REVA UNIVERSITY

BENGALURU, INDIA

Computer Organization and Architecture Lab

B20CI0402

Fourth Semester CSE / AIDS

2022-2023 Even Semester

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Name	
SRN	
Branch	
Semester	
Section	
Academic Year	

CONTENTS

COURSE OVERVIEW:

Computer organization and architecture is the science and art of selecting and interconnecting hardware components to create a computer that meets functional, performance, and cost goals. Computer organization defines the constituent parts of the system, how they are interconnected, and how they interoperate in order to implement the architectural specification. In this course, student will learn the basics of hardware components from basic arithmetic units to memory and I/O devices, instruction set architectures and assembly language, anddesigns to improve performance.

COURSE OBJECTIVE (S):

The objectives of this course are to:

1. Explain architecture of ARM processor and write simple assembly programs.
2. Demonstrate the translation of assembly instructions into their binary representation.
3. Describe and understand the processor memory hierarchy.
4. Discuss basic understanding of interrupts, I/O devices, and I/O protocols

COURSE OUTCOMES (COs)

After the completion of the course, the student will be able to:

CO#	Course Outcomes	POs	PSOs
CO1	Make use of ARM processor instruction set for developing simple assembly programs.	1,2,5,9,10,12	1
CO2	Interpret the functional architecture of computing systems.	1,2,5, 9,10,12	1
CO3	Identify the issue related to instruction set architecture, memory unit and control unit and I/O functions.	1,2,5, 9,10,12	1
CO4	Develop a real world application using parallel processing concepts.	1,2,4, 9,10,12	1,2,3
CO5	Summarize the common components and organization of memory unit and I/O functions.	1,2,4, 9,10,12	1
CO6	Experiment with the multitasking of different programs.	1,2,4,9,10,12	1,2,3

BLOOM'S LEVEL OF THE COURSE OUTCOMES

CO#	Bloom's Level					
	Remember (L1)	Understand (L2)	Apply (L3)	Analyze (L4)	Evaluate (L5)	Create (L6)
CO1			√			
CO2		√				
CO3			√			
CO4			√			
CO5		√				
CO6			√			

COURSE ARTICULATION MATRIX

CO#/ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	2	3			3				3	3		2	3		
CO2	3	3			2				3	3		2	3		
CO3	3	3			2				3	3		2	3		
CO4	3	3		2					3	3		2	3	3	3
CO5	3	3		2					3	3		2	3		
CO6	3	3		2					3	3		2	3	3	3

Note: 1-Low, 2-Medium, 3-High

PART-A Assembly Language Programs.

Experiment No 1a.

Aim: Write an assembly language program to realize the given expressions

i)A=B+C-D ii)A=4A+B iii)X=3X+4Y+9Z

i) A=B+C-D

```
        AREA   RESET, CODE
        ENTRY
        MOV  R0, #00              ; move immediate data 00 into R0 (A)
        MOV  R1, #20              ; B
        MOV  R2, #30              ; C
        MOV  R3,#10         ; D
        ADD  R0,R2,R1       ; A=B+C
        SUB   R0,R0,R3      ; A=A-D
STOP    B   STOP
        END
```

Output : R0 = 0x00000028

ii)A=4A+B

```
        AREA   RESET, CODE
        ENTRY
        MOV  R0,#2                  ; A
        MOV  R1,#5                  ; B
        ADD  R0,R1,R0,LSL #2    ;Multiply R0(A) by 4 and add R1(B) and store in R0(A)
STOP    B   STOP
        END
```

Output : R0 = 0x0000000D

iii)X=3X+4Y+9Z

```
        AREA   RESET, CODE
        ENTRY
        MOV R1,  #2                 ; Let X = 2
        MOV R2,  #3                 ; Let Y = 3
        MOV R3,  #4                 ; Let Z = 4
        ADD  R1, R1, R1, LSL #1     ; Multiply R1(X) by 2 and Add with R1(X) Store result in R1
                                        (computation of 3X)(Barrel shift used)
        MOV R2, R2, LSL #2          ; Multipy R2(y) by 4 and Move the result to R2(Y)
                                      (computation of 4Y)
        ADD  R3, R3, R3, LSL #3     ; Mutiply R3(Z) by 8 and Add with R3(Z) and place result in
                                      R3(Z)  (computation of 9Z)
        ADD  R1, R1, R2             ; X = 3X + 4Y
        ADD  R1, R1, R3             ; X = X + 9Z
STOP    B   STOP
        END
```

Output : R1 = 0x00000036

Experiment No 1b.

Aim: Write an assembly language program to add two 64-bit numbers.

```
        AREA RESET, CODE                ;Defining code area stsrt point 0x00000000
        ENTRY                           ;entry point should followed be instructions
        LDR R0,=VALUE1                  ;load R0 with address VALUE1
        LDR R1,[R0]                     ;load R1 with contents pointed by R0
        LDR R2,[R0,#4]                  ;load R2 with contents pointed by [R0 + 4]
        LDR R0,=VALUE2
        LDR R3,[R0]
        LDR R4,[R0,#4]
        ADDS R6,R2,R4                   ;add with status contents of R4 and R2 and store in R6
                                          (lower order 32 bit number)
        ADC R5,R1,R3                    ;add with carry contents of R1 and R3 and store in R5
                                          (higher order 32 bit number)
        LDR R0,=RESULT                  ;load R0 with address RESULT (Data Memory RAM
                                          Start address = 0x40000000)
        STR R6,[R0]                     ;store the contents of R6 (lower order) to data memory
                                          pointed by R0
        STR R5,[R0,#4]                  ;store the contents of R5 (higher order) to data
                                          memory Pointed by [R0 + 4]
STOP   B   STOP                         ;infinite loop here to stop execution
VALUE1 DCD &BBBBBBBB,&AAAAAAAA          ;data first 64 bit number stored at location
VALUE1
VALUE2 DCD &CCCCCCCC,&FFFFFFFF          ;data second 64 bit number stored at location
VALUE2
        AREA MEMORY,DATA                ;defining RAM area
RESULT SPACE 8                          ;Reserve 8 bytes of RAM cleared to zero at
location RESULT
        END                                     ;End of program
```

Output: R5=0x88888888, R6=0xAAAAAAA9

Output in memory location : 0x40000000 = A9 AA AA AA 88 88 88 88

Experiment No 2.

Aim: Write an assembly language program to find average of N 32-bit numbers.

```
        AREA RESET, CODE
        ENTRY

        MOV R3, #0
        MOV R4, #0
        LDR    R0, =INPUTS
        LDR    R1, =OUTPUTS
CONT
        LDR    R2, [R0]              ; Load R2 with first element from INPUTS pointed by R0
        ADD R4, R4, R2               ; R4 hold cumulative sum
        ADD R0, R0, #4               ; Make R0 to point next element
        ADD R3, R3, #1               ; increment counter (R3) by 1
        CMP  R3, #5                  ; Compare R3 to Number of elements
        BNE CONT                     ; Branch if not equal to CONT
        MOV    R2, #5                ; Set R2 to number of elements (Divisor)
        MOV    R3, #0                ; R3 = 0 (quotient)
REPT  SUBS    R4, R4, R2             ; Subtract Dividend(R4) - Divisor(r2) with status (N - flag)
        ADDPL R3, R3, #1             ; Add 1 to R3(quotient) if N flag is zero ( if subtration result was +ve)
        BPL REPT                     ; Branch to REPT if N = 0
        ADDMI R4, R4, R2             ; Add R2(dividend) to R4 if N flag is one ( if subtration result was –
                                       ve) (R4 = Remainder)

        STR    R3, [R1]              ; Put the quotient to data memory pointed by R1 (OUTPUTS)
        STR R4, [R1, #4]             ; Put the Remainder to data memory pointed by [R1 + 4] (OUTPUTS)
STOP  B  STOP

INPUTS    DCD    01,02,03,04,06
        AREA  MEMORY, DATA
OUTPUTS  SPACE  4
        END
```

Output : Quotient R3=0x00000003, Remainder R4=0x00000001

Output in Memory location : 0x40000000 = 03 00 00 00 01 00 00 00

Experiment No 3.

Aim: Write an assembly language program to find number of occurrences of a number in a given List, using linear search method.

```
        AREA RESET, CODE
        ENTRY
         LDR R0,=ARR              ; R0 set to point memory location ARR
         MOV R1, #0               ; R1 set as Loop Iterator
         MOV R7, #0               ; R7 will hold Number of occurences of key in the list ARR
        MOV R4, #4                ; key

  LOP   CMP R1, #10               ; compare R1 with number of elements
        BEQ EXT                   ; Branch to EXT if equal
        LDR R3, [R0]              ; Load R3 with element from ARR pointed by R0
        CMP R3, R4                ; compare key element (R4) with R3 (elements from ARR)
        BEQ MOVE1                 ; Branch if equal to MOVE1
        B CNT                     ; else Branch to CNT

MOVE1   ADD R7, R7, #1    ; Increment the occurence counter by 1
         B  CNT                   ; Branch to CNT

   CNT  ADD R0, R0, #4              ; Set the pointer R0 to next element in ARR
        ADD R1, R1, #1              ; Increment the loop iterator (R1) by 1
        B LOP                       ; Branch to LOP
EXT                       ; Exit label
STOP B STOP
ARR DCD 4,-5,-1,4,4,0,4,-8,7,4
        END
```

Output : R7 = 0x00000005

Experiment No 4.

Aim: Write an assembly language program to count number of ones(1s) in a given 32-bit binary number.

```
        AREA  RESET,CODE
        ENTRY
        LDR R0,=0xAAAAAABB          ;Load the Number in R0
        MOV R1,#0                   ;Clear R1 to Zero
LOOP
        MOVS R0, R0, LSR #1         ;Right Shift R0 by 1 and move with status new value in R0
        ADDCS R1, R1, #1            ;Add 1 to R1 if carry flag was set
        CMP R0, #0                  ;Check if R0 has become zero
        BNE LOOP                    ;If R0 not Zero branch back to LOOP
EXIT   B  EXIT
        END
```

Output : R1=0x00000012.

Experiment No 5.

Aim: Write an assembly language program to find factorial of a given 32-bit number using procedure.

```
        AREA RESET,CODE
        ENTRY
        LDR R0,=VALUE           ;Set R0 to point location value
        BL FACT                 ;Call subroutine FACT
        LDR R1,=0X40000000      ;Load R1 with RAM address
        STR R5,[R1]             ;Store the contents of R5 to RAM
STOP B STOP
FACT                            ;FACT subroutine
        LDR R2,[R0]             ;Load R2 (Number) with contents from RAM location pointed by R0
        CMP R2,#00              ;Compare the Number with zero
        BEQ END1                ;If equal to zero goto END1
        MOV R3,R2               ;Save the number in R3 from R2
LOOP
        SUB R2,R2,#01           ;Decrement R2 by 1
        CMP R2,#00              ;Compare R2 with Zero
        MULNE R3,R2,R3          ;Multiply R2 and R3 and store the result in R3
        BNE LOOP                ;Branch on not equal to LOOP
        MOV R5,R3               ;Store the factorial in R5
        B END2                      ;Unconditionalbrancj to END2
END1
        MOV R5,#0X01            ;Factorial zero = 1
END2
        BX R14                      ;Return to main program.
        AREA IO,DATA
VALUE SPACE 4
        END
```

If input number is 5

Output : R5=0x00000078

Output in memory location : 0x40000000 = 78

PART-B Hardware Part

1. INTRODUCTION

The 2in 1 IoT Development Kit is designed to be a comprehensive tool to accelerate learning IoT Fundamentals and apply the concepts to get a Hands-On experience by performing different experiments. It can also be used by students and faculty alike to implement projects based on IoT concepts. Flexibility is at the heart of the Kit, and accordingly the students and faculty can not only use the hardware on the board in imaginative ways, but also innovatively write programs to try their own use cases.

This user manual provides the descriptions of the features and also examples of use cases, to guide the students to use it effectively.

The 2 in 1 IoT Development Kit can have an Arduino UNO R3 or a Raspberry Pi 3 B+ controllers, based on the order. In cases, where it was ordered with both, it can be used with one controller at a time. The kit has both open source micro-controllers. It helps the students to perform IoT related experiments based on either Arduino or Raspberry pi 3 B+. The kit has on board sensors which help the students to develop number of applications on IoT. As the kit has on board ESP8266 Wi-Fi module the students can send the different sensors information to the cloud. Using Raspberry Pi 3 B+ one can login to the kit from a remote location using SSH tools. One can flash code into the Arduino using Arduino IDE. The Raspberry Pi 3 B+ works on Raspbian OS which is installed on to the SD Card. Using Arduino IDE one can write the program in Embedded C Language. For Raspberry Pi 3 B+ one can write the programs in different Languages like Python, Embedded C, Node JS, Java, etc.

1. IoT DEVELOPMENT KIT SPECIFICATIONS

Fig. 2: 1-in-1 Iot Development Kit.

Hardware Description:

1. 16×2 LCD Display
2. Humidity sensor
3. Light sensor
4. Ultrasonic sensor
5. Gas sensor
6. Soil Moisture sensor
7. Bluetooth
8. Raspberry Pi 3 B+
9. 7 – Segment display
10. Power Supply
11. UART
12. Relay
13. Buzzer
14. Stepper Motor Interface
15. 4*4 Matrix Keypad
16. LED and Switch
17. WiFi ESP8266
18. ADC MCP3008
19. RTC
20. Arduino Uno

Specifications:

- Board is driven using either Raspberry Pi 3 B+ or Arduino UNO R3 controllers.
- Facilitates MQTT or HTTP application protocol connection to various IoT cloud platforms such as ThingSpeak, AWS IoT, Microsoft Azure, IBM Bluemix, etc.
- It can be connected to a Private Cloud including X trans Cloud.
- On board voltage regulator for generating 3.3V from +5V input through power supply /adapter.
- On board Wi-Fi connectivity through ESP8266 for Arduino and internal Wi-Fi connectivity for Raspberry Pi 3 B+.
- On board digital and analog sensors like Temperature & Humidity, Gas, Light, Ultrasonic distance sensor and Soil moisture sensor.
- On board ADC MCP3008 for Raspberry Pi 3 B+ to connect analog sensors and internal ADC for Arduino.
- On board 16×2 character LCD.
- On board 4×4 keypad matrixes.
- On board 32.768 kHz quartz crystal RTC DS1307 powered by coin cell battery of 3V.
- 4 LED's and 4 Switches.
- 2-digit 7 segment display.
- On board stepper motor driver to drive 5V stepper motor.
- One Buzzer.
- Connector slot for external Bluetooth module for Arduino.
- Provided extra connectors for connecting external analog and digital sensors.
- I2C, SPI communication protocols.
- On board UART connector.
- UART Software serial communication.
- On board Relay to drive external device.
- Several of the outputs to the on-board controller can be accomplished through events from cloud such as turning on LED’s or sending a test message to the display is possible.
- Outputs can be programmed to be event or threshold driven.
- Performance of IoT experiments and design / implementation of student projects possible including provision to write their own programs.

3. HARDWARE DETAILS

Arduino UNO R3

Arduino is an open-source hardware, software and content platform with a global community.

Specifications:

- Micro-controller: ATmega328.
- Operating Voltage: 5V.
- Input Voltage (recommended): 7-12V.
- Input Voltage (limits): 6-20V.
- Digital I/O Pins: 14 (of which 6 provide PWM output).
- Analog Input Pins: 6.
- DC Current per I/O Pin: 40 mA.
- DC Current for 3.3V Pin: 50 mA.
- Flash Memory: 32 KB of which 0.5 KB used by boot-loader.
- SRAM: 2 KB (ATmega328).
- EEPROM: 1 KB (ATmega328).
- Clock Speed: 16 MHz

Fig. 3.1: Arduino UNO R3 Board.

Raspberry Pi 3 B+

Specifications:

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU.
- 1GB RAM.
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board.
- 100 mbps Base Ethernet.
- 40-pin extended GPIO.
- 4 USB 2.0 ports.
- 4 Pole stereo output and composite video port.
- Full size HDMI.
- CSI camera port for connecting a Raspberry Pi 3 B+ camera.
- DSI display port for connecting a Raspberry Pi 3 B+ touchscreen display.
- Micro SD port for loading your operating system and storing data.
- Upgraded switched Micro USB power source up to 2.5A.

Fig. 3.2: Raspberry Pi 3 B+.

4. INSTALLATION PROCEDURE

Install the Arduino Software (IDE) on Windows PCs

Get the latest version from https://www.arduino.cc/en/Main/Software. You can choose between the Installer (.exe) and the Zip packages.

When the download finishes, proceed with the installation and please allow the driver installation process when you get a warning from the operating system.

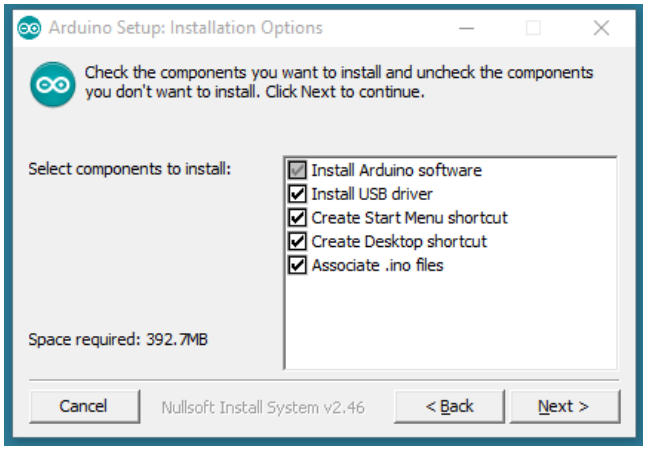


Fig. 4.1: Arduino initial setup.

Choose the components to install.

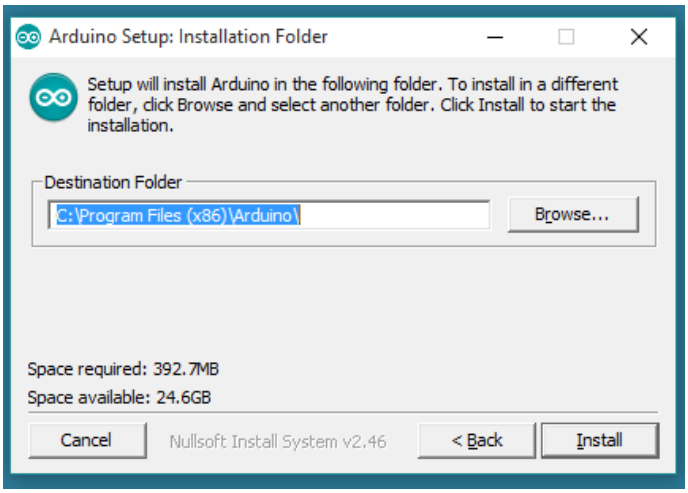


Fig. 4.2: Select folder Arduino installation.

Choose the installation directory (we suggest to keep the default one)

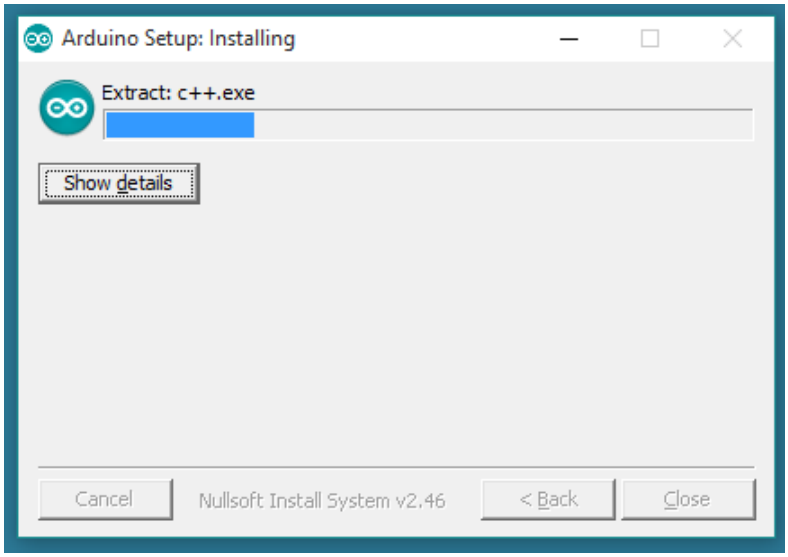


Fig. 4.3: Arduino IDE installing

The process will extract and install all the required files to execute properly the Arduino Software (IDE)

Using Arduino Uno with Arduino Desktop IDE

Connect Uno board with an A B USB cable; sometimes this cable is called a USB printer cable.

Fig. 4.4: A B USB Cable.

Install the Board Drivers

If you used the Installer, Windows – from XP up to 10 – will install drivers automatically as soon as you connect your board.

Troubleshooting:

If you downloaded and expanded the Zip package or, for some reason, the board wasn't properly recognized, please follow the procedure below.

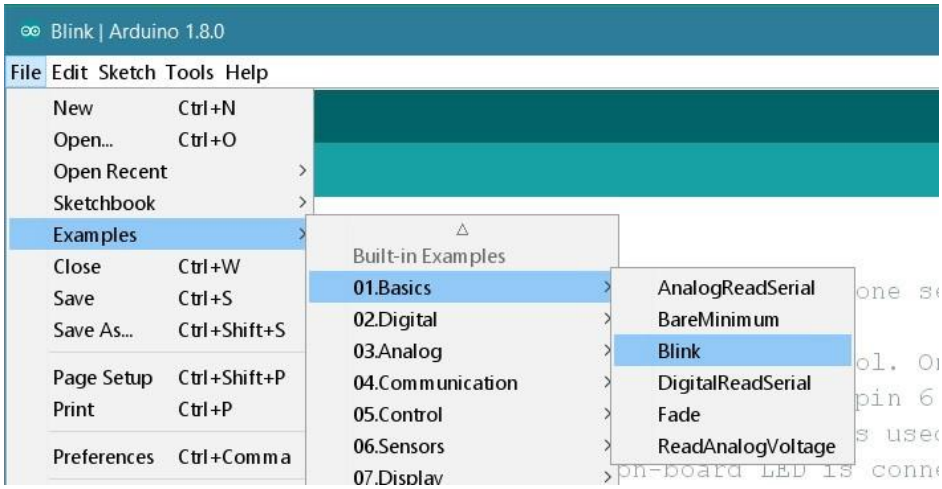
- Click on the Start Menu, and open up the Control Panel.
- While in the Control Panel, navigate to System and Security. Next, click on System. Once the System window is up, open the Device Manager.
- Look under Ports (COM & LPT). You should see an open port named "Arduino UNO (COMxx)". If there is no COM & LPT section, look under "Other Devices" for "Unknown Device".

- Right click on the "Arduino UNO (COMxx)" port and choose the "Update Driver Software" option.
- Next, choose the "Browse my computer for Driver software" option.
- Finally, navigate to and select the driver file named "arduino.inf", located in the "Drivers" folder of the Arduino Software download (not the "FTDI USB Drivers" sub- directory). If you are using an old version of the IDE (1.0.3 or older), choose the Uno driver file named "Arduino UNO.inf"
- Windows will finish up the driver installation from there.

Getting Started with Sketch

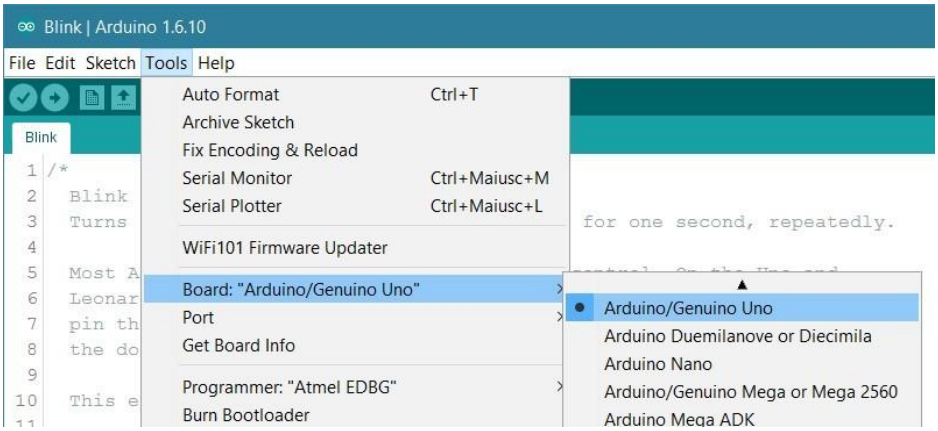
Open the LED blink example sketch: File > Examples >01.Basics > Blink.

Fig. 4.5: Arduino IDE Examples



Select the Board Type and Port

You'll need to select the entry in the Tools > Board menu that corresponds to your Arduino or Genuino board.



Select the serial device of the board from the Tools | Serial Port menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports).

To find out, you can disconnect your board and re-open the menu; the entry that disappears should be the Arduino or Genuino board. Reconnect the board and select that serial port.

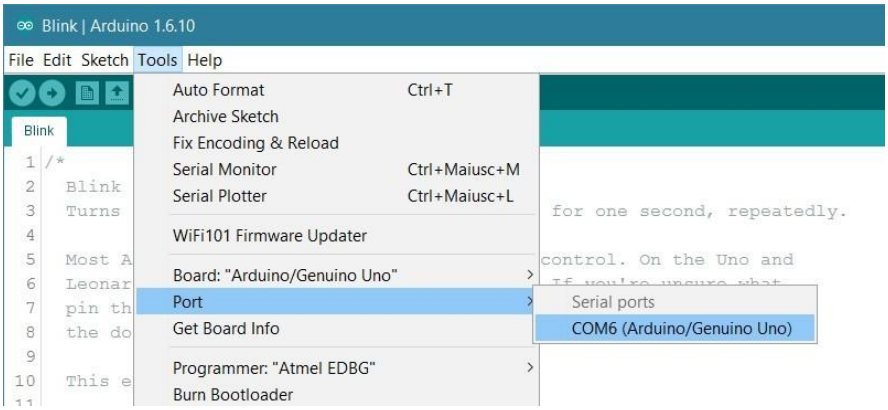


Fig. 4.7: Arduino IDE Select Port

Upload the Program

Now, simply click the "Upload" button in the environment. Wait for few seconds - you should see the RX and TX LEDs on the board flashing. If the upload is successful, the message "Done uploading." will appear in the status bar.

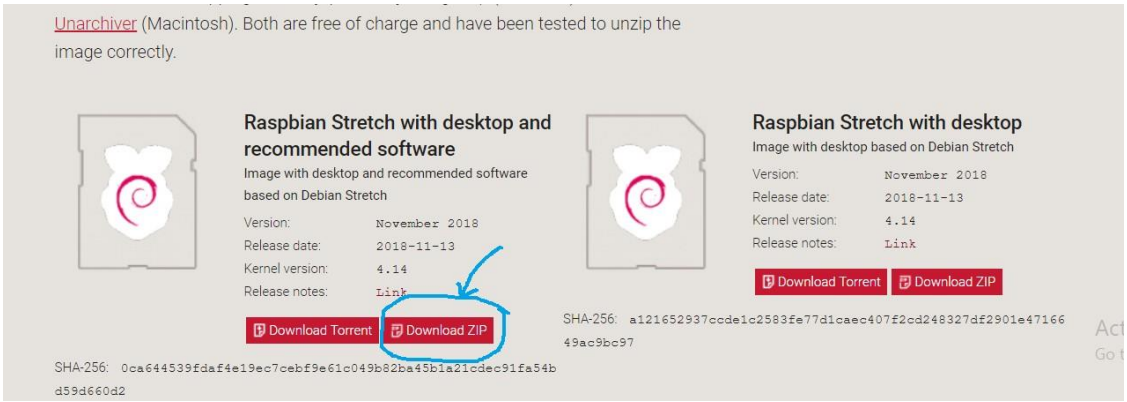
Fig: 4.8 Arduino IDE Upload Project

A few seconds after the upload finishes, you should see the pin 13 LED on the board starts blinking (in orange). If it does, congratulations!

Installing Raspbian OS for Raspberry Pi 3 B+

Using Raspbian Stretch OS Image Downloaded in. zip format

1. Download the image file in the form Zip file from the following link https://www.raspberrypi.org/downloads/raspbian/



2. Download the win32 Disk Imager Software to write the image file to SD Card from the following link. https://sourceforge.net/projects/win32diskimager/
3. Now insert the SD Card in SD card slot and connect it to PC/Laptop.
4. Open Win32 Disk Imager, choose the .img or image file you want to write as Image File and choose the SD drive as Device and press Write.
5. The write may take a while. Once it is done, remove the SD card and insert it into the device you want to use with.

Initial Setup for Raspberry Pi 3 B+

1. Once Raspberry Pi 3 B+ is booted up to GUI, Connect the keyboard and Mouse to USB slots.
2. Change the following
   a. language: English
   b. Country and Time Zone: India and Kolkata
3. From menu-->Preferences-->Raspberry Pi 3 B+
   Configuration. Then Go To Interfaces
   Enable Camera, SSH, VNC, SPI, I2C, Serial Port. Don't Enable Remaining Things.
4. Connect to WiFi.
5. Check the date and Time.
6. Open the Terminal.
   Execute following default commands.
   a. sudo apt-get update
   b. sudo apt-get upgrade
7. Now you can write your programs at Thonny Python
   IDE. Menu → Programming → Thonny Python
   IDE.
   File → New ….. You can write your Program here and save.
8. Execute the program at terminal only by following command
   sudo python <program-name>.py

Experiment No 1.

Aim: Write a program to display the text "REVA UNIVERSITY" on LCD display using Microcontroller board.

Hardware Details: LCD Display

A Liquid Crystal Display is an electronic display module. It displays 16 characters per line in 2 such lines. Each character in LCD displays 5×7pixel matrix. Here is the LCD is interfaced using 4 – bit mode.

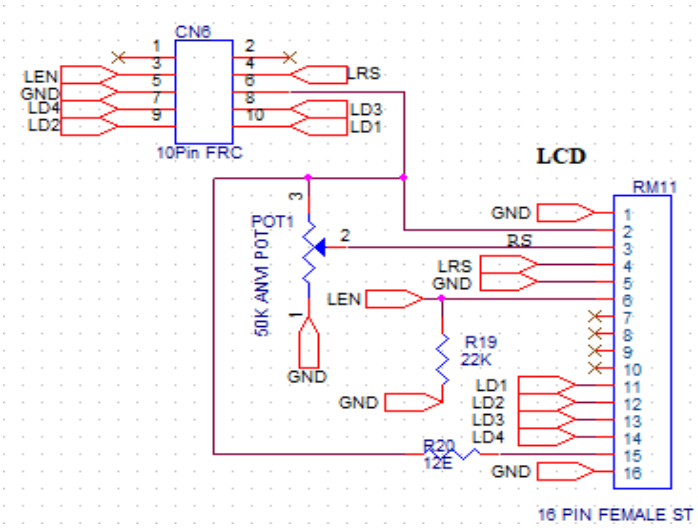


Fig.: Schematic pin diagram of LCD Interface.

Pin	Function	Name
1	Ground (0V)	Ground
2	Supply voltage; 5V (4.7V – 5.3V)	Vcc
3	Contrast adjustment; the best way is to use a variable resistor.	Vo / VEE
4	Selects command register when low, and data register when high	RS (Register Select)
5	Low to write to the register; High to read from the register	Read/write
6	Sends data to data pins when a high to low pulse is given;	Enable
7-14	8-bit data pins	DB0-DB7
15	Back-light VCC (5V)	LED+
16	Back-light Ground (0V)	LED-

Arduino Program

```
/*
 LiquidCrystal LCD - The circuit connections to the controller:
 * LCD RS pin to digital pin 16
 * LCD Enable pin to digital pin 17
 * LCD D4 pin to digital pin 13
 * LCD D5 pin to digital pin 12
 * LCD D6 pin to digital pin 11
 * LCD D7 pin to digital pin 10
 * Connect 10 pin frc cable from CN3 to CN6 for LCD interface

*/

#include <LiquidCrystal.h>      // Arduino LCD library

// initialize the library by associating LCD interface pin with the arduino pin number it is connected to

const int rs = 16, en = 17, d4 = 13, d5 = 12, d6 = 11, d7 = 10;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
  lcd.begin(16, 2);         // set up the LCD's number of columns and rows:
}

void loop() {
  lcd.setCursor(0,0);                          //Position the cursor to 0 column , 0 row
  lcd.print(" REVA UNIVERSITY ");     //Print message on the LCD
  lcd.setCursor(0,1);                          //Position the cursor to 0 column, 1 row
  lcd.print(" BENGALURU     ");           //Print message on the LCD
  delay(3000);                                 //Delay for 3 seconds

  lcd.clear();

  lcd.setCursor(0,0);                          //Position the cursor to 0 column , 0 row
  lcd.print("DEPT.OF COMPUTER");      //Print message on the LCD
  lcd.setCursor(0,1);                          //Position the cursor to 0 column, 1 row
  lcd.print(" SCIENCE & ENGG ");         //Print message on the LCD
  delay(3000);                                 //Delay for 3 seconds
}
```

Experiment No 2.

Aim: Develop a program to demonstrate UP/DOWN counter from 00 to99 on 7-segment display using microcontroller board.

Hardware Details: Seven Segment Display

A 7-segment display is used for displaying decimal numerals. The seven-segment display consists of 7 LED's, so called 7-segment display. 7-segment display is manufactured in a rectangular fashion.

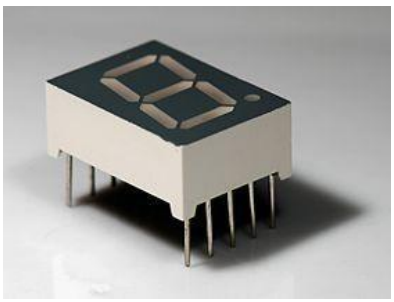


Fig. 7-Segment Display (reference part)

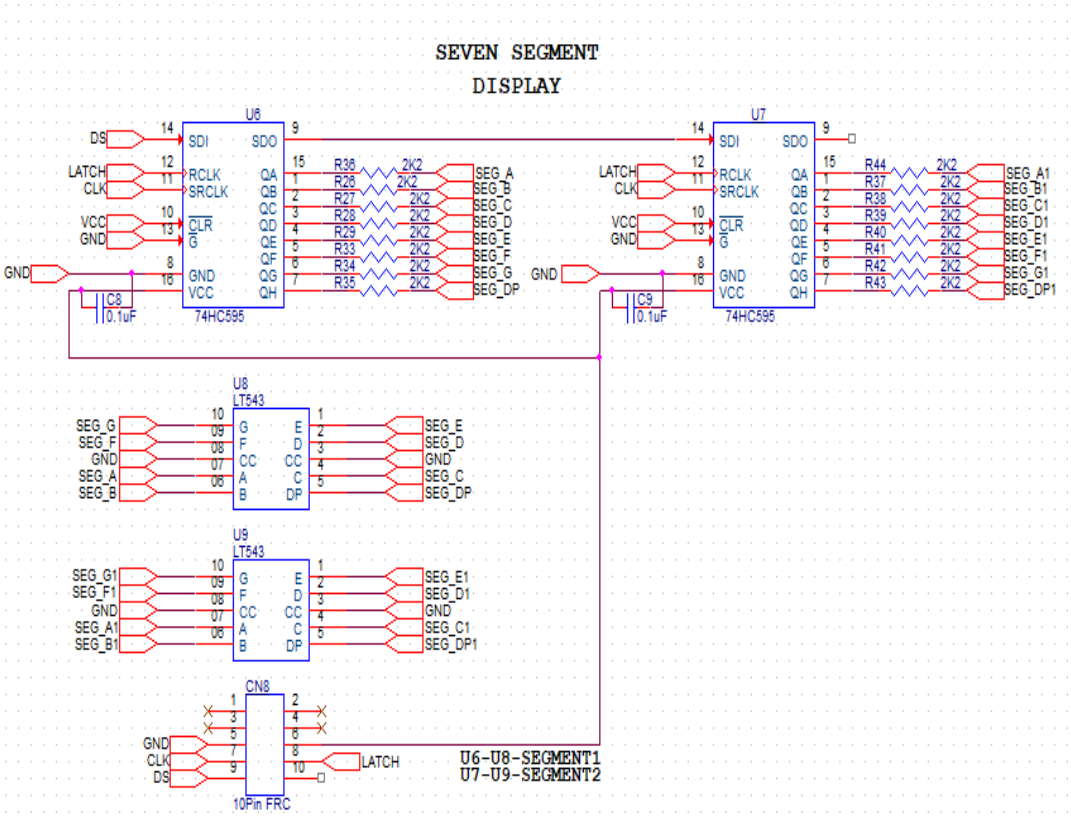


Fig. : Schematic pin diagram of 7-Segment Display.

74HC595 Shift Register

For saving the pin number for controlling a 7-segment display, a shift register is used as a serial-to-parallel converter to send signals to the display. That is, we serially send 8 bits of data, which represents the way we want to turn on the display, by one signal pin into the shift register and the register can output the corresponding data pattern to its 8 output pinsat once (parallel). Signal pins we need to take care about when using a 74HC595.

Input Pins

- 1 input data pin (DS)
- 1 clock pin for input data (SHCP)
- 1 clock pin for output data (STCP)

Output Pins

- 8 data pins (Q0 ~ Q7)

Control Pins

- 1 output enable pin (OE)
- 1 data reset pin (MR)

Arduino Program

```
// CREATE SHIFT REGISTER OBJECT (NUMBER OF SHIFT REGISTERS, DATA, CLOCK, LATCH)
// CONNECT CN3 to CN8
#include <ShiftRegister74HC595.h>
#define DATA 12
#define CLOCK 10
#define LATCH 11
ShiftRegister74HC595 sr (2, 12,10,11);
uint8_t numberB[] = {B00111111,        //0
```

```
            B00000110,      //1
            B01011011,      //2
            B01001111,     //3
            B01100110,      //4
            B01101101,      //5
            B01111101,     //6
            B00000111,     //7
            B01111111,     //8
            B01101111       //9
           };

void setup()
{

}

void loop()
{
 for(int i=0;i<10;i++)
 {
  for (int j = 0; j < 10; j++)
  {
    uint8_t pinValues[] = {numberB[i],numberB[j]};
    sr.setAll(pinValues);
    delay(500);
  }
 }
}
```

Experiment No 3.

Aim: Design a program to interface and control the stepper motor to rotate in the specified direction using microcontroller board.

Hardware Details: Stepper Motor

Stepping motor is a brush-less synchronous electric motor that converts digital pulses into mechanical shaft rotation. The motor's position can then be commanded to move and hold at one of these steps without any position sensor for feedback (an open-loop controller), as long as the motor is carefully sized to the application in respect to torque and speed.

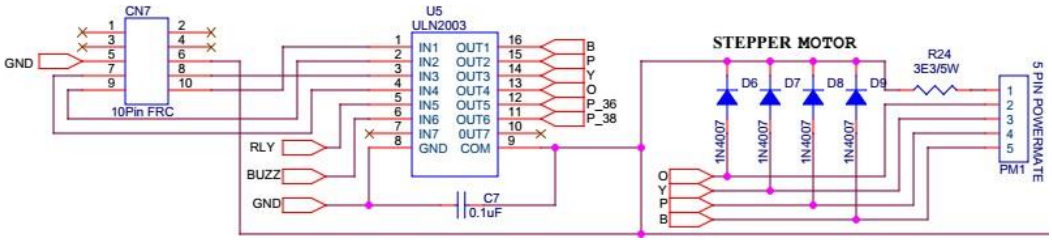


Fig.: Schematic Pin Diagram of Stepper Motor Interface.

Arduino Program

```
// Connect CN7-CN9 on the kit for Stepper Motor

#include<Stepper.h>
int stepsPerRevolution=200;
int motorspeed=10;
Stepper myStepper(stepsPerRevolution, 10,12,11,13);


void setup(){
  Serial.begin(9600);
  myStepper.setSpeed(motorspeed);
}
void loop(){
  myStepper.step(stepsPerRevolution);
  delay(500);
  myStepper.step(-stepsPerRevolution);


  delay(500);
}
```

Experiment No 4.

Aim: Write a program to interface 4X4 keypad with the microcontroller.

Hardware Details: 4x4 Keypad Matrix

The Purpose of having the 4*4 Keypad Matrix is to reduce the number of input pins. The order of matrix increases with the decreases of the number of inputs.

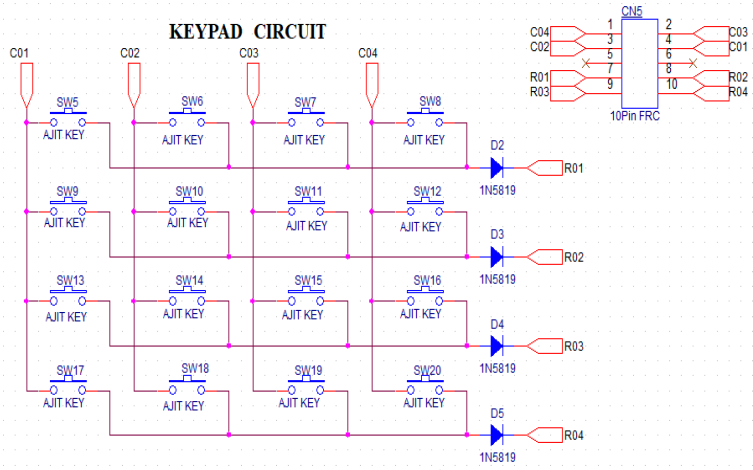


Fig. : Schematic Pin Diagram of 4*4 Keypad Matrix.

Arduino Program

```
/*
*  Example testing sketch to Display keypad values on Seven Segment display
*  By pressing the key on key on keypad same value is displayed on both Seven segment displays
* Connect CN10 - CN5  //Connect the 4*4 keypad
* Connect CN3  - CN8  //Connect 7 Segment Display
*/
#include <ShiftRegister74HC595.h>
#include <Keypad.h>
const byte numRows= 4; //number of rows on the keypad
const byte numCols= 4; //number of columns on the keypad

//keymap defines the key pressed according to the row and columns just as appears on the keypad
char keymap[numRows][numCols]=
```

```
{
{'0', '4', '8', 'c'},
{'1', '5', '9', 'd'},
{'2', '6', 'a', 'e'},
{'3', '7', 'b', 'f'}
};

char hexvalues[6] = {B01011111, B01111100,B00111001,B01011110, B01111001,B01110001};
//Seve values from Characters 'a' to 'f'

ShiftRegister74HC595 sr (2,12,10,11);
uint8_t  numberB[] = {B00111111,    //0
                      B00000110,      //1
                      B01011011,      //2
                      B01001111,      //3
                      B01100110,      //4
                      B01101101,      //5
                      B01111101,       //6
                      B00000111,       //7
                      B01111111,      //8
                      B01101111       //9
                     };
byte colPins[numRows] = {2,3,4,5}; //Rows 0 to 3
byte rowPins[numCols]= {6,7,8,9}; //Columns 0 to 3

Keypad myKeypad= Keypad(makeKeymap(keymap), rowPins, colPins, numRows, numCols);

      void setup()
      {
        Serial.begin(9600);
      }
```

```
void loop()
  {
       char keypressed = myKeypad.getKey();
       Serial.println("keypressed : "+String(keypressed));
       if (keypressed != NO_KEY)
        {
          Serial.print(keypressed);
          int i = keypressed - '0';
          if(i<10)
          {
          uint8_t pinValues[] = {numberB[i],numberB[i]};
          sr.setAll(pinValues);
          delay(100);
          }
          else
          {
          i = keypressed - 'a';
          uint8_t pinValues[] = {hexvalues[i],hexvalues[i]};
          sr.setAll(pinValues);
          delay(100);
          }
        }
  }
```

Experiment No 5.

Aim: Control on/off status of LED using switches with microcontroller.

Hardware Details: Switches and LED's

Switch and LED's are combination of 4 LED's and 4 switches which are programmed to operate LED's using the switches.

Fig.: Schematic Pin Diagram of Switches and LED's.

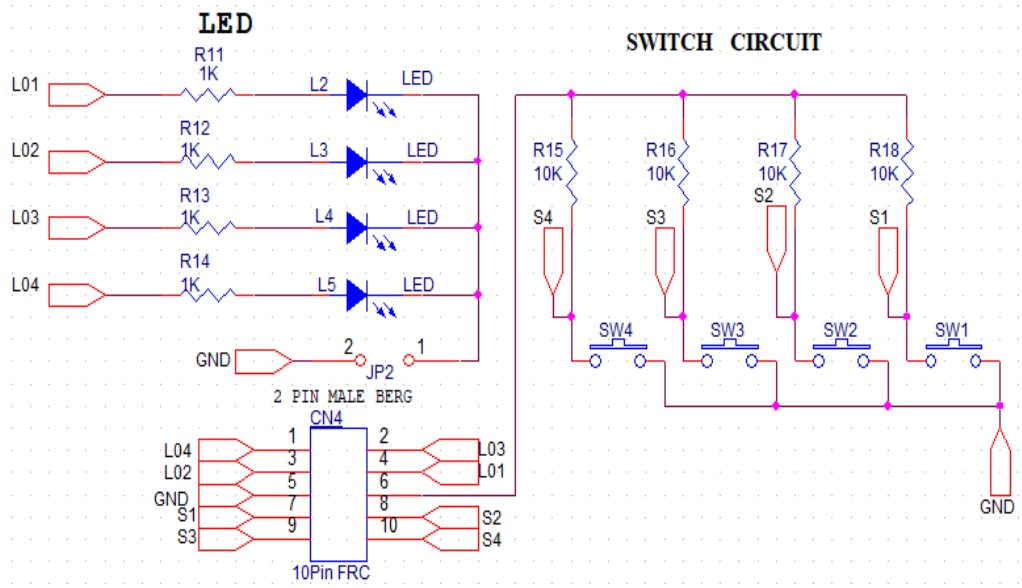


Fig.: Schematic Pin Diagram of Switches and LED's.

Note: In the program given below, we programmed two LED's and two switch.

Arduino Program

```
//Switches and LEDs
//Connect CN4 to CN9 for LED

const int buttonPin1 = 11;          // number of pushbutton 3 pin
int buttonState1 = LOW;             // set the default variable value for pushbutton3 status
const int ledPin1 =  A3;            // number of the LED 3 pin


const int buttonPin2 = 10;          // number of pushbutton 4 pin
int buttonState2 = LOW;             // set the default variable value for pushbutton4 status
const int ledPin2 =  A2;            // number of the LED 4 pin
```

```
void setup() {                          // Set Pins to Outputs Or Inputs
  pinMode(buttonPin1, INPUT);          // initialize the pushbutton pins as an inputs:
  pinMode(ledPin1, OUTPUT);            // initialize the LED pins as an outputs:
  pinMode(buttonPin2, INPUT);          // initialize the pushbutton pins as an inputs:
  pinMode(ledPin2, OUTPUT);            // initialize the LED pins as an outputs:
  Serial.begin(9600);                  // initialize serial communication at 9600 baud
}

void loop() {

  buttonState1 = digitalRead(buttonPin1);      // read current states of the pushbutton value:
  buttonState2 = digitalRead(buttonPin2);      // read current states of the pushbutton value:

// check if the pushbutton is pressed buttonState# == HIGH/LOW
  // if pressed change buttonState == HIGH to turn on ledPin#
  // else if buttonState == LOW then digitalWrite(ledPin#, LOW) Keeps Led off.


if (buttonState1 == HIGH) {                //check buttonState
digitalWrite(ledPin1, LOW);                //if HIGH turn LED on:
}
else {
digitalWrite(ledPin1, HIGH);               // turn LED off:
delay(10);
Serial.println(buttonState1);              //Print buttonState to serial
}

if (buttonState2 == HIGH) {                //check buttonState
digitalWrite(ledPin2, LOW);                //if HIGH turn LED on:
}
else {
digitalWrite(ledPin2, HIGH);               // turn LED off:
delay(10);
Serial.println(buttonState2);              //Print buttonState to serial
}
}
```

Experiment No 6.

Aim: Determine the rise in temperature using temperature sensor and microcontroller board.

Hardware Details: Temperature and Humidity Sensor

The DHT11 is a digital temperature and humidity sensor. It uses a humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin. It's fairly simple to use, but requires careful timing to grab data. You can only get new data from it once every 2 seconds, so when using our library, sensor readings can be up to 2 seconds old.

Specifications:

Operating Voltage: 3.5V to 5.5V.
Operating current: 0.3mA (measuring) 60uA (standby)
Output: Serial data. Temperature
Range: 0°C to 50°C. Humidity
Range: 20% to 90%
Resolution: Temperature and Humidity both are 16-bit. Accuracy: ±1°C and ±1%

Fig. : Schematic Pin Diagram of Temperature and Humidity Sensor

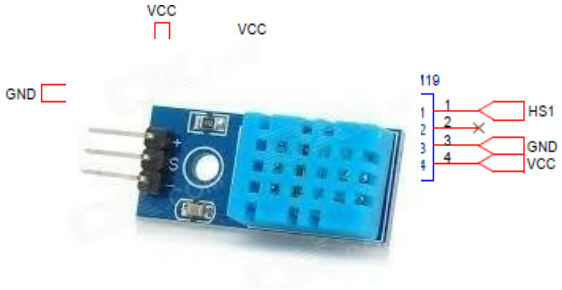


Fig.: Temperature and Humidity Sensor.

Arduino Program

```
/*
  * LCD RS pin to digital pin 16
 * LCD Enable pin to digital pin 17
 * LCD D4 pin to digital pin 13
 * LCD D5 pin to digital pin 12
 * LCD D6 pin to digital pin 11
```

```
 * LCD D7 pin to digital pin 10
 * Connect 10 pin frc cable from CN3 to CN6 for LCD interface
 * Connect RM2 to RM19 for temperature sensor
*/

#include <LiquidCrystal.h>          // Arduino LCD library
#include <dht.h>                    // Arduino Temperature and Humidity Sensor library

#define DHT11_PIN 4                 //define temperature sensor pin

dht DHT;                            //Create a dht variable

// initialize the library by associating LCD interface pin with the arduino pin number it is connected to

const int rs = 16, en = 17, d4 = 13, d5 = 12, d6 = 11, d7 = 10;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
  lcd.begin(16, 2);                 // set up the LCD's number of columns and rows:
}

void loop() {
  lcd.setCursor(0,0);                       //Position the cursor to 0 column , 0 row
  lcd.print(" WELCOME TO REVA ");           //Print message on the LCD
  lcd.setCursor(0,1);                       //Position the cursor to 0 column, 1 row
  lcd.print(" BENGALURU - 64 ");            //Print message on the LCD
  delay(3000);                              //Delay for 3 seconds
  DHT.read11(DHT11_PIN);                    //Read the temperature sensor
  lcd.clear();                              //Clear the LCD
  lcd.setCursor(0,0);                       //Position the cursor to 0 column , 0 row
  lcd.print("Temp = ");
  lcd.print(DHT.temperature);               //Display Temperature value
  lcd.print(" C");
  lcd.setCursor(0,1);                       //Position the cursor to 0 column, 1 row
  lcd.print("Humid = ");
  lcd.print(DHT.humidity);                  //Display Humidity value
  lcd.print("%");
  delay(3000);                              //Delay for 3 seconds
}
```

Experiment No 7.

Aim: Determine the leakage of gas using microcontroller board.

Hardware Details: Gas Sensor

Gas Sensors (MQ2) are useful to detect the gas leakage (home and industry). It is suitable for detecting the Smoke, CO, H2, LPG, CH4, Alcohol. Due to fast response time and its high sensitivity, measurement can be taken as soon as possible. Potentiometer is used to adjust the sensitivity of the sensor.

Fig.: Schematic Pin Diagram of GAS Sensor.

Fig.: GAS Sensor

Specifications:

- Operating Voltage is +5V.
- Can be used to Measure or detect LPG, Alcohol, Propane, Hydrogen, CO and even methane.
- Analog output voltage: 0V to 5V.
- Digital Output Voltage: 0V or 5V (TTL Logic).
- Preheat duration 20 seconds.
- Can be used as a Digital or analog sensor.
- The Sensitivity of Digital pin can be varied using the potentiometer.

Stepper Motor

Stepping motor is a brush-less synchronous electric motor that converts digital pulses into mechanical shaft rotation. The motor's position can then be commanded to move and hold at one of these steps without any position sensor for feedback (an open-loop controller), as long as the motor is carefully sized to the application in respect to torque and speed.

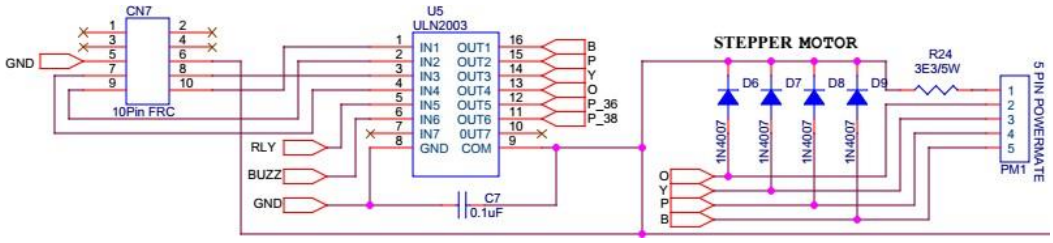


Fig.: Schematic Pin Diagram of Stepper Motor Interface.

```
//Gas sensor and Stepper motor
// Connect RM5 - RM22 on the kit for Gas Sensor
// Connect CN7-CN9 on the kit for Stepper Motor

#include <AccelStepper.h>              //Arduino Stepper Motor library
#define MotorInterfaceType 4          //A 4 wire stepper motor

// Motor pin definitions
#define motorPin1 13     // IN1 on the ULN2003 driver 1
#define motorPin2 12    // IN2 on the ULN2003 driver 1
#define motorPin3 11     // IN3 on the ULN2003 driver 1
#define motorPin4 10    // IN4 on the ULN2003 driver 1

//Define Gas Pin
int gas_pin = A0;                 // Arduino pin A0

// Initialize with pin sequence IN1-IN3-IN2-IN4 for using the AccelStepper with 28BYJ-48
AccelStepper stepper( MotorInterfaceType, motorPin1, motorPin3, motorPin2, motorPin4);

void setup() {
  pinMode(gas_pin, INPUT);             //Set the Gas senssor to input mode;
  stepper.setMaxSpeed(1000.0);         //Set the maximum speed for the motor
  Serial.begin(9600);                  // Set the serial monitor baud rate
}

void loop() {
  int gas_value = analogRead(gas_pin);                    // read the the gas sensor value
  int motorSpeed = map(gas_value, 0, 1023, 0, 1000);      // map the gas value range to motor speed
  if (gas_value >= 500){                                   // gas threshold value
     stepper.run();                                       //start the motor
     if (motorSpeed > 0)
       stepper.setSpeed(motorSpeed);                       // set the motor speed
  }
  else
     stepper.stop();                                       //stop the motor
  Serial.print("Gas Value : ");                            // display the gas value on serial monitor
  Serial.println(gas_value);
}
```