SETHU INSTITUTE OF TECHNOLOGY

(An Autonomous Institution | Accredited with 'A' Grade by NAAC)

PULLOOR, KARIAPATTI - 626 115

DEPARTMENT OF INFORMATION TECHNOLOGY

LAB RECORD

SEMESTER: VII

19UIT702 – INTERNET OF THINGS SYSTEMS AND APPLICATIONS (INTEGRATED COURSE)

Prepared By,

Dr.V.Akilandeswari, ASP/IT

Approved By,

Dr.M. Poomani@Punitha,HOD/IT

SETHU INSTITUTE OF TECHNOLOGY

PULLOOR-626 115, KARIAPATTI TALUK.VIRUDHUNAGAR DISTRICT

PRACTICAL RECORD

BONAFIDE CERTIFICATE

Certified that this the bonafiderecord of work done by

.....

in the .....

Laboratory during the year.....

Staff In-charge

Head of the Department

Reg.No:

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

Internal Examiner

External Examiner

INSTITUTE VISION

- To promote excellence in technical education and scientific research for the benefit of the society.

INSTITUTE MISSION

- To provide quality technical education to fulfill the aspiration of the student and to meet the needs of the Industry.
- To provide holistic learning ambience.
- To impart skills leading to employability and entrepreneurship.
- To establish effective linkage with industries.
- To promote Research and Development activities.
- To offer services for the development of society through education and technology.

CORE VALUES

- Quality
- Commitment
- Innovation
- Team work
- Courtesy

QUALITY POLICY

- To provide Quality technical education to the students
- To produce competent professionals and contributing citizens
- To contribute for the upliftment of the society

DEPARTMENT VISION

- To promote excellence in producing competent IT Professionals to serve the society through technology and research

DEPARTMENT MISSION

- Producing competent professionals in information and communication technologies.
- Educating the students with the state of art computing environment and pedagogical innovations
- Encouraging entrepreneurship and imparting skills for employability
- Establishing collaboration with IT and allied industries
- Promoting research in information and communication technology to improve the quality of human life.
- Offering beneficial service to the society by imparting knowledge and providing IT solutions.

Core Values

- Quality
- Responsibility
- Novelty
- Team work
- Courtesy

PROGRAM EDUCATIONAL OBJECTIVES (PEO)

- Exhibit proficiency in analyzing, designing and developing IT based solutions to cater to the needs of the Industry**{Technical Competence}**
- Provide professional expertise to the industry and society with effective communication and ethics**{Professionalism}**
- Engage in lifelong learning for professional development and research **{Life-Long Learning}**

PROGRAM OUTCOMES (PO)

a. Apply the knowledge of Mathematics, Basic Science, Computer and communication Fundamentals to solve complex problems in Information Technology. **[Engineering Knowledge]**

b. Identify, formulate, review research literature and analyze complex problems reaching concrete conclusions using principles of mathematics , Engineering sciences and Information Technology**[Problem Analysis]**

c. Design solution for complex information and communication engineering problems and design system components or processes that meet with realistic constraints for public health and safety, cultural, societal and environment considerations. **[Design/Development of Solutions]**

d. Conduct investigations of complex Information technology related problems using research based knowledge and research methods including design of experiments,analysis and interpretation of data to provide valid conclusions through synthesis of information.**[Conduct investigations of complex problems]**

e. Create, select and apply appropriate techniques, resources and modern IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations. **[Modern Tool Usage]**

f. Apply reasoning informed by contextual knowledge to assess societal, health, safety, legal and cultural issues and consequent responsibilities relevant to professional engineering practice .**[The Engineer and Society]**

g. Understand the impact of professional engineering solutions in societal and environmental contexts and demonstrate the knowledge of and need for sustainable development.**[Environment and sustainability]**

h. Apply ethical principles and commit to professional ethics and responsibilities through the norms of professional engineering practice .**[Ethics]**

i. Function effectively as an individual and as a member or leader in diverse teams and in multidisciplinary settings.**[Individual and Team Work]**

j. Communicate effectively with the engineering community and the society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations and give and receive clear instructions.**[Communication]**

k. Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work, as a member /or leader in a team, to manage projects in multi-disciplinary environment. **[Project Management and Finance]**

l. Recognize the need for, and have the preparation and ability to engage in independent and Life-long learning in broadest context of technological change. **[Life-long Learning]**

PROGRAM SPECIFIC OUTCOMES (PSO)

1. Design software solutions using programming skills and computing technologies.
2. Design and implement data communication system using various IT components.

LIST OF EXPERIMENTS

S. No	Date	LIST OF EXPERIMENTS	MARKS	SIGNATURE
1 a)		Create an IOT enabled environment using Arduino software in microchip studio/Thinker cad/Arduino IDE and perform necessary software installation in embedded platform.		
1 b)		Demonstrate the IOT programming by configuring the buzzer as output device with the Arduino microcontroller and turn it ON and OFF at an interval of 1 second by using the Microchip Studio IDE/ Thinker cad/Arduino IDE.		
2 a)		Implement IOT program to blink LED by selecting the Atmel based microcontroller card in the Arduino Integrated Development Environment/ Thinker cad/Microship studio IDE.		
2 b)		Implement the IOT programming by interfacing the Bar-graph LEDs and the interrupt switch to toggle the status of 2 Bar-graph LEDs depending on whether the interrupt switch is pressed or releases using Microchip Studio IDE/ Thinker cad/Arduino IDE.		
3 a)		Implement the IOT program by representing the same numeral of up to 8-bits in different number systems on the inbuilt LCD and increment that number every 500 milliseconds loop through this infinitely.		
3 b)		Study the Installation process of the operating systems for Raspberry Pi / Beagle board and describe the process of OS installation on Raspberry – Pi/ Beagle board		
4 a)		Implement the IOT program by interfacing the LCD with microcontroller to display the moving and scrolling and static text/string on LCD using		

		ATmega2560 controller in Microchip Studio IDE.		
4 b)		Implement the IOT program by using the LCD of ATmega 2560 in Microchip studio IDE to represent the same numeral of up to 8-bits in different number systems.		
4 c)		Study Connectivity and Configuration of Raspberry-Pi/ Beagle Board circuit with basic peripherals, LEDs, Understanding GPIO and its use in program.		
4 d)		Implement the program to blink five LEDs using Arrays. All the five LEDS will light after one other.		
5		Mini Project		

1. **a) Create an IOT enabled environment using Arduino software in microchip studio/Thinker cad/Arduino IDE and perform necessary software installation in embedded platform.**

Aim

To create an IOT enabled environment using arduino software in microchip/Thinker cad/Arduino IDE and Perform necessary software installation in embedded platform.

Arduino Software

The Arduino IDE (Integrated Development Environment) is used to write the computer code and upload this code to the physical board. The Arduino IDE is very simple and this simplicity is probably one of the main reason Arduino became so popular. We can certainly state that being compatible with the Arduino IDE is now one of the main requirements for a new microcontroller board. Over the years, many useful features have been added to the Arduino IDE and you can now managed third-party libraries and boards from the IDE, and still keep the simplicity of programming the board. The main window of the Arduino IDE is shown below, with the simple simple Blink example.

Procedure

Step 1: Select the version of Arduino IDE from the https://www.arduino.cc/en/software based on your requirement.

Step 2: Download and install the Arduino IDE 2.0 on your Windows

2.1 : To install the Arduino IDE 2.0 on a Windows computer, simply run the file downloaded from the software page.

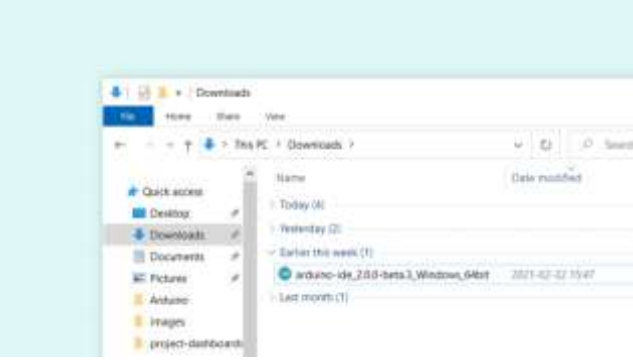


Figure 1.1 Running the Installation file

2.2: Follow the instructions in the installation guide. The installation may take several minutes.

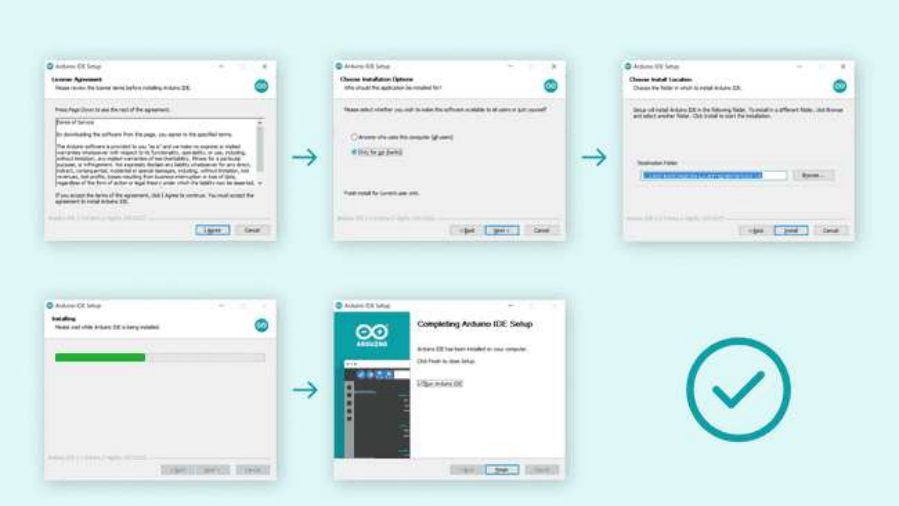
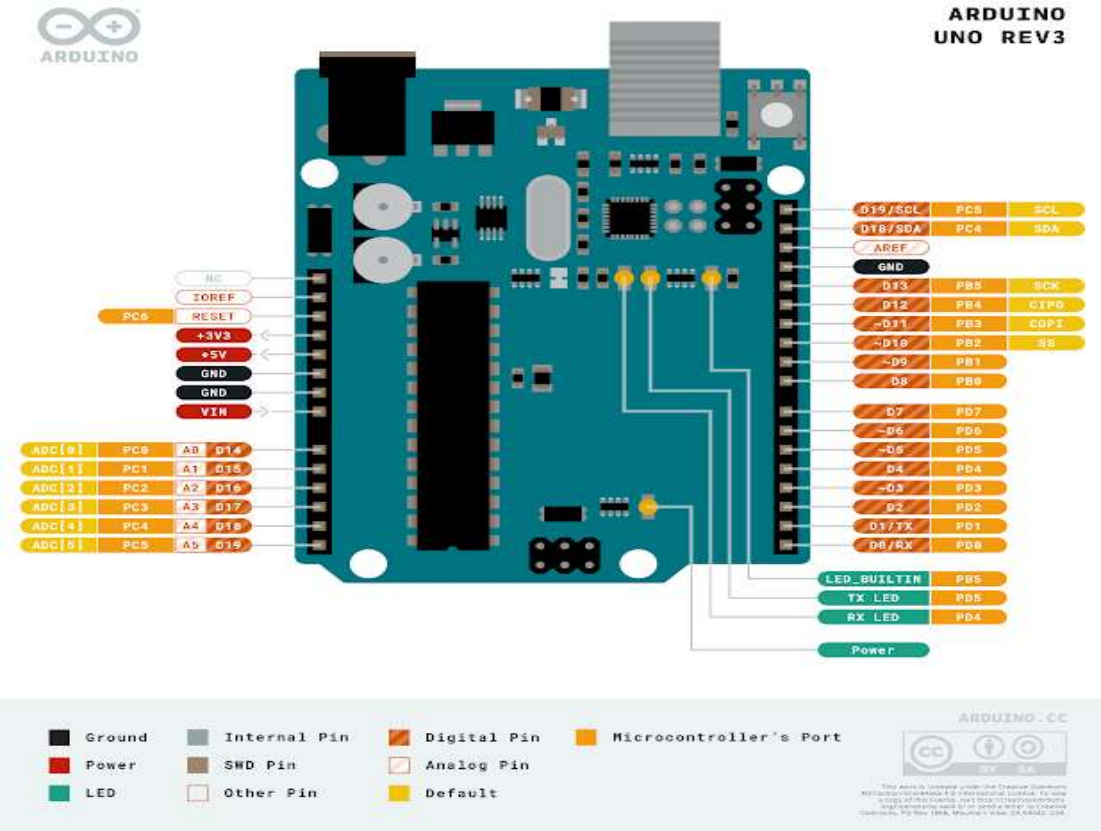


Figure 2.2 Instructions for installing the IDE 2.0

Arduino UNO REV3



Result

Thus the IOT enabled environment using Arduino software is created.

1. **b) Demonstrate the IOT programming by configuring the buzzer as output device with the Arduino microcontroller and turn it ON and OFF at an interval of 1 second by using the Microchip Studio IDE/ Thinker cad/Arduino IDE.**

Aim

To write the IOT programming by configuring the buzzer as output device with the Arduino microcontroller and turn it ON and OFF at an interval of 1 second by using the Microchip Studio IDE/ Thinker cad/Arduino IDE.

Procedure

Step 1: Before starting we need to know about tone() and noTone() function.

tone()

Step 2: Generates a square wave of the specified frequency (and 50% duty cycle) on a pin. A duration can be specified, otherwise the wave continues until a call to noTone(). The pin can be connected to a piezo buzzer or other speaker to play tones.

Syntax :

tone(pin_number, frequency)

noTone()

Stops the generation of a square wave triggered by tone().

Has no effect if no tone is being generated.

noTone(pin_number)

Step 1: The setup part is same as the setup part of Buzzer without *tone()*. So refer the Step -1 of Buzzer without tone().

Step 2: Next call the tone() function. It have two parameters. First one is the pin number which the buzzer is attached. Here it is 11. And second parameter is the frequency in Hertz. And set the frequency as 200 Hz.

tone(11,200);

Step 3: Then add a delay of 0.5 seconds.

delay(1000);

Step 4: Then call the function noTone(). This will help release the pin from tone() function. This function have only one parameter. It is the pin number which the buzzer attached.

noTone(11)

Step 5: Again add a delay of 1 seconds.

```
delay(1000);
```

Step 6 : Then upload the code to Arduino Uno.

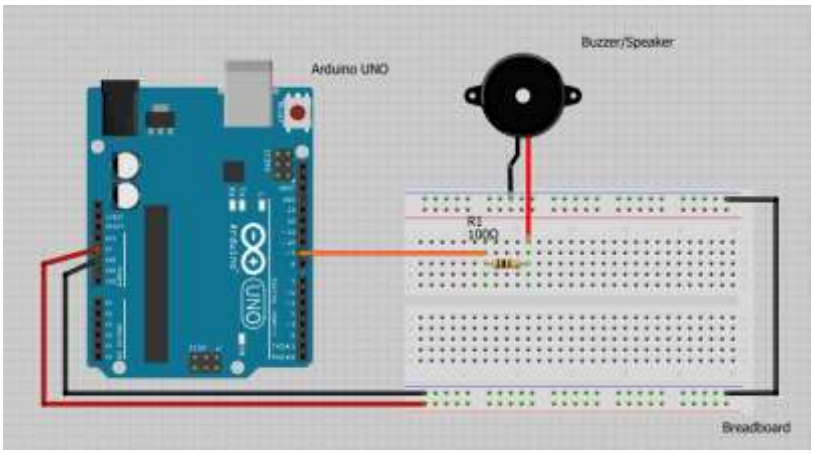
Connection

1. Arduino Uno - Buzzer
2. GND - GND (The short leg)
3. 5V - Vcc (The long leg)
4. For prototyping purpose use the Male/Female Jumber wire.
5. Next I am going to interface the buzzer with *tone()* function

Program

```
void setup() {
        pinMode(11,OUTPUT);
}
void loop() {
        tone(11,200);
        delay(1000);
        noTone(11);
         delay(1000);
}
```

Output:



Result Thus, IOT programming is configured by using the buzzer as output device with the Arduino microcontroller and turn it ON and OFF at an interval of 1 second by using the Arduino IDE.

2 a) Implement IOT program to blink LED by selecting the Atmel based microcontroller card in the Arduino Integrated Development Environment/ Thinker cad/Microship studio IDE.

Aim

To write a IOT program to blink LED by selecting the Atmel based microcontroller card/ Arduino Integrated Development Environment/ Thinker cad/Microship studio IDE.

Procedure

Step 1 : Initialize LED_BUILTIN pin as an output pin with the line.

pinMode(LED_BUILTIN, OUTPUT);

Step 2 : In the main loop, you turn the LED on with the line:

digitalWrite(LED_BUILTIN, HIGH);

Step 3: Supply 5 volts to the LED anode. That creates a voltage difference across the pins of the LED, and lights it up. Then you turn it off with the line:

Step 4 : *digitalWrite(LED_BUILTIN, LOW);* It takes the LED_BUILTIN pin back to 0 volts, and turns the LED off. In between the on and the off, you want enough time for a person to see the change, so the delay() commands tell the board to do nothing for 1000 milliseconds, or one second. When you use the delay() command, nothing else happens for that amount of time.

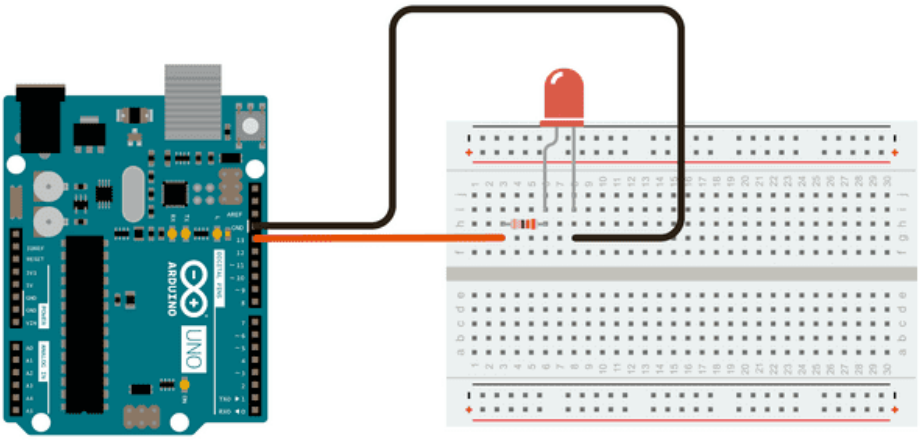
Step 5: Upload the code to the Arudino uno.

Program

```
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
```

}

Output



Result

Thus IOT program is created to blink LED by selecting the Atmel based microcontroller card in the Arduino Integrated Development Environment/ Thinker cad/Microship studio IDE.

2 b) Implement the IOT programming by interfacing the Bar-graph LEDs and the interrupt switch to toggle the status of 2 Bar-graph LEDs depending on whether the interrupt switch is pressed or releases using Microchip Studio IDE/ Thinker cad/Arduino IDE.

Aim:

To write a IOT programming by interfacing the Bar-graph LEDs and the interrupt switch to toggle the status of 2 Bar-graph LEDs depending on whether the interrupt switch is pressed or releases using Microchip Studio IDE/ Thinker cad/Arduino IDE.

Procedure:

Step 1: Complete the program as per the instructions present as comments in the C file.

Complete the pre-written functions to achieve the following as per the main function:

Step 2: Only the 2nd, 6th and 8th Bar-graph LED pins are configured as Output and only the 2nd LED is turned ON.

Step 3: The Interrupt Switch is configured as Input and the pull-up resistor for it is activated.

Step 4:The 6th LED on the Bar-graph is turned ON.Check whether the Interrupt Switch is pressed or not.

Step 5: If the Interrupt Switch is pressed, only the 2nd LED will turn OFF and the 8th LED will turn ON.

Step 6: If the Interrupt Switch is not pressed, the 2nd LED will remain ON and only 8th LED will turn OFF.

Connections:

Interrupt Switch : PE7

[There are two switches present on Firebird-V. One is Reset Switch and the other is Interrupt Switch. In this experiment we will be using the Interrupt Switch which is connected to an External Interrupt Pin (PE7 / INT7) and hence the name.]

Bar-graph LED:

LED 1 --> PJ0

LED 2 --> PJ1

LED 3 --> PJ2

LED 4 --> PJ3

LED 5 --> PJ4

LED 6 --> PJ5

LED 7 --> PJ6

LED 8 --> PJ7

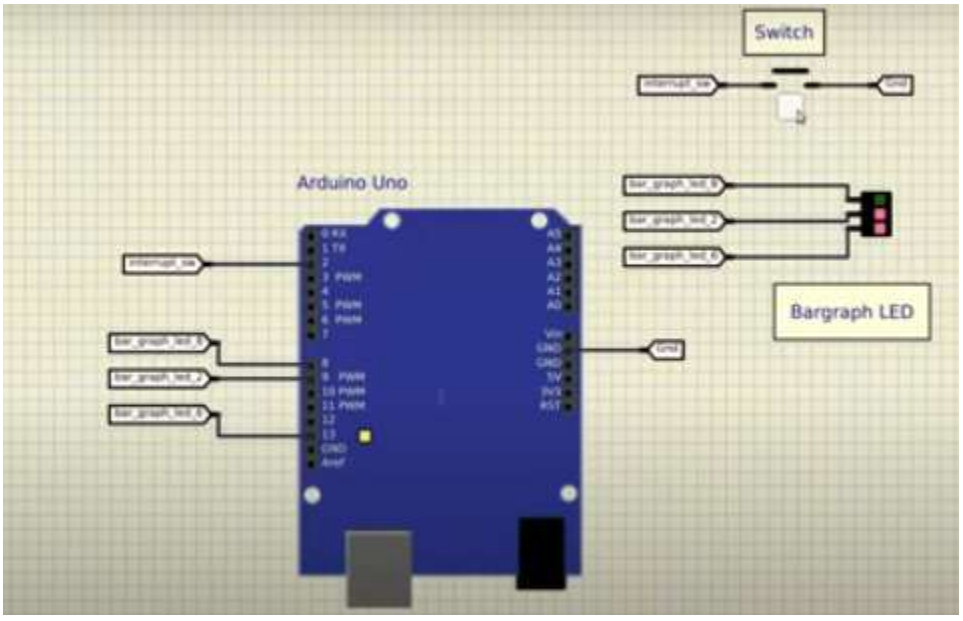
Program:

```
#include "firebird_simulation.h"                    // Header file included that contains
    macro definitions essential for Firebird V robot
#include <util/delay.h>                                          // Standard AVR Delay
    Library
#include <stdbool.h>                                        // Standard C Library for Boolean
    Type


void bar_graph_led_pins_config(void) {
    // << NOTE >> : Use Masking and Shift Operators here

    // Make **ONLY** 2nd, 6th and 8th Bar-graph LED pins as output
    bar_graph_led_ddr_reg    |= ;

    // Set **ONLY** 2nd Bar-graph LED as high (ON)
    bar_graph_led_port_reg   |= ;
}


void interrupt_sw_pin_config(void) {
    // << NOTE >> : Use Masking and Shift Operators here

    // Makes **ONLY** Interrupt Switch pin as input
    interrupt_sw_ddr_reg      &= ;

    // Makes **ONLY** Interrupt Switch pin internally pull-up
    interrupt_sw_port_reg     |= ;
}


        bool interrupt_switch_pressed(void)
        {

        }
```

```
void turn_on_bar_graph_led(unsigned char led_pin) {
    // << NOTE >> : Use Masking and Shift Operators here

    // Set **ONLY** a particular Bar-graph LED pin as high (ON)
}


void turn_off_bar_graph_led(unsigned char led_pin) {
    // << NOTE >> : Use Masking and Shift Operators here

    // Set **ONLY** a particular Bar-graph LED pin as low (OFF)
}


int main(void) {

    // << NOTE >> : You are not allowed to modify or change anything inside this
    function

    bar_graph_led_pins_config();
            // Initialize the 2nd, 6th and 8th Bar-graph LEDs
    interrupt_sw_pin_config();
                    // Initialize the Interrupt Switch

    turn_on_bar_graph_led(bar_graph_led_6_pin);
            // Turn ON 6th Bar-graph LED

    while (1)
    {
      if ( interrupt_switch_pressed() )
      {
            turn_off_bar_graph_led(bar_graph_led_2_pin);                    //
    Turn OFF 2nd Bar-graph LED
```

```
                turn_on_bar_graph_led(bar_graph_led_8_pin);
          // Turn ON 8th Bar-graph LED

                _delay_ms(50);

          }
          else
          {
                turn_on_bar_graph_led(bar_graph_led_2_pin);
          // Turn ON 2nd Bar-graph LED
                turn_off_bar_graph_led(bar_graph_led_8_pin);                    //
       Turn OFF 8th Bar-graph LED
          }
       }
}
```

Output



Result: Thus the IOT programme by interfacing the Bar-graph LEDs and the interrupt switch to toggle the status of 2 Bar-graph LEDs depending on whether the interrupt switch is pressed or releases using Microchip Studio IDE/ Thinker cad/Arduino IDE is created.

3 a) Implement the IOT program by representing the same numeral of up to 8-bits in different number systems on the inbuilt LCD and increment that number every 500 milliseconds loop through this infinitely.

Aim:

To represent the same numeral of up to 8-bits in different number systems on the inbuilt LCD and increment that number every 500 milliseconds loop through this infinitely.

Algorithm

Step 1: The LCD has already been initialized by the function lcd_port_config i.e. all relevant pins (PC0-PC2, PC4 - PC7) in the PORTs have been configured by setting the appropriate values in relevant registers.

Step 2: The lcd_init function initializes the LCD in 4-bit mode.

Step 3: The above two functions are declared in the lcd.h header file, which is already included in the project folder.

Step 4: The lcd_print_wireframe function prints the wireframe onto the LCD, these locations will remain static and not change when the numerals are incremented. Refer to the function documentation/comments in the C file for more details.

Step 5: A variable named counter of data type unsigned char (this means a maximum value of 255) has been initialized to zero.

Step 6: Inside the while loop:

a. Display the numerals in different number systems on the LCD using the functions lcd_print_binary, lcd_print_octal, lcd_print_decimal, lcd_print_hexadecimal.
b. Subsequently increment the counter variable.
c. All numbers should be printed with padded zeros for the number of spaces (cells) they have been alloted according to the lcd_print_wireframe function. For e.g.,
d. 003 instead of 3 in decimal and octal

e. 0F instead of F in hexadecimal and 00001010 instead of 1010 in binary

f. The counter variable will wrap around to 0 after you increment it beyond 255. Thus, due to the above fact the program will always give valid results in the infinite while loop.

Connection Details

LCD Connections:

RS --> PC0

RW --> PC1

EN --> PC2

DB7 --> PC7

DB6 --> PC6

DB5 --> PC5

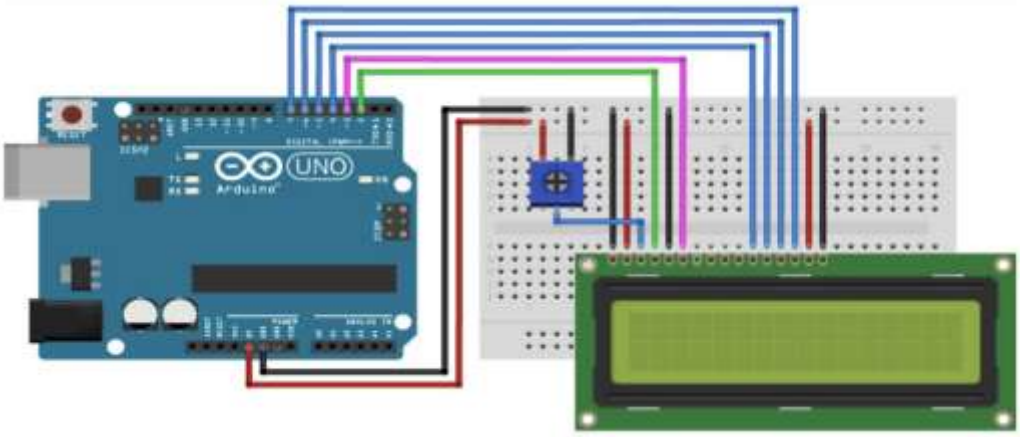
DB4 --> PC4

Program

```
#include <LiquidCrystal.h>
// Create an LCD object. Parameters: (RS, E, D4, D5, D6, D7):
LiquidCrystal lcd = LiquidCrystal(2, 3, 4, 5, 6, 7);
//LiquidCrystal lcd = LiquidCrystal(7, 6, 5, 4, 3, 2);

//LiquidCrystal lcd(2, 3, 4, 5, 6, 7);
void setup() {
  // Specify the LCD's number of columns and rows. Change to (20, 4) for a 20x4
LCD:
  lcd.begin(16, 2);
  lcd.clear();
}
void loop() {
  // Set the cursor on the third column and the first row, counting starts at 0:
 //lcd.setCursor(2, 0);
  // Print the string 'Hello World!':
  lcd.print("Hello World!");
  // Set the cursor on the third column and the second row:
  //lcd.setCursor(2, 1);
  // Print the string 'LCD tutorial':
```

```
 // lcd.print("LCD tutorial");
}
```

Output



Result: Thus, the IOT program is created for representing the same numeral of up to 8-bits in different number systems on the inbuilt LCD and increment that number every 500 milliseconds loop through this infinitely.

3 b) Study the Installation process of the operating systems for Raspberry Pi / Beagle board and describe the process of OS installation on Raspberry – Pi/ Beagle board

Aim

To Study the Installation process of the operating systems for Raspberry Pi / Beagle board and describe the process of OS installation on Raspberry – Pi/ Beagle board

RASPBERRY PI: Raspberry Pi is a credit card sized micro processor available in different models with different processing speed starting from 700 MHz. Whether you have a model B or model B+, or the very old version, the installation process remains the same. People who have checked out the official Raspberry Pi website, But using the Pi is very easy and from being a beginner, one will turn pro in no time. So, it's better to go with the more powerful and more efficient OS, the Raspbian. The main reason why Raspbian is extremely popular is that it has thousands of pre built libraries to perform many tasks and optimize the OS. This forms a huge advantage while building applications.

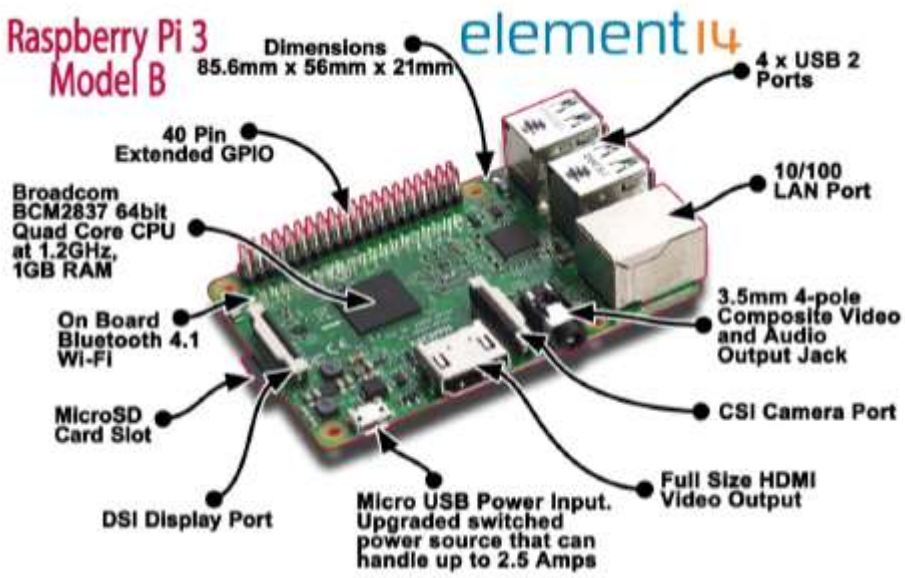


Figure 3.1 Raspberry Pi Elements

As for the specifications, the Raspberry Pi is a credit card-sized computer powered by the Broadcom BCM2835 system-on-a-chip (SoC). This SoC includes a 32-bit ARM1176JZFS processor, clocked at 700MHz, and a Videocore IV GPU. It also has 256MB of RAM in a POP package above the SoC. The Raspberry Pi is powered by a 5V micro USB AC charger or at least 4 AA batteries (with a bit of hacking). While the ARM CPU delivers real-world performance similar to that of a 300MHz Pentium 2, the Broadcom GPU is a very capable graphics core capable of hardware decoding several high definition video formats. The

Raspberry Pi model available for purchase at the time of writing — the Model B — features HDMI and composite video outputs, two USB 2.0 ports, a 10/100 Ethernet port, SD card slot, GPIO (General Purpose I/O Expansion Board) connector, and analog audio output (3.5mm headphone jack). The less expensive Model A strips out the Ethernet port and one of the USB ports but otherwise has the same hardware. Raspberry Pi Basics: installing Raspbian and getting it up and running.

1 Downloading Raspbian and Image writer.

You will be needing an image writer to write the downloaded OS into the SD card (micro SD card in case of Raspberry Pi B+ model). So download the "win32 disk imager" from the website.

2 Writing the image

Insert the SD card into the laptop/pc and run the image writer. Once open, browse and select the downloaded Raspbian image file. Select the correct device, that is the drive representing the SD card. If the drive (or device) selected is different from the SD card then the other selected drive will become corrupted. SO be careful.

After that, click on the "Write" button in the bottom. As an example, see the image below, where the SD card (or micro SD) drive is represented by the letter "G:\"

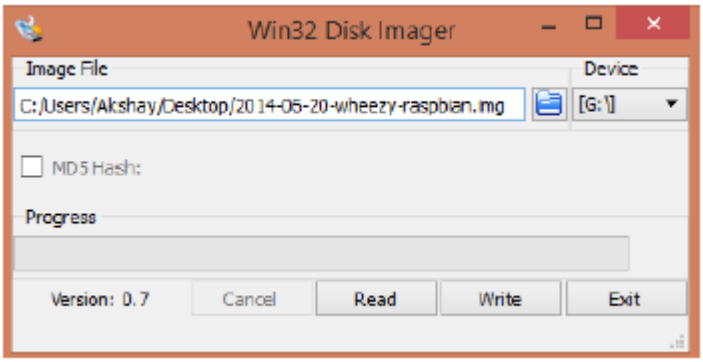


Figure 3.2 OS Installation

Once the write is complete, eject the SD card and insert it into the Raspberry Pi and turn it on. It should start booting up.

3 Setting up the Pi

Please remember that after booting the Pi, there might be situations when the user credentials like the "username" and password will be asked. Raspberry Pi comes with a default user name and password and so always use it whenever it is being asked. The credentials are:

login: pi

password: raspberry

When the Pi has been booted for the first time, a configuration screen called the "Setup Options" should appear and it will look like the image below.

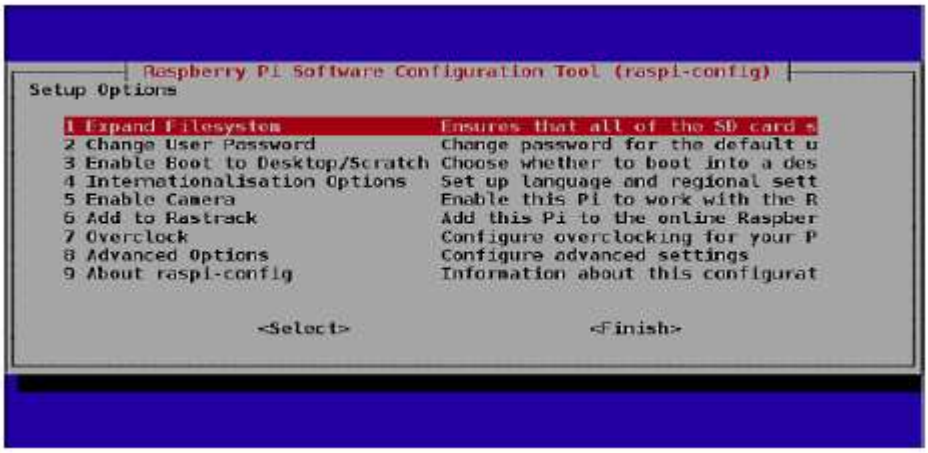


Figure 3.3 Raspberry Configuration

If you have missed the "Setup Options" screen, its not a problem, you can always get it by typing the following command in the terminal.

sudo raspi-config

Once you execute this command the "Setup Options" screen will come up as shown in the image above.

Now that the Setup Options window is up, we will have to set a few things. After completing each of the steps below, if it asks to reboot the Pi, please do so. After the reboot, if you don't get the "Setup Options" screen, then follow the command given above to get the screen/window.

The first thing to do: select the first option in the list of the **setup options window**, that is select the "**Expand Filesystem**" option and hit the enter key. We do this to make use of all the space present on the SD card as a full partition. All this does is, expand the OS to fit the whole space on the SD card which can then be used as the storage memory for the Pi

Select the third option in the list of the setup options window, that is select the "**Enable Boot To Desktop/Scratch" option** and hit the enter key. It will take you to another window called the "**choose boot option**" window that looks like the image below.

In the "choose **boot option window**", select the second option, that is, "**Desktop Log in as user 'pi' at the graphical desktop**" and hit the enter button. Once done you will be taken back to the "**Setup Options**" page, if not select the "OK" button at the bottom of this window

and you will be taken back to the previous window. We do this because we want to boot into the desktop environment which we are familiar with. If we don't do this step then the Raspberry Pi boots into a terminal each time with no GUI options. Once, both the steps are done, select the **"finish"** button at the bottom of the page and it should reboot automatically. If it doesn't, then use the following command in the terminal to reboot.

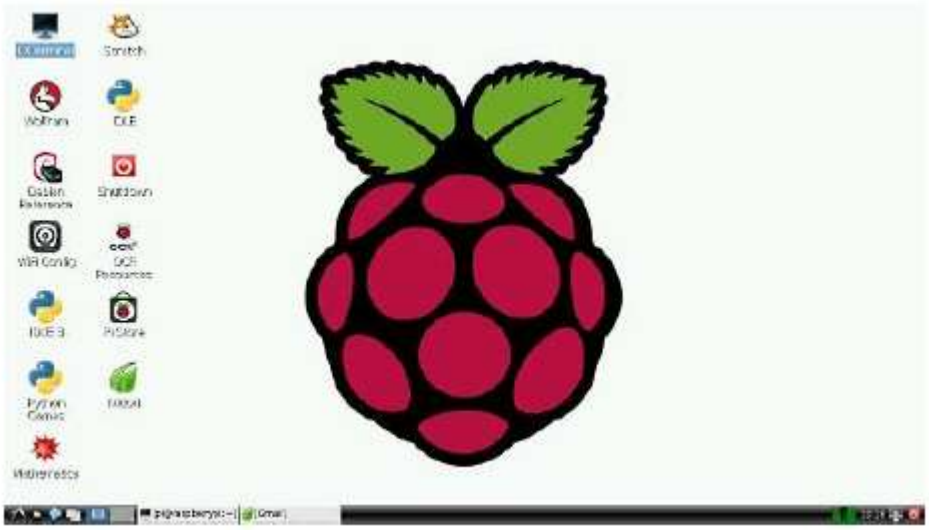


Figure 3.4 Raspberry Desktop

4 a) Implement the IOT program by interfacing the LCD with microcontroller to display the moving and scrolling and static text/string on LCD using ATmega2560 controller in Microchip Studio IDE.

Aim:

To write IOT program by interfacing the LCD with microcontroller to display the moving and scrolling and static text/string on LCD using ATmega2560 controller in Microchip Studio IDE.

Algorithm:

Step 1: complete the program as per the instructions present as comments in the C file, Experiment-4.c.

Step 2: These functions are declared and defined in lcd.h and lcd.c respectively. Also, some other functions present in lcd.h and lcd.c are listed below:

Procedure:

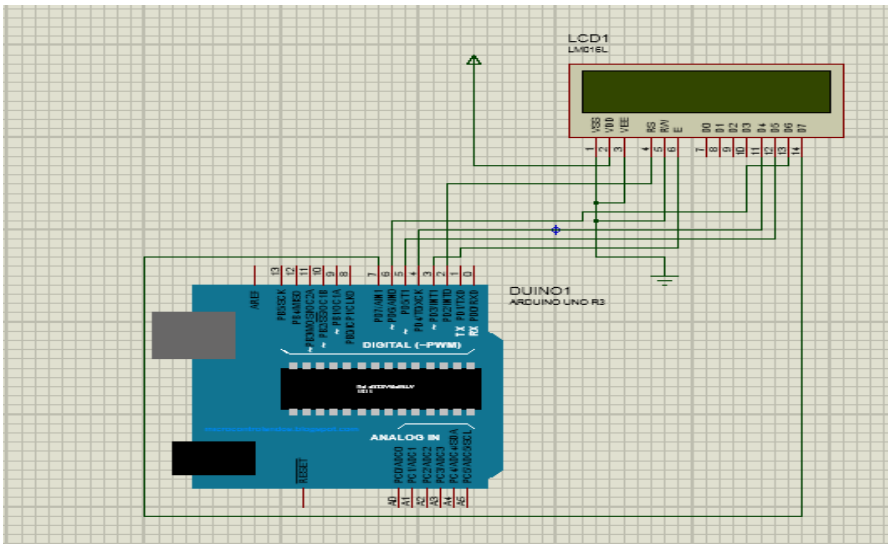
```
void lcd_port_config();
// This function configures the LCD port as output
void lcd_init();
// This function initialises the LCD in 4bit mode
void lcd_cursor(char row, char column);
// This function sets the cursor to the specified position
void lcd_wr_char(char row, char coloumn, char alpha_num_char);
// This function prints an alpha-numeric character at specified (row, column) position on
    LCD
void lcd_wr_command(unsigned char cmd);
// This function is used to send command to LCD
void lcd_string(char row, char column, char* str);
// This function prints the given string on the LCD at the specified (row, column) position
void lcd_numeric_value(char row, char column, int val, int digits);
// This function prints any integer value or value in a variable as integer on the specified
```

Program

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(7,6,5,4,3,2);

void setup(){
  Serial.begin(9600);
  lcd.begin(16,2);
```

```
  lcd.setCursor(0,0);
}
void loop(){
  Serial.println("Welcome To");
  delay(1000);
  lcd.begin(16,2);
  lcd.setCursor(3,0);
  lcd.print("Welcome To");
   delay(1000);
 // lcd.begin(16,2);
  //lcd.setCursor(0,1);
  //lcd.print("Sethu college");
 // delay(1000);
  lcd.begin(16,2);
  lcd.setCursor(4,1);
  lcd.print("IOT LAB");
  delay(1000);
}
```

Output



Result:

Thus the IOT program by using the LCD of ATmega 2560 in Microchip studio IDE to represent the same numeral of up to 8-bits in different number systems is implemented.

4 b) Implement the IOT program by using the LCD of ATmega 2560 in Microchip studio IDE to represent the same numeral of up to 8-bits in different number systems.

Aim:

To Implement the IOT program by using the LCD of ATmega 2560 in Microchip studio IDE to represent the same numeral of up to 8-bits in different number systems.

Algorithm:

Step 1: Lcd8_Clear() & Lcd4_Clear() : Calling these functions will clear the 16×2 LCD display screen when interfaced with 8 bit and 4 bit mode respectively.

Step 2: Lcd8_Set_Cursor() & Lcd4_Set_Cursor() : These function will set the cursor position on the LCD screen by specifying its row and column. By using these functions we can change the position of character and string displayed by the following functions.

Step 3: Lcd8_Write_Char() & Lcd4_Write_Char() : These functions will write a single character to the LCD screen and the cursor position will be incremented by one.

Step 4: Lcd8_Write_String() & Lcd8_Write_String() : These function will write string or text to the LCD screen and the cursor positon will be incremented by length of the string plus one.

Step 5: Lcd8_Shift_Left() & Lcd4_Shift_Left() : This function will shift data in the LCD display without changing data in the display RAM.

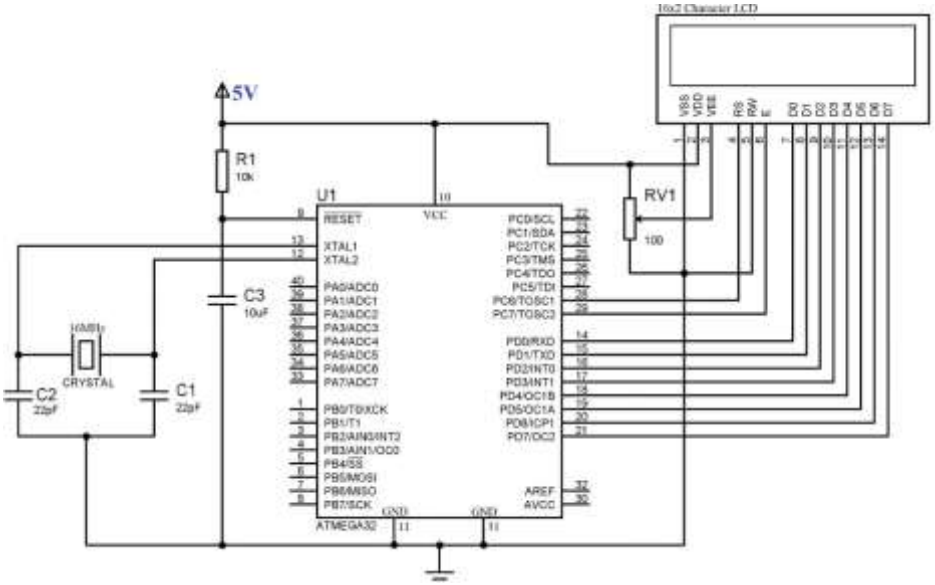
Step 6: Lcd8_Shift_Right() & Lcd8_Shift_Right() : This function will shift data in the LCD display without changing data in the display RAM.

Program:

```
#ifndef F_CPU
#define F_CPU 16000000UL // 16 MHz clock speed
#endif

#define D0 eS_PORTD0
#define D1 eS_PORTD1
#define D2 eS_PORTD2
#define D3 eS_PORTD3
#define D4 eS_PORTD4
#define D5 eS_PORTD5
#define D6 eS_PORTD6
#define D7 eS_PORTD7
#define RS eS_PORTC6
#define EN eS_PORTC7
```

```
#include <avr/io.h>
#include <util/delay.h>
#include "lcd.h" //Can be download from the bottom of this article
int main(void)
{
  DDRD = 0xFF;
  DDRC = 0xFF;
  int i;
  Lcd8_Init();
  while(1)
  {
    Lcd8_Set_Cursor(1,1);
    Lcd8_Write_String("electroSome LCD Hello World");
    for(i=0;i<15;i++)
    {
      _delay_ms(500);
      Lcd8_Shift_Left();
    }
    for(i=0;i<15;i++)
    {
      _delay_ms(500);
      Lcd8_Shift_Right();
    }
    Lcd8_Clear();
    Lcd8_Write_Char('e');
    Lcd8_Write_Char('S');
    _delay_ms(2000);
  }
}
```

Output:



Result:

Thus the IOT program by using the LCD of ATmega 2560 in Microchip studio IDE to represent the same numeral of up to 8-bits in different number systems is implemented.

4 c) Study Connectivity and Configuration of Raspberry-Pi/ Beagle Board circuit with basic peripherals, LEDs, Understanding GPIO and its use in program.

Aim:

Study of connectivity and configuration of Raspberry-Pi circuit with basic peripherals, LEDS. Undersatnding GPIO and its use in program.

GPIOA powerful feature of the Raspberry Pi is the row of GPIO (general-purpose input/output) pins along the top edge of the board. A 40-pin GPIO header is found on all current Raspberry Pi boards (unpopulated on Pi Zero and Pi Zero W). Prior to the Pi 1 Model B+ (2014), boards comprised a shorter 26-pin header.

GPIO pins

Assignment : 3Aim : Study of connectivity and configuration of Raspberry-Pi circuit with basic peripherals, LEDS. Undersatnding GPIO and its use in program.GPIOA powerful feature of the Raspberry Pi is the row of GPIO (general-purpose input/output) pins along the top edge of the board. A 40-pin GPIO header is found on all current Raspberry Pi boards (unpopulated on Pi Zero and Pi Zero W). Prior to the Pi 1 Model B+ (2014), boards comprised a shorter 26-pin header.GPIO pinsAny of the GPIO pins can be designated (in software) as an input or output pin and used for a wide range of purposes.

Note: the numbering of the GPIO pins is not in numerical order; GPIO pins 0 and 1 arepresent on the board (physical pins 27 and 28) but are reserved for advanced use (see below)

Figure 4.2 GPIO

GPIO:

1. Act as both digital output and digital input.
2. **Output**: turn a GPIO pin high or low.
3. **Input**: detect a GPIO pin high or low

4. **Installing GPIO library:**
5. Open terminal
6. Enter the command "sudoapt-**get install python-dev"** to install python development
7. Enter the command **"sudoapt-get install python-rpi.gpio"** to install GPIO library.
8. **Basic python coding:**
9. Open terminal enter the command
10. **sudo nano filename.py**

This will open the nano editor where you can write your code

- Ctrl+O : Writes the code to the file
- Ctrl+X : Exits the editor
- Blinking LED Code:
- import RPi.GPIO as GPIO #GPIO library import time
- GPIO.setmode(GPIO.BOARD) # Set the type of board for pin numbering
- GPIO.setup(11, GPIO.OUT) # Set GPIO pin 11as output pin
- for i in range (0,5): GPIO.output(11,True) # Turn on GPIO pin 11
- time.sleep(1)
- GPIO.output(11,False)
- time.sleep(2)
- GPIO.output(11,True)
- GPIO.cleanup()

Power Pins

The header provides 5V on Pin 2 and 3.3V on Pin 1. The 3.3V supply is limited to 50mA. The 5V supply draws current directly from your microUSB supply so can use whatever is left over after the board has taken its share. A 1A power supply could supply up to 300mA once the Board has drawn 700mA.

Basic GPIO

The header provides 17 Pins that can be configured as inputs and outputs. By default they are all configured as inputs except GPIO 14 & 15.In order to use these pins you must tell the system whether they are inputs or outputs. This can be achieved a number of ways and it depends on how you intend to control them. I intend on using Python.

SDA & SCL:

The 'DA' in SDA stands for data, the 'CL' in SCL stands for clock; the S stands for serial. You can do more reading about the significance of the clock line for various types of computer bus, You will probably find I2C devices that come with their own userspace drivers and the linux kernel includes some as well. Most computers have an I2C bus, presumably for some of the purposes listed by wikipedia, such as interfacing with the RTC (real time clock) and configuring memory. However, it is not exposed, meaning you can't attach anything else to it, and there are a lot of interesting things that could be attached -- pretty much any kind of common sensor (barometers, accelerometers, gyroscopes, luminometers, etc.) as well as output devices and displays. You can buy a USB to I2C adapter for a normal computer, but they cost a few hundred dollars. You can attach multiple devices to the exposed bus on the pi.

UART, TXD & RXD: This is a traditional serial line; for decades most computers have had a port for this and a port for parallel.1 Some pi oriented OS distros such as Raspbian by default boot with this serial line active as a console, and you can plug the other end into another computer and use some appropriate software to communicate with it. Note this interface does not have a clock line; the two pins may be used for full duplex communication (simultaneous transmit and receive).

PCM, CLK/DIN/DOUT/FS: PCM is is how uncompressed digital audio is encoded. The data stream is serial, but interpreting this correctly is best done with a separate clock line (more lowest level stuff).

SPI, MOSI/MISO/CE0/CE1: SPI is a serial bus protocol serving many of the same purposes as I2C, but because there are more wires, it can operate in full duplex which makes it faster and more flexible.

Raspberry Pi Terminal Commands

[sudo apt-get update] - Update Package Lists
[sudo apt-get upgrade] - Download and Install Updated Packages
[sudo raspi-config] - The Raspberry Pi Configuration Tool
[sudo apt-get clean] - Clean Old Package Files
[sudo reboot] - Restart your Raspberry Pi
[sudo halt] - Shut Down your Raspberry Pi

4 d) Implement the program to blink five LEDs using Arrays. All the five LEDS will light after one other.

Aim

To Implement the program to blink five LEDs using Arrays. All the five LEDS will light after one other.

Algorithm

Step 1: Setup routine runs once when you press reset.

Step 2: initialize the digital pin as an output

Step 3: Loop routine runs over and over again forever

3.1 // turn the LED on (HIGH is the voltage level)

3.2 wait for 1/2 a second

3.3 turn the LED off by making the voltage LOW

3.4 wait for 1/2 a second

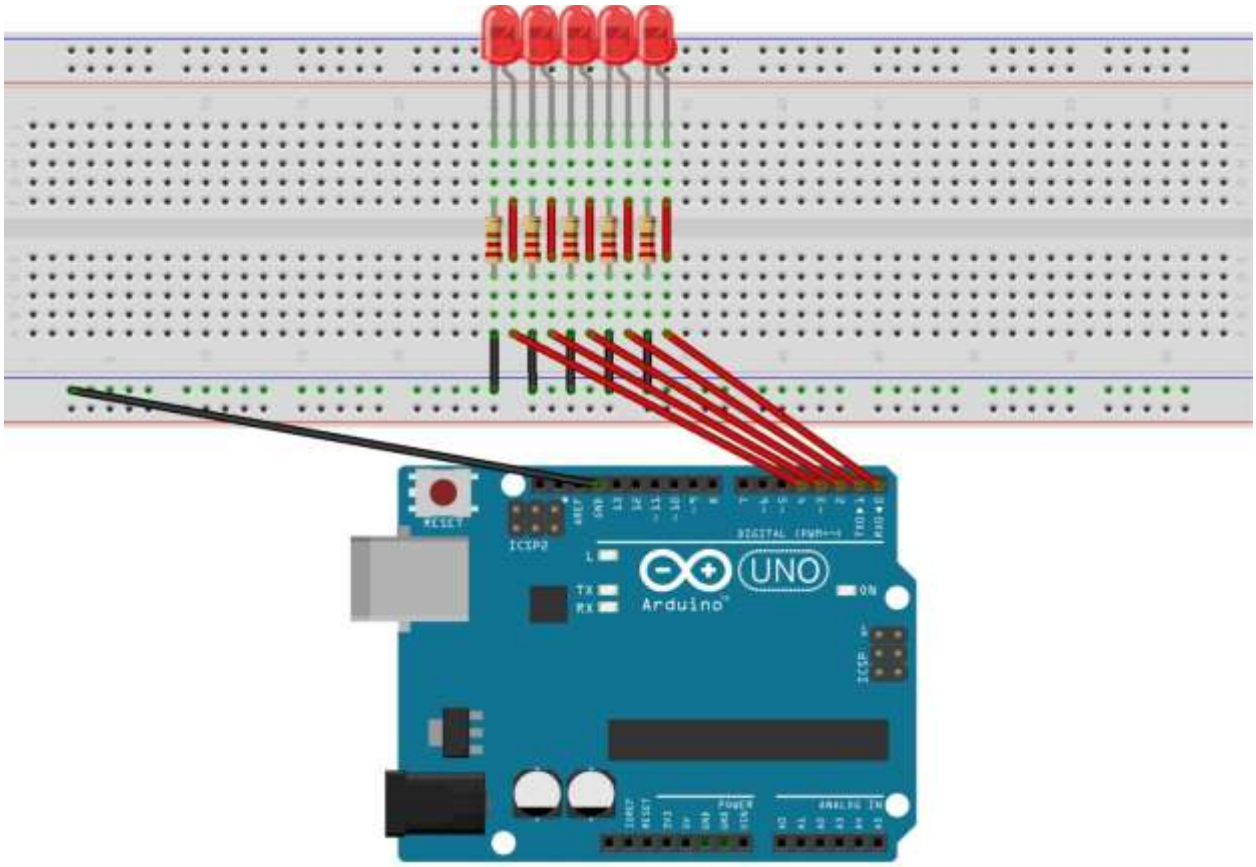
Step 4: Repeat upto LED 5

Program

```
int led = 13;
int led2 = 12;
int led3 = 11;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
  pinMode(led2, OUTPUT);
pinMode(led3, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(100);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(100);
  {digitalWrite(led2, HIGH);
```

```
delay(100);
digitalWrite(led2, LOW);
delay(100);}
{digitalWrite(led3, HIGH);
delay(100);
digitalWrite(led3, LOW);
delay(100);}// wait for a second
}
```

Output



\

Result

Thus the IOT program to blink five LEDs using Arrays is implemented with five LEDS that will light after one other.