circuit Diagram



.

Procedure:

- Connect the components on a breadboard as mentioned above and connect the Arduino to the PC/laptop.
- Open Aurdino IDE and open a new editor.
- Enter the required code save it as a "ino" file.
- In the "Tools" menu go to the "Port" tab and select the appropriate port.
- Upload the program into the Arduino and verify the output.

Program:

To make the built-in LED blink:

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

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

To make the externally connected LED blink:

```
int led = 13;

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

// the loop function runs over and over again forever
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

Result:

A program has been written to make the built-in and externally connected LED blink and the output verified.

Exp. No.: 26 **INTERFACING A WATER-LEVEL SENSOR WITH AN ARDUINO**

Date:

Aim:

To interface a water-level sensor with an Arduino and to write a program to measure the water level.

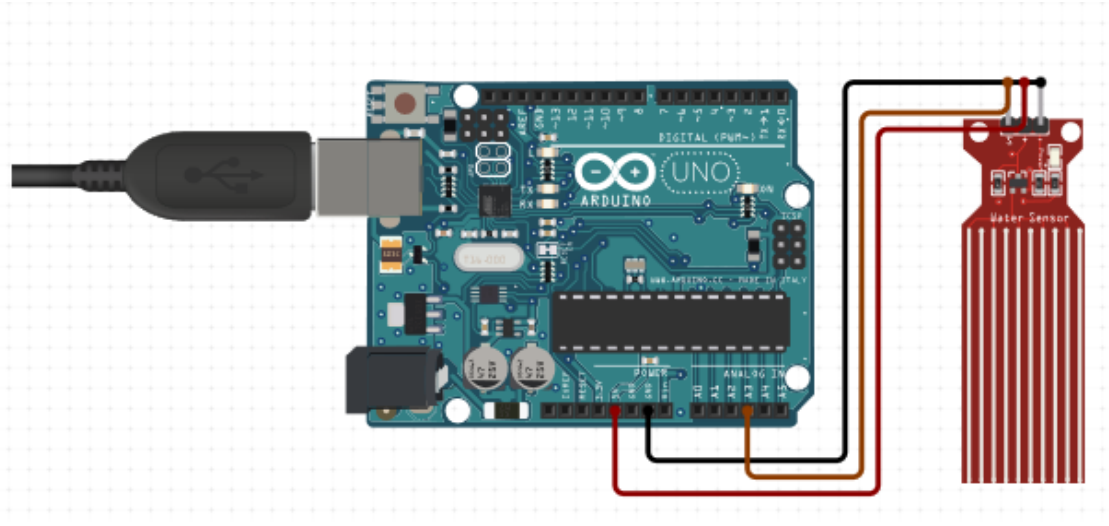
Components required:

- PC or laptop
- Arduino UNO
- Water-level sensor
- Breadboard
- Connecting wires

Software required:

- Arduino IDE

Circuit Diagram:



Procedure:

6. Connect the components on a breadboard as mentioned above and connect the Arduino to the PC/laptop.
7. Open Arduino IDE and open a new editor.
8. Enter the required code save it as an “ino” file.
9. In the “Tools” menu go to the “Port” tab and select the appropriate port.
10. Upload the program into the Arduino and verify the output on the serial monitor in Arduino IDE.

Program:

```
int sensorPin = A3;
int sensorValue = 0;
int value;
void setup() {
// put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(sensorPin, INPUT);
}
void loop() {
  // put your main code here, to run repeatedly:
  //sensorValue = analogRead(sensorPin);
  value = analogRead(sensorPin);
 if (value<=480){
    Serial.println("Water level: 0mm - Empty!");
  }
else if (value>480 && value<=530){
    Serial.println("Water level: 0mm to 5mm");
  }
  else if (value>530 && value<=615){
    Serial.println("Water level: 5mm to 10mm");
  }
  else if (value>615 && value<=660){
    Serial.println("Water level: 10mm to 15mm");
  }
else if (value>660 && value<=680){
    Serial.println("Water level: 15mm to 20mm");
  }
else if (value>680 && value<=690){
    Serial.println("Water level: 20mm to 25mm");
  }
```

```
else if (value>690 && value<=700){
   Serial.println("Water level: 25mm to 30mm");
  }
else if (value>700 && value<=705){
   Serial.println("Water level: 30mm to 35mm");
  }
else if (value>705){
   Serial.println("Water level: 35mm to 40mm");
  }
  delay(2000);
}
```

Result:

A water-level sensor was interfaced with an Arduino and a program to measure the water level was written and executed.

Exp. No.: 27 **MQ-6 GAS SENSOR INTERFACING WITH ARDUINO**

Date:

Aim:

To interface a MQ-6 Gas sensor with an Arduino.

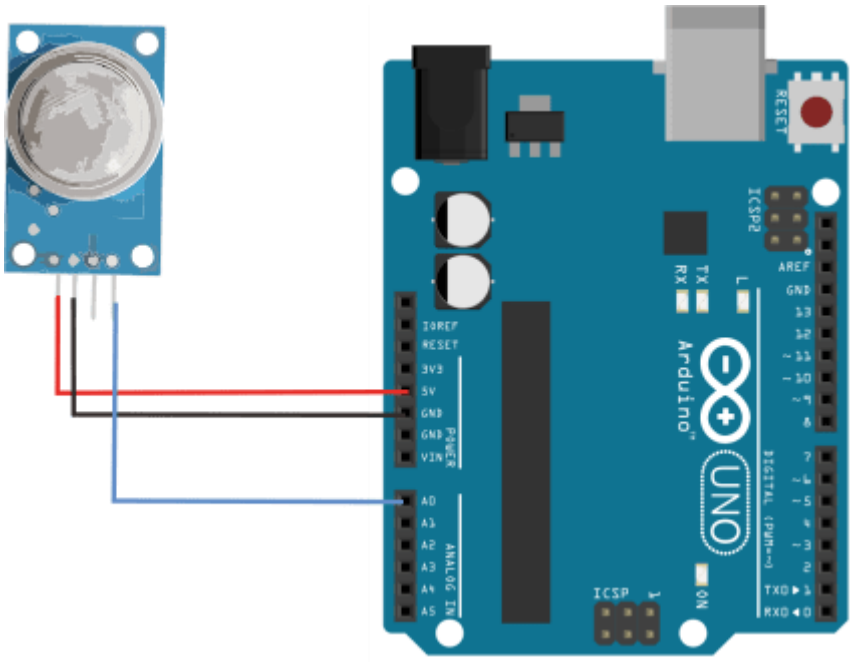
Components required:

- PC or laptop
- Arduino UNO
- MQ-6 GAS SENSOR
- Breadboard
- Connecting wires

Software required:

- Arduino IDE

Connections:



VCC - 5V

GND - GND

S - Analog pin0

Procedure:

- Connect the components on a breadboard as mentioned above and connect the Arduino to the PC/laptop.
- Open Arduino IDE and open a new editor.
- Enter the required code save it as an "ino" file.
- In the "Tools" menu go to the "Port" tab and select the appropriate port.
- Upload the program into the Arduino and verify the output on the serial monitor in Arduino IDE.

Program:

```
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1000);
}
```

Result: