RAJYA VOKKALIGARA SANGHA (R)

BANGALORE INSTITUTE OF TECHNOLOGY

K.R Road, V.V Pura, Bengaluru - 560 004 | www.bit-bangalore.edu.in

Recognised By All India Council for Technical Education (AICTE), New Delhi

Affiliated to Visvesvaraya Technological University (VTU), Belagavi

Accredited by National Board of Accreditation UG Programs (9)

Accredited by National Assessment and Accreditation Council (NAAC)

ARIIA ATAL RANKING OF INSTITUTIONS ON INNOVATION ACHIEVEMENTS

QS I-GAUGE - GOLD

INSTITUTION'S INNOVATION COUNCIL (Ministry of HRD Initiative)

INTERNET OT THINGS LAB MANUAL

ಬೆಂಗಳೂರು ತಾಂತ್ರಿಕ ಮಹಾವಿದ್ಯಾಲಯ
BANGALORE INSTITUTE OF TECHNOLOGY
BIT

Department of Electronics & Telecommunication Engineering

IoT (Internet of Things) Lab Manual (21EC581)

Ability Enhancement Course

Name:____

USN:____

Year/Semester:_____________________________________

Bangalore Institute of Technology

Dept. of Electronics and Telecommunication Engineering

Vision of the institute

Establish and develop the Institute as the Centre of higher learning, ever abreast with expanding horizon of knowledge in the field of Engineering and Technology with entrepreneurial thinking, leadership excellence for life-long success and solve societal problems.

Mission of the institute

- Provide high quality education in the Engineering disciplines from the undergraduate through doctoral levels with creative academic and professional programs.
- Develop the Institute as a leader in Science, Engineering, Technology, Management and Research and apply knowledge for the benefit of society.
- Establish mutual beneficial partnerships with Industry, Alumni, Local, State and Central Governments by Public Service Assistance and Collaborative Research.
- Inculcate personality development through sports, cultural and extracurricular activities and engage in social, economic and professional challenges.

Vision of the department

Empower every student to be Creative and Productive in the field of Electronics and Telecommunication Engineering by imparting excellent Technical Education and inculcating Human Values.

Mission of the department

- To make our Students acquaint with the Global requirements such as Problem Solving Skills, Cultural Sensitivity, Ethical Behaviour and Social Responsibility.
- To motivate our Students to pursue Higher Education and engage in continuous up-gradation of their Professional Skills.
- To encourage students to develop Communication Skills, Professional Values, and Positive Attitude that in turn leads to fostering Leadership Qualities.

Syllabus:

IoT (Internet of Things) Lab			
Course Code	21EC581	CIE Marks	50
Teaching Hours/Week (L: T:P: S)	0:0:2:0	SEE Marks	50
Credits	1	Exam Hours	2
Course objectives: • To impart necessary and practical knowledge of components of Internet of Things • To develop skills required to build real-life IoT based projects.			
Experiments			
1. i) To interface LED/Buzzer with Arduino/Raspberry Pi and write a program to 'turn ON' LED for 1 sec after every 2 seconds.			
ii) To interface Push button/Digital sensor (IR/LDR) with Arduino/Raspberry Pi and write a program to 'turn ON' LED when push button is pressed or at sensor detection.			
2. i) To interface DHT11 sensor with Arduino/Raspberry Pi and write a program to print temperature and humidity readings.			
ii) To interface OLED with Arduino/Raspberry Pi and write a program to print temperature and humidity readings on it.			
3. To interface motor using relay with Arduino/Raspberry Pi and write a program to 'turn ON' motor when push button is pressed.			
4. To interface Bluetooth with Arduino/Raspberry Pi and write a program to send sensor data to smartphone using Bluetooth.			
5. To interface Bluetooth with Arduino/Raspberry Pi and write a program to turn LED ON/OFF when '1'/'0' is received from smartphone using Bluetooth.			
6. Write a program on Arduino/Raspberry Pi to upload temperature and humidity data to Thingspeak cloud.			
7. Write a program on Arduino/Raspberry Pi to retrieve temperature and humidity data from Thingspeak cloud.			
8. To install MySQL database on Raspberry Pi and perform basic SQL queries.			
9. Write a program on Arduino/Raspberry Pi to publish temperature data to MQTT broker.			
10. Write a program to create UDP server on Arduino/Raspberry Pi and respond with humidity data to UDP client when requested.			
11. Write a program to create TCP server on Arduino/Raspberry Pi and respond with humidity data to TCP client when requested.			
12. Write a program on Arduino/Raspberry Pi to subscribe to MQTT broker for temperature data and print it.			

Introduction to Internet of Things

IoT (Internet of Things) is a revolutionary concept that refers to the network of interconnected physical objects, devices, and sensors that communicate and exchange data over the internet. These objects, often called "things," are embedded with various technologies such as sensors, actuators, and communication interfaces, enabling them to collect, transmit, and receive data. Here's an explanation of IoT:

Key Components and Characteristics of IoT:

1. **Things/Devices:** These are the physical objects equipped with sensors, actuators, and communication modules. These devices can be anything from smart thermostats, wearable fitness trackers, industrial machines, to environmental sensors, and more. They gather data from their surroundings or perform actions based on commands.
2. **Sensors and Actuators:** Sensors collect data from the environment, measuring variables such as temperature, humidity, pressure, motion, and more. Actuators are responsible for carrying out actions based on commands received from the network or other devices, such as turning on/off lights or adjusting the thermostat.
3. **Connectivity:** IoT devices rely on various communication technologies to connect to the internet and other devices. This can include Wi-Fi, cellular networks, Bluetooth, Zigbee, LoRa, or Ethernet, depending on the device's range, power consumption, and bandwidth requirements.
4. **Data Transmission:** Data collected by IoT devices is transmitted to central processing units, such as cloud servers or edge computing platforms. This data can be transmitted in real-time or stored locally for later transmission, depending on the use case.
5. **Data Processing:** The collected data is processed to extract valuable insights, trends, and patterns. This can involve analytics, artificial intelligence, and machine learning to make informed decisions.
6. **Cloud/Edge Computing:** IoT data can be processed either in the cloud (centralized) or at the edge (closer to the device). Edge computing reduces latency and allows for real-time decision-making, while cloud computing offers scalability and accessibility from anywhere.
7. **Applications:** IoT applications are software programs or interfaces that enable users to interact with IoT data and control devices. These can take the form of web-based dashboards, mobile apps, or automation systems.

Key Principles of IoT:

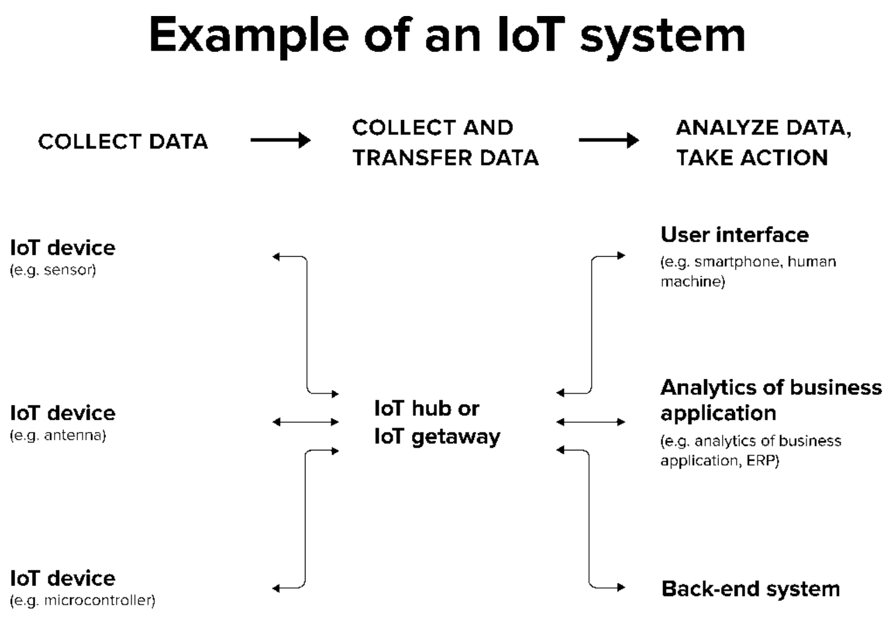
1. **Interconnectivity:** IoT thrives on the idea that virtually any device can be connected to the internet, allowing for data sharing and remote control. This interconnectivity facilitates automation and real-time monitoring.
2. **Sensing and Data Collection:** IoT devices are equipped with sensors that collect data from the physical world. This data can be environmental data, usage statistics, or other relevant information.
3. **Data Analysis:** The collected data is analyzed to derive valuable insights, enabling businesses and individuals to make informed decisions, optimize processes, and improve efficiency.
4. **Remote Control and Automation:** IoT allows for remote monitoring and control of devices. This can range from adjusting home thermostat settings from a smartphone to remotely managing industrial equipment.

Applications of IoT:

IoT has a wide range of applications across various industries, including:

- **Smart Homes:** Home automation, security systems, and energy management.
- **Smart Cities:** Traffic management, waste management, and environmental monitoring.
- **Industrial IoT (IIoT):** Predictive maintenance, asset tracking, and process optimization.
- **Healthcare:** Remote patient monitoring and medical device connectivity.
- **Agriculture:** Precision farming and livestock monitoring.
- **Logistics and Supply Chain:** Inventory management and asset tracking.

How does Internet of Thing (IoT) Work?
The Internet of Things (IoT) works by connecting physical devices, sensors, and objects to the internet, enabling them to collect, transmit, and receive data. The core principles of how IoT works involve data sensing, connectivity, data processing, and user interaction:


Example of an IoT system
COLLECT DATA
COLLECT AND TRANSFER DATA
ANALYZE DATA, TAKE ACTION
IoT device (e.g. sensor)
IoT device (e.g. antenna)
IoT device (e.g. microcontroller)
IoT hub or IoT getaway
User interface (e.g. smartphone, human machine)
Analytics of business application (e.g. analytics of business application, ERP)
Back-end system

1.
1.
1.
1.
1.
1.
1.
1.
1.
1.
1.
1.

1. **Sensing Data:** IoT devices are equipped with various sensors that capture data from the physical world. These sensors can measure parameters like temperature, humidity, pressure, light, motion, and more. Some devices also include cameras and microphones for visual and audio data.
2. **Data Transmission:** IoT devices use communication technologies to transmit the data they collect to other devices or central processing units. The choice of communication method depends on factors such as range, power consumption, and bandwidth requirements. Common communication technologies include Wi-Fi, cellular networks (3G, 4G, 5G), Bluetooth, Zigbee, LoRa, and more.
3. **Data Processing:** The data collected by IoT devices is sent to central processing units, such as cloud servers or edge computing platforms. Here, the data is processed and analyzed. This processing can include data filtering, aggregation, storage, and, in many cases, the application of machine learning or artificial intelligence algorithms to extract meaningful insights from the data.
4. **Storage:** Processed data is often stored in databases or data lakes, making it accessible for future analysis and historical reference.
5. **Decision-Making:** Based on the processed data, IoT systems can make informed decisions. These decisions can range from simple actions like turning on a light when a motion sensor is triggered to

complex decisions in industrial IoT (IIoT), such as predicting equipment maintenance needs.

6. **User Interaction:** IoT applications and interfaces enable users to interact with the system. Users can monitor and control devices, view data through web-based dashboards or mobile apps, and receive alerts or notifications. Some systems also allow voice control or integration with other software applications.
7. **Automation:** IoT can enable automation based on pre-defined rules or data-driven triggers. For example, a smart thermostat can automatically adjust the temperature based on occupancy and ambient conditions.
8. **Security and Privacy:** Security measures are crucial in IoT to protect data, devices, and the network from unauthorized access and potential vulnerabilities. This includes encryption, authentication, and access control.

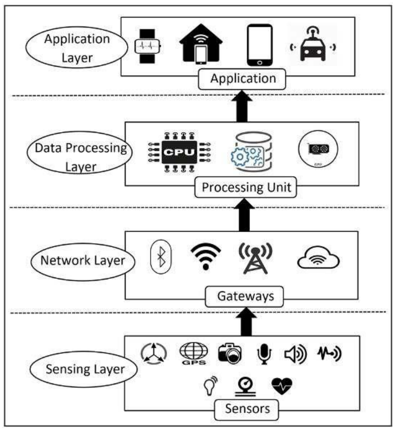
IoT Ecosystem: IoT systems are part of a larger ecosystem that includes:

- **Devices/Things:** These are the IoT devices and sensors that collect data and perform actions.
- **Connectivity:** Communication networks and protocols that allow devices to transmit dataover the internet.
- **Cloud/Edge:** Centralized cloud platforms or edge computing environments for data processing and storage.
- **Applications:** Software interfaces that enable users and other systems to interact with IoTdata and devices

IoT Architecture

1. IoT (Internet of Things) architecture defines the structure and components of an IoT system,outlining how devices, data, and processes interact. It provides a blueprint for building a functional and secure IoT ecosystem. Here's an explanation of the typical architecture of IoT:
2. Devices/Things: These are the physical objects or devices embedded with sensors, actuators, and communication modules. IoT devices can include a wide range of objects, from environmental sensors to industrial machinery and wearable devices. They collect data from their surroundings or perform actions based on commands received.
3. Sensors and Actuators: Sensors are responsible for collecting data from the physical world.They measure variables such as temperature, humidity, pressure, motion, light, and more.
4. Actuators carry out actions based on commands received from the network or other devices.Examples include turning on lights or adjusting thermostat settings.
5. Connectivity: IoT devices rely on various communication technologies to connect to theinternet and other devices. The choice of connectivity depends on the specific use case.
6. Common IoT communication technologies include Wi-Fi, cellular networks (3G, 4G, 5G),Bluetooth, Zigbee, LoRa, and Ethernet.
7. Data Transmission: Data collected by IoT devices is transmitted to central processing units,such as cloud servers or edge computing platforms. Data can be transmitted in real-time or stored locally for later transmission, depending on the usecase.
8. Data Processing: Processed data is essential for extracting valuable insights from the collected information. Data processing can involve filtering, aggregation, and applying machine learning algorithms.
9. Data processing can occur in the cloud (centralized) or at the edge (closer to the device),depending on the architecture and requirements.
10. Cloud/Edge Computing: The data is sent to cloud platforms or edge computing environments for further processing, storage, and analysis.
11. Cloud computing offers scalability and accessibility from anywhere, while edge computingreduces latency and allows for real-time decision-making.

12. Applications: IoT applications are software programs or interfaces that enable users to interact with IoT data and control devices.



These applications can take the form of web-based dashboards, mobile apps, or automationsystems.

1. User Interaction: End-users interact with the IoT system through web interfaces, mobile apps, voice commands, or other interfaces.
Users can monitor data, control devices, and receive alerts or notifications.

2. Security and Privacy: Security measures are crucial to protect against data breaches, unauthorized access, and device tampering. IoT security includes encryption, authentication, access control, and regular updates to address vulnerabilities.

3. External Integration: IoT systems may integrate with other systems, such as enterprise resource planning (ERP) systems, customer relationship management (CRM) systems, or third- party APIs for extended functionality and data sharing.

IoT Protocols

IoT (Internet of Things) protocols are communication standards that enable devices and systems in the IoT ecosystem to exchange data reliably and efficiently. These protocols play a crucial role in ensuring that devices from different manufacturers can communicate and work together seamlessly. Here are some common IoT protocols:

MQTT (Message Queuing Telemetry Transport):

- MQTT is a lightweight and efficient publish-subscribe messaging protocol.
- It is well-suited for IoT due to its low overhead, making it ideal for low-power and low- bandwidth devices.
- Devices can publish messages to topics, and other devices can subscribe to these topics toreceive data.

HTTP (Hypertext Transfer Protocol) and HTTPS (HTTP Secure):

- HTTP is a standard protocol for transmitting data over the web.
- IoT devices can communicate with web servers and cloud services using HTTP or itssecure version, HTTPS.
- It's commonly used for interactions between IoT devices and cloud platforms or webapplications.

CoAP (Constrained Application Protocol):

- CoAP is a lightweight, UDP-based protocol designed for constrained IoT devices.
- It is suitable for low-power, low-data-rate applications, and it is often used in resource-constrained environments.

AMQP (Advanced Message Queuing Protocol):

- AMQP is a robust and reliable messaging protocol that ensures guaranteed messagedelivery.
- It is suitable for IoT applications that require reliable communication, such as industrialand enterprise IoT use cases.

DDS (Data Distribution Service):

- DDS is a protocol for real-time, data-centric communication between IoT devices andsystems.
- It is commonly used in industrial and mission-critical applications where low latency andhigh reliability are essential.

Bluetooth and Bluetooth Low Energy (BLE):

- Bluetooth is a wireless communication protocol often used in short-range IoTapplications.
- BLE, a low-power variant of Bluetooth, is well-suited for battery-operated IoT devicesand smartphone connectivity.

LoRaWAN (Long Range Wide Area Network):

- LoRaWAN is designed for long-range, low-power communication.
- It is suitable for IoT applications like remote environmental monitoring and assettracking.

Zigbee:

- Zigbee is a low-power, low-data-rate wireless communication protocol.
- It is commonly used in home automation and industrial IoT applications.

Modbus:

- Modbus is a communication protocol widely used in industrial automation.
- It is suitable for connecting industrial devices and sensors to control systems and SCADA(Supervisory Control and Data Acquisition) systems.

DDS (Device Device Server):

- DDS is a lightweight protocol that facilitates peer-to-peer communication between IoTdevices.
- It is suitable for applications where low latency and efficient data exchange are critical.

IoT Software

Embedded systems have less storage and processing power, their language needs are different. The most commonly used operating systems for such embedded systems are Linux or UNIX-likeOSs like Ubuntu Core or Android.
IoT software encompasses a wide range of software and programming languages from general- purpose languages like C++ and Java to embedded-specific choices like Google ‘s Go language or Parasail.

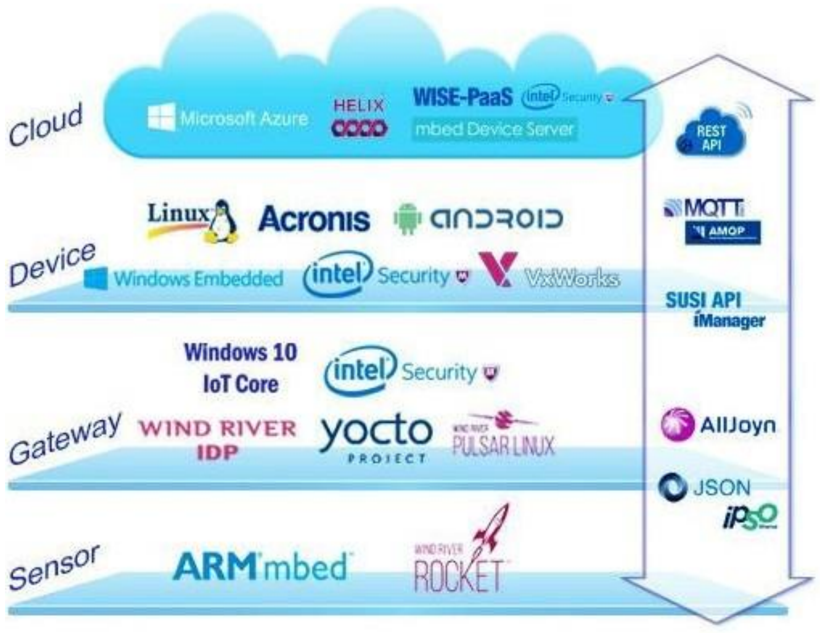


Fig: IoT Hardware | IoT Software

Few IoT Software’s are

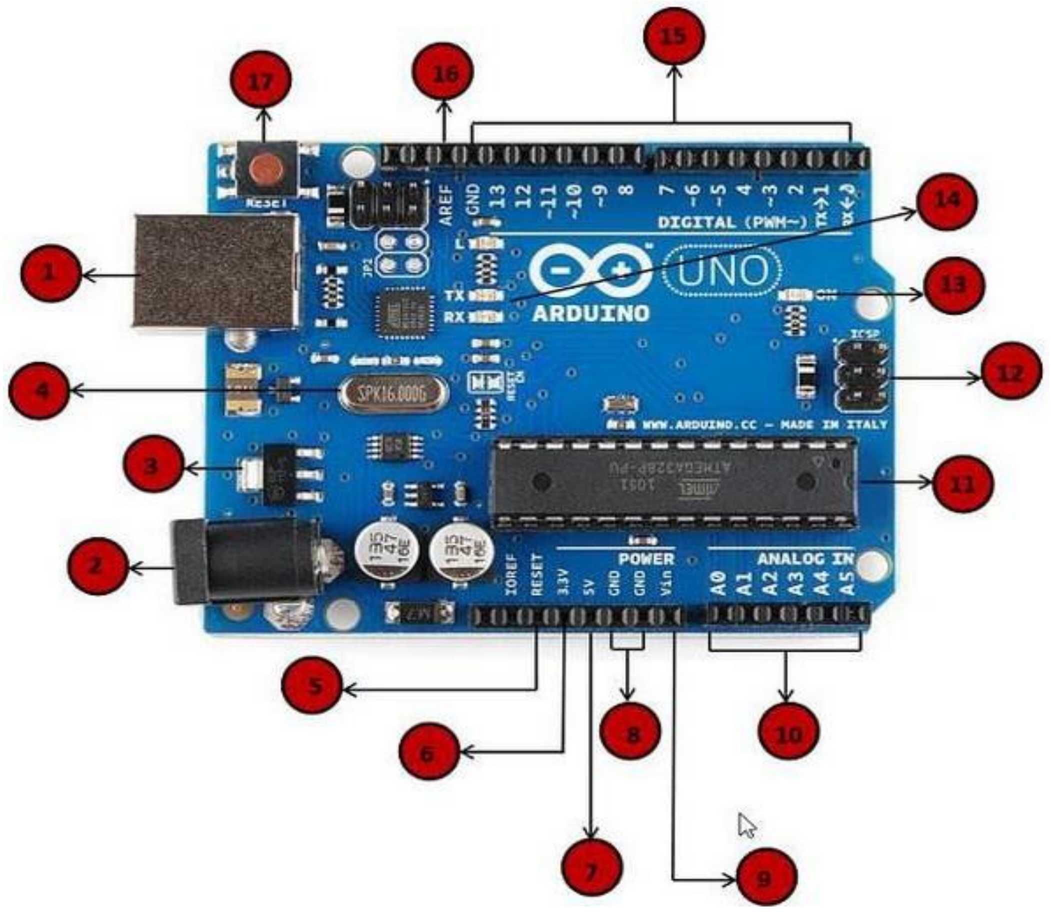
C & C++: The C programming language has its roots in embedded systems—it even got its start for programming telephone switches. It‘s pretty ubiquitous, that is, it can be used almost everywhere and many programmers already know it. C++ is the object-oriented version of C, which is a language popular for both the Linux OS and Arduino embedded IoT software systems.These languages were basically written for the hardware systems which makes them so easy to use.

Java: While C and C++ are hardware specific, the code in JAVA is more portable. It is more like a write once and read anywhere language, where you install libraries, invests time in writing codes once and you are good to go.

Python: There has been a recent surge in the number of python users and has now becomeone of the ―go-to‖ languages in Web development. Its use is slowly spreading to the embedded control and IoT world—specially the Raspberry Pi processor. Python is an interpreted language, which is, easy to read, quick to learn and quick to write. Also, it‘s a powerhouse for serving data-heavy applications.

Introduction to Arduino Uno

Arduino UNO is a Micro controller which is the most popular board in the Arduino board family. It is the best board to get started with electronics and coding.
The Layout of the Arduino UNO:



Power USB:

Arduino board can be powered by using the USB cable from your computer. All you need to do is connect the USB cable to the USB connection (1).

Power (Barrel Jack):

Arduino boards can be powered directly from the AC mains power supply by connecting it to the Barrel Jack (2).

Voltage Regulator:

The function of the voltage regulator is to control the voltage given to the Arduino board and stabilize the DC voltages used by the processor and other elements.

Crystal Oscillator:

The crystal oscillator helps Arduino in dealing with time issues. How does Arduino calculate time? The answer is, by using the crystal oscillator. The number printed on top of the Arduino crystal is 16.000H9H. It tells us that the frequency is 16,000,000 Hertz or 16 MHz.

Arduino Reset:

You can reset your Arduino board, i.e., starts your program from the beginning. You can reset the UNO board in two ways. First, by using the reset button (17) on the board. Second, you can connect an external reset button to the Arduino pin labeled RESET (5).

Pins (3.3, 5, GND, Vin)

3.3V (6) − Supply 3.3 output volt
5V (7) − Supply 5 output volt
Most of the components used with Arduino board works fine with 3.3 volt and 5 volt.
GND (8)(Ground) − There are several GND pins on the Arduino, any of which can be used to ground your circuit.
Vin (9) − This pin also can be used to power the Arduino board from an external power source, like AC mains power supply.

Analog pins :

The Arduino UNO board has 6− analog input pins A0 through A5. These pins can read the signal from an analog sensor like the humidity sensor or temperature sensor and convert it into a digital value that can be read by the microprocessor.

Main microcontroller(IC-ATmega328P):

Each Arduino board has its own microcontroller (11). You can assume it as the brain of your board. The main IC (integrated circuit) on the Arduino is slightly different from board to board. The microcontrollers are usually of the ATMEL Company. You must know what IC your board has before loading up a new program from the Arduino IDE. This information is available on the top of the IC. For more details about the IC construction and functions, you can refer to the data sheet.

ICSP pin
Mostly, ICSP (12) is an AVR, a tiny programming header for the Arduino consisting of MOSI, MISO, SCK, RESET, VCC, and GND. It is often referred to as an SPI (Serial Peripheral Interface), which could be considered as an "expansion" of the output. Actually, you are slaving the output device to the master of the SPI bus.

Power LED indicator
This LED should light up when you plug your Arduino into a power source to indicate that your board is powered up correctly. If this light does not turn on, then there is something wrong with the connection.

TX and RX LEDs
On your board, you will find two labels: TX (transmit) and RX (receive). They appear in two places on the Arduino UNO board. First, at the digital pins 0 and 1, to indicate the pins responsible for serial communication. Second, the TX and RX led (13). The TX led flashes with different speed while sending the serial data. The speed of flashing depends on the baud rate used by the board. RX flashes during the receiving process.

Digital I/O
The Arduino UNO board has 14 digital I/O pins (15) (of which 6 provide PWM (Pulse Width Modulation) output. These pins can be configured to work as input digital pins to read logic values (0 or 1) or as digital output pins to drive different modules like LEDs, relays, etc. The pins labeled "~" can be used to generate PWM.

AREF
AREF stands for Analog Reference. It is sometimes, used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

Installing and work with Arduino IDE

AIM: Demonstrating the installing of Arduino IDE in laptops or Desktops.

REQUIREMENTS:

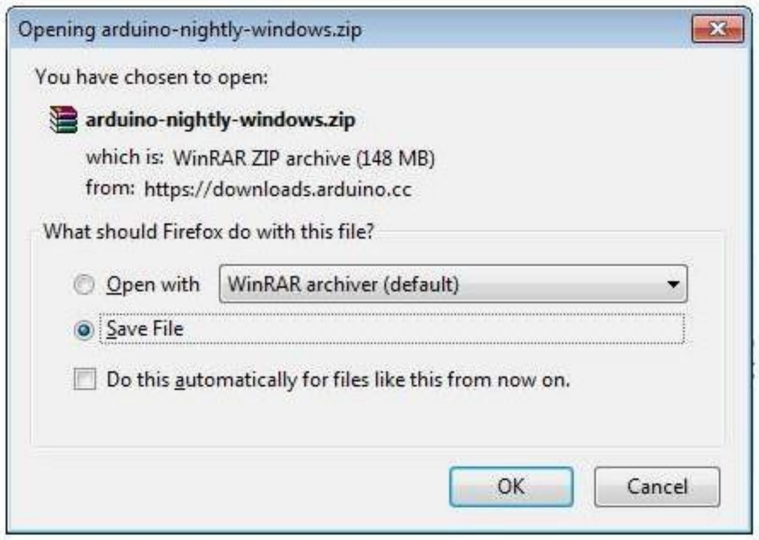
Arduino UNO Board,
USB A to USB−B Cable.
Arduino IDE

DESCRIPTION:

We will learn how to set up the Arduino IDE on our computer and prepare the board to receive the program via USB cable.

Step 1
First you must have your Arduino board (you can choose your favorite board) and a USB cable. In case you use Arduino UNO, Arduino Duemilanove, Nano, Arduino Mega 2560, or Diecimila, you will need a standard USB cable (A plug to B plug), the kind you would connect to a USB printer as shown in the following image.
Step 2 − Download Arduino IDE Software.

You can get different versions of Arduino IDE from the Download page on the Arduino Official website. You must select your software, which is compatible with your operating system (Windows, IOS, or Linux). After your file download is complete, unzip the file.



Opening arduino-nightly-windows.zip

You have chosen to open:

arduino-nightly-windows.zip

which is: WinRAR ZIP archive (148 MB)
from: https://downloads.arduino.cc

What should Firefox do with this file?

Open with WinRAR archiver (default)
[✓ Save File]
Do this automatically for files like this from now on.

Step 3 − Power up your board.

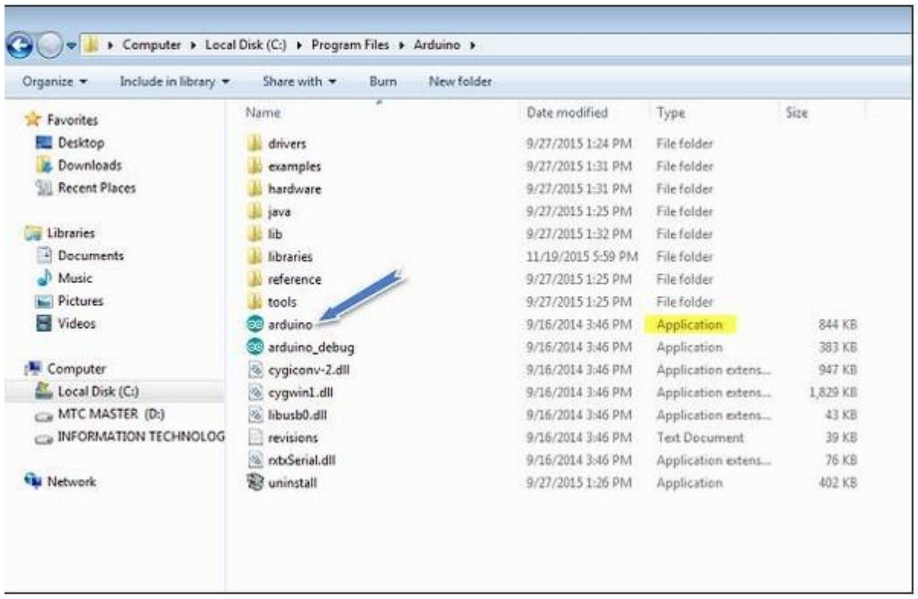
The Arduino Uno, Mega, Duemilanove and Arduino Nano automatically draw power from either, the USB connection to the computer or an external power supply. If you are using an Arduino Diecimila, you have to make sure that the board is configured to draw power from the USB connection. The power source is selected with a jumper, a small piece of plastic that fits onto two of the three pins between the USB and power jacks. Check that it is on the two pins closest to the USB port.

Connect the Arduino board to your computer using the USB cable. The green power LED (labeled PWR) should glow.

Step 4 − Launch Arduino IDE.

After your Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Double−click the icon to start the IDE.

Step 5 − Open your first project.



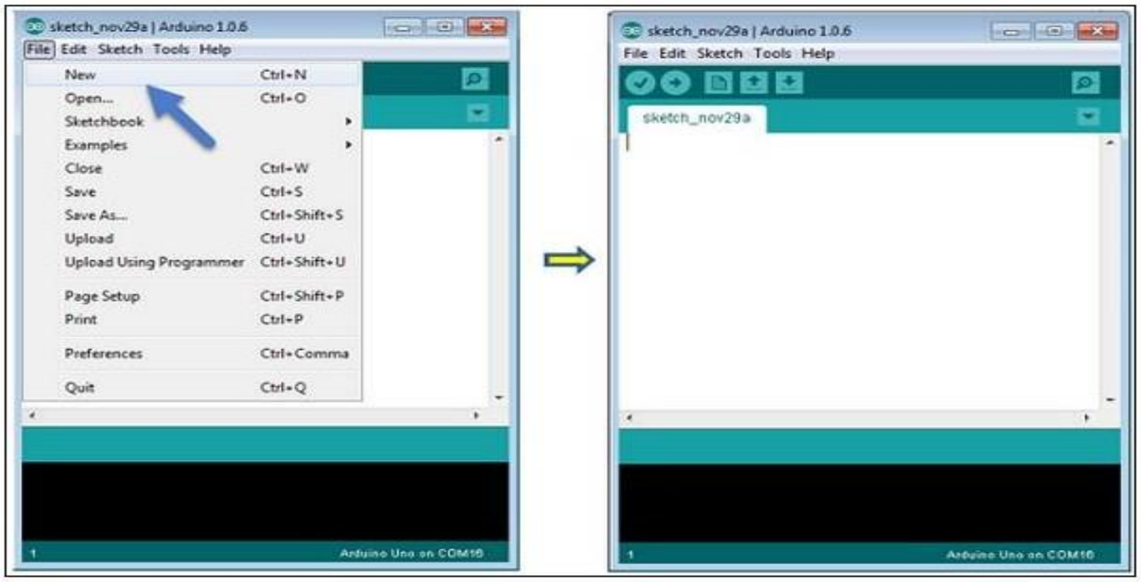
Name	Date modified	Type	Size
drivers	9/27/2015 1:24 PM	File folder	
examples	9/27/2015 1:31 PM	File folder	
hardware	9/27/2015 1:31 PM	File folder	
java	9/27/2015 1:25 PM	File folder	
lib	9/27/2015 1:32 PM	File folder	
libraries	11/19/2015 5:59 PM	File folder	
reference	9/27/2015 1:25 PM	File folder	
tools	9/27/2015 1:25 PM	File folder	
arduino	9/16/2014 3:46 PM	Application	844 KB
arduino_debug	9/16/2014 3:46 PM	Application	383 KB
cygiconv-2.dll	9/16/2014 3:46 PM	Application extens...	947 KB
cygwin1.dll	9/16/2014 3:46 PM	Application extens...	1,829 KB
libusb0.dll	9/16/2014 3:46 PM	Application extens...	43 KB
revisions	9/16/2014 3:46 PM	Text Document	39 KB
rxtxSerial.dll	9/16/2014 3:46 PM	Application extens...	76 KB
uninstall	9/27/2015 1:26 PM	Application	402 KB

Once the software starts, you have two options −

Create a new project.

Open an existing project example.

To create a new project, select File → New.

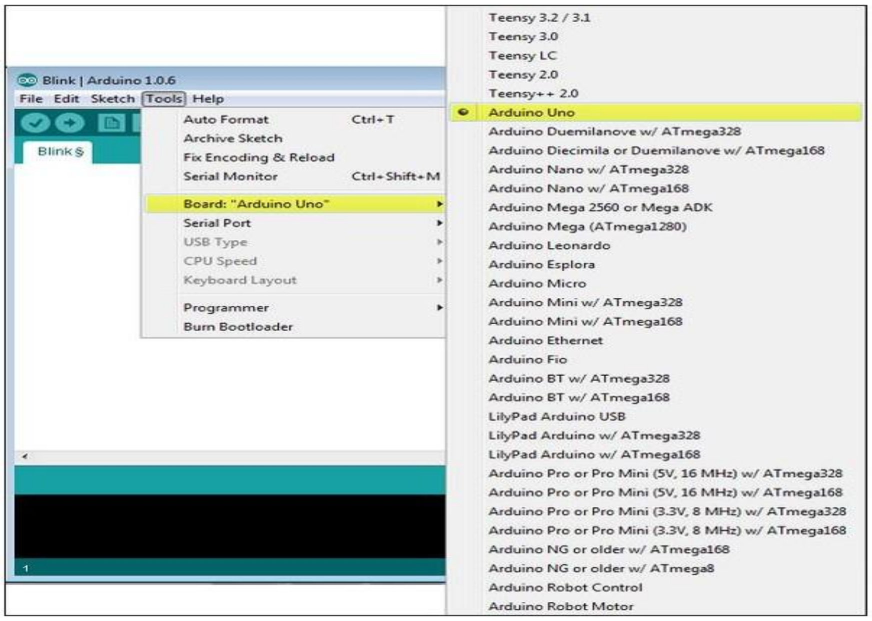


To open an existing project example, select File → Example → Basics → Blink.

Here, we are selecting just one of the examples with the name Blink. It turns the LED on and off with some time delay. You can select any other example from the list.

Step 6 − Select your Arduino board.

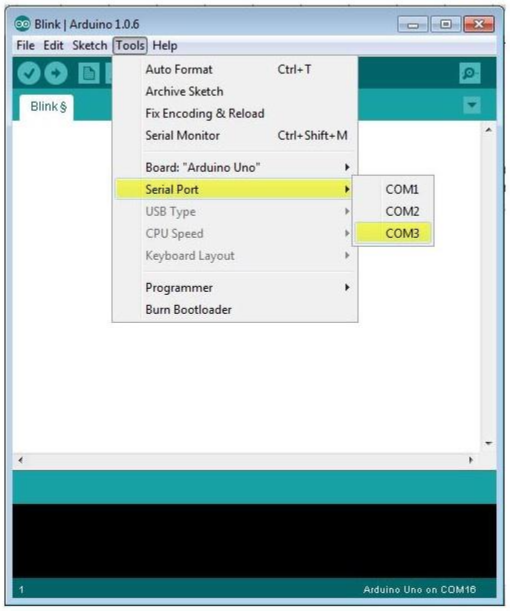
To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer.
Go to Tools → Board and select your board.


Blink | Arduino 1.0.6
File Edit Sketch Tools Help
Blink §
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board: "Arduino Uno"
Serial Port
USB Type
CPU Speed
Keyboard Layout
Programmer
Burn Bootloader
Teensy 3.2 / 3.1
Teensy 3.0
Teensy LC
Teensy 2.0
Teensy++ 2.0
Arduino Uno
Arduino Duemilanove w/ ATmega328
Arduino Diecimila or Duemilanove w/ ATmega168
Arduino Nano w/ ATmega328
Arduino Nano w/ ATmega168
Arduino Mega 2560 or Mega ADK
Arduino Mega (ATmega1280)
Arduino Leonardo
Arduino Esplora
Arduino Micro
Arduino Mini w/ ATmega328
Arduino Mini w/ ATmega168
Arduino Ethernet
Arduino Fio
Arduino BT w/ ATmega328
Arduino BT w/ ATmega168
LilyPad Arduino USB
LilyPad Arduino w/ ATmega328
LilyPad Arduino w/ ATmega168
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega168
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega168
Arduino NG or older w/ ATmega168
Arduino NG or older w/ ATmega8
Arduino Robot Control
Arduino Robot Motor

Here, we have selected Arduino Uno board according to our tutorial, but you must select the name matching the board that you are using.

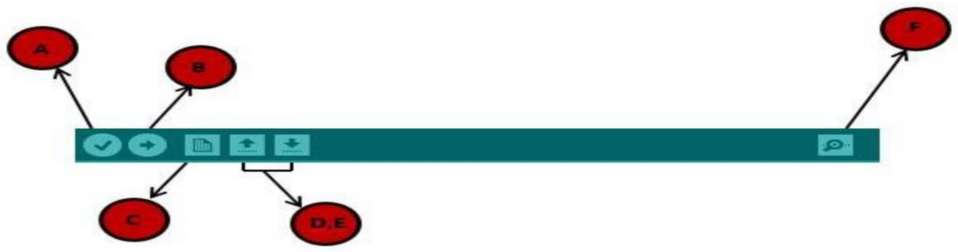
Step 7 − Select your serial port.

Select the serial device of the Arduino board. Go to Tools → Serial Port menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port.



Step 8 − Upload the program to your board.

Before explaining how we can upload our program to the board, we must demonstrate the function of each symbol appearing in the Arduino IDE toolbar.



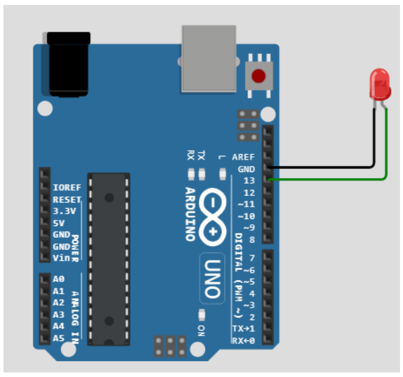
A − Used to check if there is any compilation error.
B − Used to upload a program to the Arduino board.
C − Shortcut used to create a new sketch.
D − Used to directly open one of the example sketch.
E − Used to save your sketch.
F − Serial monitor used to receive serial data from the board and send the serial data to the board.
Now, simply click the "Upload" button in the environment. Wait a few seconds; you will see the RX and TX LEDs on the board, flashing. If the upload is successful, the message "Done uploading" will appear in the status bar.

Experiment No: 1.a
To interface LED/Buzzer with Arduino and write a program to ‘turn ON’ LED for 1 sec after every 2 seconds.

Apparatus Required:

Sl No	Components	Quantity
1	Arduino UNO	1
2	LED or Buzzer	1
3	USB cable A to B	1
4	Connecting wires	2

Circuit Diagram:

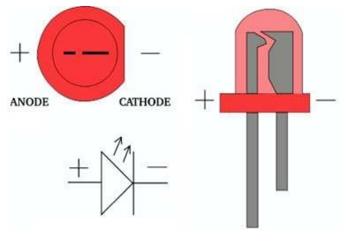


Pin Connection	
Arduino UNO	**LED**
GND	Cathode
D13	Anode

Theory:

LED Stands for Light Emitting Diode is a semiconductor device that emits light when an electric current passes through it. LEDs are widely used for various applications due to their energy efficiency, long lifespan, and versatility. Here are some key points about LEDs.

Basic Operation: LEDs work on the principle of electroluminescence. When electrons and holes (positive counterparts of electrons) recombine within the semiconductor material, they release energy in the form of photons, which produces light.



Item	Min	Max	Unit
Forward Current	20	30	mA
Forward Voltage	1.8	2.2	V

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
}
```

Alternatively:

```
const int ledPin = 8;
void setup() {
// initialize digital pin ledPin as an output.
pinMode(ledPin, OUTPUT);
}
// the loop function runs over and over again forever
void loop() {
digitalWrite(ledPin, HIGH); // turn the LED on (HIGH is the voltage level)
delay(1000); // wait for a second
digitalWrite(ledPin, LOW); // turn the LED off by making the voltage LOW
delay(1000); // wait for a second
}
```

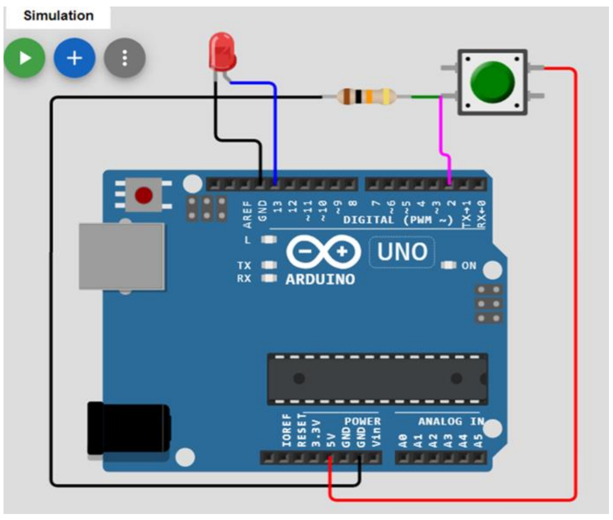
Result: Turn ON and OFF of LED automatically by Arduino Uno is observed

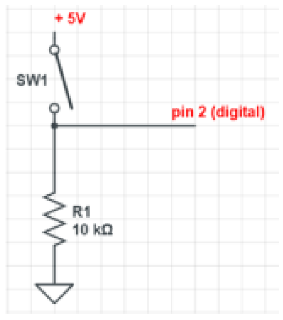
Experiment No: 1.b
To interface Push button with Arduino and write a program to 'turn ON' LED when push button is pressed or at sensor detection.

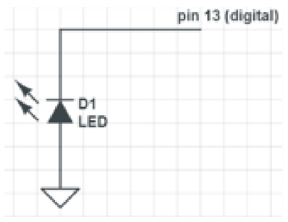
Component Required:

Sl No	Components	Quantity
1	ARDUINO UNO	1
2	LED	1
3	USB cable A to B	1
4	Connecting wires	--
5	Push Button	1
6	Bread Board	1
7	Resistance (10 kΩ)	1

Circuit Diagram







Program:

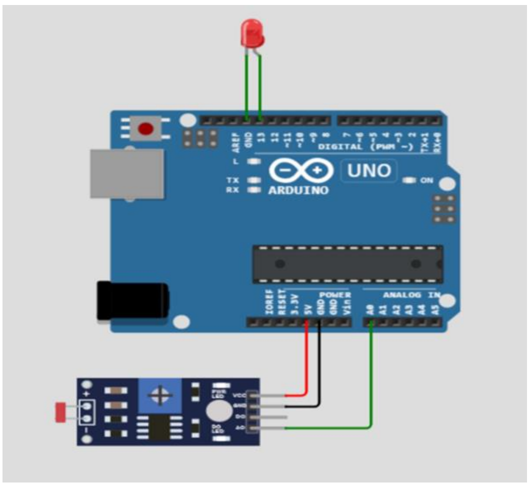
```
const int pushPin = 2;  // the number of the pushbutton pin
const int ledPin = 13;    // the number of the LED pin
 // variables will change:
int buttonState = 0;  // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
              pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
              pinMode(pushPin, INPUT);
}
 void loop() {
  // read the state of the pushbutton value:
              buttonState = digitalRead(pushPin);
   // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
              // turn LED on:
    digitalWrite(ledPin, HIGH);
              }
              else {
              // turn LED off:
              digitalWrite(ledPin, LOW);
              }
}
```

Result: Turn ON and OFF of LED automatically as well as controlled Switching operation ofLED by using push button done using Arduino Uno.

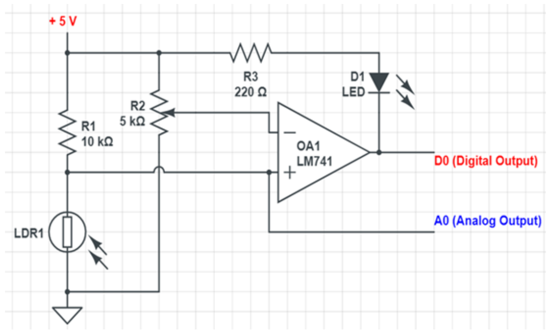
Experiment No: 1.c
To interface Digital sensor (IR/LDR) with Arduino and write a program to 'turn ON' LED at sensor detection.

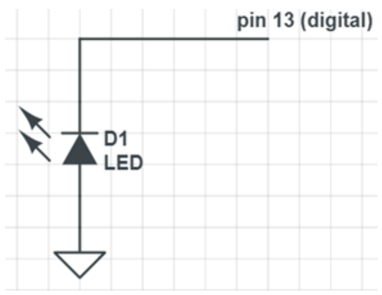
Component Required:

Sl No	Components	Quantity
1	ARDUINO UNO	1
2	LED	1
3	USB cable A to B	1
4	Connecting wires	--
5	LDR Sensor Module	1
6	Bread Board	1
7	Resistance (10 kΩ)	1



ARDUINO UNO	**LDR Sensor Module**
GND	GND
5V	VCC
D2	D0: Digital output

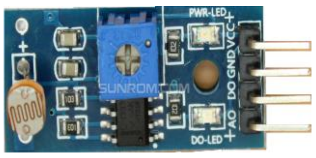




Theory:

LDR sensor module is a low-cost digital sensor as well as analog sensor module, which is capable to measure and detect light intensity.
This sensor also is known as the Photoresist or sensor. This sensor has an on-board LDR(Light Dependent Resistor), that helps it to detect light.
This sensor module comes with 4 terminals. Where the "DO" pin is a digital output pin and the "AO" pin is an analog output pin. The output of the module goes high in the absence of light and it becomes low in the presence of light.
The sensitivity of the sensor can be adjusted using the on-board potentiometer.

Specifications:
Operating Voltage: 3.3V to 5V DC
Operating Current: 15ma
Output Digital - 0V to 5V, Adjustable trigger level from preset
Output Analog - 0V to 5V based on light falling on the LDR
LEDs indicating output and power
PCB Size: 3.2cm x 1.4cm
LM393 based design



Program:

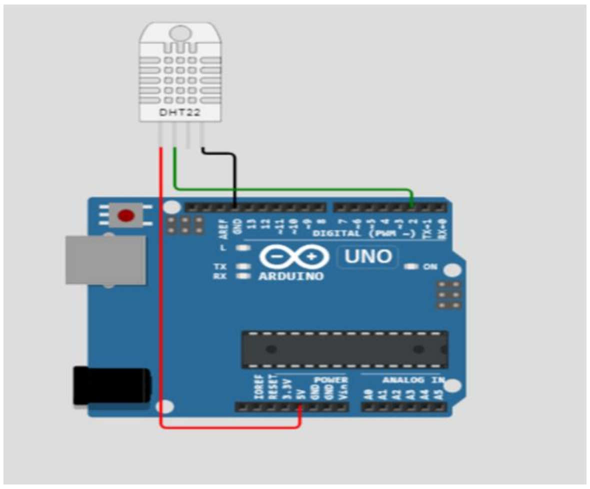
```
/* interfacing LDR sensor with Arduino | LDR Sensor Arduino Code for Digital Output */
// Declare your LDR sensor out pin connected Arduino pin "D2"

 int LDRSensor = 2;

void setup()
 {
  //Initialize Sensor (pin2) as an INPUT.
  pinMode (LDRSensor, INPUT);
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
  //Define baud rate for serial communication
  Serial.begin (9600);
 }

void loop()
 {
  //Read Digital output value from sensor using digitalRead()function
  int Sensordata = digitalRead (LDRSensor);
  //Print the sensor value on your serial monitor window
  if (Sensordata==HIGH) {
// do stuff if the condition is true
  Serial.print("Room is Dark");
  Serial.println();
  Serial.print("Sensor value:");
  Serial.println(Sensordata);
  Serial.println();
  digitalWrite(LED_BUILTIN, HIGH);
  Serial.print("Light is Switched ON");
  Serial.println();

}

else {
// do stuff if the condition is false
  Serial.print("Room is Bright");
  Serial.println();
  Serial.print("Sensor value:");
  Serial.println(Sensordata);
  Serial.println();
  digitalWrite(LED_BUILTIN, LOW);
  Serial.print("Light is Switched OFF");
  Serial.println();

}
    //Delay for 1 second to get clear output on the serial monitor
  delay(1000);
 }
```

Experiment No.: 2.a
To interface DHT11 sensor with Arduino and write a program to print temperature and humidity readings

Component Required:

Sl No	Components	Quantity
1	ARDUINO UNO	1
2	DHT11 Temperature sensor	1
3	USB cable A to B	1
4	Connecting wires	--

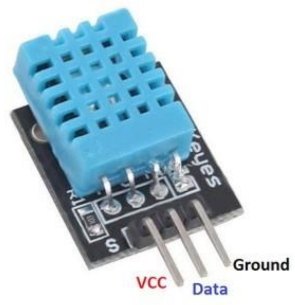
Circuit Diagram



ARDUINO UNO	**DHT11**
GND	GND
5v	VCC
D2	DATA

Theory

The **DHT11 sensor** is a **low-cost** digital temperature and humidity sensor. It operates ata **voltage of 3.3V to 5V** and can measure temperatures rangingfrom **0°C to 50°C** with an accuracy of ±2°C. Additionally, it can measure relative humidity ranging from **20% to 90%** with an accuracy of ±5% .



The humidity sensing capacitor has two electrodes with a moisture holding substrate as a dielectric between them. Change in the capacitance value occurs with the change in humidity levels. The IC measure, process this changed resistance values and change them into digital form. For measuring temperature this sensor uses a Negative Temperature coefficient thermistor, which causes a decrease in its resistance value with increase in temperature.

DHT11 Specifications

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

Libraries are a collection of code that makes it easy for you to connect to a sensor, display,module, etc. For example, the Liquid Crystal library makes it easy to talk to character LCD displays.

There are thousands of libraries available for download directly through the Arduino IDE, andyou can find all of them listed at the Arduino Library Reference.

Steps to Add DHT11 Sensor Library

1) Open your Arduino IDE and go to **Sketch** > **Include Library** > **Manage Libraries**. The Library Manager should open.
2) Search DHT then find the DHT Sensor library by Adafruit
3) Click install button to install library
4) If ask click on install all button to install library dependencies

Program:

```
#include "DHT.h"
#define DHTPIN 2     // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11   // DHT 11
// Initialize DHT sensor.
 DHT dht(DHTPIN, DHTTYPE);
void setup() {
  Serial.begin(9600);
  Serial.println(F("DHTxx test!"));
  dht.begin();
}

void loop() {
  // Wait a few seconds between measurements.
  delay(2000);
  // Reading temperature or humidity takes about 250 milliseconds
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
  float h = dht.readHumidity();
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();
  // Read temperature as Fahrenheit (isFahrenheit = true)
  float f = dht.readTemperature(true);
  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }

// Compute heat index in Fahrenheit (the default)
  float hif = dht.computeHeatIndex(f, h);
  // Compute heat index in Celsius (isFahreheit = false)
  float hic = dht.computeHeatIndex(t, h, false);
  Serial.print(F("Humidity: "));
  Serial.print(h);// Print Humidity value
  Serial.println();
  Serial.println();
  Serial.print(F("%  Temperature: "));
  Serial.print(t);
  Serial.print(F("°C, "));
  Serial.print(f);
  Serial.print(F("°F : "));  // Print Temperature (Celsius /Fahrenheit) value
  Serial.println();
  Serial.println();
  Serial.print(F("Heat index: "));
  Serial.print(hic);
  Serial.print(F("°C "));
  Serial.print(hif);
  Serial.println(F("°F"));// Print Heat index (Celsius /Fahrenheit) value
  Serial.println();
  Serial.println();
   }
```

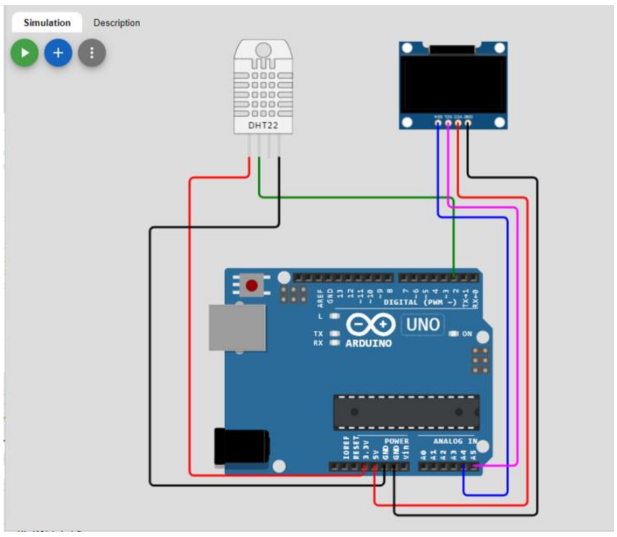
Result: Successfully interfaced DHT11 with Arduino Uno and measured values of temperature and humidity readings are displayed on serial monitor window.

Experiment No.: 2.b
To interface OLED with Arduino and write a program to print temperature and humidity readings on it.

Component Required:

Sl No	Components	Quantity
1	ARDUINO UNO	1
2	DHT11 Temperature sensor	1
3	USB cable A to B	1
4	Connecting wires	--
5	OLED Display 128×64, SSD1306 I2C	1

Circuit Diagram

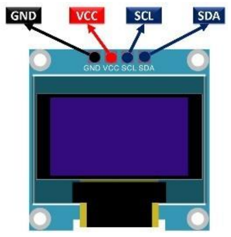


ARDUINO UNO	**DHT 11/22**	**OLED Pins**
3.3V/5V	**VCC**	**VCC**
GND	**GND**	**GND**
D2	**DATA**	**--**
A4	**--**	**SDA**
A5	**--**	**SCK**

Theory

The SSD1306 OLED I2C 128X64 OLED Display module is a small monochrome organic light-emitting diode (OLED) display that is controlled through an I2C interface. It has a display resolution of 128×64 pixels, and the SSD1306 is the controller chip that manages the display. It's commonly used for display purposes in various electronics projects and is compact low power, and easily readable in low light conditions.

Specification of OLED	
Size	0.96 inch
Terminals	4
Pixels	128×64
Communication	I2C only
VCC	3.3V-5V
Operating Temperature	-40℃ to +80℃



FEATURES

- Resolution: 128 x 64 dot matrix panel
- Power supply
  - V_{DD} = 1.65V to 3.3V for IC logic
  - V_{CC} = 7V to 15V for Panel driving
- For matrix display
  - OLED driving output voltage, 15V maximum
  - Segment maximum source current: 100uA
  - Common maximum sink current: 15mA
  - 256 step contrast brightness current control
- Embedded 128 x 64 bit SRAM display buffer
- Pin selectable MCU Interfaces:
  - 8-bit 6800/8080-series parallel interface
  - 3 /4 wire Serial Peripheral Interface
  - I^2C Interface
- Screen saving continuous scrolling function in both horizontal and vertical direction
- RAM write synchronization signal
- Programmable Frame Rate and Multiplexing Ratio
- Row Re-mapping and Column Re-mapping
- On-Chip Oscillator
- Chip layout for COG & COF
- Wide range of operating temperature: -40°C to 85°C

To control the OLED display you need the adafruit_SSD1306.h and the adafruit_GFX.h libraries. Follow the next instructions to install those libraries.

1. Open your Arduino IDE and go to **Sketch** > **Include Library** > **Manage Libraries**.The Library Manager should open.
2. Type "**SSD1306**" in the search box and install the SSD1306 library from Adafruit.
3. Click install button to install library
4. If ask click on install all button to install library dependencies
5. After installing the SSD1306 library from Adafruit, type "**GFX**" in the search box andinstall the library.
6. If ask click on install all button to install library dependencies
7. After installing the libraries, restart your Arduino IDE
8. Find the **Adafruit_SSD1306.h** file in the Arduino Library folder. Generally, it is located at **Documents\Arduino\libraries** on windows systems. There you will find the Adafruit_SSD1306.h file inside the Adafruit_SSD1306 folder.

Program

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <DHT.h>

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

#define DHTPIN 2 // pin connected to DHT11 sensor
#define DHTTYPE DHT22
// create SSD1306 display object connected to I2C
Adafruit_SSD1306 oled(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
DHT dht(DHTPIN, DHTTYPE);

String displayString;

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

  // initialize OLED display with address 0x3C for 128x64
  if (!oled.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    while (true);
  }

  delay(2000);         // wait for initializing
  oled.clearDisplay(); // clear display

  oled.setTextSize(2);      // text size
  oled.setTextColor(WHITE); // text color
  oled.setCursor(0, 10);    // position to display

  dht.begin();        // initialize DHT11 the temperature and humidity sensor

  displayString.reserve(10); // to avoid fragmenting memory when using String
}
```

```
void loop() {
  float humi  = dht.readHumidity();    // read humidity
  float tempC = dht.readTemperature(); // read temperature

  // check if any reads failed
  if (isnan(humi) || isnan(tempC)) {
    displayString = "Failed";
  } else {
    displayString  = String(tempC, 1); // one decimal places
    displayString += "°C";
    displayString += String(humi, 1); // one decimal places
    displayString += "%";
  }
  Serial.println(displayString);    // print the temperature in Celsius to Serial Monitor
  oledDisplayCenter(displayString); // display temperature and humidity on OLED

    }



void oledDisplayCenter(String text) {
  int16_t x1;
  int16_t y1;
  uint16_t width;
  uint16_t height;

  oled.getTextBounds(text, 0, 0, &x1, &y1, &width, &height);

  // display on horizontal and vertical center
  oled.clearDisplay(); // clear display
  oled.setCursor((SCREEN_WIDTH - width) / 2, (SCREEN_HEIGHT - height) / 2);
  oled.println(text); // text to display
  oled.display();
}
```

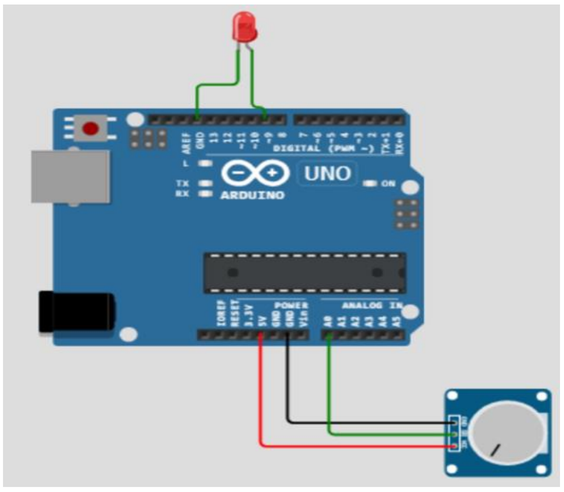
Result: Successfully interfaced DHT11 with Arduino Uno and measured temperature and humidity readings were displayed on OLED

Experiment No: 3
To interface motor using relay with Arduino and write a program to ‘run the motor at different speeds.

Component Required:

Sl No	Components	Quantity
1	ARDUINO UNO	1
2	Motor driver	1
3	USB cable A to B	1
4	Connecting wires	--
5	Potentiometer	1
6	DC motor	1

Circuit Diagram



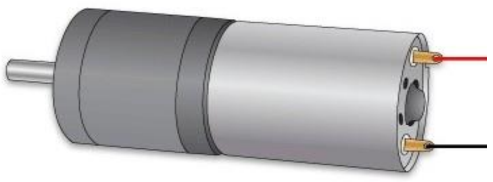
Note: Connect DC Motor in place of LED

Theory

DC Motor: A DC motor is a machine which converts DC electrical energy into mechanical energy. DC motors normally have just two leads, one positive and one negative. If you connectthese two leads directly to a battery, the motor will rotate. If you switch the leads, the motor will rotate in the opposite direction.

DC Motor Specification:

Operating Voltage (V): 12 Rated Speed (RPM): 200 Rated Torque (kg-cm): 1.5Load Current (A): 0.3 No Load Current (A): 0.06



The DC motor speed in general is directly proportional to the supply voltage, so if reduce the voltage from 9 volts to 4.5 volts then our speed become half of what it originally had. But in practice, for changing the speed of a dc motor we cannot go on changing the supply voltage all the time. The speed controller PWM for a DC motor works by varying the average voltage supplied to the motor

Program

```
const int ledPin = 9;
void setup() {
  Serial.begin(9600);

  pinMode(ledPin, OUTPUT);  // sets the pin as output
}
void loop() {
  val = analogRead(analogPin);  // read the input pin
  analogWrite(ledPin, val / 4); //analogRead values go from 0to 1023,analogWrite values from 0to 255
  Serial.print("Analog Value : ");
  Serial.print(val);
  Serial.println();
  delay(1000);
}
```

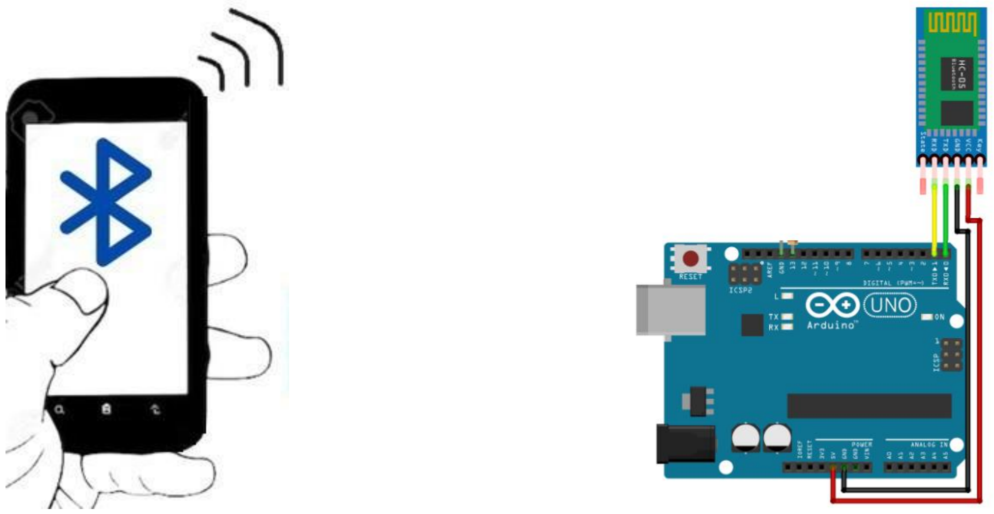
Result: DC motor running at different speeds is observed

Experiment No: 4
To interface Bluetooth with Arduino and write a program to turn LED ON/OFF when '1'/'0' is received from smartphone using Bluetooth.

Component Required:

Sl No	Components	Quantity
1	Arduino Uno	1
2	Bluetooth Module HC-05	1
3	USB cable A to B	1
4	Connecting wires	--
5	LED	1
6	Smart Phone with Arduino Bluetooth Controller app	1

Circuit Diagram

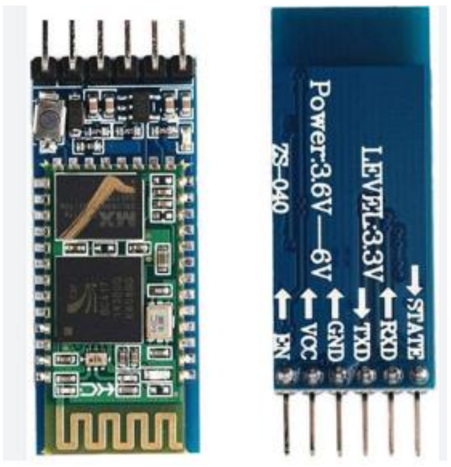


Pin Connection details	
Arduino Pins	**Bluetooth Module**
5 V	VCC
GND	GND
Rx	Tx
Tx	RX

Theory

Bluetooth is a low-power wireless connectivity technology used to stream audio, transfer dataand broadcast information between devices.
Operating frequency is: 2.4 to 2.485 GHz(UHF)

The HC-05 is a popular module which can add two-way (full-duplex) wireless functionality. It can be used to communicate between two microcontrollers like Arduino or communicate with any device with Bluetooth functionality like a Phone or Laptop.
There are many android applications that are already available which makes this process a lot easier. The module communicates with the help of USART at 9600 baud rate hence it is easy to interface with any microcontroller that supports USART.



HC-05 Pin out Configuration	
Pin Name	**Description**
Enable / Key	This pin is used to toggle between Data Mode (set low) and AT commandmode (set high). By default, it is in Data mode
Vcc	Powers the module. Connect to +5V Supply voltage
Ground	Ground pin of module, connect to system ground.
TX – Transmitter	Transmits Serial Data. Everything received via Bluetooth will be given outby this pin as serial data.
RX – Receiver	Receive Serial Data. Every serial data given to this pin will be broadcastedvia Bluetooth
State	The state pin is connected to on board LED, it can be used as feedback tocheck if Bluetooth is working properly.
LED	Indicates the status of Module • Blink once in 2 sec: Module has entered Command Mode • Repeated Blinking: Waiting for connection in Data Mode • Blink twice in 1 sec: Connection successful in Data Mode
Button	Used to control the Key/Enable pin to toggle between Data and commandMode

Connecting the Android device to the HC-05 creates a serial communication channel very similar to the serial monitor in the Arduino IDE. This means we need a Bluetooth version of the serial monitor.

Procedure to connect Bluetooth with mobile app

1. Download “Arduino **Bluetooth controller** “app from play store and install in your mobile.

   https://play.google.com/store/apps/details?id=com.giumig.apps.bluetoothserialmonitor
2. Pair your phone with HC-05. for doing this go to *Settings->Bluetooth->Scan device->select HC-05* as KLSX (**Check backside of module X =0 1 2 3 4 5 6….example KLS1)** and pair it.Pass code to pair is ‘1234’.
3. Now, open the app and connect the HC-05 module in **terminal mode**.
4. Type “0” in terminal to turn OFF LED.
5. Type “1” in terminal to turn ON LED
6. Output: If you send ‘0’ it will make to LED ON and if you send ‘1’ it will make the LED OFF. The Arduino code also sends the current state of LED also.

Program

```
const int led = 13;
void setup() {
  Serial.begin(9600);
  pinMode(led, OUTPUT);
}
void loop() {
  if(Serial.available()>0){
    char data = Serial.read();
    switch(data){
      case '0':
        digitalWrite(led, LOW);
         Serial.println("LED IS OFF");
         break;
      case '1':
        digitalWrite(led, HIGH);
        Serial.println("LED IS ON");
        break;
      default:
        break;
    }
  }
}
```

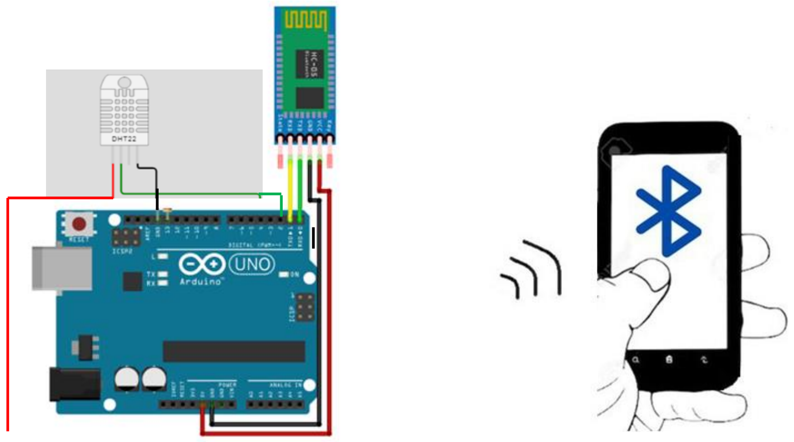
Result: An Arduino program written and verified to turn LED ON/OFF when '1'/'0' is receivedfrom smartphone using Bluetooth.

Experiment No: 5
To interface Bluetooth with Arduino and write a program to send sensor data to smartphone using Bluetooth.

Component Required:

Sl No	Components	Quantity
1	ARDUINO UNO	1
2	HC-05 Bluetooth Module	1
3	USB cable A to B	1
4	Connecting wires	1
5	Smart Phone with RC Controlled Bluetooth app	1

Circuit Diagram



Pin Connection details:

Arduino Pins	Bluetooth Module	Sensor DHT11
5 V	VCC	
GND	GND	GND
3.3 V	--	VCC
D2	--	DATA
Rx	Tx	
Tx	RX	

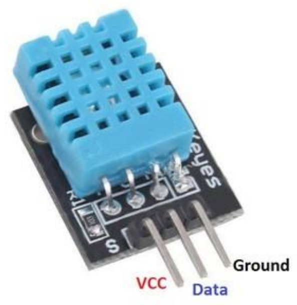
Theory:

The HC-05 is a popular module which can add two-way (full-duplex) wireless functionality. Youcan use this module to communicate between two microcontrollers like Arduino or communicatewith any device with Bluetooth functionality like a Phone or Laptop. There are many android applications that are already available whichmakes this process a lot easier. The module communicates with the help of USART at 9600 baud rate hence it is easy to interface with any microcontroller that supports USART. So if you looking for a Wireless module that could transfer data from your computer or mobile phone to microcontroller or vice versa then this module might be the right choice for you.

HC-05 Technical Specifications

- Serial Bluetooth module for Arduino and other microcontrollers
- Operating Voltage: 4V to 6V (Typically +5V)
- Operating Current: 30mA
- Range: <100m
- Works with Serial communication (USART) and TTL compatible
- Follows IEEE 802.15.1 standardized protocol
- Uses Frequency-Hopping Spread spectrum (FHSS)
- Can operate in Master, Slave or Master/Slave mode
- Can be easily interfaced with Laptop or Mobile phones with Bluetooth
- Supported baud rate: 9600,19200,38400,57600,115200,230400,460800.

The **DHT11 sensor** is a **low-cost** digital temperature and humidity sensor. It operates ata **voltage of 3.3V to 5V** and can measure temperatures rangingfrom **0°C to 50°C** with an accuracy of ±2°C. Additionally, it can measure relative humidity ranging from **20% to 90%** with an accuracy of ±5% .



Program:

```
#include "DHT.h"
#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
void setup() {
 Serial.begin(9600);
 dht.begin();}

void loop()
{
 readSensor();

 }
void readSensor() {
 float h = dht.readHumidity();
 float t = dht.readTemperature();
 if (isnan(h) || isnan(t)) {
 Serial.println("Failed to read from DHT sensor!");
 return;
 }
 Serial.print("Humidity: ");
 Serial.print(h);
 Serial.print(" %\t");
 Serial.println();
 Serial.println();
 Serial.print("Temperature: ");
 Serial.print(t);
 Serial.print(" *C ");
 Serial.println();
 Serial.println();
 delay(2000);
}
```

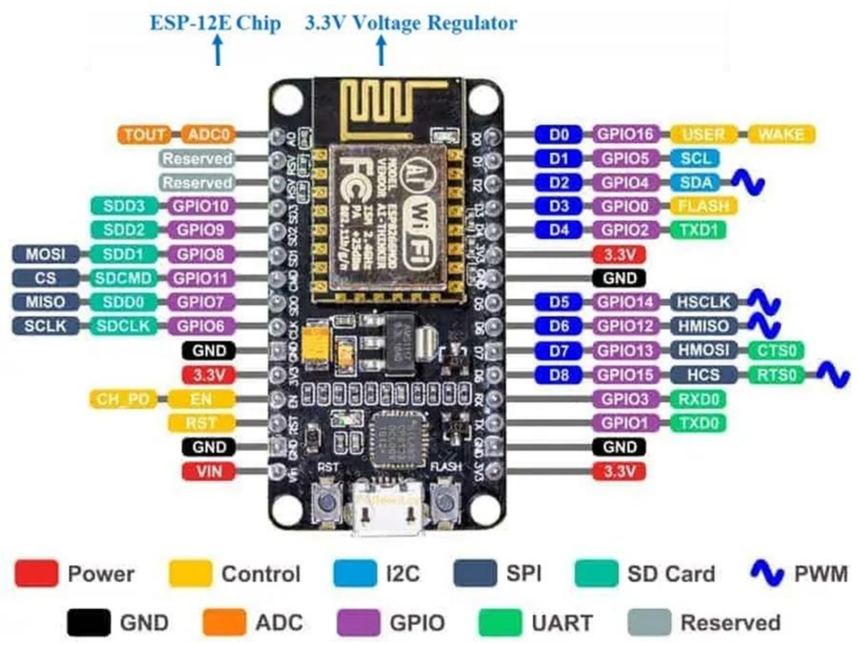
Result: Successfully Interfaced with Bluetooth and sent sensor DHT11 Temperature data tosmartphone using Bluetooth.

Introduction to Node MCU ESP8266 module

The NodeMCU (*N*ode *M*icro-Controller *U*nit) is an open-source software and hardware development environment built around an inexpensive System-on-a-Chip (SoC) called the ESP8266. The ESP8266, designed and manufactured by Espressif Systems, contains the crucial elements of a computer: CPU, RAM, networking (WiFi), and even a modern operating system and SDK. That makes it an excellent choice for Internet of Things (IoT) projects of all kinds.

NodeMCU ESP8266 Specifications & Features and Pin out

- Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
- Operating Voltage: 3.3V
- Input Voltage: 7-12V
- Digital I/O Pins (DIO): 16
- Analog Input Pins (ADC): 1
- UARTs: 1
- SPIs: 1
- I2Cs: 1
- Flash Memory: 4 MB
- SRAM: 64 KB
- Clock Speed: 80 MHz
- USB-TTL based on CP2102 is included onboard, Enabling Plug n Play
- PCB Antenna
- Small Sized module to fit smartly inside your IoT projects



PIN	CODE
A0	A0
GPIO 16	D0
GPIO 5	D1
GPIO 4	D2
GPIO 0	D3
GPIO 2	D4
GPIO14	D5
GPIO 12	D6
GPIO 13	D7
GPIO 15	D8
GPIO 9	SD2
GPIO10	SD3
GPIO3	Rx
GPIO1	Tx

Steps to install Node MCU

1. Download and install Arduino IDE
2. Open the IDE and follow this path. **File -> preferences -> Additional board manager URL.**
3. Now paste the URL in the dialog box **: http://arduino.esp8266.com/stable/package_esp8266com_index.json**
4. Then, click the "**OK**" button.
5. Now follow this path**. Tools -> Board -> Boards Manager**
6. Search for **ESP8266** and install the "**ESP8266 by ESP8266 Community**"
7. After this, restart your Arduino IDE.
8. Then, go to **Tools** > **Board** and check that you have ESP8266 boards available.

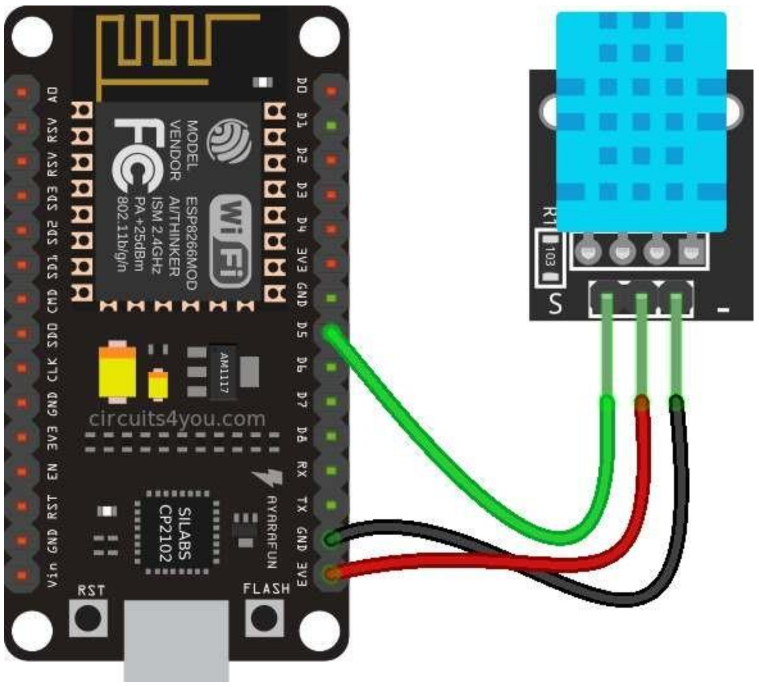
First, make sure you have an ESP8266 selected in **Tools** > **Board**. If you're using the ESP8266-12E NodeMCU Kit as shown in previous pictures, select the **NodeMCU 1.0 (ESP-12E Module)** option.

Experiment No: 6
Write a program on Arduino to upload temperature and humidity data to thingspeak cloud.

Component Required:

Sl No	Components	Quantity
1	ESP8266 12E Node MCU Kit	1
2	Temperature sensor DHT11	1
3	USB cable A to B	1
4	Connecting wires	1
5	Breadboard	1

Circuit diagram



Pin Connection details	
NODE MCU	**DHT11**
GND	GND
3V3	VCC
D5/D4	DATA

Theory:

ThingSpeak is an open-source Internet of Things (IoT) application and API that allows users to collect and store sensor data in the cloud and perform analytics on that data. It allows users to create "channels" to collect data from multiple sensors, and also has built-in support for visualizing and analyzing the data. ThingSpeak can be used for a variety of applications, such as monitoring environmental conditions, tracking the location of assets, and controlling devices remotely. It is available for free and also has paid subscription plans for additional features and support. The device that sends the data must be configured with the correct channel information, such as the channel ID and write API key.

- 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.

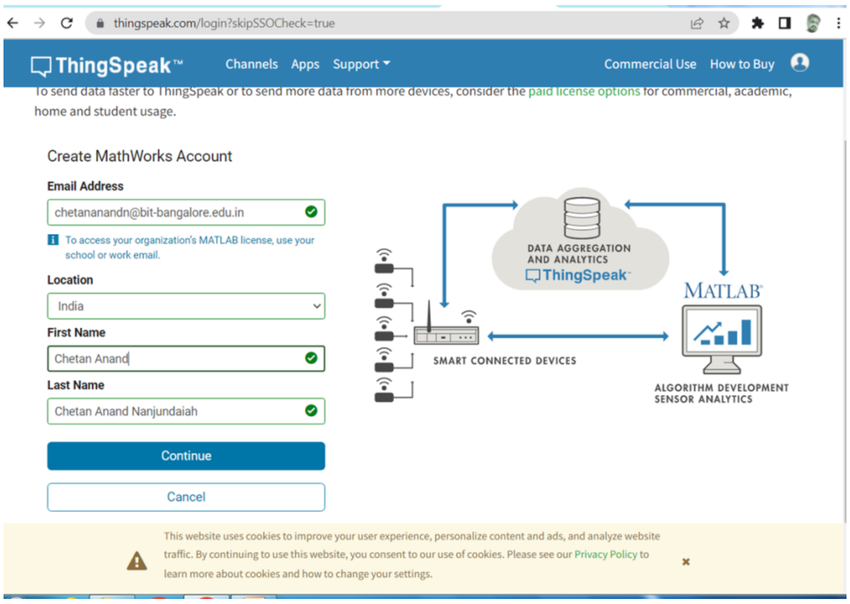
The core element of ThingSpeak is a 'ThingSpeak Channel'.
A channel stores the data that we send to ThingSpeak and comprises of the below elements:

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 sign up and create a channel.
- Once we have a channel, we can send the data, allow ThingSpeak to process it and also retrieve the same.

Creating a ThingSpeak Channel

Step 1: Open https://thingspeak.com/ and click on the 'Get Started Now' button on the center of the page and you will be redirected to the sign-up page(you will reach the same page when you click the 'Sign Up' button on the extreme right).
Fill out the required details and click on the 'Create Account' button


thingspeak.com/login?skipSSOCheck=true

To send data faster to ThingSpeak or to send more data from more devices, consider the paid license options for commercial, academic, home and student usage.

Create MathWorks Account

Email Address: chetananandn@bit-bangalore.edu.in

To access your organization's MATLAB license, use your school or work email.

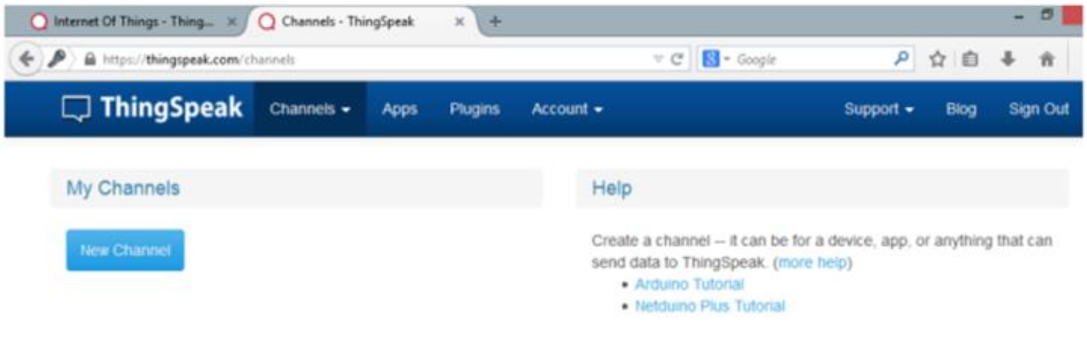
Location: India

First Name: Chetan Anand

Last Name: Chetan Anand Nanjundaiah

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

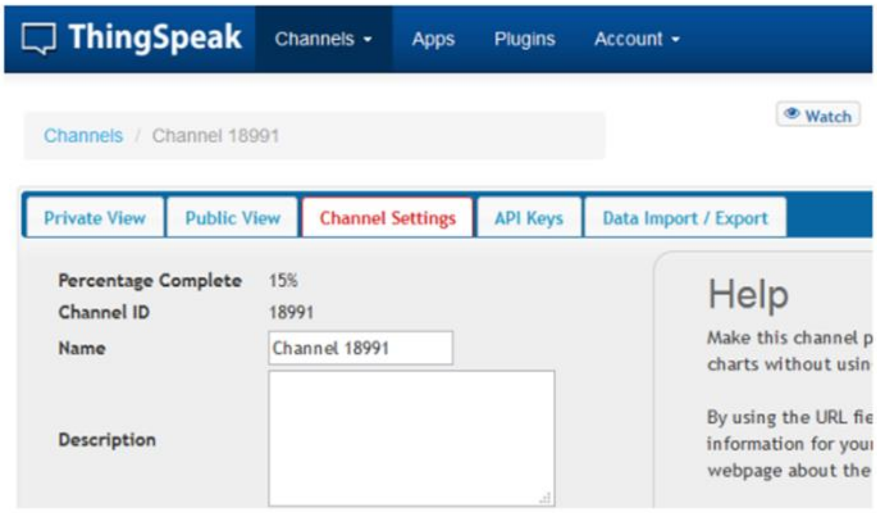
Now you should see a page with a confirmation that the account was successfully created. The confirmation message disappears after a few seconds and the final page should look as in the below screen:



Create a channel -- it can be for a device, app, or anything that can send data to ThingSpeak. (more help)

- Arduino Tutorial
- Netduino Plus Tutorial

Step 2: Go ahead and click on 'New Channel'. You should see a page like the below:



Percentage Complete	15%
Channel ID	18991
Name	Channel 18991
Description	

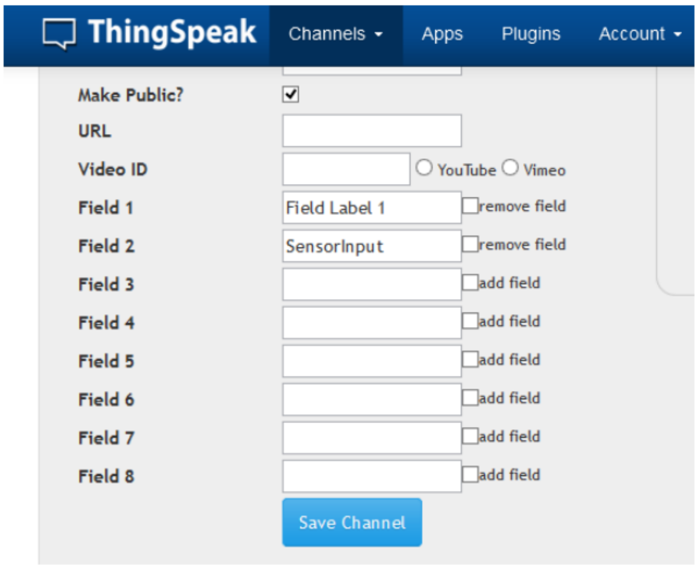
Help

Make this channel p
charts without usin

By using the URL fie
information for you
webpage about the

Change the name to fit your need

Add a description corresponding to the channel



Make Public?	✓	
URL		
Video ID		○ YouTube ○ Vimeo
Field 1	Field Label 1	☐ remove field
Field 2	SensorInput	☐ remove field
Field 3		☐ add field
Field 4		☐ add field
Field 5		☐ add field
Field 6		☐ add field
Field 7		☐ add field
Field 8		☐ add field

Fields 1 to 8 - These are the fields which correspond to the data sent by a sensor or a 'thing'. A field has to be added before it can be used to store data. By default, Field 1 is added.

Once you have edited the fields, click on '**Save Channel**' button.

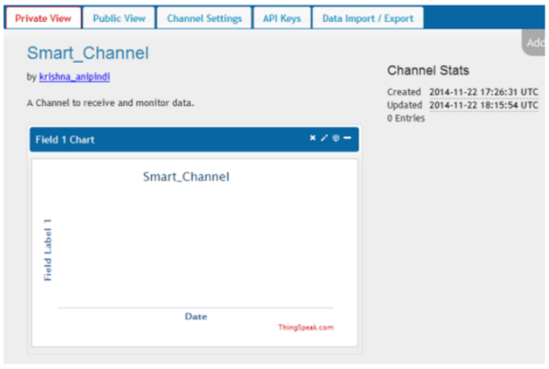
Latitude, longitude and elevation:
These fields correspond to the location of a 'thing' and are especially significant for moving things.

Make Public?
- If the channel is made public, anyone can view the channel's data feed and the corresponding charts. If this check box is not checked, the channel is private, which means for every read or write operation, the user has to pass a corresponding API key.

URL :
This can be the URL of your blog or website and if specified, will appear on the public view of the channel

Video ID:
This is the ID corresponding to your YouTube or Vimeo ID. If specified, the video appears on the public view of the channel.

Step 3: 'Private View' tab is defaulted:



Smart_Channel

by krishna_anipindi

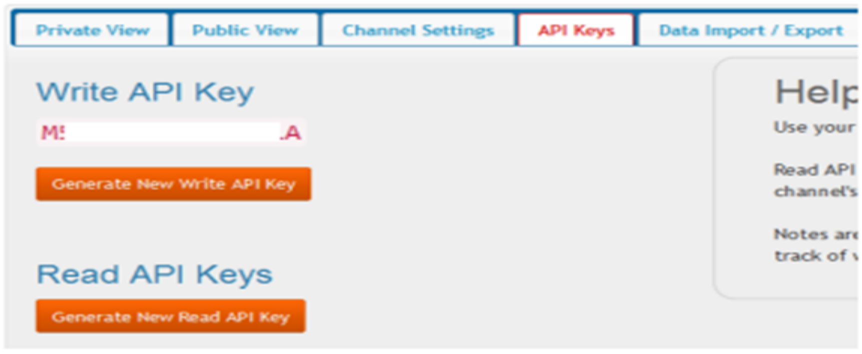
A Channel to receive and monitor data.

Channel Stats

Created 2014-11-22 17:26:31 UTC
Updated 2014-11-22 18:15:54 UTC
0 Entries

The Private View shows a chart corresponding to each of the fields that we have added. Now click on the 'Public View' tab. This should look exactly similar to the what we see in the 'Private View' tab since our channel is public.

Step 4: click on the 'API Keys' tab



Write API Key

M! [redacted] A

Read API Keys

Help

Use your

Read API
channel's

Notes are
track of

The write API key is used for sending data to the channel
The read API key(s) is used to read the channel data

Share the Read API keys with people who are approved and authorized to view your channel.

Step 5: Installing the ThingSpeak Library

To send or receive sensor readings to ThingSpeak, we'll use the ThingSpeak Arduino library.
Go to Sketch > Include Library > Manage Libraries… and search for "ThingSpeak" in the Library Manager.
Install the ThingSpeak library by **MathWorks**

Code

```
#include <DHT.h> #include
<ESP8266WiFi.h>

const char* ssid = "SYS123";
const char* password ="12345678";
String apiKey = "TG2DHQ689TQX9MVV"; //Write API Key from website
const char* server = "api.thingspeak.com";


#define DHTPIN 14 // 14-D5, 2-D4
#define DHTTYPE DHT11
DHT dht(DHTPIN,   DHTTYPE);
float humidity;
float temp;

WiFiClient client; //need to communication with the ThingSpeak server

void setup() {
  Serial.begin(9600);
  dht.begin(); WiFi.begin(ssid, password);
  Serial.print("Wifi connecting to ...SSID:");
  Serial.println(ssid); Serial.print("connecting");

  while(WiFi.status()!=WL_CONNECTED){delay(500);
    Serial.print(".");
  }

 Serial.println("Wifi connected successfully");
 Serial.print("nodemcu Ip Adress");
 Serial.println(WiFi.localIP());

}
```

```
void loop()
            { temp=dht.readTemperature();
              humidity= dht.readHumidity();

  if(client.connect(server, 80)){

     String postStr = apiKey;
     postStr += "&field1=";
     postStr += String(temp);
     postStr += "&field2=";
     postStr += String(humidity);
     postStr += "\r\n\r\n";

     client.print("POST /update HTTP/1.1\n");
     client.print("Host: api.thingspeak.com\n");
     client.print("Connection: close\n");
     client.print("X-THINGSPEAKAPIKEY: "+apiKey+"\n");
     client.print("Content-Type: application/x-www-form-urlencoded\n");
     client.print("Content-Length: ");
     client.print(postStr.length());
     client.print("\n\n");
     client.print(postStr);
     delay(1000); //necessary for posting to ThingSpeak

     Serial.print("Temperature = ");
     Serial.println(temp,2);
     Serial.print("Humidity = ");
     Serial.print(humidity, 2);
     Serial.println (" %");
     Serial.println("Data sent to ThingSpeak");
   }
   client.stop();
   Serial.println("Waiting for 15sec...");
   delay(15000); //ThingSpeak needs min. 15sec delay between each data post
}
```

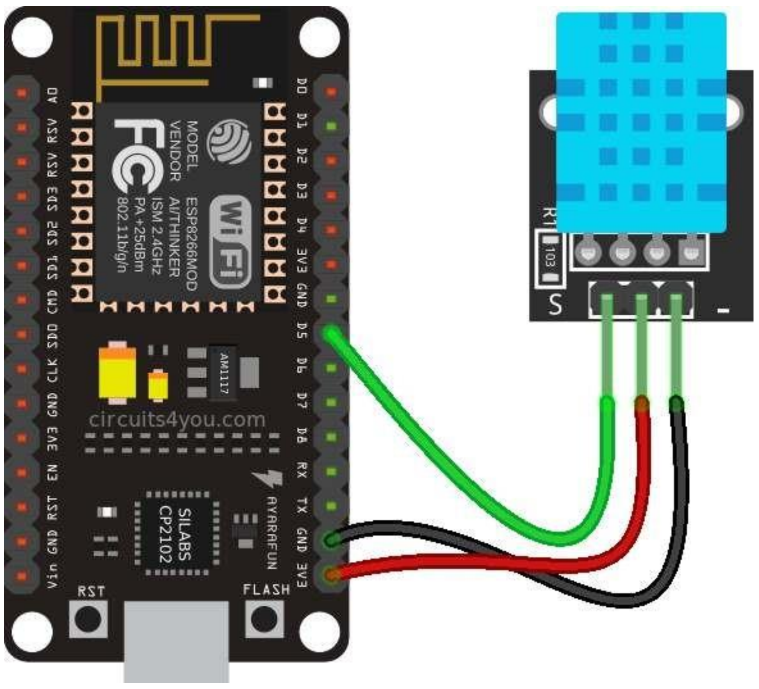
Result: Program written on Arduino IDE to upload temperature and humidity data to thingspeakcloud.

Experiment No: 7
Write a program on Arduino to retrieve temperature and humidity data from Thingspeak cloud.

Component Required:

Sl No	Components	Quantity
1	Node MCU	1
2	Temperature sensor DHT11	1
3	USB cable	1
4	Connecting wires	--
5	Breadboard	1

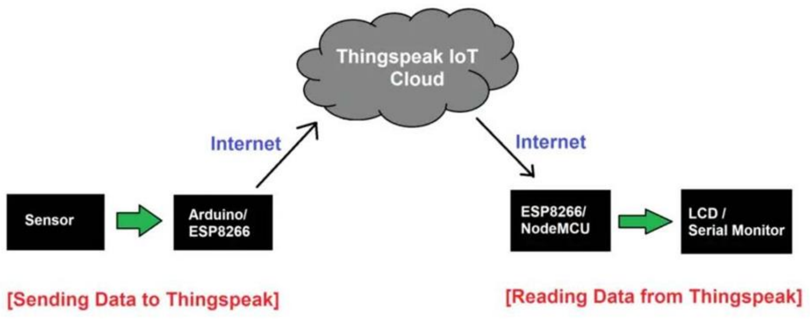
Circuit diagram



Pin Connection details	
NODE MCU	**DHT11**
GND	GND
3V3	VCC
D5	DATA

Theory:

To read values from Thingspeak we need to upload some data in real time, to do this, first upload temperature and humidity data to Thingspeak using previous experiment using NodeMCU 8266



Channel Settings that you need to do read data

1. Go to your Thingspeak account and do the following setting to receive temperature and humidity data.
2. Go to channel setting put 'tick' mark for both filed 1 and filed 2 and scroll down to bottom and save it.
3. You need your channel ID to read the fields on your channel you wish to read so that copy your channel id and paste in the code
4. You need your **Read API key** from your channel and copy **Read API key**.
5. Use this **Read API key** in our code.

Write following program and upload in the Node MCU82666
After successful upload Open the serial monitor; you will be able to see the values read from your channel.

Code:

```
#include <ThingSpeak.h>
#include <ESP8266WiFi.h>

const char* ssid = "SYS123";
const char* password ="12345678";

//---------Channel Details  //
// enter your Channel ID
unsigned long counterChannelNumber = 2380293;
// enter your Read API Key
const char * myCounterReadAPIKey = "Q56EYHWYZIXXZ31F";
const int FieldNumber1 = 1; // The field you wish to read
const int FieldNumber2 = 2; // The field you wish to read

WiFiClient  client; //need to communication with the ThingSpeak server
void setup()
{
Serial.begin(9600);
WiFi.begin(ssid, password);

WiFi.mode(WIFI_STA);
ThingSpeak.begin(client);


while(WiFi.status()!=WL_CONNECTED){ delay(500);
Serial.println("Wifi connecting ");
}
Serial.println("Wifi connected successfully ");
}

void loop(){
//  Channel 1 //
long temp = ThingSpeak.readLongField(counterChannelNumber, FieldNumber1, myCounterReadAPIKey);
int statusCode = 0;
statusCode = ThingSpeak.getLastReadStatus();
if (statusCode == 200)
{
Serial.print("Temperature: ");
Serial.println(temp);
}
else
{
Serial.println("Unable to read channel / No internet connection");
}
delay(100);
//  Channel 2 //
long humidity = ThingSpeak.readLongField(counterChannelNumber, FieldNumber2,myCounterReadAPIKey);
statusCode = ThingSpeak.getLastReadStatus();
if (statusCode == 200)
{
Serial.print("Humidity: ");
Serial.println(humidity);
}
else
{
Serial.println("Unable to read channel / No internet connection");
}
delay(100);
//  End of Channel 2  //
}
```

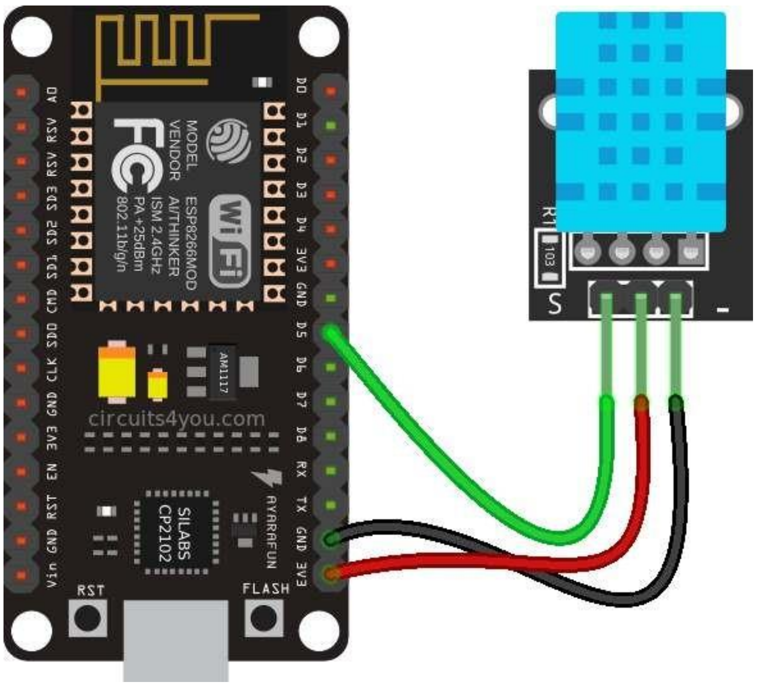
Result: Program on Arduino IDE written to retrieve temperature and humidity data from Thingspeak cloud.

Experiment No: 8
Write a program on NodeMCU to publish temperature data to MQTT broker.

Component Required:

Sl No	Components	Quantity
1	Node MCU	1
2	Temperature sensor DHT11	1
3	USB cable	1
4	Connecting wires	--
5	Breadboard	1

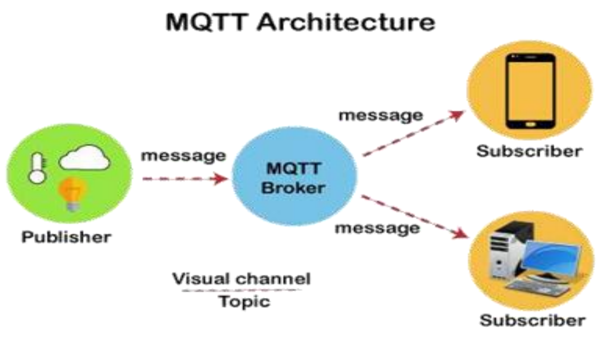
Circuit diagram


circuits4you.com
ESP8266MOD
AI/THINKER
SILABS
CP2102
RST
FLASH
D0 D1 D2 D3 D4 3V3 GND D5 D6 D7 D8 RX TX GND 3V3
A0 RSV RSV SD3 SD2 SD1 CMD SDO CLK GND 3V3 EN RST GND Vin
S

Pin Connection details	
NODE MCU	**DHT11**
GND	GND
3V3	VCC
D3 / D5	DATA

Theory:

MQTT stands for Message Queuing Telemetry Transport. MQTT is a simple messaging protocol, designed for constrained devices with low bandwidth. So, it's the perfect solution to exchange data between multiple IoT devices.
Devices publish messages on a specific topic. All devices that are subscribed to that topic receive the message.



In a publish and subscribe system, a device can publish a message on a topic, or it can be subscribed to a particular topic to receive messages.
The MQTT broker is responsible for receiving all messages, filtering the messages, deciding who is interested in them, and then publishing the message to all subscribed clients.

The MQTT broker is the central point of communication, and it is in charge of dispatching all messages between the senders and the rightful receivers. A client is any device that connects to the broker and can publish or subscribe to topics to access the information. A topic contains the routing information for the broker. Each client that wants to send messages publishes them to a certain topic, and each client that wants to receive messages subscribes to a certain topic. The broker delivers all messages with the matching topic to the appropriate clients.

- ThingSpeak™ has an MQTT broker at the URL mqtt3.thingspeak.com and port 1883.
- The ThingSpeak broker supports both MQTT publish and MQTT subscribe

In this Experiment, we will create a setup that allows a NODE MCUESP8266 board to send data to another MCU ESP 8266, using MQTT (Message Queuing Telemetry Transport). The sender device simply publishes a message to a broker service, which then can be subscribed to by a receiver device.
The data we will send consists of readings from a DHT11 sensor, including temperature and humidity data, from a NODE MCU ESP8266 to another NODE MCU. This experiment utilizes the broker test.mosquitto.org, an open-source service that is free for anyone to use.

Code:

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <DHT.h>
const char *ssid =  "SYS123";     // replace with your wifi ssid and wpa2 key
const char *password =  "12345678"; //
const char* mqtt_server = "test.mosquitto.org"; // MQTT broker address
const int mqtt_port = 1883;
const char* temperature_topic = "home/temp"; // MQTT topic to publish temperature data

#define DHTPIN D3
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

WiFiClient espClient; // establish a network connection over Wi-Fi.
PubSubClient client(espClient); // enables MQTT communication on the Arduino platform.
void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
}
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    if (client.connect("ESP8266Client")) {
      Serial.println("connected");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}

void setup() {
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, mqtt_port);
  dht.begin();
}
```

```
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

    float temperature = dht.readTemperature();
  if (!isnan(temperature))
   { // Check if temperature reading is valid
    Serial.print("Temperature: ");
    Serial.println(temperature);
    char tempString[8];
    dtostrf(temperature, 6, 2, tempString); // Convert float to string
    client.publish(temperature_topic, tempString);//Publish temperature data to MQTT
broker
  } else {
    Serial.println("Failed to read temperature from DHT sensor");
  }

  delay(5000); // Publish temperature data every 5 seconds
}
```

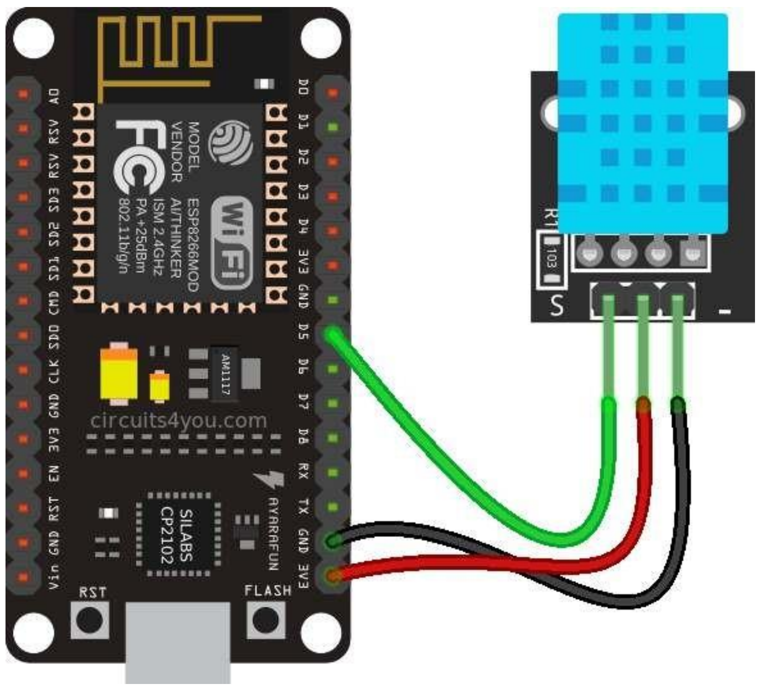
Result: Program written on Node MCU to publish temperature data to MQTT broker.

Experiment No: 9
Write a program on NodeMCU to subscribe to MQTT broker for temperature data and print it.

Component Required:

Sl No	Components	Quantity
1	Node MCU	1
2	Temperature sensor DHT11	1
3	USB cable	1
4	Connecting wires	--
5	Breadboard	1

Circuit diagram



Pin Connection details	
NODE MCU	**DHT11**
GND	GND
3V3	VCC
D5	DATA

Theory:

The MQTT Protocol was first introduced in 1999,as a light- weight **publish** and **subscribe** system. It is particularly useful for devices with low-bandwidth, where we can send commands, sensor values or messages over the Internet with little effort.

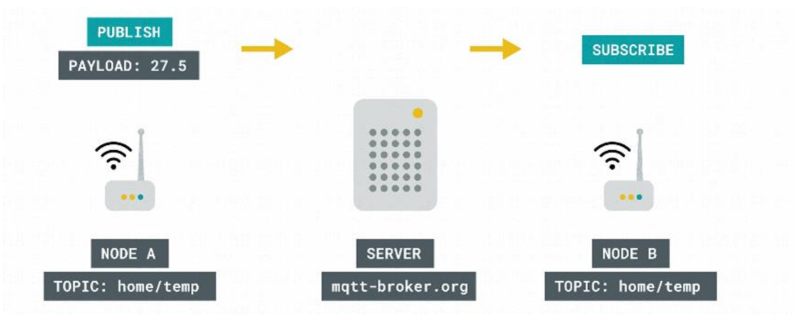
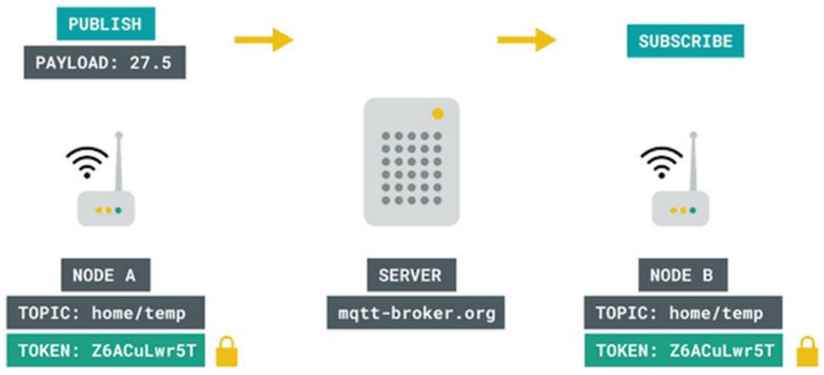


Figure gives basic explanation on how it works as a node, for example and Arduino with a Wi-Fi module, sends a payload to a broker.
A broker is a kind of "middle-point" server, that essentially stores payloads sent to it, in something called **topics**.
A topic, is a definition of what type of data it contains, it could for example be "humidity" or " temperature".
Another node can then subscribe to this information, from the broker, and voilà, data has been moved from Node A to Node B over the Internet.

One way to protect the data is for example, by using a **token**, something that is quite common when working with various IoT services. For instance, if we are publishing something to a broker, anyone that has the URL, e.g. **randombroker.org/randomtopic** can subscribe to it. But if we add a unique token on both sides, they wouldn't be able to.
These tokens could for example be **Z6ACuLwr5T**, which is not exactly something easy to guess.



Code:

```
#include <ESP8266WiFi.h>
#include <ArduinoMqttClient.h>

const char *ssid =  "SYS123";     // replace with your wifi ssid and wpa2 key
const char *password =  "12345678"; // password of your ssid

const char broker[] = "test.mosquitto.org";
int port  = 1883;
const char topicT[] = "home/temp";

WiFiClient wifiClient;
MqttClient mqttClient(wifiClient);

void setup()
{ Serial.begin(9600);
Serial.print("Wifi connecting to  ");
Serial.print(ssid);
WiFi.begin(ssid, password);

while(WiFi.status()!=WL_CONNECTED)
{ Serial.println("connecting  ");
delay(500);
}
Serial.println("Wifi connected successfully");
Serial.print("Attempting to connect to the MQTT broker: ");
Serial.println(broker);

if (!mqttClient.connect(broker, port))
{ Serial.print("MQTT connection failed! Error code = ");
Serial.println(mqttClient.connectError());

while (1);
}
Serial.println("You're connected to the MQTT broker!");
Serial.println();

// set the message receive callback
mqttClient.onMessage(onMqttMessage);
Serial.print("Subscribing to topic: ");
Serial.println(topicT);
Serial.println();
// subscribe to a topic
mqttClient.subscribe(topicT);

Serial.print("Topic Tempertaure: ");
Serial.println(topicT);
Serial.println();

}
```

```
void loop()
{ mqttClient.poll();
delay(1000);
}
void onMqttMessage(int messageSize)
{
// received message, print out the topic and contents
Serial.println("Received a message with topic '");
Serial.print(mqttClient.messageTopic());
Serial.print("', length ");
Serial.print(messageSize);
Serial.println(" bytes:");

// use the Stream interface to print the contents
while (mqttClient.available())
{
Serial.print((char)mqttClient.read());
}
Serial.println();
Serial.println();
delay(1500);
}
```

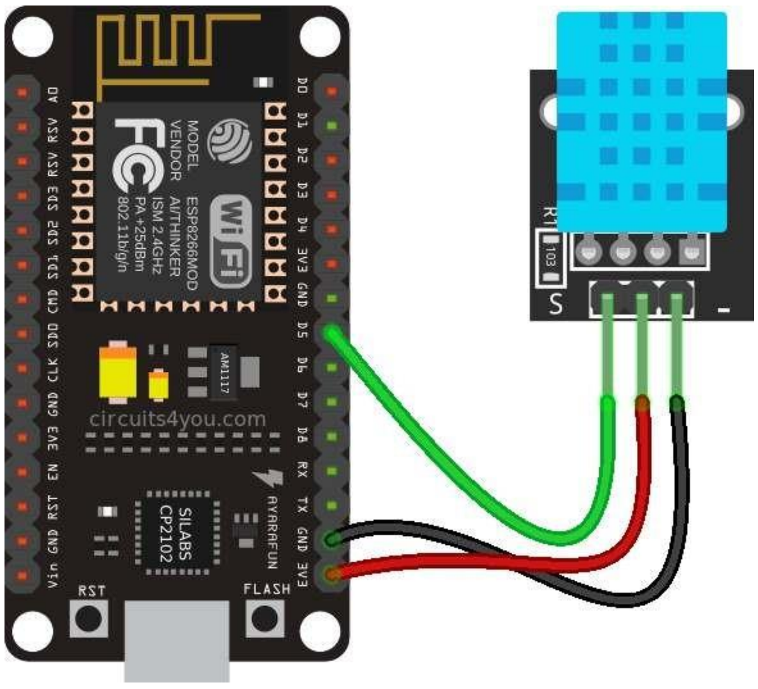
Result: Program written on NodeMCU to subscribe to MQTT broker to print temperature and humidity data.

Experiment No: 10
Write a program to create UDP server on NodeMCU and respond with humidity data to UDP client when requested

Component Required:

Sl No	Components	Quantity
1	Node MCU	1
2	Temperature sensor DHT11	1
3	USB cable	1
4	Connecting wires	--
5	Breadboard	1

Circuit diagram



Pin Connection details	
NODE MCU	**DHT11**
GND	GND
3V3	VCC
D4 / D5	DATA

Theory:

User Datagram Protocol (UDP) is a network communication protocol that operates at the transport layer of the Internet Protocol (IP) suite. It is a connectionless and lightweight protocol designed for fast and efficient data transmission, but it does not provide the same level of reliability and error-checking as Transmission Control Protocol (TCP).
UDP is a lightweight, connectionless, and fast protocol that prioritizes low-latency data transmission over reliability. It is suitable for applications where occasional packet loss or out- of-order delivery can be tolerated, and real-time communication is essential.
However, for applications that require guaranteed delivery and error recovery, TCP is a better choice.

```
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>
#include <DHT.h>

#define DHTPIN D4
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

float temp;
char packet[1024];
const char* ssid = "SYS123";
const char* password ="12345678";
const char* udpServerIP = "192.168.137.1";//Replace with the actual IP address
unsigned int port = 5129;
WiFiUDP Udp; // Create a UDP object
void setup()
{
Serial.begin(9600);
dht.begin();
WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED)
{ delay(500);
Serial.println("Connecting to WiFi...");
}
Serial.println("Wifi connected successfully");
}
```

```
void loop()
{ Udp.beginPacket(udpServerIP, port);

// Read temperature from DHT sensor and convert it to a string
temp = dht.readTemperature();
//confirm value in serila monitor
Serial.print("Temperature in degree Cel. :");
Serial.println(temp);


String tempStr = String(temp);

// Copy the temperature string into the packet buffer
tempStr.toCharArray(packet, 1024);

// Send the packet over UDP
Udp.write(packet);

// End the UDP packet and send it
Udp.endPacket();
delay(500);
}
```

Python code for UDP Server

```
import socket

UDP_IP = "192.168.137.1"
UDP_PORT = 5129

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((UDP_IP, UDP_PORT))

while True:
      received_data, addr = sock.recvfrom(1024)
      print(received_data)
```

Procedure

To run Python code for UDP Server

1) Install python software
2) Open python IDLE
3) In python IDLE go to **File** **New File** (it opens new script windows) **Type the code**
4) Click on **Run** button and **Save** the program from script window
5) See the output from IDLE shell (command prompt)

Result: Python program used to create UDP server and Arduino program written to respond with humidity data to UDP client when requested.