“EMPOWERMENT THROUGH TECHNOLOGICAL EXCELLENCE”

GenbaSopanraoMoze College of Engineering,

Balewadi, Pune-411045

Department of Artificial Intelligence & Machine Learning

Academic Year: 2024-25

SEM-I

Lab Manual

Subject: IOT with AI

(318546)

Class: TE

Subject In-charge: Prof. Prachi Admane

Prerequisites: Basic : Basic Electronics Engineering Computer Programming

Course Objectives :

• Hardware platforms and operating systems commonly used in IoT systems.

• Help the students in providing a good learning environment and also work with real time problems faced in day to day life.

Course Outcomes: On completion of the course, students will be able to–

CO1: Understand IOT Application Development using Raspberry Pi/ Beagle board/ Arduino board

CO2: Develop and modify the code for various sensor based applications using wireless sensor modules and working with a variety of modules like environmental modules.

CO3: Make use of Cloud platform to upload and analyze any sensor data

GENERAL LABORATORY INSTRUCTIONS

This laboratory is designed for hands-on experience with Internet of Things technologies, including sensors, actuators, microcontrollers, and network communication. Our goal is to provide practical experience in building, programming, and troubleshooting IoT devices and systems.

- Students are instructed to come in IOT laboratory on time
- Students should be punctual to the lab, if not the conducted experiments will not be repeated.
- Do sign on laboratory log to maintain the attendance.
- Be cautious when handling electrical components. Ensure that all devices are powered off before making connections or adjustments.
- Before starting any experiment or project, thoroughly read the provided instructions and guidelines.
- Handle all components with care to avoid damage. Use appropriate tools for connections and adjustments.
- Keep detailed records of your setup, configuration, code, and results. This is crucial for troubleshooting and reporting.
- Prepare and submit any required reports or documentation based on your experiment or project.
- While leaving the lab shut down your PC

List Of Assignments

Subject:- IOT with AI **Class:- TE**

Practical (PR) : 04hrs/week PR : 25 Marks

TW : 25Marks

Index

Sr. No.	List of Assignments
1	Study of Arduino and other microcontroller (History & Elevation)
2	Study of different operating systems for Arduino. Understanding the process of OS installation
3	Write an application to read temperature from the environment. If temperature crosses threshold value then it notifies with buzzer
4	Write a program using Arduino to control LED (One or more ON/OFF). Or Blinking.
5	Create a program so that when the user enters ‘b’ the green light blinks, ‘g’ the green light is illuminated ‘y’ the yellow light is illuminated and ‘r’ the red light is illuminated.
6	Write a program that asks the user for a number and outputs the number squared that is entered.
7	Write a program to control the color of the LED by turning 3 different potentiometers. One will be read for the value of Red, one for the value of Green, and one for the value of Blue.
8	Write a program read the temperature sensor and send the values to the serial monitor on the computer.
9	Write a program so it displays the temperature in Fahrenheit as well as the maximum and minimum temperatures it has seen.
10	Study of ThingSpeak – an API and Web Service for the Internet of Things.

IOT Lab No.1

Assignment no.1

Title:- Study of Raspberry-Pi/ Beagle board/Arduino and other microcontroller

Objective:- To study Raspberry-Pi/ Beagle board/Arduino and other microcontroller

Theory:-

In IoT applications the Arduino is used to collect the data from the sensors/devices to send it to the internet and receives data for purpose of control of actuators.

Examples of IoT: -

1) Apple Watch and Home Kit.

2) Smart Refrigerator.

3) Smart Refrigerator.

4) Smart cars.

5) Google Glass.

6) Smart thermostats.

A) Raspberry-Pi:- The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote teaching of basic computer science in schools and in developing countries. It does not include peripherals (such as keyboards and mice). The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. The Raspberry Pi is a credit-card-sized computer that costs between $5 and $35. It's available anywhere in the world, and can function as a proper desktop computer or be used to build smart devices. A Raspberry Pi is a general-purpose computer, usually with a Linux operating system, and the ability to run multiple programs. Raspberry Pi is like the brain. Its primary advantage comes in processing higher level processing capability. It’s a single board computer.

Here are the various components on the Raspberry Pi board:

- ARM CPU/GPU -- This is a Broadcom BCM2835 System on a Chip (SoC) that's made up of an ARM central processing unit (CPU) and a Video core 4 graphics processing unit (GPU). The CPU handles all the computations that make a computer work (taking input, doing calculations and producing output), and the GPU handles graphics output.

• GPIO -- These are exposed general-purpose input/output connection points that will allow the real hardware hobbyists the opportunity to tinker.

• RCA -- An RCA jack allows connection of analog TVs and other similar output devices.

• Audio out -- This is a standard 3.55-millimeter jack for connection of audio output devices such as headphones or speakers. There is no audio in.

• LEDs -- Light-emitting diodes, for your entire indicator light needs.

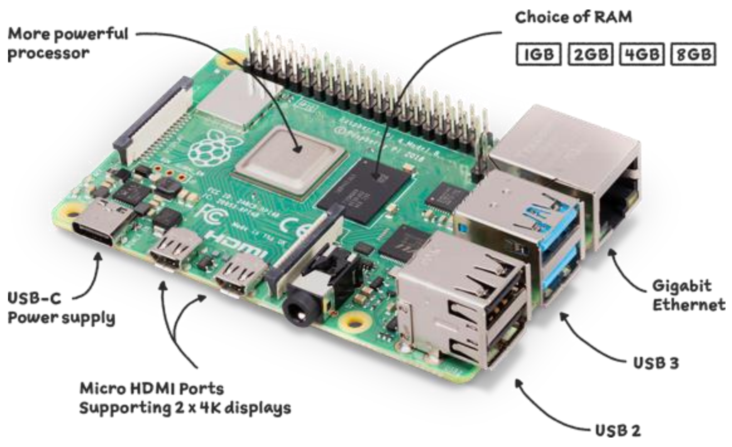
• USB -- This is a common connection port for peripheral devices of all types (including your mouse and keyboard). Model A has one, and Model B has two. You can use a USB hub to expand the number of ports or plug your mouse into your keyboard if it has its own USB port.

• HDMI -- This connector allows you to hook up a high-definition television or other compatible device using an HDMI cable.

• Power -- This is a 5v Micro USB power connector into which you can plug your compatible power supply.

• SD card slot -- This is a full-sized SD card slot. An SD card with an operating system (OS) installed is required for booting the device. They are available for purchase from the manufacturers, but you can also download an OS and save it to the card yourself if you have a Linux machine and the wherewithal.

• Ethernet -- This connector allows for wired network access and is only available on the Model B.



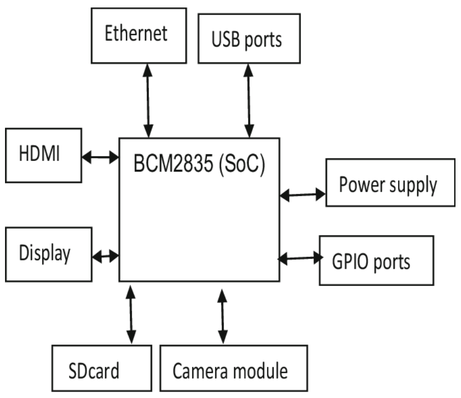
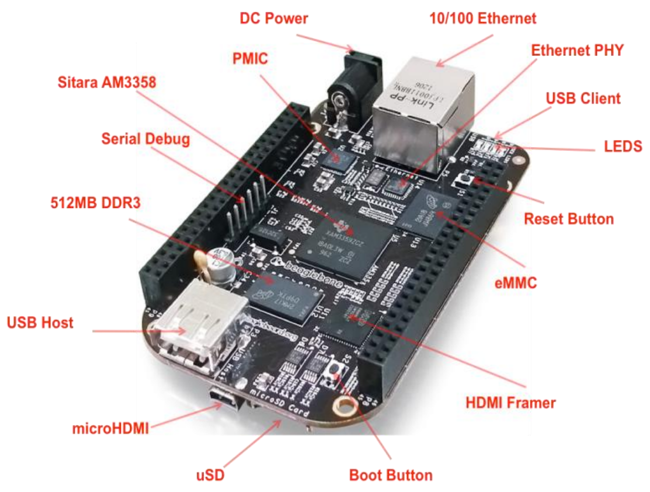


Fig1: Raspberry pi board

Fig 2: Architecture of Raspberry pi

B) Beagle board:- The Beagle Board is a low-power open-source single-board computer produced by Texas Instruments in association with Digi-Key and Newark element14. The Beagle Board was also designed with open source software development in mind, and as a way of demonstrating the Texas Instrument's OMAP3530 system-on-a-chip.]The board was developed by a small team of engineers as an educational board that could be used in colleges around the world to teach open source hardware and software capabilities. It is also sold to the public under the Creative Commons share-alike license. The board was designed using Cadence OrCAD for schematics and Cadence Allegro for PCB manufacturing; no simulation software was used. Beagle Bone Black is a low-cost, open source, community-supported development platform for ARM® Cortex™-A8 processor developers and hobbyists. Boot Linux in under 10-seconds and get started on Sitara™ AM335x ARM Cortex-A8 processor development in less than 5 minutes with just a single USB cable



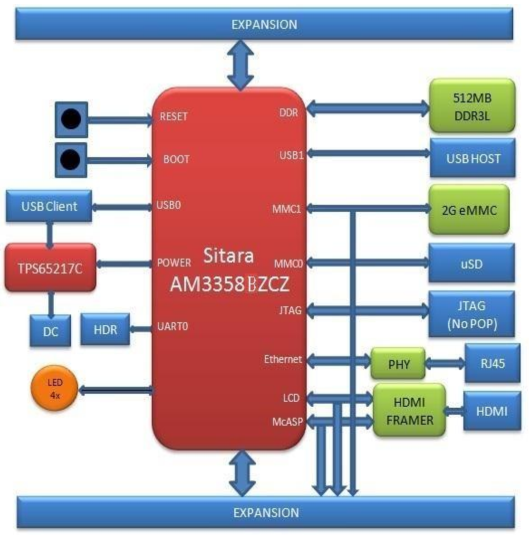


Fig1: Beagle Board Fig2: Architecture of Beagle board

Here are the various components on the Beagle board: Processor: AM335x 1GHz ARM® Cortex-A8

- 512MB DDR3 RAM
- 4GB 8-bit eMMC on-board flash storage
- 3D graphics accelerator
- NEON floating-point accelerator
- 2x PRU 32-bit microcontrollers Connectivity
- USB client for power & communications
- USB host

• Ethernet

• HDMI

• 2x 46 pin headers Software Compatibility

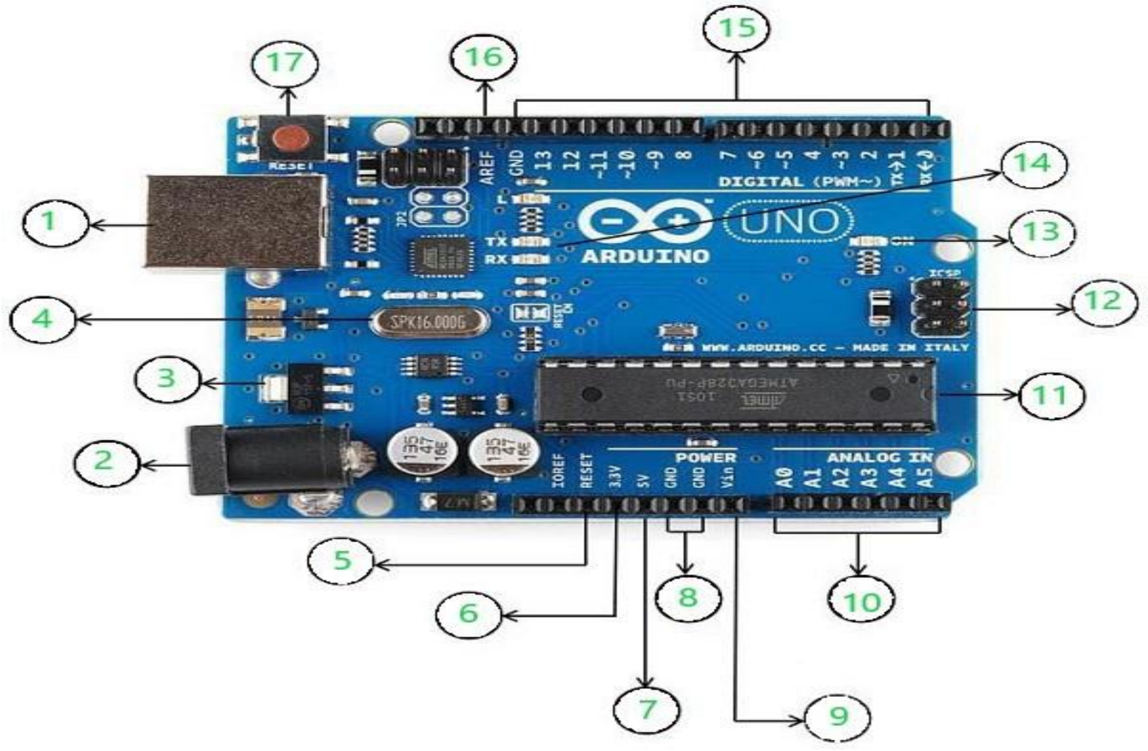
• Debian

• Android

• Ubuntu

• Cloud9 IDE on Node.js w/ BoneScript library

C) Arduino:- 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 I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable. It can be powered by the USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts. The word "uno" means "one" in Italian and was chosen to mark the initial release of Arduino Software.



Features of the Arduino

1. Arduino boards are able to read analog or digital input signals from different sensors and turn it into an output such as activating a motor, turning LED on/off, connect to the cloud and many other actions.

2. The board functions can be controlled by sending a set of instructions to the microcontroller on the board via Arduino IDE.

3. Arduino IDE uses a simplified version of C++, making it easier to learn to program.

4. Arduino provides a standard form factor that breaks the functions of the microcontroller into a more accessible package.

Various components on the Arduino board:

Microcontrollers :-

ATmega328P (used on most recent boards)

ATmega168 (used on most Arduino Diecimila and early Duemilanove)

ATmega8 (used on some older board)

Digital Pins

In addition to the specific functions listed below, the digital pins on an Arduino board can be used for general purpose input and output via the pinMode(), digitalRead(), and digitalWrite() commands. Each pin has an internal pull-up resistor which can be turned on and off using digitalWrite() (w/ a value of HIGH or LOW, respectively) when the pin is configured as an input. The maximum current per pin is 40 mA.

Analog Pins

In addition to the specific functions listed below, the analog input pins support 10-bit analog-to-digital conversion (ADC) using the analogRead() function. Most of the analog inputs can also be used as digital pins: analog input 0 as digital pin 14 through analog input 5 as digital pin 19. Analog inputs 6 and 7 (present on the Mini and BT) cannot be used as digital pins.

Power Pins

VIN (sometimes labelled "9V"). The input voltage to the Arduino 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. Note that different boards accept different input voltages ranges, please see the documentation for your board. Also note that the LilyPad has no VIN pin and accepts only a regulated input.

Other Pins

- AREF. Reference voltage for the analog inputs. Used with analogReference().
- Reset. (Diecimila-only) Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.
- Analog Reference pin (orange)
- Digital Ground (light green)
- Digital Pins 2-13 (green)
- Digital Pins 0-1/Serial In/Out - TX/RX (dark green) - These pins cannot be used for digital i/o (digitalRead and digitalWrite) if you are also using serial communication (e.g. Serial.begin).
- Reset Button - S1 (dark blue)
- In-circuit Serial Programmer (blue-green)
- Analog In Pins 0-5 (light blue)
- Power and Ground Pins (power: orange, grounds: light orange)
- External Power Supply In (9-12VDC) - X1 (pink)
- Toggles External Power and USB Power (place jumper on two pins closest to desired supply) - SV1 (purple)
- USB (used for uploading sketches to the board and for serial communication between the board and the computer; can be used to power the board) (yellow)

Conclusion: - Thus, we have studied of IoT platforms such as Raspberry-Pi/Beagle board/Arduino.

IOT Lab No.2

Assignment no.2

Title:- Study of different operating systems for Raspberry-Pi /Beagle board/Arduino. Understanding the process of OS installation

Objective:- To study operating systems for platforms such as Raspberry-Pi/Beagle board/Arduino.

Theory:-

1) **Raspberry-Pi:** - The Pi can run the official Raspbian OS, Ubuntu Mate, Snappy Ubuntu Core, the Kodibased media centers OSMC and LibreElec, the non-Linux based Risc OS (one for fans of 1990s Acorn computers). It can also run Windows 10 IoT Core, which is very different to the desktop version of Windows, as mentioned below.

OS which install on Raspberry-Pi: Raspbian, Ubuntu MATE, Snappy Ubuntu, Pidora, Linutop, SARPi, Arch Linux ARM, Gentoo Linux,

How to install Raspbian on Raspberry-Pi:

Step 1: Download Raspbian

Step 2: Unzip the file. The Raspbian disc image is compressed, so you'll need to unzip it. The file uses the ZIP64 format, so depending on how current your built-in utilities are, you need to use certain programs to unzip it.

Step 3: Write the disc image to your microSD card. Next, pop your microSD card into your computer and write the disc image to it. The process of actually writing the image will be slightly different across these programs, but it's pretty self-explanatory no matter what you're using. Each of these programs will have you select the destination (make sure you've picked your microSD card!) and the disc image (the unzipped Raspbian file). Choose, double-check, and then hit the button to write.

Step 4: Put the microSD card in your Pi and boot up. Once the disc image has been written to the microSD card, you're ready to go! Put that sucker into your Raspberry Pi, plug in the peripherals and power source, and enjoy. The current edition to Raspbian will boot directly to the desktop. Your default credentials are username pi and password raspberry.

2) BeagleBone Black:- The BeagleBone Black includes a 2GB or 4GB on-board eMMC flash memory chip. It comes with the Debian distribution factory pre-installed. You can flash new operating systems including Angstrom, Ubuntu, Android, and others.

1. Os which install on BeagleBone Black: Angstrom, Android, Debian, Fedora, Buildroot, Gentoo, Nerves Erlang/OTP, Sabayon, Ubuntu, Yocto, MINIX 3

   How to install Debian on BeagleBone Black:
   Step 1: Download Debian img.xz file.
   Step 2: Unzip the file.
   Step 3: Insert your MicroSD (uSD) card into the proper slot. Most uSD cards come with a full-sized SD card that is really just an adapter. If this is what you have then insert the uSD into the adapter, then into your card reader.
   Step 4: Now open Win32 Disk imager, click the blue folder icon, navigate to the debian img location, and double click the file. Now click Write and let the process complete. Depending on your processor and available RAM it should be done in around 5 minutes.
   Step 5: Alright, once that's done, you'll get a notification pop-up. Now we're ready to get going. Remove the SD adapter from the card slot, remove the uSD card from the adapter. With the USB cable disconnected insert the uSD into the BBB.
   Step 6: Now, this next part is pretty straight forward. Plug the USB cable in and wait some more. If everything is going right you will notice that the four (4) leds just above the USB cable are doing the KIT impression. This could take up to 45 minutes, I just did it again in around 5 minutes. Your mileage will vary. Go back and surf reddit some more.
   Step 7: If you are not seeing the leds swing back and forth you will need to unplug the USB cable, press and hold down the user button above the uSD card slot (next to the 2 little 10 pin ICs) then plug in the USB cable. Release the button and wait. You should see the LEDs swinging back and forth after a few seconds. Once this happens it's waiting time. When all 4 LEDs next to the USB slot stay lit at the same time the flash process has been completed.
   Step 8: Remove the uSD card and reboot your BBB. You can reboot the BBB by removing and reconnecting the USB cable, or hitting the reset button above the USB cable near the edge of the board.
   Step 9: Now using putty, or your SSH flavor of choice, connect to the BBB using the IP address 192.168.7.2. You'll be prompted for a username. Type root and press Enter. By default, there is no root password. I recommend changing this ASAP if you plan on putting your BBB on the network. To do this type password, hit enter, then enter your desired password. You will be prompted to enter it again to verify.

3) Arduino: - The Arduino itself has no real operating system. You develop code for the Arduino using the Arduino IDE which you can download from Arduino - Home. Versions are available for Windows, Mac and Linux. The Arduino is a constrained microcontroller. Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board. You are literally writing the "firmware" when you write the code and upload it. It's both good and its bad.

Conclusion: - Thus, we have studied of how to install operating systems for platforms such as Raspberry-Pi/Beagle board/Arduino. Arduino IDE (Integrated Development Environment)

The Arduino Software (IDE) is easy-to-use and is based on the Processing programming environment. The Arduino Integrated Development Environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in functions from C and C++. The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board. The Arduino Software (IDE) – contains: • A text editor for writing code • A message area • A text consoles • A toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.

Conclusion:- We have studied the operating systems for platforms such as Raspberry-Pi/Beagle board/Arduino.

IOT Lab No.3

Assignment no.3

Title:- Write an application to read temperature from the environment. If temperature crosses threshold value then it notifies with buzzer

Objective:- To study an application to read temperature from the environment. If temperature crosses threshold value then it notifies with buzzer

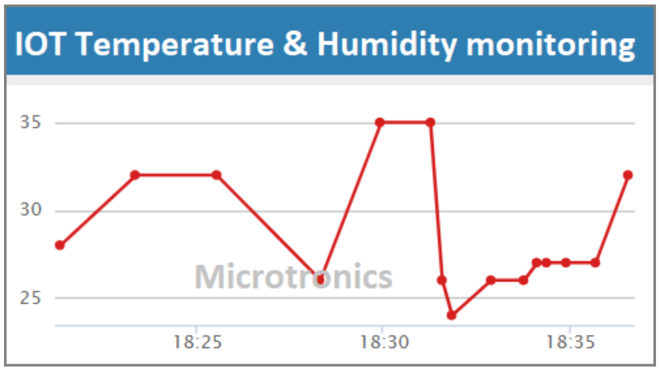
Theory:-

We can remotely monitor the Temperature and Humidity of any system or environment using the IOT Temperature and Humidity monitoring system using Arduino. This real-time and historical data of humidity and temperature can be viewed from anywhere in the world.

Hardware implementation:

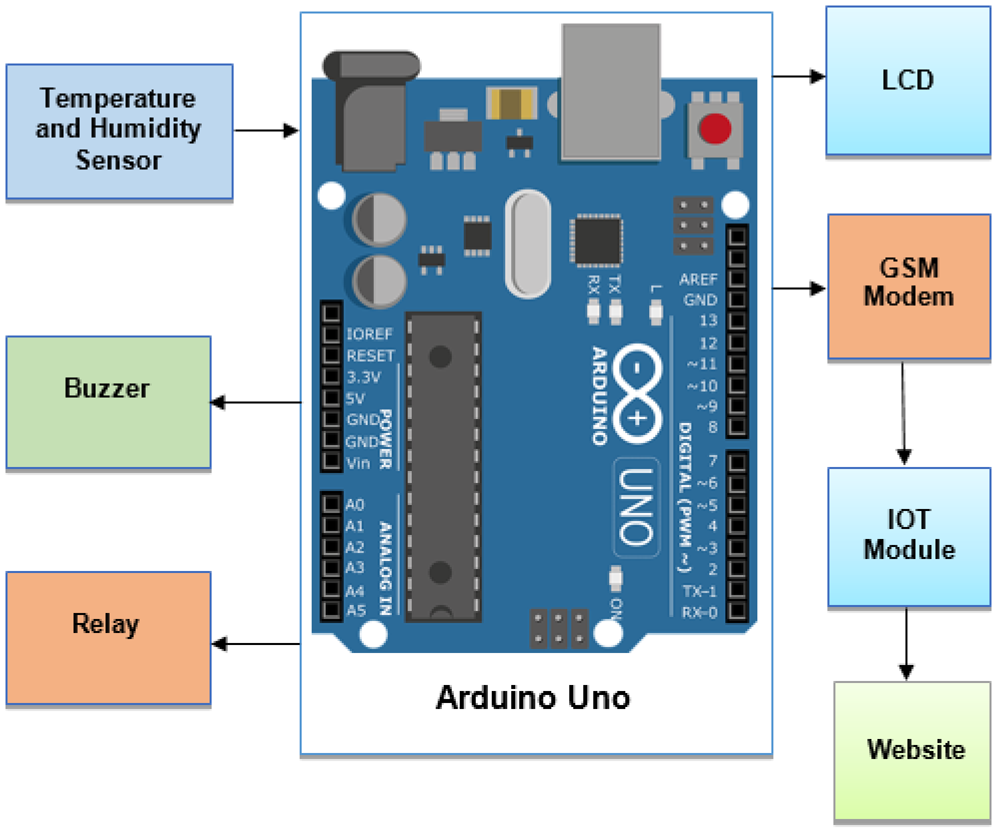
- We have used a DHT11 sensor in this project to monitor the temperature and humidity. DHT11 sensor continuously reads the temperature and humidity levels in the surrounding environment. DHT11 sensor gives the values of real-time temperature and humidity to Arduino Uno.
- Arduino uno displays these 2 values on a 16 by 2 LCD display.
- Arduino also checks if these 2 values cross their respective threshold values. If the threshold level is crossed then a buzzer and a relay are activated.
- SMS alert is sent to the user which contains the values of Temperature and/or Humidity.
- After a pre-defined periodic interval, the sensor data is sent over the cloud using IOT. The Arduino board establishes a connection to the internet using a GSM module. We have used the ThingSpeak IOT platform to receive, store, and display the values of temperature and humidity. Values are shown in the form of a Graph.
- GSM modem is used in this project which is used to send SMS as well as to send Temperature and Humidity data over IOT.

Software used in IOT based temperature and humidity monitoring system



1. We have used Arduino IDE to write the embedded program for the IOT temperature and humidity monitoring project.
2. Eagle PCB layout software is used to prepare the PCB layout of the IOT temperature and humidity monitoring system.

Block diagram of the IOT Temperature and Humidity monitoring system using Arduino project:



Conclusion:- Thus, we have studied an application to read temperature from the environment.

IOT Lab No.4

Assignment no.4

Title:- Write a program using Arduino to control LED (One or more ON/OFF). Or Blinking.

Objective:- To write a program using Arduino to control LED (One or more ON/OFF). Or Blinking. Material required:-

An LED is a simple diode that emits light in a forward bias. LED-blinking program on the Arduino IDE and download it to the microcontroller board. The program simply turns ON and OFF LED with some delay between them.
LEDs Light Emitting Diodes . When a voltage is given to a PN Junction Diode, electrons, and holes recombine in the PN Junction and release energy in the form of light (Photons). An LED's electrical sign is comparable to that of a PN Junction Diode. When free electrons in the conduction band recombine with holes in the valence band in forward bias, energy is released in the form of light.

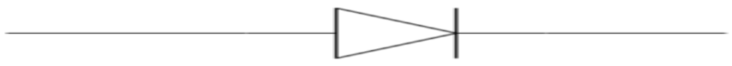


Figure – Diode

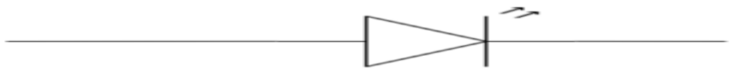
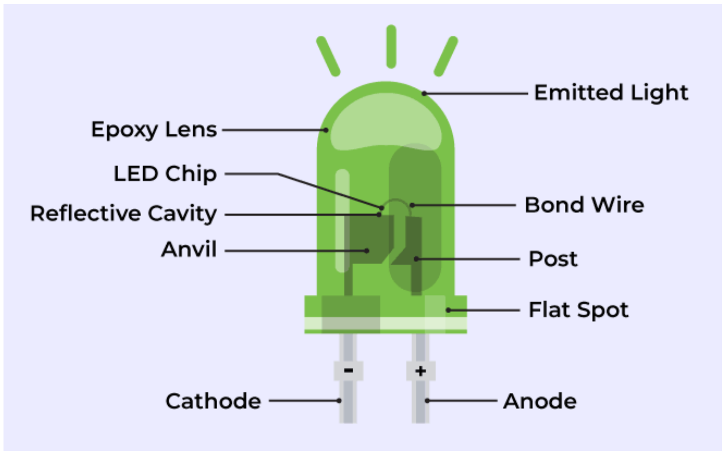


Figure – Light Emitting Diode (LED)

Structure of an LED



Structure of LED

The flow of charge carriers (electrons and holes) across the P-N junction drives the activity of an LED. When a forward voltage (anode positive in comparison to the cathode) is applied, electrons and holes recombine at the junction, releasing energy in the form of photons (light). The semiconductor chip is linked to external terminals known as the anode (+) and the cathode (-). The anode is linked to the P-region, and the cathode to the N-region.

Blinking an LED

Blinking an LED is an introductory Arduino project in which we control an LED using Arduino. LED blinking refers to the process of continuously turning an LED (Light Emitting Diode) and off in a repetitive pattern. It is a simple and common demonstration in electronics and microcontroller-based projects.

Working Procedure

setup() and loop() are two fundamental Arduino functions for controlling the behavior of your board. The Arduino framework automatically calls these functions, which form the foundation of any Arduino program.

The setup() function is only called once when the Arduino board boots up or is reset. Its goal is to set pin modes, initialize variables, and execute any other necessary setup tasks before the main loop begins. This function can be used to configure settings that should only be changed once over the board's lifespan.

The loop() function is the heart of an Arduino program. After the setup() function is executed, the loop() function starts running repeatedly until the Arduino is powered off or reset. It contains the main code that performs the desired tasks, controls the board, user input. Whatever is included in the loop() function will be executed in a continuous loop, allowing the Arduino to perform its intended functions continuously.

In the code, we have declared two integers, **LED pin** and **delayT. LEDpin** represents the pin number of the Arduino where LEDs need to be connected, and **delayT** is an integer variable for the delay() function. The delay() function accepts values in milliseconds.

Components Required

1. 1 X LED
2. 1 X Resistor, 330 Ohm
3. Breadboard
4. Arduino UNO R4 or earlier versions.
5. Jumper wires

Components Description

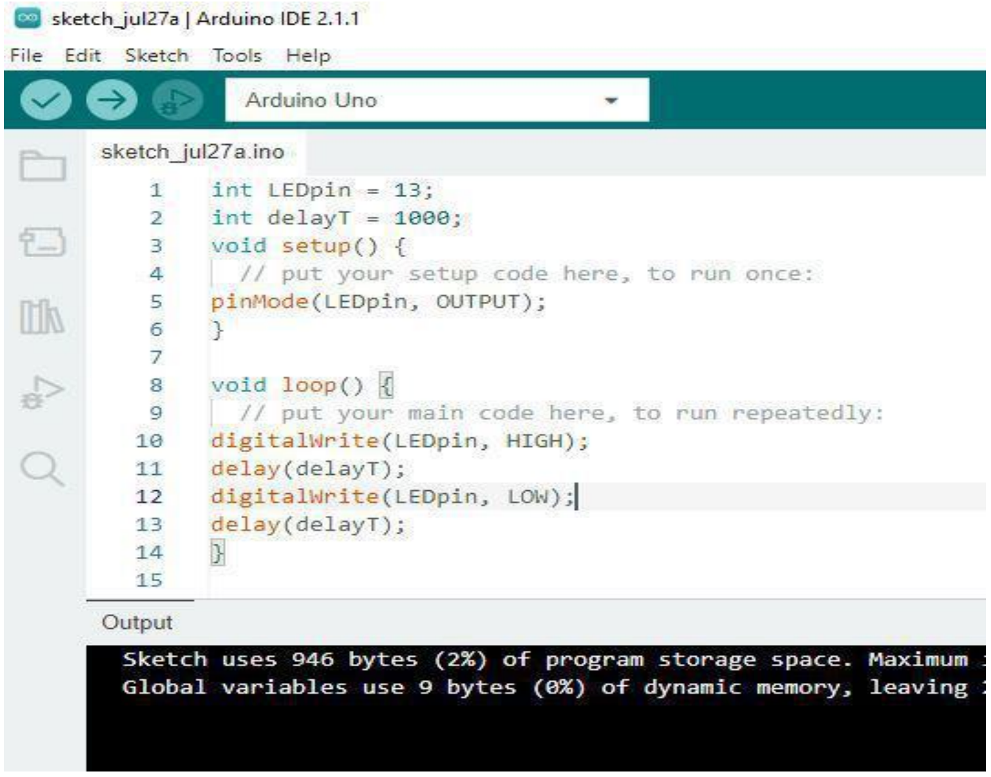
1. **1 X LED:** We are controlling only one LED in this program.
2. **1 X Resistor, 330 Ohm:** For every LED, we need one current limiting resistor.
3. **Breadboard:** A breadboard is a fundamental tool used in electronics and prototyping to build and test circuits without soldering.
4. **Arduino UNO R4** or earlier versions.
5. **Jumper wires:** Jumper wires are simple electrical wires with connectors on both ends used to create connections between various electronic components or points on a circuit on a breadboard.

Arduino Code

C++

```
int LEDpin = 13;
int delayT = 1000;
void setup() {
  // put your setup code here, to run once:
pinMode(LEDpin, OUTPUT);
}
void loop() {
  // put your main code here, to run repeatedly:
digitalWrite(LEDpin, HIGH);
delay(delayT);
digitalWrite(LEDpin, LOW);
delay(delayT);
}
```

Deployment Using Arduino IDE


sketch_jul27a | Arduino IDE 2.1.1
File Edit Sketch Tools Help
Arduino Uno
sketch_jul27a.ino
Output

```
int LEDpin = 13;
int delayT = 1000;
void setup() {
  // put your setup code here, to run once:
pinMode(LEDpin, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
digitalWrite(LEDpin, HIGH);
delay(delayT);
digitalWrite(LEDpin, LOW);
delay(delayT);
}
```

```
Sketch uses 946 bytes (2%) of program storage space. Maximum
Global variables use 9 bytes (0%) of dynamic memory, leaving
```

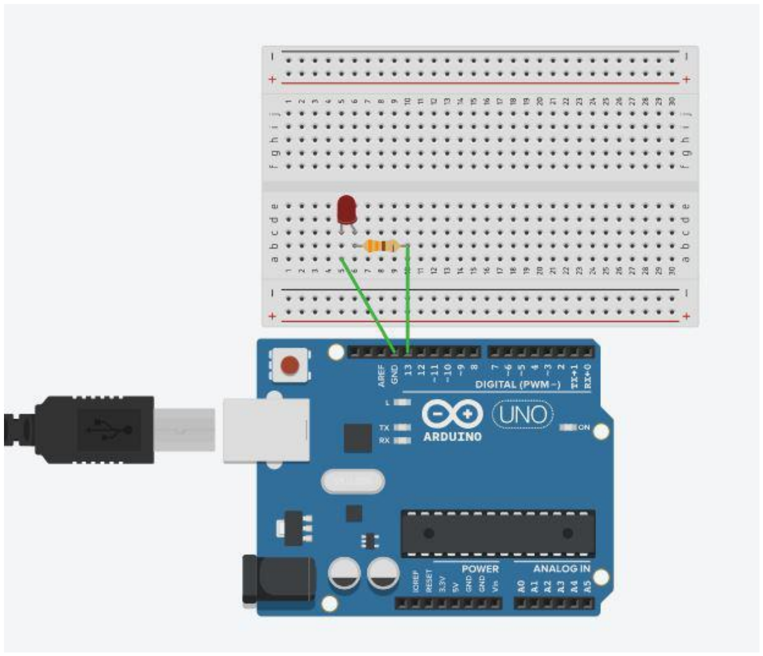
Circuit Diagram

In the circuit diagram, we used one 330-ohm resistor in series with the LED. This resistor is also called a current-limiting resistor. The Anode of the LED (the longer pin) is connected to one end of the resistor, and the cathode (the shorter pin) is connected to the ground. The other end of the resistor is connected to the Arduino pin. A step-by-step explanation is as follows:

1. **LED Connections:** Connect the LED to the breadboard. The LED has two legs, the longer of which is the anode (positive) and the shorter of which is the cathode (negative).
2. **Resistor Connection:** Insert one end of the resistor into the same row of the breadboard as the LED's Anode. The resistor's other end should be connected to the Arduino's digital output pin.
3. **Ground (GND) Connection:** Connect a jumper wire from the same row as the LED's cathode to any Arduino board GND (Ground) pin. This connects the circuit to the ground of the Arduino.

The circuit is now complete. Here's how it works:

When you upload a simple Arduino program that controls the LED, the microcontroller on the Arduino board executes the program, and the LED will blink according to the code you wrote.



Applications and Uses of LED Blinking

The LED blinking project is an important and straightforward method that can be utilized for a wide range of applications in microcontroller-based projects like :

- **Security Systems:** To check the status of security systems
- **Warning Signals:** In battery-operated devices, LED blinking can be used to indicate low battery levels.
- In Testing and debugging.
- **Status Indication:** LEDs can be used to indicate different states of a system. For example, in a home automation project, an LED might blink to indicate whether a device is connected to an internet network or not.

Conclusion

The Arduino LED blinking project provides a hands-on introduction to microcontroller's hardware interfaces, and programming ideas

IOT Lab No.5

Assignment no.5

Title:- Create a program so that when the user enters ‘b’ the green light blinks, ‘g’ the green light is illuminated ‘y’ the yellow light is illuminated and ‘r’ the red light is illuminated.

Objective:- To write a program to enters ‘b’ the green light blinks, ‘g’ the green light is illuminated ‘y’ the yellow light is illuminated and ‘r’ the red light is illuminated.

- **Hardware Requirement:** Arduino, LED, 220/100 ohm resistor etc.
- **Software Requirement**: Arduino IDE

- **Theory:**

The problem statement is like Arduino traffic light, a fun little project that you can build in under an hour. Here's how to build your own using an Arduino, and how to change the circuit for an advanced variation.

What You Need to Build an Arduino Traffic Light

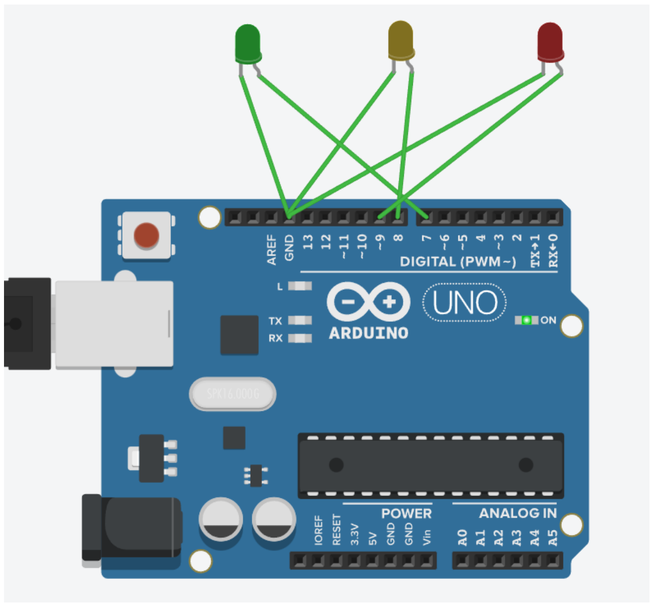
Controller Apart from the basic Arduino, you'll need:

- 1 x 10k-ohm resistor
- 1 x pushbutton switch
- 6 x 220-ohm resistors
- A breadboard
- Connecting wires
- Red, yellow and green

  LEDs Arduino Traffic

  Light: The Basics

Let's start small. A basic, single traffic light is a good place to start. Here's the circuit: Connect the anode (long leg) of each LED to digital pins eight, nine, and ten (via a 220- ohm resistor). Connect the cathodes (short leg) to the Arduino's ground.



```
char mychar;

void setup()
{
   Serial.begin(9600);
   pinMode(7,OUTPUT);
   pinMode(8,OUTPUT);
   pinMode(9,OUTPUT);
}

void loop()
{
   Serial.println("Enter which colour you want to Blink\n1.'b' for green colour \n2.'g' for yellow
colour\n3.'r' for red colour");

   while(Serial.available() >= 0)
   {
      mychar = Serial.read();
      if (mychar == 'b')
      {
         digitalWrite(7, HIGH);
         delay(100);
         digitalWrite(7, LOW);
         delay(100);
         digitalWrite(8, LOW);
         digitalWrite(9, LOW);
       }
       if (mychar == 'g')
       {
            digitalWrite(7, LOW);
            digitalWrite(8, HIGH);
            delay(100);
            digitalWrite(8, LOW);
            delay(100);
            digitalWrite(9, LOW);
            break;
          }
          if (mychar == 'r')
          {
             digitalWrite(7, LOW);
             digitalWrite(8, LOW);
             digitalWrite(9, HIGH);
             delay(100);
             digitalWrite(9, LOW);
             delay(100);
          }
     }
}
```

Conclusion:- Thus we have created a program that the user enters ‘b’ the green light illuminated, ‘g’ the green light is illuminated ‘y’ the yellow light is illuminated and ‘r’ the red light is illuminated.

IOT Lab No.6

Assignment no.6

Title:- Write a program that asks the user for a number and outputs the number squared that is entered.

Objective:- To write a program that asks the user for a number and outputs the number squared that is entered.

- **Outcome:** Connectivity, configuration and serial communication with Arduino.
- **Hardware Requirement:** Arduino, USB Cable etc.
- **Software Requirement**: Arduino IDE

Theory:

Arduino serial monitor for beginners in electronics. Send and receive data between the serial monitor window on a computer and an Arduino. The serial monitor is a utility that is part of the Arduino IDE. Send text from an Arduino board to the serial monitor window on a computer. In addition, send text from the serial monitor window to an Arduino board. Communications between the serial monitor and Arduino board takes place over the USB connection between the computer and Arduino.

- **Demonstration of the Arduino Serial Monitor for Beginners**

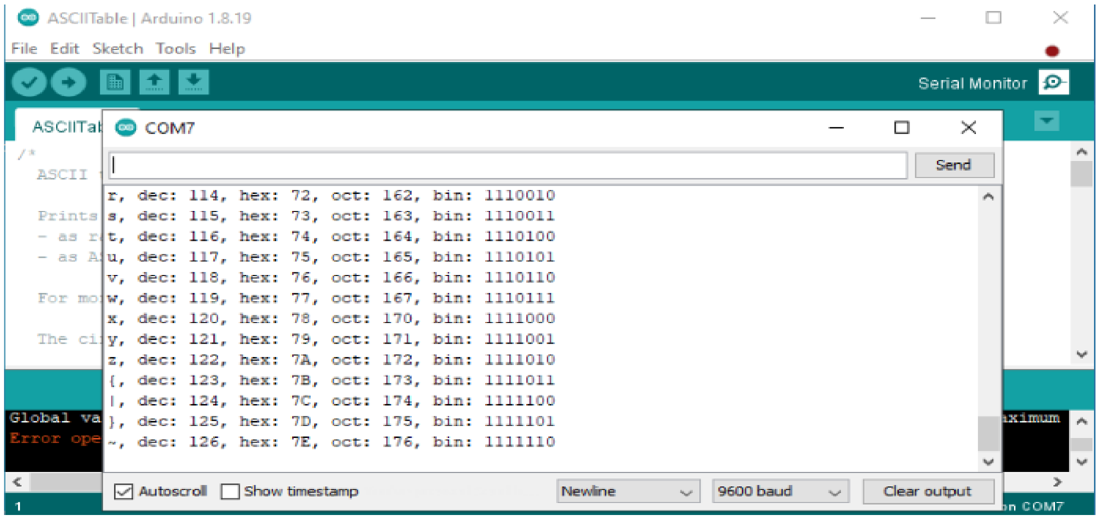
Part 2 of this Arduino tutorial for beginners shows how to install the Arduino IDE. In addition, it shows how to load an example sketch to an Arduino. It is necessary to know how to load a sketch to an Arduino board in this part of the tutorial. Therefore, first finish the previous parts of this tutorial before continuing with this part. A sketch loaded to an Arduino board demonstrates how the serial monitor works in the sub-sections that follow.

- **Load an Example Sketch that uses the Serial Monitor to an Arduino Board**

Start the Arduino IDE application. Select File → Examples → 04.Communication → ASCIITable from the top Arduino IDE menu bar. As a result, the ASCIITable example sketch opens in a new Arduino IDE window. Upload the ASCIITable example sketch to the Arduino Uno or MEGA 2560 board.
After the ASCIITable sketch is uploaded, nothing is seen to happen. This is because this example sketch sends text out of the USB port of the Arduino board. Because there is nothing running on the computer to receive this text, nothing is seen.

- **How to Open the Arduino Serial Monitor Window for Beginners**

The following image shows the location of the serial monitor window icon on the Arduino IDE toolbar. A red dot near the top right of the image shows the serial monitor toolbar icon location.



```
r, dec: 114, hex: 72, oct: 162, bin: 1110010
s, dec: 115, hex: 73, oct: 163, bin: 1110011
t, dec: 116, hex: 74, oct: 164, bin: 1110100
u, dec: 117, hex: 75, oct: 165, bin: 1110101
v, dec: 118, hex: 76, oct: 166, bin: 1110110
w, dec: 119, hex: 77, oct: 167, bin: 1110111
x, dec: 120, hex: 78, oct: 170, bin: 1111000
y, dec: 121, hex: 79, oct: 171, bin: 1111001
z, dec: 122, hex: 7A, oct: 172, bin: 1111010
{, dec: 123, hex: 7B, oct: 173, bin: 1111011
|, dec: 124, hex: 7C, oct: 174, bin: 1111100
}, dec: 125, hex: 7D, oct: 175, bin: 1111101
~, dec: 126, hex: 7E, oct: 176, bin: 1111110
```

Click the Serial Monitor icon near the top right of the Arduino IDE to open the serial monitor window. The above image shows the serial monitor window opened, and on top of the Arduino IDE window. Because the ASCIITable example is loaded on the Arduino board, when the serial monitor window opens, the Arduino sends text to the serial monitor window. This is also because opening the serial monitor window resets the Arduino board, causing the ASCIITable sketch to run from the beginning again.
The ASCIITable sketch sends text out of the USB port of the Arduino. Because the serial monitor is connected to the USB port, it receives the text and displays it in the big receive area of the window. As a result, text scrolls on the serial monitor window for a while. The text then stops because the Arduino has finished sending text. Use the right scrollbar in the serial monitor window to scroll up. Scrolling up reveals all of the text that the Arduino sent.

- **What to do When Junk Characters are Displayed**

When junk, or garbage characters, or even nothing is displayed in the serial monitor, it is usually because of an incorrect baud rate setting. Look at the bottom of the serial monitor in the above image. Notice the value 9600 baud in a box. This is the baud setting of communications between the Arduino and serial monitor. The ASCIITable, and most other built-in example sketches, set the Arduino to communicate at 9600 baud. If your serial monitor window shows a different baud rate, change it to 9600 baud. Do this by clicking the baud drop-down list. Select 9600 baud on the list that drops down.

- **Reset the Arduino Board with the RESET Button**

Press and release the RESET button on the Arduino board and the ASCIITable sketch runs from the beginning again. As a result of the reset, the same text scrolls down the serial monitor window and then stops again. The RESET button is the only push button on the Arduino Uno or MEGA 2560.
Pushing the RESET button in holds the board in reset. This means that the sketch currently loaded on the board stops running. Releasing the RESET button takes the board out of reset. As a result, the sketch currently loaded on the Arduino starts running from the beginning again.

- **Clear the Serial Monitor Window Receive Area**

The red dot in the image below shows the location of the Clear output button at the bottom of the serial monitor window. Click the Clear output button and text is cleared from the receive area of the serial monitor window. Reset the Arduino, and the receive area fills with text from the ASCIITable sketch again.
Serial Monitor Window Clear Output Button

- **What the ASCIITable Sketch Does**

ASCII stands for American Standard Code for Information Interchange. ASCII is a standard way that uses numbers to represent various characters. For example, the decimal number 65 represents the letter A. Another example is the decimal number 125 represents a closing brace: }. This allows computers to send and receive text by sending and receiving numbers. For example when a computer receives the number 65, it knows to display the letter A.
The ASCIITable sketch sends the numbers 33 through to 126 out of the USB port. This results in the printable text characters from the ASCII table displayed in the serial monitor window. In addition to the ASCII characters, the number that represents each character is displayed. Each number is shown in four different numbering systems. These are the decimal, hexadecimal, octal and binary number systems. In the serial monitor window, these number systems are abbreviated to dec, hex, oct and bin.

Conclusion: Thus we have create a program that asks the user for a number and outputs the number squared that is entered.

IOT Lab No.7

Assignment no.7

Title:- Write a program to control the color of the LED by turning 3 different potentiometers. One will be read for the value of Red, one for the value of Green, and one for the value of Blue.

Objective:- To write a program to control the color of the LED by turning 3 different potentiometers. One will be read for the value of Red, one for the value of Green, and one for the value of Blue.

- **Outcome:** Connectivity, configuration and control of LED using Arduino circuit under different conditions.
- **Hardware Requirement:** Arduino, LED, 220 ohm resistor etc.
- **Software Requirement**: Arduino IDE

- **Theory:**

The problem statement is like Arduino traffic light, a fun little project that you can build in under an hour. Here's how to build your own using an Arduino, and how to change the circuit for an advanced variation.

What You Need to Build an Arduino Traffic Light

Controller Apart from the basic Arduino, you'll need:

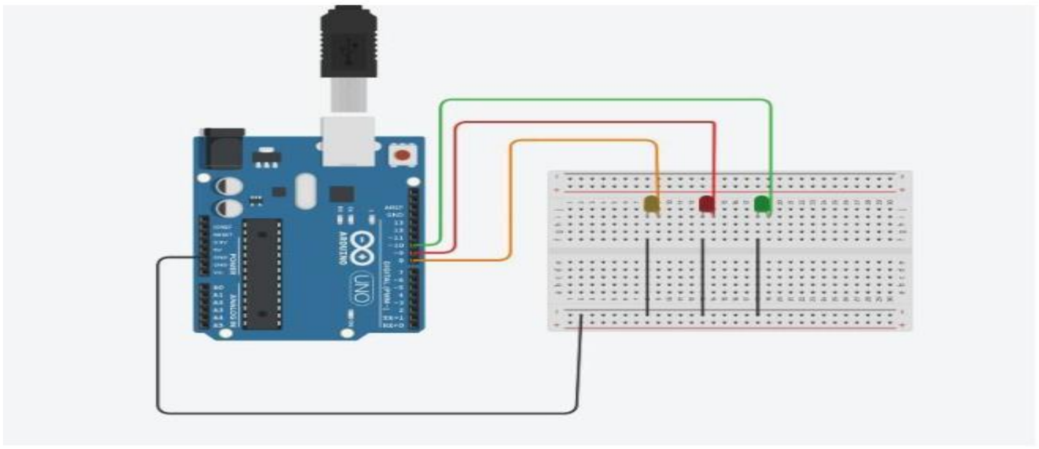
- 1 x 10k-ohm resistor
- 1 x pushbutton switch
- 6 x 220-ohm resistors
- A breadboard
- Connecting wires
- Red, yellow and green

  LEDs Arduino Traffic

  Light: The Basics

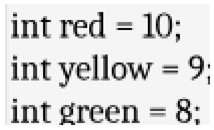
Let's start small. A basic, single traffic light is a good place to start. Here's the circuit:

Connect the anode (long leg) of each LED to digital pins eight, nine, and ten (via a 220- ohm resistor). Connect the cathodes (short leg) to the Arduino's ground.



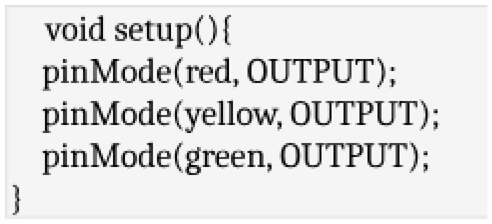
Code for the Arduino Traffic Light

Start by defining variables so that you can address the lights by name rather than a number. Start a new Arduino project, and begin with these lines:



```
int red = 10;
int yellow = 9;
int green = 8;
```

Next, let’s add the setup function, where you'll configure the red, yellow and green LEDs to be outputs. Since you have created variables to represent the pin numbers, you can now refer to the pins by name instead:



```
  void setup(){
  pinMode(red, OUTPUT);
  pinMode(yellow, OUTPUT);
  pinMode(green, OUTPUT);
}
```

The **pinMode** function configures the Arduino to use a given pin as an output. You have to do this for your LEDs to work at all. Now for the actual logic of the traffic light. Here's the code you need. Add this below your variable definitions and setup function:

```
 void loop(){
 changeLights();
 delay(15000);
}
void changeLights(){
     // green off, yellow on for 3
 seconds digitalWrite(green, LOW);
 digitalWrite(yellow, HIGH);
 delay(3000);
```

```
    // turn off yellow, then turn red on for 5 seconds
 digitalWrite(yellow, LOW);
    digitalWrite(red,
 HIGH); delay(5000);
    // red and yellow on for 2 seconds (red is already on though)
 digitalWrite(yellow, HIGH);
    delay(2000);
    // turn off red and yellow, then turn on
 green digitalWrite(yellow, LOW);
    digitalWrite(red, LOW);
 digitalWrite(green, HIGH);
 delay(3000);
}
```

Upload this code to your Arduino, and run (make sure to select the correct board and port from the **Tools** > Board and **Tools** > **Port** menus). You should have a working traffic light that changes every 15 seconds, like this (sped up):

Let's break down this code. The **changeLights** function performs all the hard work. This rotates the traffic light through yellow and red, then back to green. As this gets called inside the **loop** function, the Arduino will run this code forever, with a 15-second pause every time.

The **changeLights** function consists of four distinct steps:

- Green on, yellow off
- Yellow off, red on
- Yellow on, red on
- Green on, red off, yellow off

These four steps replicate the process used in real traffic lights. For each step, the code is very similar. The appropriate LED gets turned on or off using **digitalWrite**. This is an Arduino function used to set output pins to HIGH (for on), or LOW (for off).

After enabling or disabling the required LEDs, the **delay** makes the Arduino wait for a given amount of time. Three seconds in this case.

Going Deeper: Arduino Pedestrian Crossing

Now that you know the basics, let's improve it. Add in a pushbutton for pedestrians to change the light whenever they like:

Notice how the traffic light is exactly the same as the previous example. Connect the button to digital pin 12. You'll notice that the switch has a high-impedance 10k-ohm resistor attached to it, and you may be wondering why. This is a pull-down resistor.

A switch either lets the current flow or doesn't. This seems simple enough, but in a logic circuit, the current should be always flowing in either a high or low state (remember, 1 or 0, HIGH or LOW). You might assume that a pushbutton switch that isn't actually pressed would be in a LOW state, but in fact, it's said to be 'floating', because no current gets drawn at all.

In this floating state, it's possible that a false reading will occur as it fluctuates with electrical interference. In other words, a floating switch is giving neither a reliable HIGH nor LOW reading. A pull-down resistor keeps a small amount of current flowing when the switch gets closed, thereby ensuring an accurate low state reading.

In other logic circuits, you may find a pull-up resistor instead, and this works on the same principle, but in reverse, making sure that particular logic gate defaults to high.

Now, in the loop part of the code, instead of changing the lights every 15 seconds, you're going to read the state of the pushbutton switch instead, and only change the lights when it's activated.

Code for the Arduino Pedestrian Crossing

Start by adding a new variable to store your button pin:

```
int button = 12; // switch is on pin 12
```

Now, in the setup function, add a new line to declare the switch as an input. Add a line to set the traffic lights to the green stage. Without this initial setting, they would off until the first time **changeLights** runs.

```
pinMode(button, INPUT);
digitalWrite(green, HIGH);
```

Change the entire loop function to the following instead:

```
void loop() {
        if (digitalRead(button) ==
     HIGH){ delay(15); // software
     debounce if (digitalRead(button)
     == HIGH) {
        // if the switch is HIGH, ie. pushed down - change the lights!
        changeLights();
        delay(15000); // wait for 15 seconds
     }
  }
}
```

That should do it. You may be wondering why the button checking happens twice (**digitalRead(button)**), separated by a small delay. This is debouncing. Much like the pull-down resistor for the button, this simple check stops the code detecting minor interference as a button press.

By waiting inside the **if** statement for 15 seconds, the traffic lights can't change for at least that duration. Once 15 seconds is over the loop restarts. Each restart of the loop, it reads the state of the button again, but if it isn't pressed, the **if** statement never activates, the lights never change, and the program restarts again.

Here's how this looks (sped

up): Arduino Traffic Light with

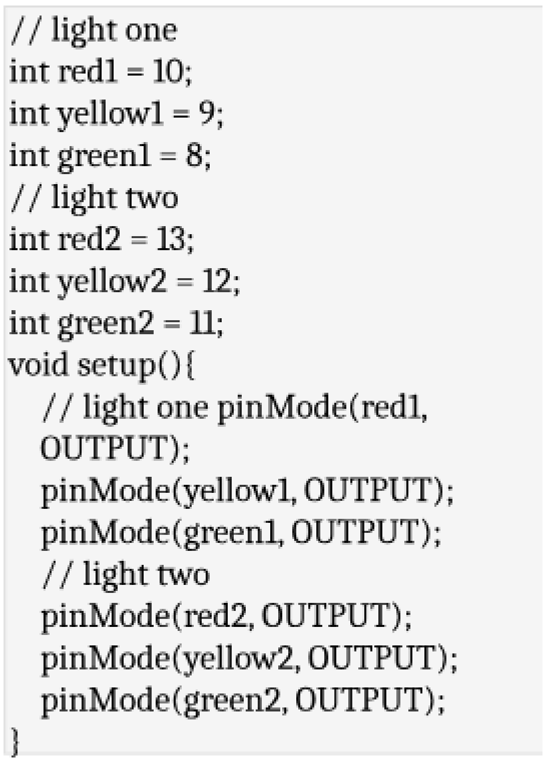
Junction

Let's try a more advanced model. Instead of a pedestrian crossing, change your circuit to have two traffic lights:

Connect the second traffic light to digital pins 11, 12, and

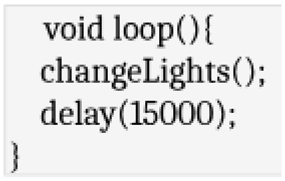
13. Code for the Arduino Traffic Light with Junction

First, assign your new traffic light pins to variables, and configure them as outputs, like in the first example:



```
// light one
int red1 = 10;
int yellow1 = 9;
int green1 = 8;
// light two
int red2 = 13;
int yellow2 = 12;
int green2 = 11;
void setup(){
   // light one pinMode(red1,
   OUTPUT);
   pinMode(yellow1, OUTPUT);
   pinMode(green1, OUTPUT);
   // light two
   pinMode(red2, OUTPUT);
   pinMode(yellow2, OUTPUT);
   pinMode(green2, OUTPUT);
}
```

Now, update your loop to use the code from the first example (instead of the pedestrian crossing):



```
  void loop(){
  changeLights();
  delay(15000);
}
```

Once again, all the work is carried out in the **changeLights** function. Rather than going **red** > **red** & **yellow** > **green**, this code will alternate the traffic lights. When one is on green, the other is on red. Here's the code:

```
void changeLights(){
      // turn both yellows on
   digitalWrite(green1, LOW);
   digitalWrite(yellow1, HIGH);
   digitalWrite(yellow2, HIGH);
   delay(5000);
      // turn both yellows off, and opposite green and red
   digitalWrite(yellow1, LOW);
      digitalWrite(red1, HIGH);
   digitalWrite(yellow2, LOW);
   digitalWrite(red2, LOW);
   digitalWrite(green2, HIGH);
```

```
     delay(5000);
     // both yellows on again
  digitalWrite(yellow1, HIGH);
  digitalWrite(yellow2, HIGH);
  digitalWrite(green2, LOW);
  delay(3000);
     // turn both yellows off, and opposite green and red
  digitalWrite(green1, HIGH);
     digitalWrite(yellow1,
  LOW); digitalWrite(red1,
  LOW); digitalWrite(yellow2,
  LOW); digitalWrite(red2,
  HIGH); delay(5000);
}
```

Conclusion: - Thus we have write a program to control the color of the LED by turning 3 different potentiometers, for the value of Red, Green, and Blue.

IOT Lab No.8

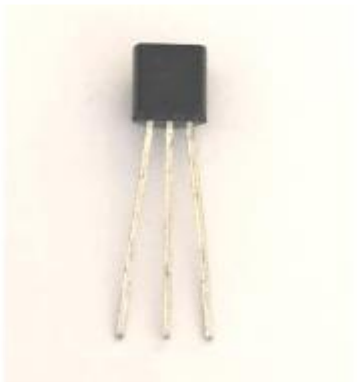
Assignment no.8

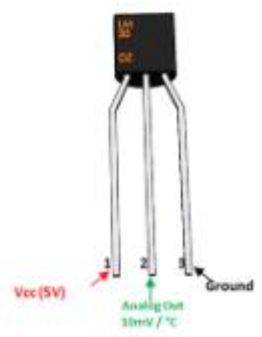
Title:- Write a program read the temperature sensor and send the values to the serial monitor on the computer.

Objective:- To write a program read the temperature sensor and send the values to the serial monitor on the computer.

Outcome: Understanding working principle of DHT11, LM35 temperature sensor.

- **Hardware Requirement:** Arduino, LED, LM35, DHT11, etc
- **Software Requirement**: Arduino IDE
- **Theory:** **LM35 Temperature Sensor**





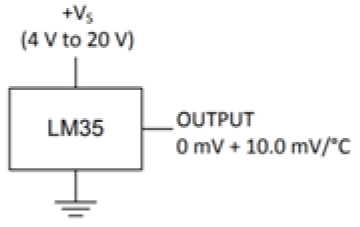
LM35 Temperature Sensor Pinout
LM35 Sensor Pinout Configuration

Pin Number	Pin Name	Description
1	Vcc	Input voltage is +5V for typical applications
2	Analog Out	There will be increase in 10mV for raise of every 1°C. Can range from -1V(-55°C) to 6V(150°C)
3	Ground	Connected to ground of circuit

- **LM35 Sensor Features**
  - Minimum and Maximum Input Voltage is 35V and -2V respectively. Typically 5V.
  - Can measure temperature ranging from -55°C to 150°C
  - Output voltage is directly proportional (Linear) to temperature (i.e.) there will be a rise of 10mV (0.01V) for every 1°C rise in temperature.
  - ±0.5°C Accuracy
  - Drain current is less than 60uA
  - Low cost temperature sensor
  - Small and hence suitable for remote applications
  - Available in TO-92, TO-220, TO-CAN and SOIC package

- **LM35 Temperature Sensor Equivalent**
  LM34, DS18B20, DS1620, LM94022

  How to use LM35 Temperature Sensor:
  LM35 is a precession Integrated circuit Temperature sensor, whose output voltage varies, based on the temperature around it. It is a small and cheap IC which can be used to measure temperature anywhere between -55°C to 150°C. It can easily be interfaced with any Microcontroller that has ADC function or any development platform like Arduino.
  Power the IC by applying a regulated voltage like +5V (VS) to the input pin and connected the ground pin to the ground of the circuit. Now, you can measure the temperature in form of voltage as shown below.



  If the temperature is 0°C, then the output voltage will also be 0V. There will be a rise of 0.01V (10mV) for every degree Celsius rise in temperature. The voltage can be converted into temperature using the below formulae.

V_{OUT} = 10 mv/°C × T

where

- V_{OUT} is the LM35 output voltage
- T is the temperature in °C

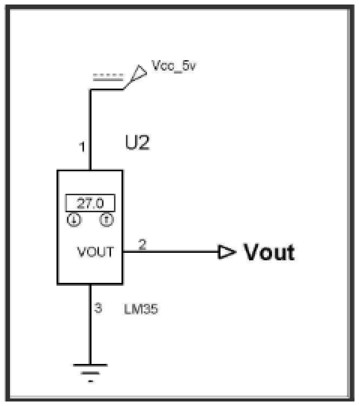
LM35 Temperature Sensor Applications

- Measuring temperature of a particular environment
- Providing thermal shutdown for a circuit/component
- Monitoring Battery Temperature
- Measuring Temperatures for HVAC applications.

-

- **How Does LM35 Sensor Work?**
  Main advantage of LM35 is that it is linear i.e. 10mv/°C which means for every degree rise in temperature the output of LM35 will rise by 10mv. So if the output of LM35 is 220 mv/0.22V the temperature will be 22°C. So if room temperature is 32°C then the output of LM35 will be 320mv i.e. 0.32V.

- **LM35 Interfacing Circuit**

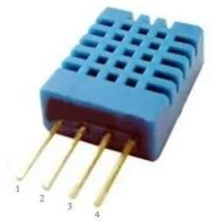


As such no extra components are required to interface LM35 to ADC as the output of LM35 is linear with 10mv/degree scale. It can be directly interfaced to any 10 or 12 bit ADC. But if you are using an 8-bit ADC like ADC0808 or ADC0804 an amplifier section will be needed if you require to measure 1°C change.
LM35 can also be directly connected to Arduino. The output of LM35 temperature can also be given to comparator circuit and can be used for over temperature indication or by using a simple relay can be used as a temperature controller.

- **DHT11 interfacing with arduino and weather station**
  DHT11 sensor is used to measure the **temperature** and **humidity**. It has a resistive humidity sensing component and a negative temperature coefficient (NTC). An 8 bit MCU is also connected in it which is responsible for its fast response. It is very inexpensive but it gives values of both temperature and humidity at a time.

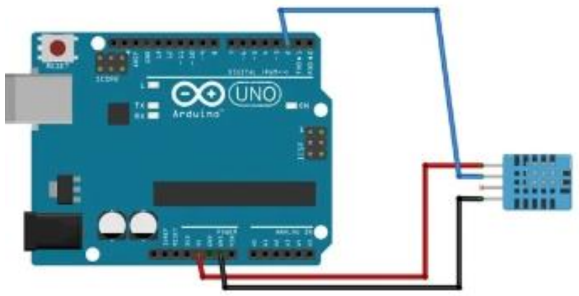
- **Specification of DHT11**
  - It has humidity range from 20 to 90% RH
  - It has temperature range from 0 – 50 C
  - It has signal transmission range of 20 m
  - It is inexpensive
  - It has fast response and it is also durable

- **DHT11 Pin out**



  - The first pin of the DHT11 is vcc pin.
  - The second pin of the DHT is Data pin.
  - The third pin is not used.
  - The fourth pin of the DHT sensor is ground.

- **DHT11 interfacing with arduino**
  First of all connect the ground and the VCC of the DHT11 temperature and humidity sensor to the ground and 5v of the **Arduino**. Then connect the data pin of the DHT11 sensor to the pin 2 of the Arduino.



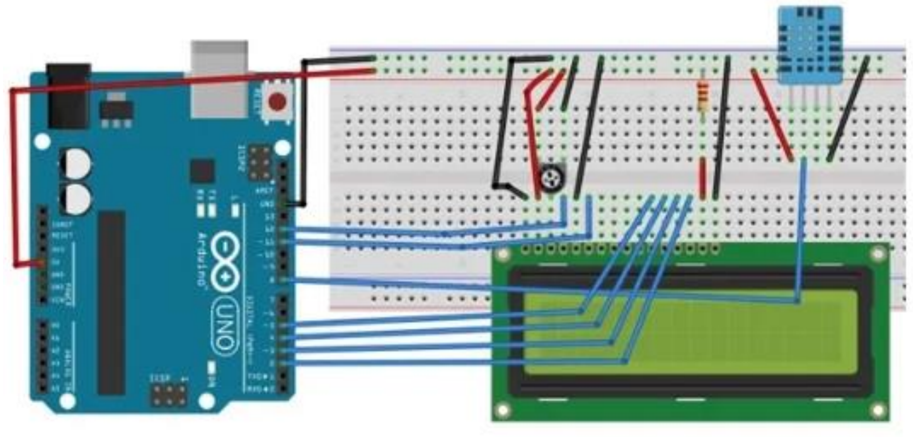
- **Installing the DHT11 Library**
  To run the following code in Arduino IDE you will first have to install the DHT library in you Arduino directory.
  Download the zip file from here and place it in your Arduino library folder. The path to Arduino library folder for my computer is
  Documents/ Arduino/ Libraries
  Unzip the downloaded file and place it in this folder.
  After copying the files, the Arduino library folder should have a new folder named DHT containing the dht.h and dht.cpp. After that copy the following code in the Arduino IDE and upload the code.

- **Code of DHT11 interfacing with arduino**

```
// Code for DHT11 Temperature and humidity sensor.
 #include " DHT.h "
 #define DHTPIN 2
 #define DHTTYPE DHT11
 DHT dht ( DHTPIN, DHTTYPE ) ;
 void setup ()
{
   Serial.begin ( 9600 ) ;
   dht.begin (  ) ;
}
 void loop ( ) {
 float humidity = dht.readHumidity ( ) ;
 float temp = dht.readTemperature ( ) ;
 if ( isnan ( t ) || isnan ( h ) ) {
            Serial.println ( " Sensor not working " ) ;
   }
 else
{


            Serial.print ( " Temp is " ) ;
            Serial.print ( temp ) ;
            Serial.println ( " *C " ) ;
            Serial.print ( " Humidity in % is : " ) ;
            Serial.print ( humidity ) ;
            Serial.print ( " % \t " ) ;
   }
  }
```

- **Weather Station using DHT11 and arduino**
  In this example we will make a weather station that will sense the humidity and temperature and will show it on the lcd attached to the Arduino. Make the circuit as shown in the diagram. The resistor in the circuit will make the black light darker. We have used the 220 ohm resistor but you can use any resistor having value near to that. The potentiometer we used in the circuit is used to set the screen contrast. We have used the 10 K ohm value but you can choose any value relative to that one.



- **Components Required**
    - Arduino Uno (you can use any)
    - 16 x 2 LCD
    - DHT11 Temperature and humidity sensor
    - 10 K ohm potentiometer
    - 220 ohm resistor

- **Code of weather station using arduino and DHT11**

```
// This code is for the weather station using the DHT11 humidity and temperature sensor.
// Install the library of the DHT before uploading the code in the Arduino IDE
#include < dht.h >
#include < LiquidCrystal.h >
LiquidCrystal lcd ( 12, 11, 5, 4, 3, 2 ) ;
dht DHT ;                   // declaring dht a variable
#define DHT11_PIN 8           // initializing pin 8 for dht
void setup ( ) {
  lcd.begin ( 16, 2 ) ;       // starting the 16 x 2 lcd
}
void loop ( )
{
int chk = DHT.read11(DHT11_PIN );
lcd.setCursor ( 0, 0 ) ;
lcd.print ( " Temperature is : " ) ;
  lcd.print ( DHT.temperature ) ;
  lcd.print ( ( char ) 223 ) ;
  lcd.print ( " C " ) ;
  lcd.setCursor ( 0 , 1 );
  lcd.print ( " Humidity is : " ) ;
  lcd.print ( DHT.humidity ) ;
  lcd.print ( " % " ) ;
  delay ( 1000 ) ;
}
```

Conclusion: - Thus,we have write a program read the temperature sensor and values send to the serial monitor on the computer.

IOT Lab No.9

Assignment no.9

Title:- Write a program so it displays the temperature in Fahrenheit as well as the maximum and minimum temperatures it has seen.

Objective:- To write program so it displays the temperature in Fahrenheit as well as the maximum and minimum temperatures it has seen.

- **DHT11 interfacing with arduino and weather station**

DHT11 sensor is used to measure the **temperature** and **humidity**. It has a resistive humidity sensing component and a negative temperature coefficient (NTC). An 8 bit MCU is also connected in it which is responsible for its fast response. It is very inexpensive but it gives values of both temperature and humidity at a time.

- **Specification of DHT11**
  - It has humidity range from 20 to 90% RH
  - It has temperature range from 0 – 50 C
  - It has signal transmission range of 20 m
  - It is inexpensive
  - It has fast response and it is also durable

- **DHT11 Pin out**

  - The first pin of the DHT11 is vcc pin.
  - The second pin of the DHT is Data pin.
  - The third pin is not used.
  - The fourth pin of the DHT sensor is ground.

- **DHT11 interfacing with arduino**

First of all connect the ground and the VCC of the DHT11 temperature and humidity sensor to the ground and 5v of the **Arduino**. Then connect the data pin of the DHT11 sensor to the pin 2 of the Arduino.

- **Installing the DHT11 Library**

  To run the following code in Arduino IDE you will first have to install the DHT library in you Arduino directory.
  Download the zip file from github and place it in your Arduino library folder.

- **Code of DHT11 interfacing with Arduino**

```
// Code for DHT11 Temperature and humidity sensor.
 #include " DHT.h "
 #define DHTPIN 2
 #define DHTTYPE DHT11         // Selecting the type of DHT sensors
 DHT dht ( DHTPIN, DHTTYPE ) ;
 void setup ( ) {
   Serial.begin ( 9600 ) ;
   dht.begin (  ) ;    // The sensor will start working
 }
 void loop ( ) {
   float humidity = dht.readHumidity ( ) ;
  float temp = dht.readTemperature ( ) ;
     if ( isnan ( t ) || isnan ( h ) ) {
            Serial.println ( " Sensor not working " ) ;
   }
 else
{


            Serial.print ( " Temp is " ) ;
            Serial.print ( temp )
            Serial.println ( " *C " ) ;     // Printing “ *C ”  on display.
            Serial.print ( " Humidity in % is : " ) ;
            Serial.print ( humidity ) ;     // Printing the humidity on display
            Serial.print ( " % \t " ) ;     // Printing “%” on display

   }
 }
```

- **Weather Station using DHT11 and arduino**

In this example we will make a weather station that will sense the humidity and temperature and will show it on the lcd attached to the Arduino. Make the circuit as shown in the diagram. The resistor in the circuit will make the black light darker. We have used the 220 ohm resistor but you can use any resistor having value near to that. The potentiometer we used in the circuit is used to set the screen contrast. We have used the 10 K ohm value but you can choose any value relative to that one.

- **Components Required**
  - Arduino Uno (you can use any)
  - 16 x 2 LCD
  - DHT11 Temperature and humidity sensor
  - 10 K ohm potentiometer
  - 220 ohm resistor

- **Code of weather station using arduino and DHT11**

```
// This code is for the weather station using the DHT11 humidity and temperature sensor.
// Install the library of the DHT before uploading the code in the Arduino IDE
#include < dht.h >
#include < LiquidCrystal.h >
LiquidCrystal lcd (12, 11, 5, 4, 3, 2 ) ;
dht DHT ;
#define DHT11_PIN 8
void setup () {
  lcd.begin ( 16, 2 ) ;
}
void loop ()
{
  int chk = DHT.read11(DHT11_PIN ) ;
  lcd.setCursor ( 0, 0 ) ;
  lcd.print ( " Temperature is : " ) ;
  lcd.print ( DHT.temperature ) ;
  lcd.print ( ( char ) 223 ) ;
  lcd.print ( " C " ) ;
  lcd.setCursor ( 0 , 1 );
  lcd.print ( " Humidity is : " ) ;
  lcd.print ( DHT.humidity ) ;
  lcd.print ( " % " ) ;
  delay ( 1000 ) ;


}
```

Temperature Scales

Thermometers measure temperature according to well-defined scales of measurement. The three most common temperature scales are the Fahrenheit, Celsius, and Kelvin scales.

- **Celsius Scale & Fahrenheit Scale**

The Celsius scale has a freezing point of water as 0ºC and the boiling point of water as 100ºC. On the Fahrenheit scale, the freezing point of water is at 32ºF and the boiling point is at 212ºF. The temperature difference of one degree Celsius is greater than a temperature difference of one degree Fahrenheit. One degree on the Celsius scale is 1.8 times larger than one degree on the Fahrenheit scale 180/100=9/5.

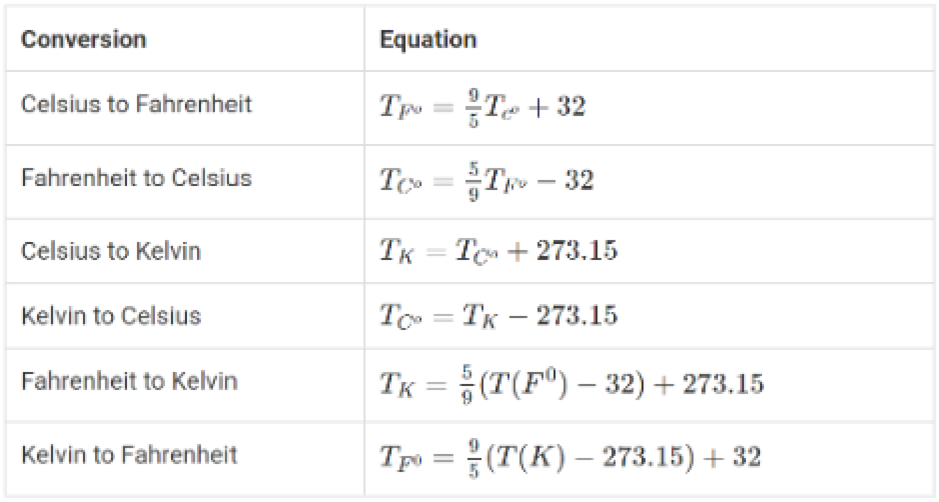
- **Kelvin Scale**

Kelvin scale is the most commonly used temperature scale in science. It is an absolute temperature scale defined to have 0 K at the lowest possible temperature, called absolute zero. The freezing and boiling points of water on this scale are 273.15 K and 373.15 K, respectively. Unlike other temperature scales, the Kelvin scale is an absolute scale. It is extensively used in scientific work. The Kelvin temperature scale possesses a true zero with no negative temperatures. It is the lowest temperature theoretically achievable and is the temperature at which the particles in a perfect crystal would become motionless.

- **Relationship Between Different Temperature Scales**

The relationship between three temperature scales is given in the table below:

Relationship between different Temperature Scales



Conversion	Equation
Celsius to Fahrenheit	$T_{F^o} = \frac{9}{5}T_{C^o} + 32$
Fahrenheit to Celsius	$T_{C^o} = \frac{5}{9}T_{F^o} - 32$
Celsius to Kelvin	$T_K = T_{C^o} + 273.15$
Kelvin to Celsius	$T_{C^o} = T_K - 273.15$
Fahrenheit to Kelvin	$T_K = \frac{5}{9}(T(F^0) - 32) + 273.15$
Kelvin to Fahrenheit	$T_{F^0} = \frac{9}{5}(T(K) - 273.15) + 32$

Conclusion:- Thus, we have write a program to displays the temperature in Fahrenheit as well as the maximum and minimum temperatures .

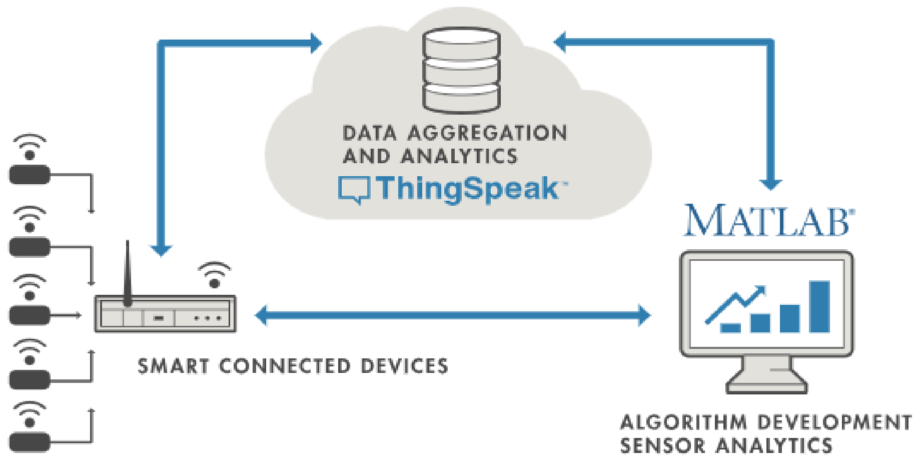
IOT Lab No.10

Assignment no.10

Title:- Study of ThingSpeak – an API and Web Service for the Internet of Things.

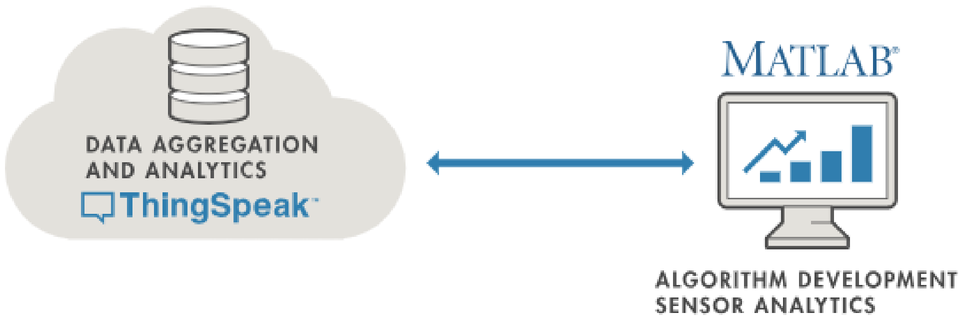
Theory:- ThingSpeak is a platform providing various services exclusively targeted for building IoT applications. It offers the capabilities of real-time data collection, visualizing the collected data in the form of charts, ability to create plugins and apps for collaborating with web services, social network and other APIs. We will consider each of these features in detail below. The core element of ThingSpeak is a 'ThingSpeak Channel'. A channel stores the data that we send to ThingSpeak and comprises of the below elements: • 8 fields for storing data of any type - These can be used to store the data from a sensor or from an embedded device. • 3 location fields - Can be used to store the latitude, longitude and the elevation. These are very useful for tracking a moving device. • 1 status field - A short message to describe the data stored in the channel. To use ThingSpeak, we need to signup and create a channel. Once we have a channel, we can send the data, allow ThingSpeak to process it and also retrieve the same. Let us start exploring ThingSpeak by signing up and setting up a channel.

.



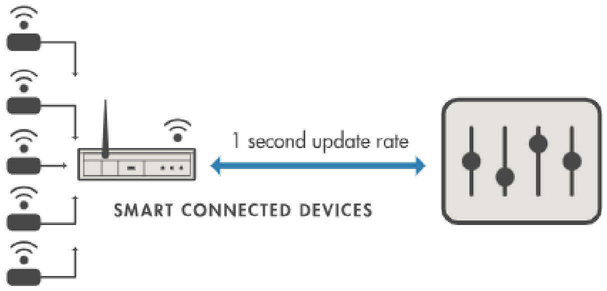
We can connect any internet-connected device with Thingspeak. When sending data from device or equipment, we can use native libraries for common embedded hardware prototyping platforms like Arduino, from machines or local gateways using REST API or an MQTT API

Access Your Data Both Online and Offline
ThingSpeak stores all the information that sends it in one central location in the cloud, so that can easily access   data for online or offline analysis. Your private data is protected with an API key that can control. When you are logged in to your ThingSpeak account, you can use the web to securely download the data stored in the cloud. We can also programmatically read data in CSV or JSON formats using a REST API call and the appropriate API key. Devices can also read data from a ThingSpeak channel by subscribing to an MQTT topic. Import data from third-party web services including climate data from NOAA, public utility data from local utility providers, and stock and pricing data from financial providers. You can use that data together with the data you are collecting from your devices and equipment to investigate correlations and develop predictive algorithms.



ThingSpeak automatically charts the data that we send it, so you can remotely monitor your devices or equipment from anywhere.View your data from any web browser or mobile device. Share read-only views of your data with the clients and colleagues that you specify. Alternatively, you can ThingSpeak to manage your data and you can build your clients and customers to log in to

ThingSpeak as fast as once every second. This not only enables near-real time monitoring of your devices, but it allows you to set up control loops from the cloud



Conclusion: We have studied about ThingSpeak in terms of access data in online and offline mode, visualize remotely, computations.