ATME College of Engineering

DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

LABORATORY MANUAL

ARDUINO AND RASPBERRY PI BASED PROJECT

BEEL456D

ACADEMIC YEAR 2024-25

SEMESTER: IV

Prepared by:

Mr. Shreeshayana R (Lab In-charge)

Verified by:

Approved by:

INSTITUTIONAL VISION AND MISSION

VISION:

Development of academically excellent, culturally vibrant, socially responsible and globally competent human resources.

MISSION:

- To keep pace with advancements in knowledge and make the students competitive and capable at the global level.
- To create an environment for the students to acquire the right physical, intellectual, emotional and moral foundations and shine as torchbearers of tomorrow's society.
- To strive to attain ever-higher benchmarks of educational excellence

DEPARTMENT VISION AND MISSION

VISION:

To create Electrical and Electronics Engineers who excel to be technically competent and fulfill the cultural and social aspirations of the society.

MISSION:

- To provide knowledge to students that builds a strong foundation in the basic principles of electrical engineering, problem solving abilities, analytical skills, soft skills and communication skills for their overall development.
- To offer outcome based technical education.
- To encourage faculty in training & development and to offer consultancy through research & industry interaction.

PROGRAMME OUTCOMES:

Engineering Graduates will be able to:

PO1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of EXPERIMENTs, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes (PSOs)

At the end of graduation, the student will be able,

PSO1: Apply the concepts of Electrical & Electronics Engineering to evaluate the performance of power systems and also to control Industrial drives using power electronics.

PSO2: Demonstrate the concepts of process control for Industrial Automation, design models for environmental and social concerns and also exhibit continuous self- learning.

Program Educational Objectives (PEOs)

PEO1: To produce competent and Ethical Electrical and Electronics Engineers who will exhibit the necessary technical and managerial skills to perform their duties in society.

PEO2: To make graduates continuously acquire and enhance their technical and socio-economic skills.

PEO3: To aspire graduates on R&D activities leading to offering solutions and excel in various career paths.

PEO4: To produce quality engineers who have the capability to work in teams and contribute to real time projects.

ARDUINO AND RASPBERRY PI BASED PROJECT

Course Code	**:**	**BEEL456D**	**CIE**	**:**	**50**
Hours/Week	**:**	**02**	**Exam Hours**	**:**	**03**
Credits	**:**	**01**	**SEE**	**:**	**50**
Examination Type(SEE)	**:**	**Practical**			

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

LIST OF EXPERIMENTS

Expt.No.	Name of the Experiment	COs
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.	CO1, CO2,CO4
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.	CO1, CO2, CO4
3.	To interface motor using relay with Arduino/Raspberry Pi and write a program to 'turn ON' motor when push button is pressed	CO2,CO4
4.	To interface Bluetooth with Arduino/Raspberry Pi and write a program to send sensor data to Smartphone using Bluetooth	CO2,CO4
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	CO2,CO4
6.	Write a program on Arduino/Raspberry Pi to upload temperature and humidity data to thing speak cloud	CO4,CO4
7.	Write a program on Arduino/Raspberry Pi to retrieve temperature and humidity data from thing speak cloud.	CO3, CO4
8.	Write a program on Arduino/Raspberry Pi to publish temperature data to MQTT broker	CO3, CO4
9.	Write a program to create UDP server on Arduino/Raspberry Pi and respond with humidity data to UDP client when requested	CO3,CO4
10.	Write a program to create TCP server on Arduino/Raspberry Pi and respond with humidity data to TCP client when requested.	CO3, CO4
11.	Write a program on Arduino/Raspberry Pi to subscribe to MQTT broker for temperature data and print it.	CO3, CO4

REFERNCE BOOKS:

1. https://www.arduino.cc
2. https://www.raspberrypi.org/
3. Course in Internet of Things (IOT) Using Arduino - NIELIT Delhi Centre
4. Vijay Madisetti, Arshdeep Bahga, Internet of Things. "A Hands on Approach", University Press
5. Dr. SRN Reddy, Rachit Thukral and Manasi Mishra, "Introduction to Internet of Things: A practical Approach", ETI Labs
6. Pethuru Raj and Anupama C Raman, "The Internet of Things: Enabling Technologies, Platforms, and Use Cases", CRC Press
7. Jeeva Jose, "Internet of Things", Khanna Publishing House, Delhi
8. Adrian McEwen, "Designing the Internet of Things", Wiley
9. Raj Kamal, "Internet of Things: Architecture and Design", McGraw Hill

COURSE OUTCOMES

At the end of the course the student will be able to:

COs	Statement	RBTL
CO-1	**Explain** the concepts of Internet of Things and its hardware and software components	**L2**
CO-2	**Evaluate** Interfacing of I/O devices, sensors & communication modules.	**L5**
CO-3	**Evaluate** Remotely monitoring data and control devices.	**L5**
CO-4	**Develop** real life IoT based projects	**L5**

Cycle of Experiments

CYCLE-1	
Expt. No.	**Name of the Experiment**
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 Bluetooth with Arduino/Raspberry Pi and write a program to send sensor data to Smartphone using Bluetooth
4.	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
5.	To interface motor using relay with Arduino/Raspberry Pi and write a program to ‘turn ON’ motor when push button is pressed

CYCLE-II	
6.	Write a program on Arduino/Raspberry Pi to upload temperature and humidity data to thing speak cloud
7.	Write a program on Arduino/Raspberry Pi to retrieve temperature and humidity data from thing speak cloud.
8.	Write a program on Arduino/Raspberry Pi to publish temperature data to MQTT broker
9.	Write a program to create UDP server on Arduino/Raspberry Pi and respond with humidity data to UDP client when requested
10.	Write a program to create TCP server on Arduino/Raspberry Pi and respond with humidity data to TCP client when requested.
11.	Write a program on Arduino/Raspberry Pi to subscribe to MQTT broker for temperature data and print it.

Table of Contents

1. Arduino Uno

The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits.

The board has 14 digital 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.

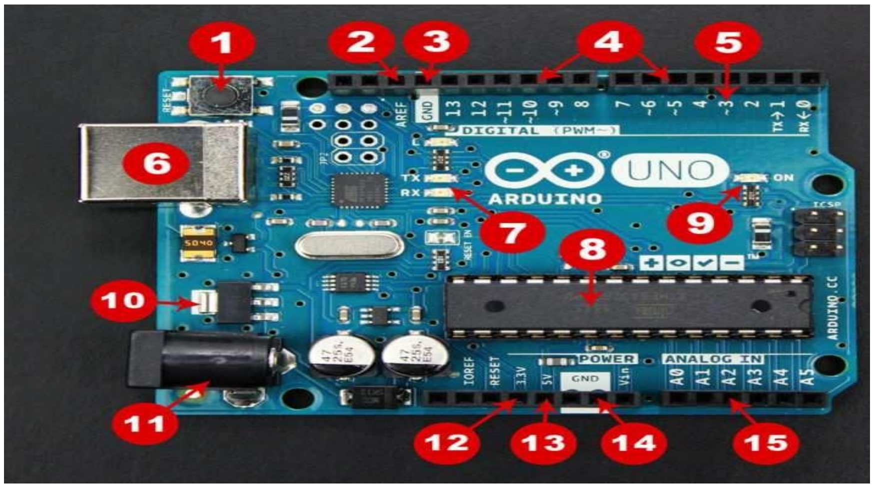


Figure-1: Arduino Uno Board

Here are the components that make up an Arduino board and what each of their functions are.

- Reset Button – This will restart any code that is loaded to the Arduino board
- AREF – Stands for "Analog Reference" and is used to set an external reference voltage
- Ground Pin – There are a few ground pins on the Arduino and they all work the same
- Digital Input/ Output – Pins 0-13 can be used for digital input or output
- PWM – The pins marked with the (~) symbol can simulate analog output
- USB Connection – Used for powering up your Arduino and uploading sketches
- TX/RX – Transmit and receive data indication LEDs
- ATmega Microcontroller – This is the brains and is where the programs are stored
- Power LED Indicator – This LED lights up anytime the board is plugged in a power source
- Voltage Regulator – This controls the amount of voltage going into the Arduino board
- DC Power Barrel Jack – This is used for powering your Arduino with a power supply
- 3.3V Pin – This pin supplies 3.3 volts of power to your projects
- 5V Pin – This pin supplies 5 volts of power to your projects

- Ground Pins – There are a few ground pins on the Arduino and they all work the same
- Analog Pins – These pins can read the signal from an analog sensor and convert it to digital.

1.1 Features of the Arduino

- 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.
- The board functions can be controlled by sending a set of instructions to the microcontroller on the board via Arduino IDE.
- Arduino IDE uses a simplified version of C++, making it easier to learn to program.
- Arduino provides a standard form factor that breaks the functions of the micro- controller into a more accessible package.

2. ARDUINO IDE (Integrated Development Environment)

2.1 Introduction:

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++.

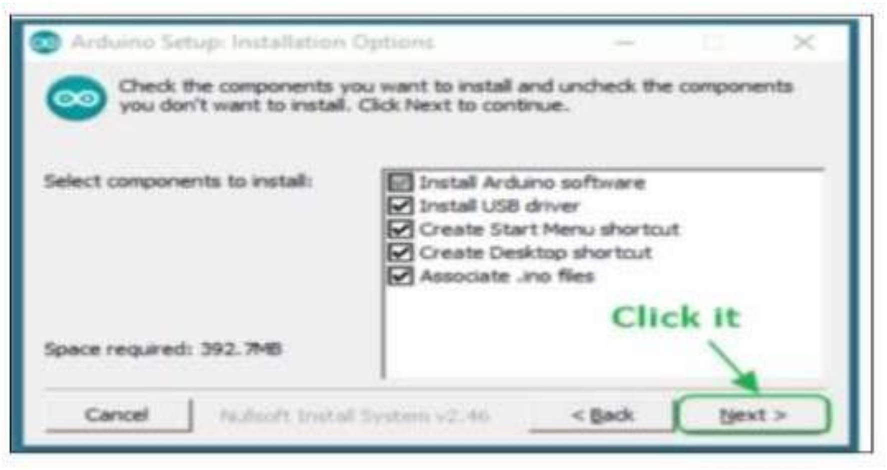
The open-source Arduino Software (IDE) makes it easy to write code and upload it to the the board. This software can be used with any Arduino board.

- 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.

2.2 INSTALLATION OF ARDUINO SOFTWARE (IDE)

Step-1: Downloading

- To install the Arduino software, download this page: http://arduino.cc/en/Main/Software and proceed with the installation by allowing the driver installation process. Or Download page on the Arduino Official website.



Arduino Setup: Installation Options

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

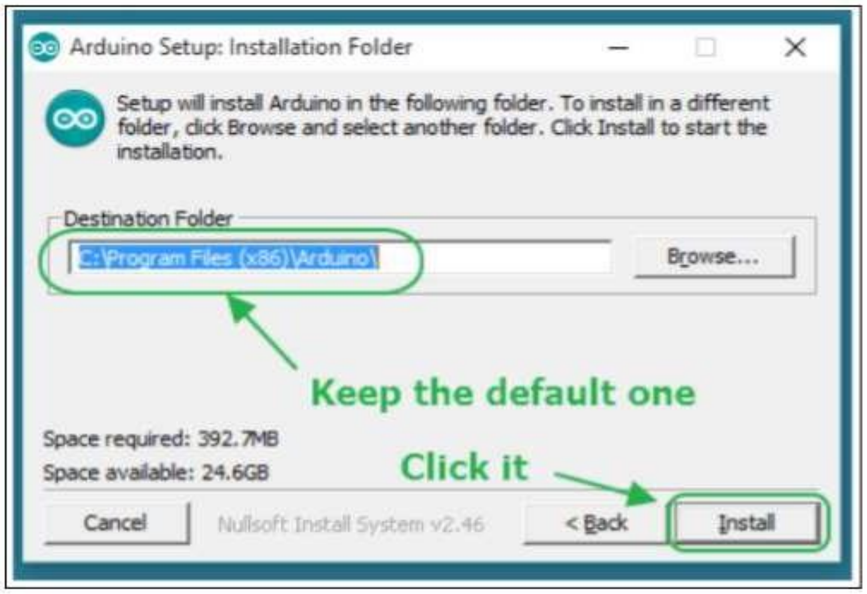
Select components to install:

- [✓] Install Arduino software
- [✓] Install USB driver
- [✓] Create Start Menu shortcut
- [✓] Create Desktop shortcut
- [✓] Associate .ino files

Space required: 392.7MB

Step-2: Directory Installation

- Choose the installation directory.



Arduino Setup: Installation Folder

Setup will install Arduino in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

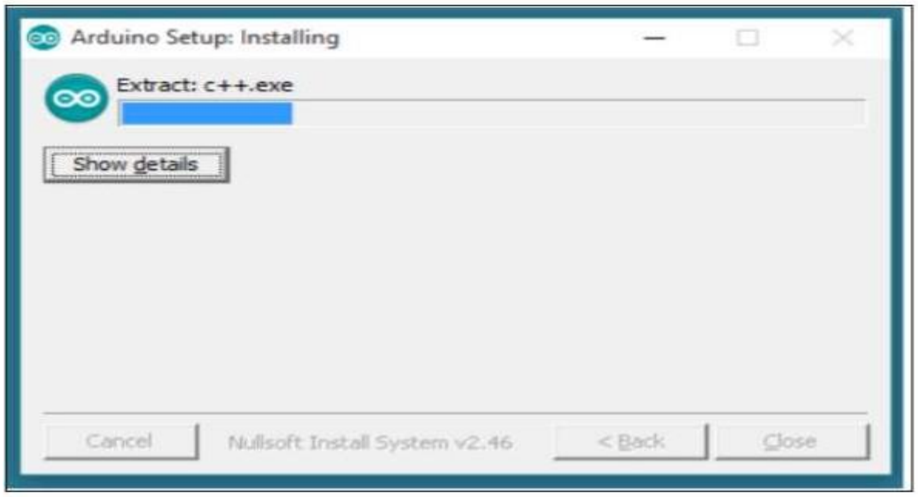
Destination Folder: C:\Program Files (x86)\Arduino\

Space required: 392.7MB

Space available: 24.6GB

Step-3: Extraction of Files

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

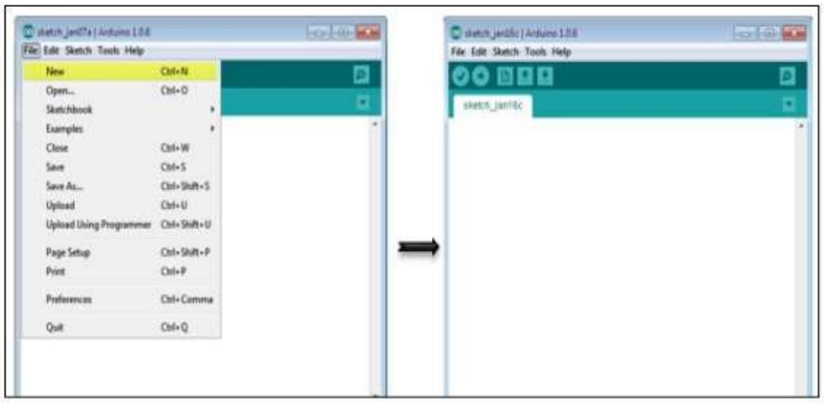


Step 4: Connecting the board

- Connect the board to the computer using the USB cable. The green power LED (labelled PWR) should go on.

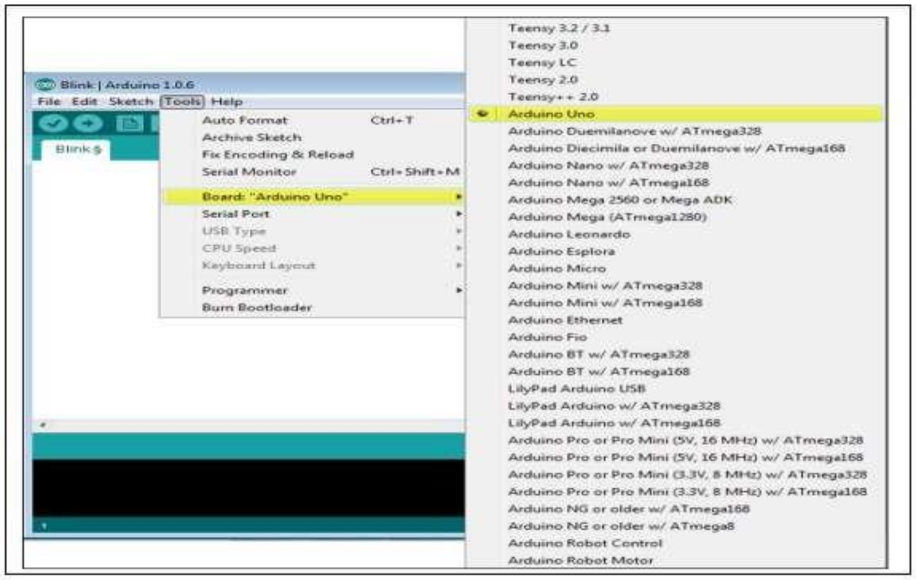
Step 5: Working on the new project

- Open the Arduino IDE software on your computer. Coding in the Arduino language will control your circuit.
- Open a new sketch File by clicking on New



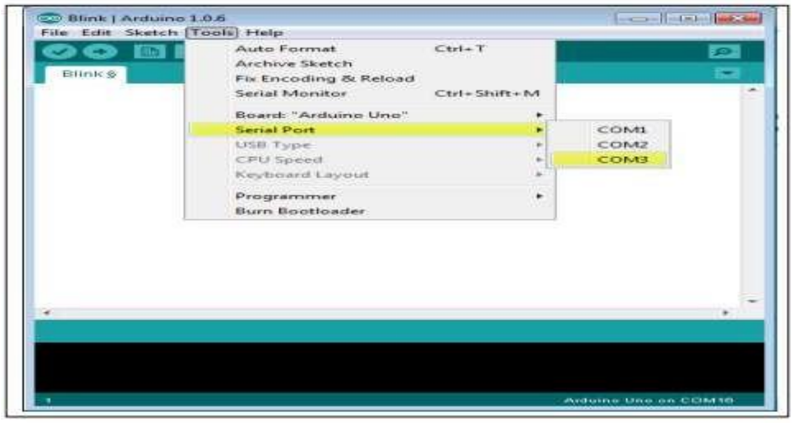
Step-6: Select your Arduino board

- 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

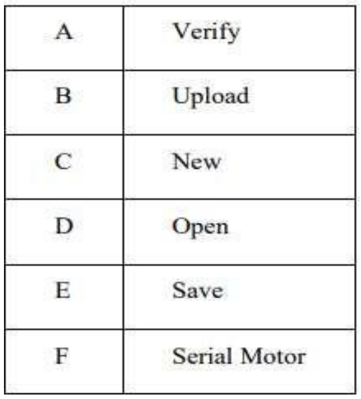
Step-7: Select your serial port

- 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

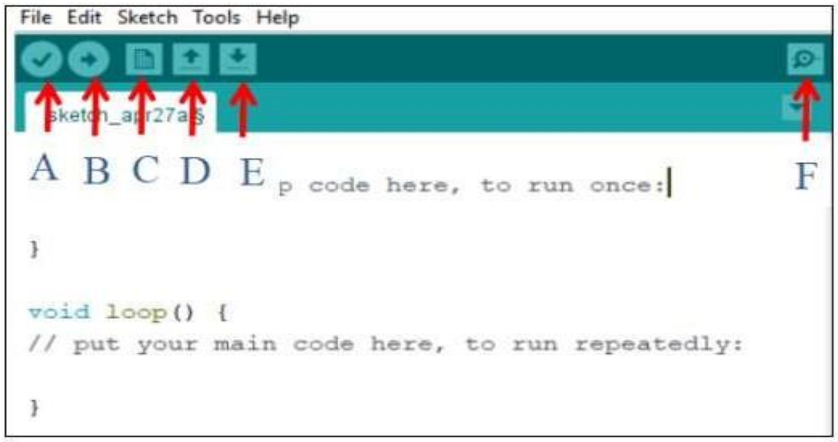


Step 8: Upload the program to your board

- 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.



A	Verify
B	Upload
C	New
D	Open
E	Save
F	Serial Motor



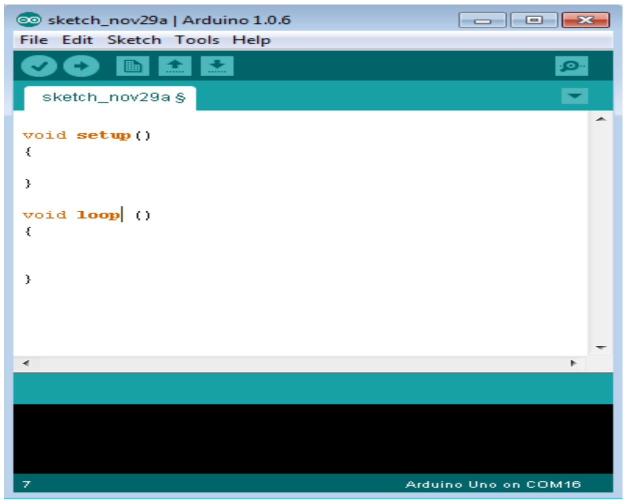
```
p code here, to run once:

}

void loop() {
// put your main code here, to run repeatedly:

}
```

2.3 Arduino - Program Structure

Procedure for writing the program in Arduino software program:

Let us start with the **Structure**. Software structure consists of two main functions −

- Setup ( ) function
- Loop ( ) function



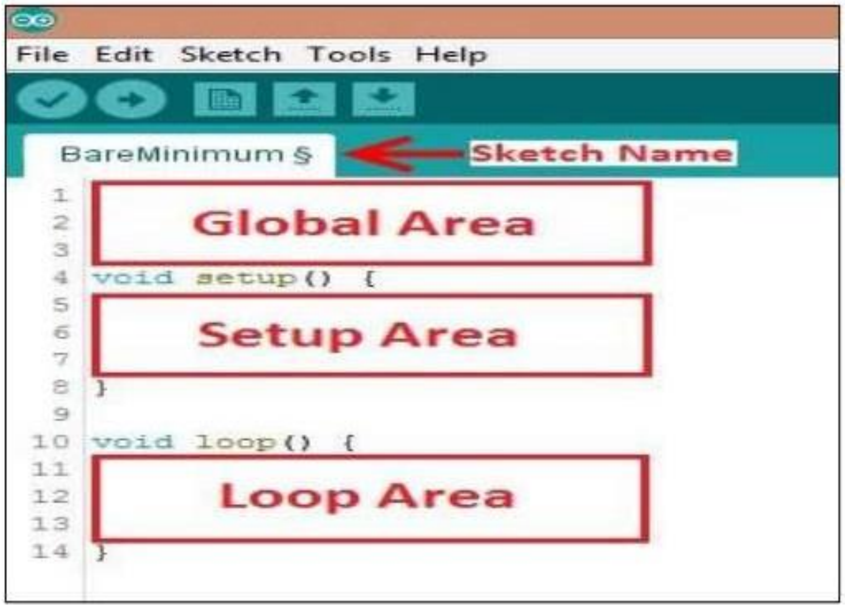
```
void setup()
{

}

void loop ()
{


}
```

- Setup( ) function

  The setup() function is called when a sketch starts. Use it to initialize the variables, pin modes, start using libraries, etc. The setup function will only run once, after each power up or reset of the Arduino board.

- Loop( ) function

  After creating a setup() function, which initializes and sets the initial values, the loop() function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.



```
void setup() {

}

void loop() {

}
```

3. Raspberry Pi

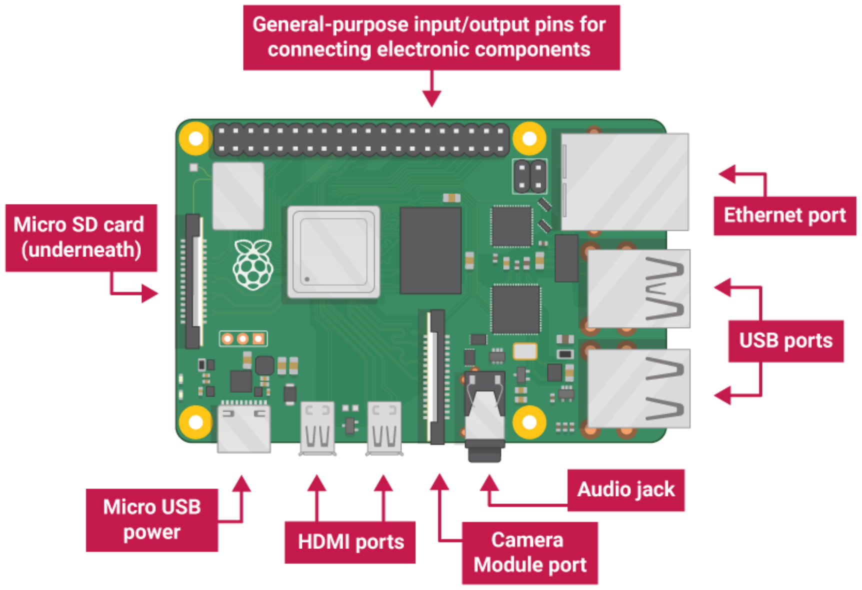
3.1 Raspberry Pi Introduction:

The Raspberry Pi is a series of small single-board computers developed by the Raspberry Pi Foundation in the UK. It was created to promote the teaching of basic computer science and programming in schools and developing countries, but it has since gained popularity among hobbyists and professionals alike for a wide range of projects.

3.2 Raspberry Pi Models:

Over the years, several models of the Raspberry Pi have been released, each offering different specifications and capabilities. Some of the popular models include:

- Raspberry Pi 1 Model A/B/A+/B+
- Raspberry Pi 2 Model B
- Raspberry Pi 3 Model B/B+
- Raspberry Pi 4 Model B
- Raspberry Pi Zero/Zero W


General-purpose input/output pins for connecting electronic components
Micro SD card (underneath)
Ethernet port
USB ports
Micro USB power
HDMI ports
Camera Module port
Audio jack

3.3 Raspberry Pi Pin Description:

The Raspberry Pi features a GPIO (General-Purpose Input/Output) header that allows it to connect to various components and peripherals. Here's a brief description of the GPIO pins on a Raspberry Pi:

A. Power Pins:

- 3.3V: Provides 3.3V power output.
- 5V: Provides 5V power output.
- GND (Ground): Ground pins.

B. GPIO Pins:

- The GPIO pins are used for digital input/output.
- GPIO pins are numbered from GPIO2 to GPIO27 on most models.
- Some pins have additional functions such as SPI, I2C, UART, etc.

C. Special Function Pins:

- I2C Pins (SDA, SCL): Used for I2C communication.
- SPI Pins (MOSI, MISO, SCLK, CE0, CE1): Used for SPI communication.
- UART Pins (TXD, RXD): Used for serial communication.
- PWM Pins: Pulse Width Modulation pins for analog-like output.

3.3 Raspberry Pi Features:

- **Processor**: The Raspberry Pi models come with different processors ranging from ARMv6 to ARMv8 architectures.
- **Memory**: The RAM size varies between models, with some models having up to 8GB of RAM.
- Storage: Most Raspberry Pi models use microSD cards for storage, but the Raspberry Pi 4 supports USB boot and has an optional USB 3.0 port for faster storage options.
- **Connectivity**: Raspberry Pi boards come with built-in Ethernet, Wi-Fi, and Bluetooth options, with the newer models offering faster and more reliable connectivity options.
- **Operating System**: The Raspberry Pi can run various operating systems, including Raspberry Pi OS (formerly Raspbian), Ubuntu, and others.
- **GPIO:** The GPIO header allows the Raspberry Pi to connect to various sensors, LEDs, motors, and other electronic components, making it suitable for a wide range of projects.
- **Expansion**: The Raspberry Pi features multiple USB ports, HDMI ports, audio jacks, and camera/display interfaces, allowing for easy expansion and connectivity with peripherals.

4. SENSORS

The sensor can be defined as a device which can be used to sense/detect the physical quantity like force, pressure, strain, light etc and then convert it into desired output like the electrical signal to measure the applied physical quantity. *In few cases, a sensor alone may not be sufficient to analyze the obtained signal. In those cases, a* ***signal conditioning unit*** *is used in order to maintain sensor's output voltage levels in the desired range with respect to the end device that we use.*

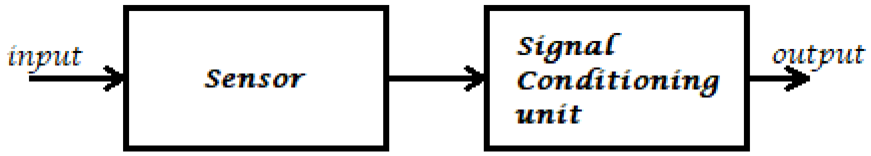


Fig. : Sensor Block

In **signal conditioning unit**, the output of the sensor may be amplified, filtered or modified to the desired output voltage. For example, if we consider a microphone it detects the audio signal and converts to the output voltage (is in terms of millivolts) which becomes hard to drive an output circuit. So, a signal conditioning unit (an amplifier) is used to increase the signal strength. But the signal conditioning may not be necessary for all the sensors like photodiode, LDR etc.

Most of the sensors can't work independently. So, sufficient input voltage should be applied to it. Various sensors have different operating ranges which should be considered while working with it else the sensor may get damaged permanently.

4.1 Types of Sensors:

Let us see the various different types of sensors that are available in the market and discuss their functionality, working, applications etc. We will discuss various sensors like:

- Light Sensor
- IR Sensor (IR Transmitter / IR LED)
- Photodiode (IR Receiver)
- Light Dependent Resistor
- Temperature Sensor
- Thermistor

- Thermocouple
- Pressure/Force/Weight Sensor
- Strain Gauge (Pressure Sensor)
- Load Cells (Weight Sensor)
- Position Sensor
- Potentiometer
- Encoder
- Hall Sensor (Detect Magnetic Field)
- Flex Sensor
- Sound Sensor
- Microphone
- Ultrasonic Sensor
- Touch Sensor
- PIR Sensor
- Tilt Sensor
- Accelerometer
- Gas Sensor

3.1.1 IR LED:

It is also called as IR Transmitter. It is used to **emit Infrared rays**. The range of these frequencies are greater than the microwave frequencies (i.e. >300GHz to few hundreds of THz). The rays generated by an infrared LED can be sensed by Photodiode explained below. The pair of **IR LED and photodiode is called IR Sensor**.

Fig.10: IR LED

3.1.2 Photo Diode (Light Sensor):

It is a semiconductor device which is ***used to detect the light rays and mostly used as IR Receiver***. Its construction is similar to the normal PN junction diode but the working principle differs from it. As we know a PN junction allows small leakage currents when it is reverse biased so, this property is used to detect the light rays. A photodiode is constructed such that light rays should fall on the PN junction which makes the leakage current increase based on the intensity of the light that we have applied. So, in this way, a photodiode can be ***used to sense the light rays*** and maintain the current through the circuit.

Using a photodiode we can build a basic automatic street lamp which glows when the sunlight intensity decreases. But the photodiode works even if a small amount of light falls on it so, care should be taken.

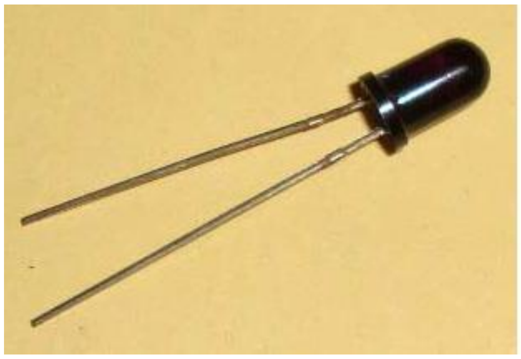


Fig.11: Photo Diode

3.1.3 LDR (Light Dependent Resistor):

As the name itself specifies that the resistor that depends upon the light intensity. It works on the principle of photoconductivity which means the conduction due to the light. It is generally made up of Cadmium sulfide. **When light falls on the LDR,** ***its resistance decreases and acts similar to a conductor and when no light falls on it, its resistance is almost in the range of MΩ or ideally it acts as an open circuit.*** One note should be considered with LDR is that it won’t respond if the light is not exactly focused on its surface.

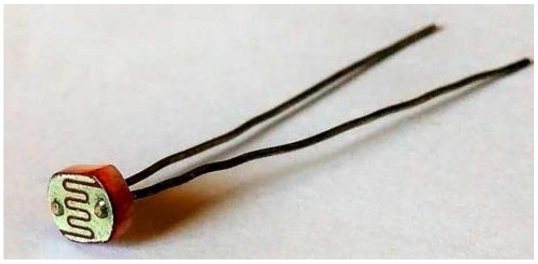


Fig.12: LDR

3.1.4 Thermistor (Temperature Sensor):

A thermistor can be used to ***detect the variation in temperature*****.** It has a negative temperature coefficient that means when the temperature increases the resistance decreases. So, the thermistor's resistance can be varied with the rise in temperature which causes more current flow through it. This change in current flow can be used to determine the amount of change in temperature. An application for thermistor is, it is used to detect the rise in temperature and control the leakage current in a transistor circuit which helps in maintaining its stability.

3.1.5 **Thermocouple (Temperature Sensor):**

Another component that can ***detect the variation in temperature*** **is a thermocouple.** In its construction, two different metals are joined together to form a junction. Its main principle is when the junction of two different metals are heated or exposed to high temperatures a potential across their terminals varies. So, the varying potential can be further used to measure the amount of change in temperature.

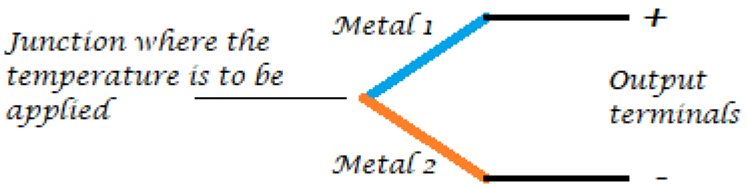


Fig.13: Thermocouple Process

3.1.6 Strain Gauge (Pressure/Force Sensor):

A strain gauge is used to ***detect pressure when a load is applied***. It works on the principle of resistance, we know that the resistance is directly proportional to the length of the wire and is inversely proportional to its cross-sectional area (R=ρl/a). The same principle can be used here to measure the load. On a flexible board, a wire is arranged in a zig-zag manner as shown in the figure below. So, when the pressure is applied to that particular board, it bends in a direction causing the change in overall length and cross-sectional area of the wire. This leads to change in resistance of the wire. The resistance thus obtained is very minute (few ohms) which can be determined with the help of the Wheatstone bridge. The strain gauge is placed in one of the four arms in a bridge with the remaining values unchanged. Therefore, when the pressure is applied to it as the resistance changes the current passing through the bridge varies and pressure can be calculated.

Strain gauges are majorly used to calculate the amount of pressure that an airplane wing can withstand and it is also used to measure the number of vehicles allowable on a particular road etc.

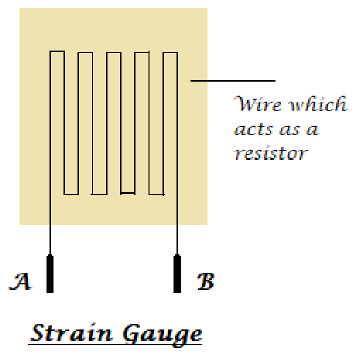


Fig.14: Strain Guage

3.1.7 Load Cell (Weight Sensor):

Load cells are similar to strain gauges which measure the physical quantity like force and give the output in form of electrical signals. When some tension is applied on the load cell it structure varies causing the change in resistance and finally, its value can be calibrated using a Wheatstone bridge.

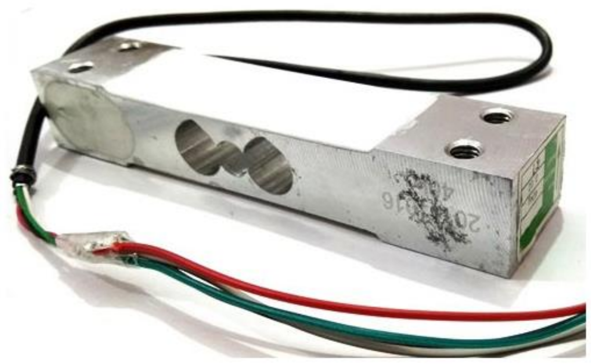


Fig.15: Load Cell

3.1.8 Potentiometer:

A potentiometer is ***used to detect the position***. It generally has various ranges of resistors connected to different poles of the switch. A potentiometer can be either rotary or linear type. In rotary type, a wiper is connected to a long shaft which can be rotated. When the shaft has rotated the position of the wiper alters such that the resultant resistance varies causing the change in the output voltage. Thus the output can be calibrated to detect the change its position.

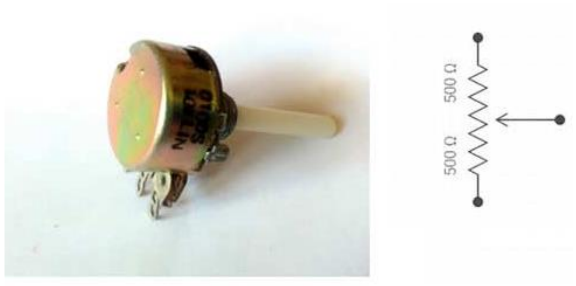


Fig.16: Potentiometer

3.1.9 Encoder:

To detect the change in the position an encoder can also be used. It has a circular rotatable disk-like structure with specific openings in between such that when the IR rays or light rays pass through it only a few light rays get detected. Further, these rays are encoded into a digital data (in terms of binary) which represents the specific position.

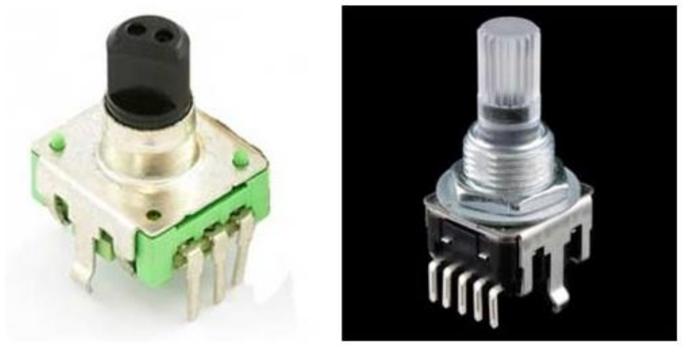


Fig.17: Encoder

3.1.10 Hall Sensor:

The name itself states that it is the sensor which works on the Hall Effect. It can be defined as when a magnetic field is brought close to the current carrying conductor (perpendicular to the direction of the electric field) then a potential difference is developed across the given conductor. Using this property a ***Hall sensor is used to detect the magnetic field*** and gives output in terms of voltage. Care should be taken that the Hall sensor can detect only one pole of the magnet.

The hall sensor is used in few smartphones which are helpful in turning off the screen when the flap cover (which has a magnet in it) is closed onto the screen.

3.1.11 Flex Sensor:

A FLEX sensor is a transducer which ***changes its resistance when its shape is changed or when it is bent***. A FLEX sensor is 2.2 inches long or of finger length. It is shown in the figure. Simply speaking the sensor terminal resistance increases when it’s bent. This change in resistance can do no good unless we can read them. The controller at hand can only read the changes in voltage and nothing less, for this, we are going to use voltage divider circuit, with that we can derive the resistance change as a voltage change.

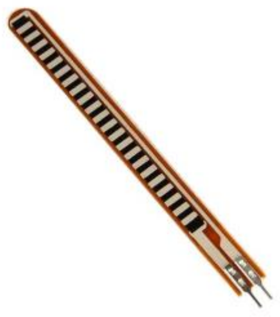


Fig.18: Flex Sensor

3.1.12 Microphone (Sound Sensor):

Microphone can be seen on all the smartphones or mobiles. It can detect the audio signal and convert them into small voltage (mV) electrical signals. A microphone can be of many types like condenser microphone, crystal microphone, carbon microphone etc. each type of microphone work on the properties like capacitance, piezoelectric effect, resistance respectively. Let us see the operation of a crystal microphone which works on the piezoelectric effect. A bimorph crystal is used which under pressure or vibrations produces proportional alternating voltage. A diaphragm is connected to the crystal through a drive pin such that when the sound signal hits the diaphragm it moves to and fro, this movement changes the position of the drive pin which causes vibrations in the crystal thus an alternating voltage is generated with respect to the applied sound signal. The obtained voltage is fed to an amplifier in order to increase the overall strength of the signal.

Fig.19: Microphone

3.1.13 Ultrasonic sensor:

Ultrasonic means nothing but the range of the frequencies. Its range is greater than audible range (>20 kHz) so even it is switched on we can’t sense these sound signals. Only specific speakers and receivers can sense those ultrasonic waves. This ultrasonic sensor is ***used to calculate the distance between the ultrasonic transmitter and the target and also used to measure the velocity of the target*.**

Ultrasonic sensor HC-SR04 can be used to measure distance in the range of 2cm-400cm with an accuracy of 3mm. Let's see how this module works. The HCSR04 module generates a sound vibration in ultrasonic range when we make the 'Trigger' pin high for about 10us which will send an 8 cycle sonic burst at the speed of sound and after striking the object, it will be received by the Echo pin. Depending on the time taken by sound vibration to get back, it provides the appropriate pulse output. We can calculate the distance of the object based on the time taken by the ultrasonic wave to return back to the sensor

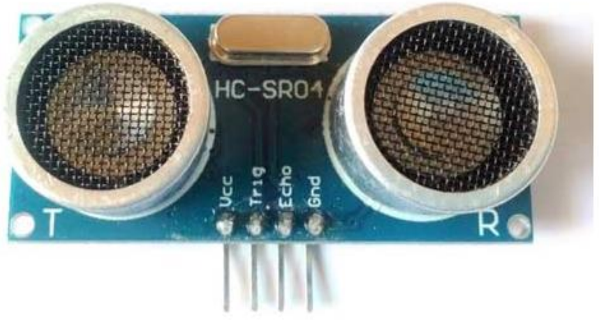


Fig.20: Ultrasonic Sensor

3.1.14 Touch Sensor:

There are two types of touch sensors ***resistive based and a capacitive based touch screens***. Let's know about working of these sensors briefly.

The ***resistive touchscreen*** has a resistive sheet at the base and a conductive sheet under the screen both of these are separated by an air gap with a small voltage applied to the sheets. When we press or touch the screen the conductive sheet touches the resistive sheet at that point causing current flow at that particular point, the software senses the location and relevant action is performed.

Whereas ***capacitive touch*** works on the electrostatic charge that is available on our body. The screen is already charged with s the all electric field. When we touch the screen a close circuit forms due to electrostatic charge that flow through our body. Further, software decides the location and the action to be performed. We can observe that capacitive touch screen won't work when wear hand gloves because there won't be conduction between the finger(s) and the screen.

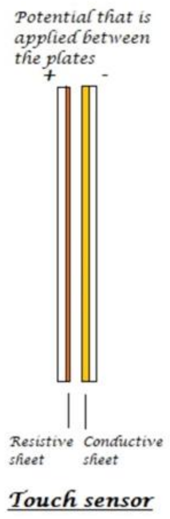


Fig.21: Touch Sensor

3.1.15 PIR sensor:

PIR sensor stands for **Passive Infrared sensor**. These are ***used to detect the motion*** of humans, animals or things. We know that infrared rays have a property of reflection. When an infrared ray hits an object, depending upon the temperature of the target the infrared ray properties changes, this received signal determines the motion of the objects or the living beings. Even if the shape of the object alters, the properties of the reflected infrared rays can differentiate the objects precisely.

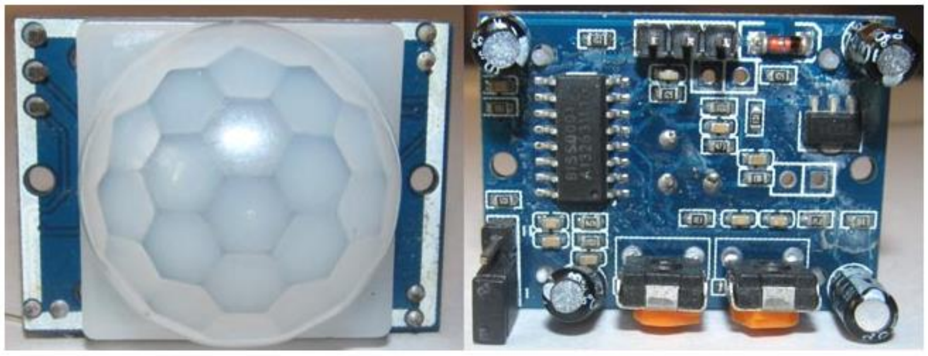


Fig.22: PIR Sensor

3.1.16 Accelerometer (Tilt Sensor):

An accelerometer sensor *can sense the tilt or movement of it in a particular direction***.** It works based on the acceleration force caused due to the earth’s gravity. The tiny internal parts of it are such sensitive that those will react to a small external change in position. It has a piezoelectric crystal when tilted causes disturbance in the crystal and generates potential which determines the exact position with respect to X, Y and Z axis.

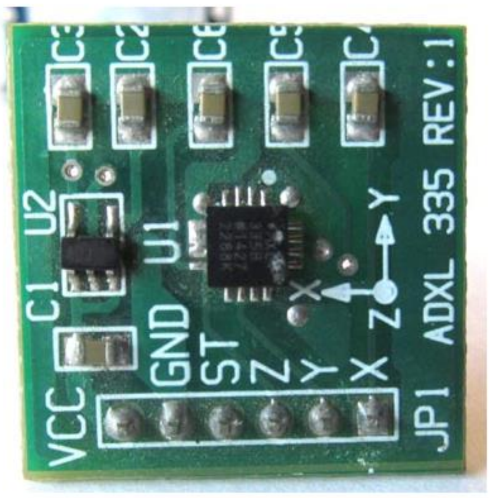


Fig.23: Accelerometer

3.1.17 Gas Sensor:

In industrial applications gas sensors plays a major role in **detecting the gas leakage**. If no such device is installed in such areas it ultimately leads to an unbelievable disaster. These gas sensors are classified into various types based on the type of gas that to be detected. Let’s see how this sensor works. Underneath a metal sheet there exists a sensing element which is connected to the terminals where a current is applied to it. When the gas particles hit the sensing element, it leads to a chemical reaction such that the resistance of the elements varies and current through it also alters which finally can detect the gas.

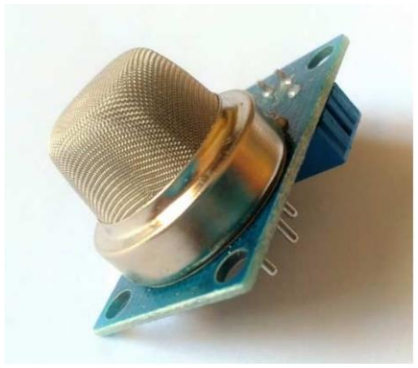


Fig.24: Gas Sensor

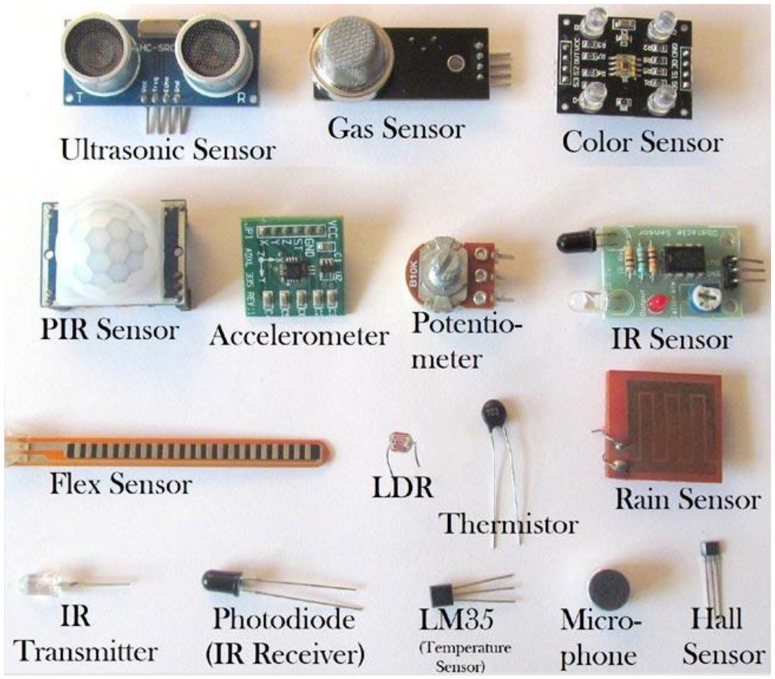


Fig.25: Basic Sensors

Experiment No. 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.

i) To interface LED/Buzzer with Arduino/Raspberry Pi and write a program to ‘turn ON’ LED for 1 sec after every 2 seconds.

Objectives:

- To implement the basic principles of interfacing an LED and buzzer with Arduino/Raspberry Pi and write a C program to turn ON the LED for 1 second after every 2 seconds.
- To implement the concept of interfacing a push button and digital sensor (IR/LDR) with Arduino/Raspberry Pi and write a C program to turn ON the LED when the push button is pressed or sensor detection occurs.

Components Required:

- Arduino UNO
- LED/ Buzzer.
- Resistor (220Ω)
- Connecting cable or USB cable.
- Breadboard.
- Jumper wires

Circuit Diagram:

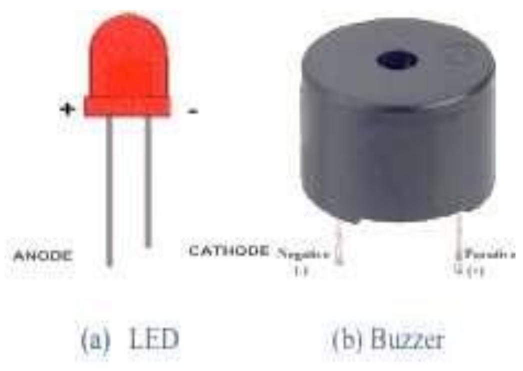


Figure-1.1 a): LED and Buzzer

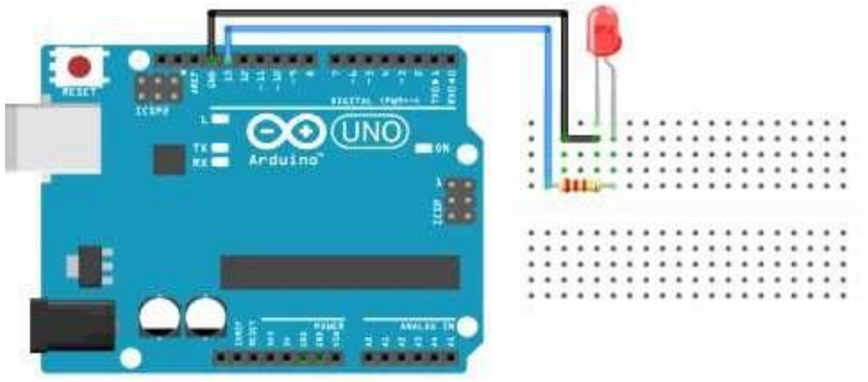


Figure-1.1 b): LED interfacing with Arduino

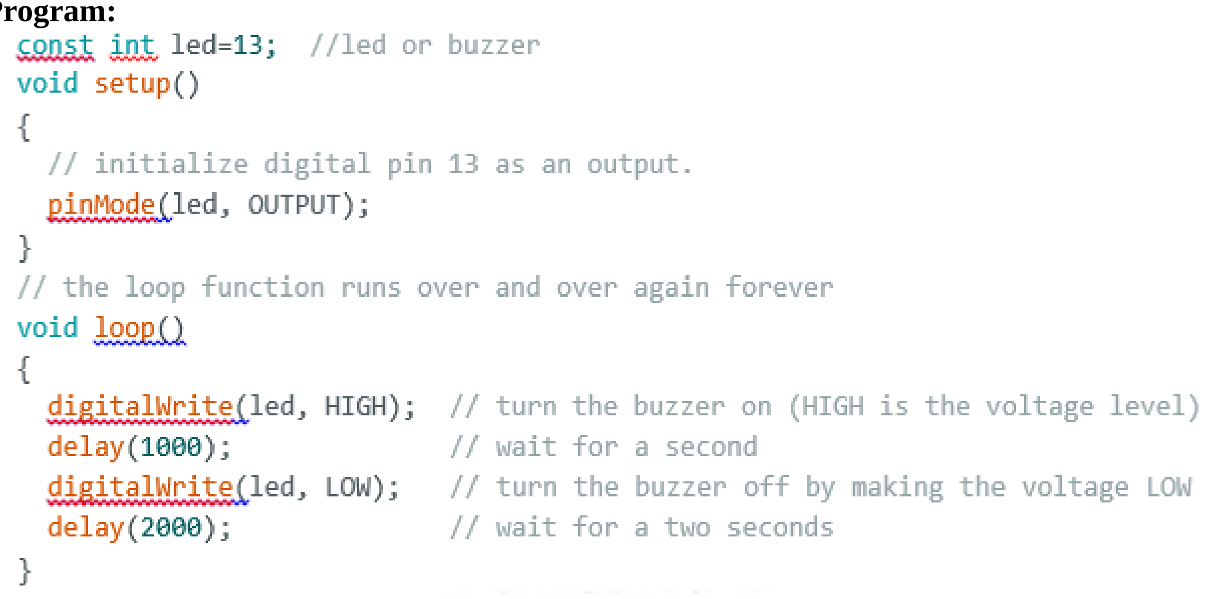
Pin Connections:

- Arduino Digital Pin (e.g., pin 13) -> One end of 220Ω Resistor -> Anode (+, longer leg) of LED/Positive (+) of Buzzer
- Cathode (-, shorter leg) of LED/Negative (-) of Buzzer -> GND Pin (Arduino)

Procedure:

- **LED/Buzzer: For LED:**
  Anode (+, longer leg): Connect to one end of the 220Ω resistor.
  Cathode (-, shorter leg): Connect to Arduino digital pin (e.g., pin 13).
  For Buzzer:
  Positive (+): Connect to one end of the 220Ω resistor.
  Negative (-): Connect to Arduino digital pin (e.g., pin 13).
- **Resistor (220Ω):**
  Connect one end of the resistor to the anode of the LED or positive terminal of the buzzer.
  Connect the other end of the resistor to the digital pin of the Arduino (e.g., pin 13).
- **Connecting Cable or USB Cable:**
  Connect one end to the Arduino UNO.
  Connect the other end to your computer for power and programming.
- **Breadboard:**
  Place the LED/Buzzer and resistor on the breadboard to ensure stability.
  Connect the jumper wires to the appropriate pins on the LED/Buzzer, resistor, and Arduino.
- **Jumper Wires:**
  Connect one end of the jumper wire from the digital pin (e.g., pin 13) of the Arduino to the other end of the resistor and anode of the LED or positive terminal of the buzzer.
  Connect another jumper wire from the cathode of the LED or negative terminal of the buzzer to the GND pin of the Arduino.

Program:



```
const int led=13;  //led or buzzer
void setup()
{
  // initialize digital pin 13 as an output.
  pinMode(led, OUTPUT);
}
// the loop function runs over and over again forever
void loop()
{
  digitalWrite(led, HIGH);  // turn the buzzer on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(led, LOW);   // turn the buzzer off by making the voltage LOW
  delay(2000);              // wait for a two seconds
}
```

Outcome:

1. Understand the basic principles of interfacing LEDs and buzzers with Arduino UNO
2. Write and Evaluate C program to control the LED and buzzer operations using Arduino UNO

Result:

LED Status	Duration in Seconds
ON	
OFF	

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.

Objective:

- To understand and implement the concept of interfacing a push button and digital sensor (IR/LDR) with Arduino/Raspberry Pi
- Write a C program to turn ON the LED when the push button is pressed or sensor detection occurs.

CASE -1: Using Push Button

Components Required:

- Arduino UNO.
- Push button.
- LED.
- Resistor (10KΩ)
- Connecting cable or USB cable.
- Breadboard.
- Jumper wires.

Circuit Diagram:

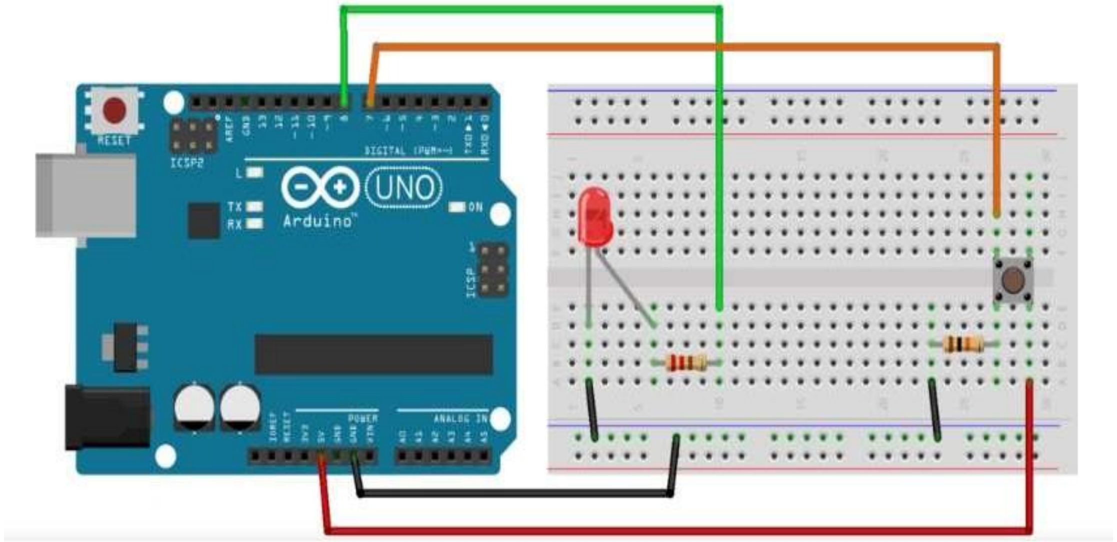


Figure-1.2: Push button interfacing with Arduino

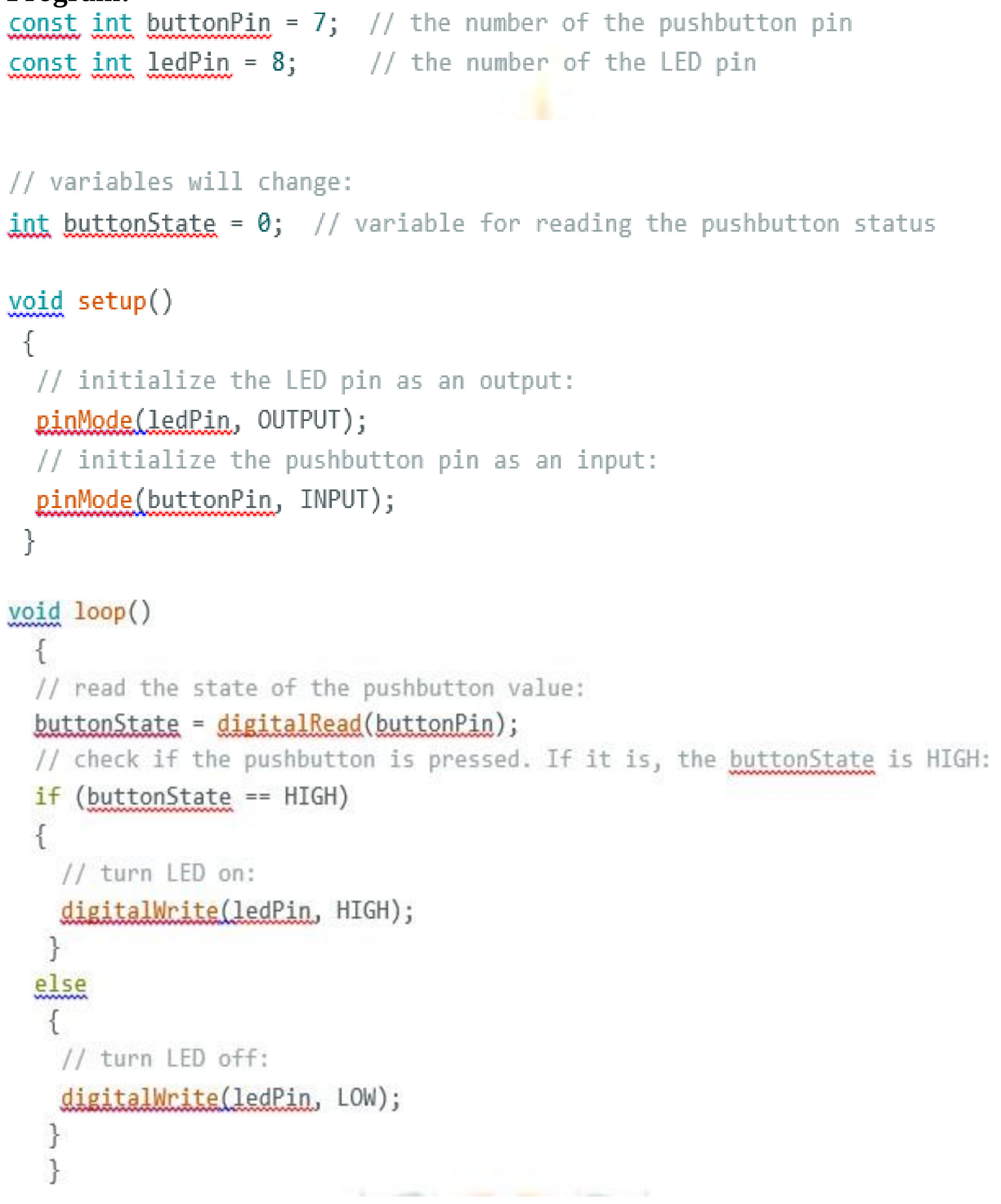
Pin Connections:

- Arduino Digital Pin (e.g., pin 7) -> One pin of Push Button
- GND Pin -> Other pin of Push Button
- 5V or VCC -> One leg of 10KΩ Resistor -> Other leg of 10KΩ Resistor -> Digital Pin of Push Button
- Arduino Digital Pin (e.g., pin 13) -> One end of 220Ω Resistor -> Anode (+, longer leg) of LED
- Cathode (-, shorter leg) of LED -> GND Pin (Arduino)

Procedure:

- **Push Button:**
  Connect one pin of the push button to Arduino digital pin (e.g., pin 7).
  Connect the other pin of the push button to GND.
  If needed, connect a 10KΩ pull-down resistor between the digital pin 7 and GND.
- **LED:**
  Anode (+, longer leg): Connect to one end of the 220Ω resistor.
  Cathode (-, shorter leg): Connect to Arduino digital pin (e.g., pin 8).
- **Resistor (10KΩ):**
  If your push button does not have a built-in resistor, connect an external 10KΩ pull-up resistor.
  Connect one leg of the 10KΩ resistor to the digital pin of the push button that is connected to the Arduino.
  Connect the other leg of the 10KΩ resistor to 5V or VCC.
- **Connecting Cable or USB Cable:**
  Connect one end to the Arduino UNO.
  Connect the other end to your computer for power and programming.
- **Breadboard:**
  Place the push button, LED, and resistor on the breadboard to ensure stability.
  Connect the jumper wires to the appropriate pins on the push button, LED, resistor, and Arduino.
- **Jumper Wires:**
  Connect one end of the jumper wire from the digital pin (e.g., pin 7) of the Arduino to one pin of the push button.
  Connect the other end of the jumper wire from the GND pin of the Arduino to the other pin of the push button.
  Connect another jumper wire from the digital pin (e.g., pin 8) of the Arduino to the anode of the LED or positive terminal of the buzzer.
  Connect another jumper wire from the cathode of the LED or negative terminal of the buzzer to GND.

Program:



```
const int buttonPin = 7;  // the number of the pushbutton pin
const int ledPin = 8;     // the number of the LED pin


// variables will change:
int buttonState = 0;  // variable for reading the pushbutton status

void setup()
 {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
 }

void loop()
  {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);
  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (buttonState == HIGH)
  {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
   }
  else
   {
    // turn LED off:
    digitalWrite(ledPin, LOW);
   }
   }
```

CASE -2: Using Infrared Sensor

Components Required:

- Arduino UNO.
- Infrared Sensor(IR)
- LED.
- Resistor (220Ω)
- Connecting cable or USB cable.
- Breadboard.
- Jumper wires.

Circuit Diagram:

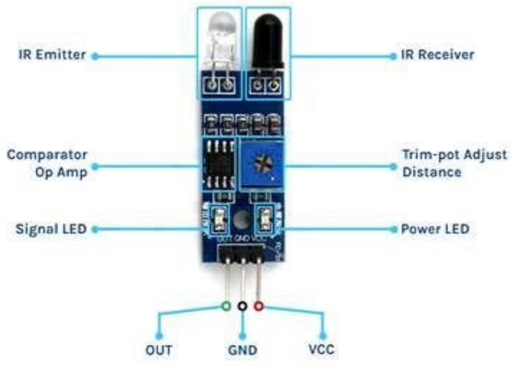


Figure 1.3 a) : IR Sensor

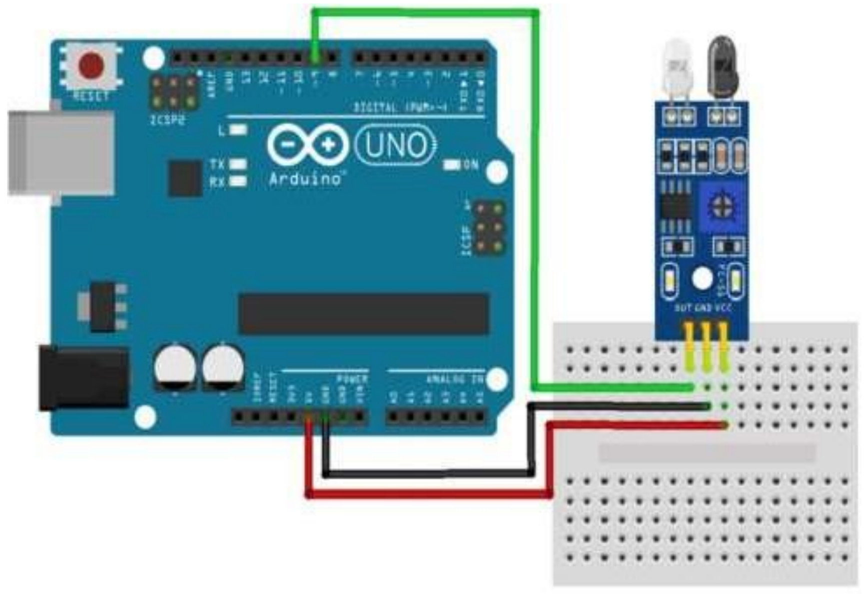


Figure 1.3 b) : Interfacing of Digital Sensor (IR) with Arduino

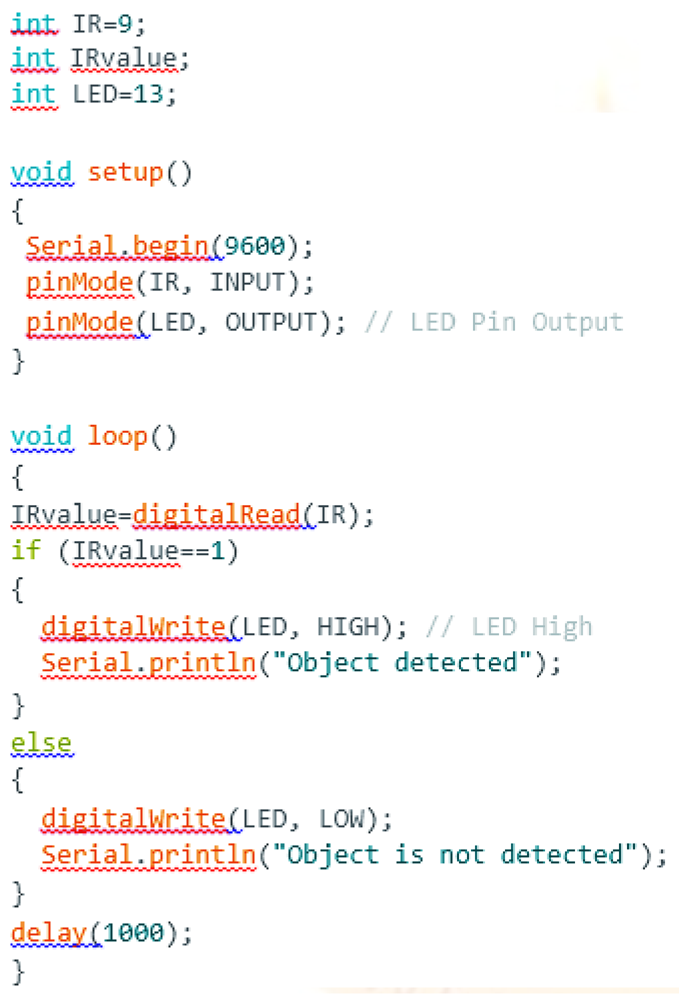
Pin Connections:

- Arduino 5V -> IR Sensor VCC
- Arduino GND -> IR Sensor GND
- Arduino Digital Pin (e.g., pin 9) -> IR Sensor OUT
- Arduino Digital Pin (e.g., pin 13) -> One end of 220Ω Resistor -> Anode (+, longer leg) of LED
- Cathode (-, shorter leg) of LED -> GND Pin (Arduino)

Procedure:

- **Infrared (IR) Sensor:**
  VCC: Connect to Arduino 5V pin.
  GND: Connect to Arduino GND pin.
  OUT: Connect to Arduino digital pin (e.g., pin 9).
- **LED:**
  Anode (+, longer leg): Connect to one end of the 220Ω resistor.
  Cathode (-, shorter leg): Connect to Arduino digital pin (e.g., pin 13).
- **Resistor (220Ω):**
  Connect one end of the resistor to the anode of the LED.
  Connect the other end of the resistor to the digital pin of the Arduino (e.g., pin 13).
- **Connecting Cable or USB Cable:**
  Connect one end to the Arduino UNO.
  Connect the other end to your computer for power and programming.
- **Breadboard:**
  Place the IR sensor, LED, and resistor on the breadboard to ensure stability.
  Connect the jumper wires to the appropriate pins on the IR sensor, LED, resistor, and Arduino.
- **Jumper Wires**:
  Connect one end of the jumper wire from the digital pin (e.g., pin 9) of the Arduino to the OUT pin of the IR sensor.
  Connect another jumper wire from the GND pin of the Arduino to the GND pin of the IR sensor.
  Connect another jumper wire from the digital pin (e.g., pin 13) of the Arduino to the anode of the LED.
  Connect another jumper wire from the cathode of the LED to GND.

Program:



```
int IR=9;
int IRvalue;
int LED=13;

void setup()
{
 Serial.begin(9600);
 pinMode(IR, INPUT);
 pinMode(LED, OUTPUT); // LED Pin Output
}

void loop()
{
IRvalue=digitalRead(IR);
if (IRvalue==1)
{
  digitalWrite(LED, HIGH); // LED High
  Serial.println("Object detected");
}
else
{
  digitalWrite(LED, LOW);
  Serial.println("Object is not detected");
}
delay(1000);
}
```

Outcome:

- Understand the concept of interfacing push buttons and digital sensors (IR/LDR) with Arduino
- Write and evaluate a C program to detect button press or sensor detection and control LED operations using Arduino

Result:

Pushbutton	LED Status (ON/OFF)
ON	
OFF	

IR Sensor Detected	LED Status (ON/OFF)
YES	
NO	

Experiment 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.

i) To interface DHT11 sensor with Arduino/Raspberry Pi and write a program to print temperature and humidity readings.

Objectives:

- To understand and implement the basic principles of interfacing a DHT11 sensor with Arduino/Raspberry Pi.
- To write a program to read temperature and humidity data from the DHT11 sensor and print the readings.

Components Required:

- Arduino UNO
- DHT11.
- Connecting cable or USB cable.
- Jumper wires

Circuit Diagram:

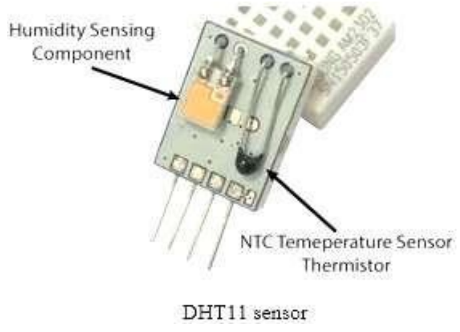

Humidity Sensing Component
NTC Temeperature Sensor Thermistor
DHT11 sensor

Figure 2.1 a) : DHT11 Sensor

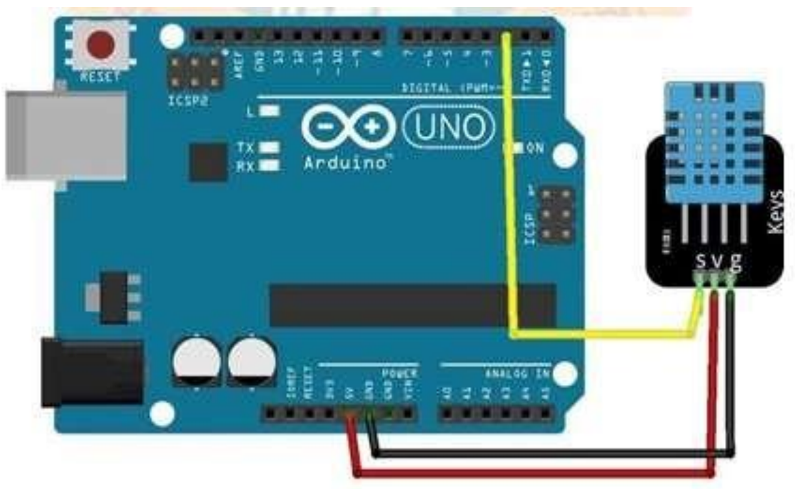


Figure 2.1 b) : DHT11 interfacing with Arduino

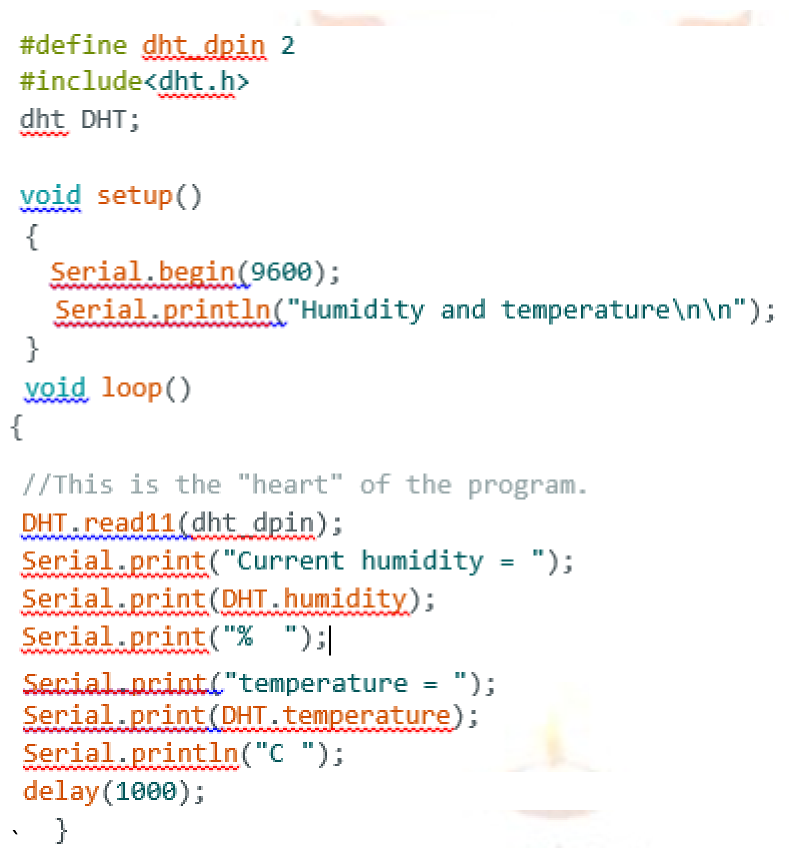
Pin Connections:

- Arduino 5V -> DHT11 VCC
- Arduino GND -> DHT11 GND
- Arduino Digital Pin (e.g., pin 2) -> DHT11 DATA

Procedure:

- Install the library for DHT in Arduino IDE.
  Open Arduino IDE and navigate to **Sketch > Include Library > Manage Libraries**.
- Search for “**DHTlib**” and install the “**DHTlib**” library in the Arduino IDE.
- **DHT11 Sensor:**
  VCC: Connect to Arduino 5V pin.
  GND: Connect to Arduino GND pin.
  DATA: Connect to Arduino digital pin (e.g., pin 2).
- **Connecting Cable or USB Cable:**
  Connect one end to the Arduino UNO.
  Connect the other end to your computer for power and programming.
- **Jumper Wires:**
- Connect the jumper wires from the Arduino to the DHT11 sensor based on the connections mentioned above.

Program:

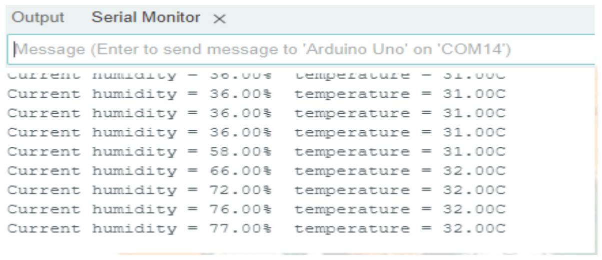


```
#define dht_dpin 2
#include<dht.h>
dht DHT;

void setup()
 {
   Serial.begin(9600);
   Serial.println("Humidity and temperature\n\n");
 }
 void loop()
{

 //This is the "heart" of the program.
 DHT.read11(dht_dpin);
 Serial.print("Current humidity = ");
 Serial.print(DHT.humidity);
 Serial.print("%  ");

 Serial.print("temperature = ");
 Serial.print(DHT.temperature);
 Serial.println("C ");
 delay(1000);
`   }
```

Outcome:

- Understand the basic principles of interfacing sensors like DHT11 with Arduino/Raspberry Pi.
- Write and evaluate a C program to read and display temperature and humidity readings using Arduino/Raspberry Pi.

Result:



```
Current humidity = 36.00%  temperature = 31.00C
Current humidity = 36.00%  temperature = 31.00C
Current humidity = 36.00%  temperature = 31.00C
Current humidity = 36.00%  temperature = 31.00C
Current humidity = 58.00%  temperature = 31.00C
Current humidity = 66.00%  temperature = 32.00C
Current humidity = 72.00%  temperature = 32.00C
Current humidity = 76.00%  temperature = 32.00C
Current humidity = 77.00%  temperature = 32.00C
```

Figure 2.2: Output Serial Monitor Window

ii) To interface OLED with Arduino/Raspberry Pi and write a program to print temperature and humidity readings on it.

Objectives:

- To understand and implement the concept of interfacing an OLED display with Arduino/Raspberry Pi.
- To write a program to display temperature and humidity readings from the DHT11 sensor on the OLED display.

Components Required:

- Arduino UNO/ Raspberry Pi.
- DHT11.
- OLED Display Module.
- Connecting cable or USB cable.
- Jumper wires

Circuit Diagram:

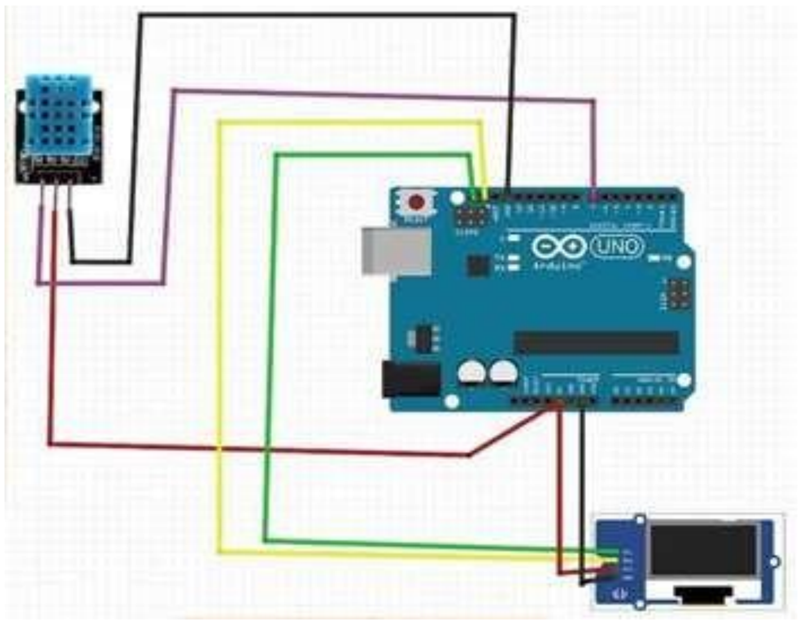


Figure 2.3: OLED interfacing with Arduino

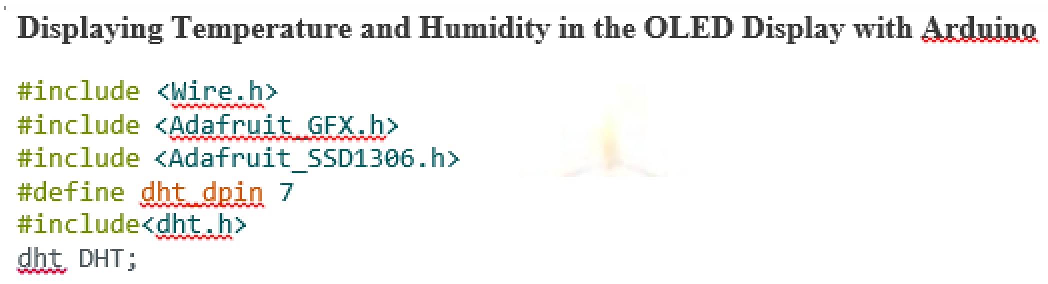
Pin Connections:

- Arduino 5V -> DHT11 VCC & OLED VCC
- Arduino GND -> DHT11 GND & OLED GND
- Arduino Digital Pin 7 -> DHT11 DATA
- Arduino A4 -> OLED SDA
- Arduino A5 -> OLED SCL

Procedure:

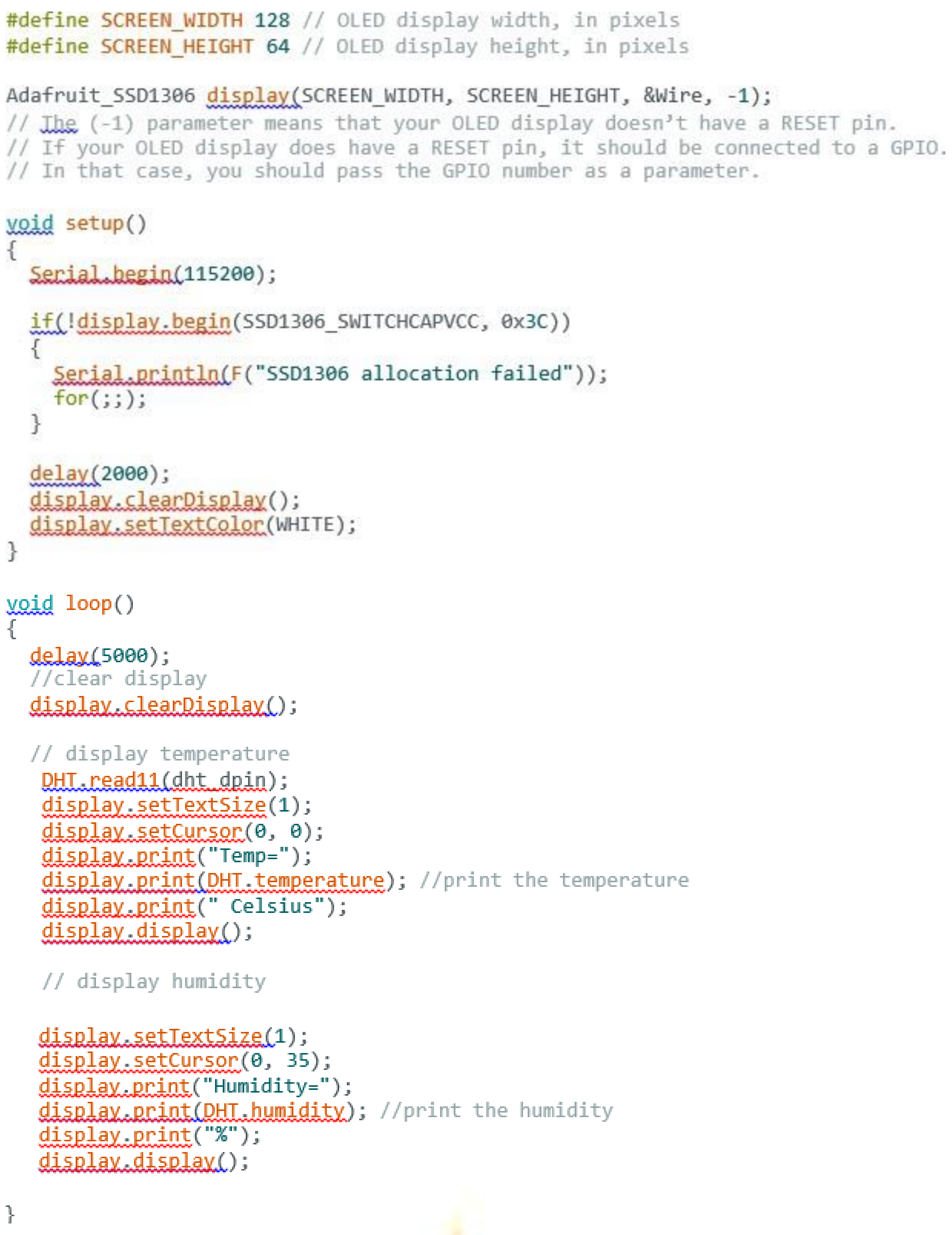
- **DHT11 Sensor:**
  VCC: **Connect to Arduino 5V pin.**
  GND: **Connect to Arduino GND pin.**
  DATA: **Connect to Arduino digital pin 7.**
- **OLED Display Module:**
  VCC: **Connect to Arduino 5V pin.**
  GND: **Connect to Arduino GND pin.**
  SDA: **Connect to Arduino A4 pin.**
  SCL: **Connect to Arduino A5 pin.**
- **Connecting Cable or USB Cable:**
  Connect one end to the Arduino UNO.
  Connect the other end to your computer for power and programming.
- **Jumper Wires:**
  Connect the jumper wires from the Arduino to the DHT11 sensor and OLED display module based on the connections mentioned above.

Program:



Displaying Temperature and Humidity in the OLED Display with Arduino

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#define dht_dpin 7
#include<dht.h>
dht DHT;
```



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

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
// The (-1) parameter means that your OLED display doesn’t have a RESET pin.
// If your OLED display does have a RESET pin, it should be connected to a GPIO.
// In that case, you should pass the GPIO number as a parameter.

void setup()
{
  Serial.begin(115200);

  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C))
  {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }

  delay(2000);
  display.clearDisplay();
  display.setTextColor(WHITE);
}

void loop()
{
  delay(5000);
  //clear display
  display.clearDisplay();

  // display temperature
   DHT.read11(dht_dpin);
   display.setTextSize(1);
   display.setCursor(0, 0);
   display.print("Temp=");
   display.print(DHT.temperature); //print the temperature
   display.print(" Celsius");
   display.display();

   // display humidity

   display.setTextSize(1);
   display.setCursor(0, 35);
   display.print("Humidity=");
   display.print(DHT.humidity); //print the humidity
   display.print("%");
   display.display();

}
```

Outcome:

- Understand the concept of interfacing displays like OLED with Arduino/Raspberry Pi.
- Write and evaluate c program to display temperature and humidity readings on an OLED display using Arduino/Raspberry Pi.

Result

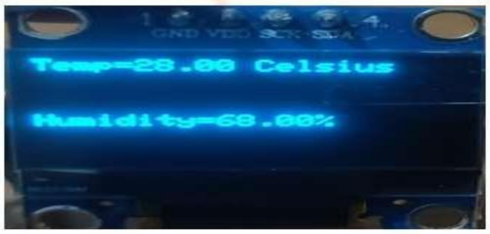


Figure 2.4: Displaying Temperature and Humidity values on OLED using Arduino

Experiment 3

To interface motor using relay with Arduino/Raspberry Pi and write a program to 'turn ON' motor when push button is pressed

Objectives:

- To understand and implement the concept of interfacing a motor using a relay with Arduino/Raspberry Pi.
- To write a C program that controls the motor to turn ON when a push button is pressed.

Components Required:

- Arduino UNO.
- Relay
- Resistor (10kΩ)
- Motor.
- Connecting cable or USB cable.
- Breadboard
- Jumper wires

Circuit Diagram:

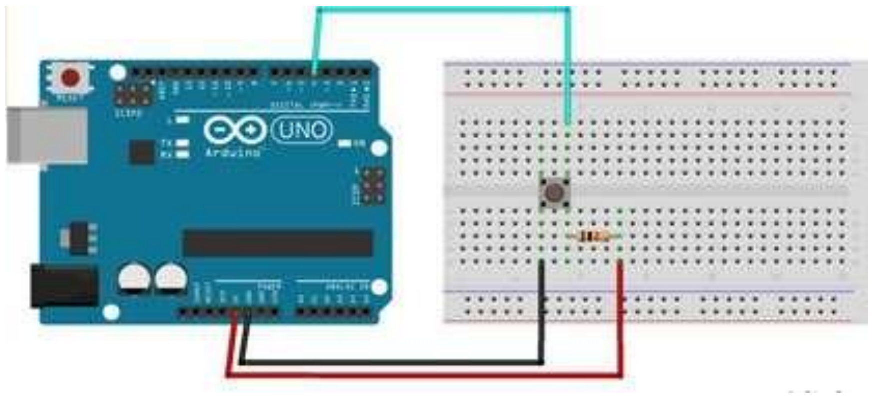


Figure 3.1: Push button interfacing with Arduino

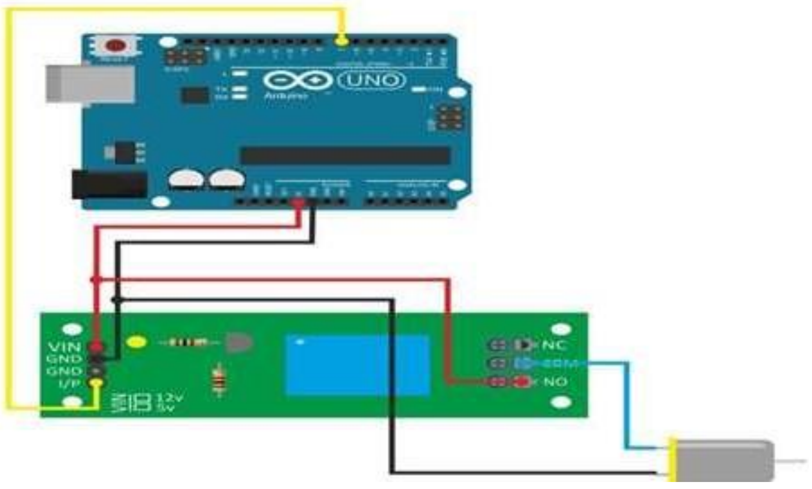


Figure 3.2: Interfacing Motor with relay using Arduino

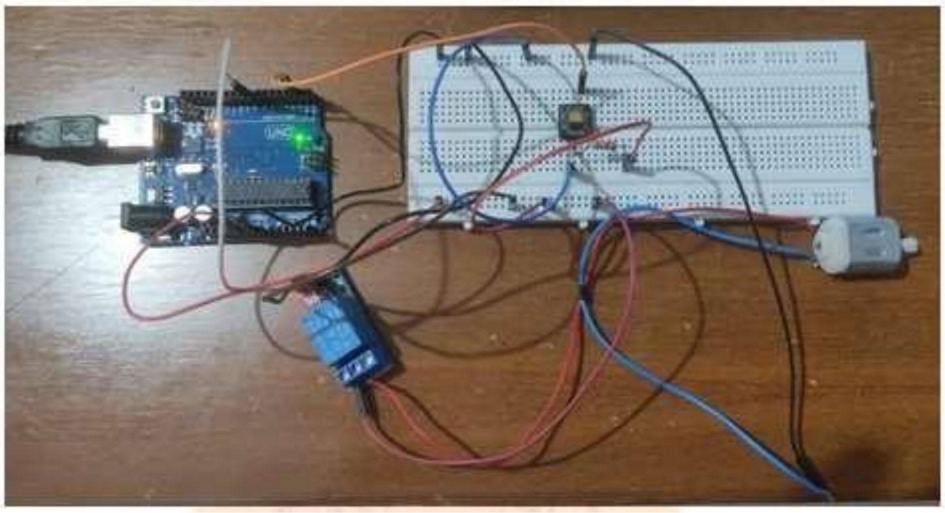


Figure 3.3: Interfacing Motor with relay and pushbutton using Arduino

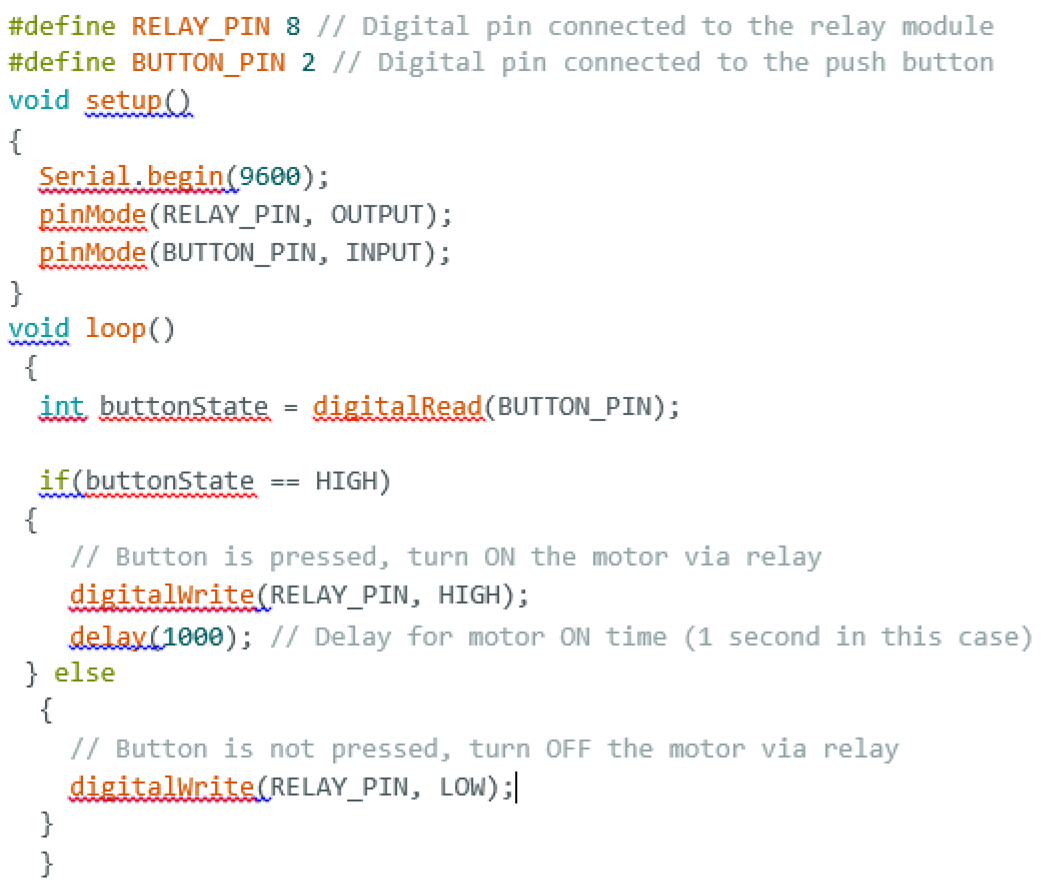
Pin Connections:

- Arduino 5V -> Relay VCC
- Arduino GND -> Relay GND
- Arduino Digital Pin 8 -> Relay IN1
- 5V Pin (Arduino) -> One leg of 10kΩ Resistor -> Other leg of 10kΩ Resistor -> Relay IN1
- Motor Positive (+) -> Relay COM
- Motor Negative (-) -> Relay NO
- Digital Pin 2 (with optional 10kΩ Resistor) -> Push Button -> GND

Procedure:

- **Relay:**
  Connect VCC of the Relay to Arduino 5V pin.
  Connect GND of the Relay to Arduino GND pin.
  Connect IN1 of the Relay to Arduino digital pin 8.
- **Resistor (10kΩ):**
  Connect one leg of the 10kΩ resistor to 5V Pin (Arduino).
  Connect the other leg of the 10kΩ resistor to Relay IN1.
- **Motor:**
  Connect Motor Positive (+) to Relay COM.
  Connect Motor Negative (-) to Relay NO.
- **Push Button:**
  Connect one pin of the push button to Arduino digital pin 2.
  Connect the other pin of the push button to GND.
  If needed, connect a 10kΩ pull-down resistor between digital pin 2 and GND.
- **Connecting Cable or USB Cable:**
  Connect one end to the Arduino UNO.
  Connect the other end to your computer for power and programming.
- **Breadboard:**
  Place the relay, resistor, motor, push button, and jumper wires on the breadboard to ensure stability.
  Connect the jumper wires to the appropriate pins on the relay, resistor, motor, push button, and Arduino.

Program:



```
#define RELAY_PIN 8 // Digital pin connected to the relay module
#define BUTTON_PIN 2 // Digital pin connected to the push button
void setup()
{
  Serial.begin(9600);
  pinMode(RELAY_PIN, OUTPUT);
  pinMode(BUTTON_PIN, INPUT);
}
void loop()
 {
  int buttonState = digitalRead(BUTTON_PIN);

  if(buttonState == HIGH)
 {
    // Button is pressed, turn ON the motor via relay
    digitalWrite(RELAY_PIN, HIGH);
    delay(1000); // Delay for motor ON time (1 second in this case)
 } else
  {
    // Button is not pressed, turn OFF the motor via relay
    digitalWrite(RELAY_PIN, LOW);
  }
  }
```

Outcome:

- Understand the basic principles of interfacing motors with Arduino/Raspberry Pi using a relay.
- Write and evaluate a C program to control the motor's operation based on the input from a push button using Arduino

Result:

Pushbutton	**Relay**	**Motor Status (ON/OFF)**
ON	ON	
OFF	OFF	

Experiment 4

To interface Bluetooth with Arduino/Raspberry Pi and write a program to send sensor data to smartphone using Bluetooth

Objective:

- To establish a Bluetooth communication link between Arduino/Raspberry Pi and a smartphone.
- To send sensor data (e.g., temperature and humidity from DHT11 sensor) from Arduino/Raspberry Pi to the smartphone via Bluetooth.

Components Required:

- Arduino UNO
- HC-05 or HC-06 Bluetooth module
- Connecting cable or USB cable
- Breadboard
- Jumper wires
- Sensor (e.g., DHT11)

Circuit Diagram:

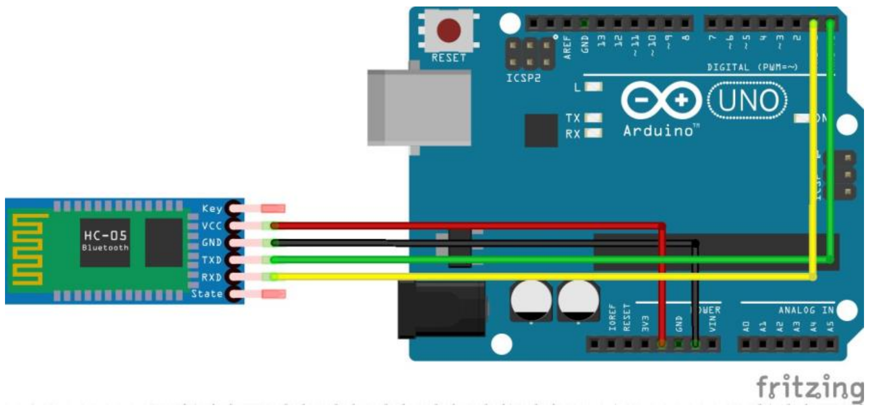


Figure 4.1: Interfacing Bluetooth with Arduino

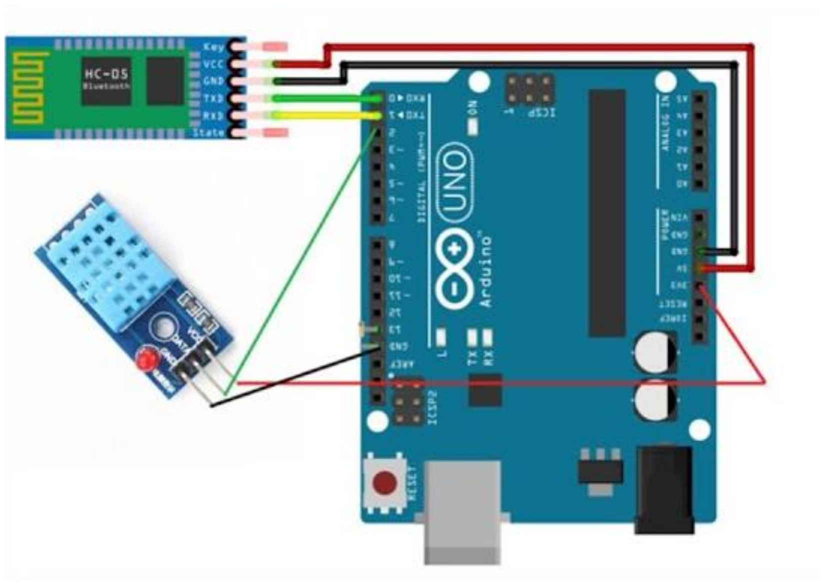


Figure 4.2: Interfacing Bluetooth and DHT11 with Arduino

Pin Connection:

- **HC-05/HC-06 Bluetooth Module:**
  VCC: Connect to Arduino 5V pin.
  GND: Connect to Arduino GND pin.
  TX (Transmit): Connect to Arduino digital pin (e.g., pin 10) via a voltage divider (for HC-05) or directly to pin 10 (for HC-06).
  RX (Receive): Connect to Arduino digital pin (e.g., pin 11) via a voltage divider (for HC-05) or directly to pin 11 (for HC-06).
- **Sensor (e.g., DHT11):**
  Connect VCC to Arduino 5V pin.
  Connect GND to Arduino GND pin.
  Connect DATA to Arduino digital pin (e.g., pin 2).

Procedure: Smartphone Application:

- Pair Bluetooth Module with Smartphone:
  Turn ON the Bluetooth on your smartphone.
  Search for available devices and pair with the Bluetooth module (e.g., HC-05/HC-06).
  Receive Data on Smartphone:

- Write a C program in Arduino IDE that connects to the Bluetooth module and reads the received data.
- Parse the received data to display temperature, humidity, and heat index.
- Perfoorm the pin connection for HC05 and DHT11 sensor on Arduino UNO and verify the sensor readings in the serial Monitor window.

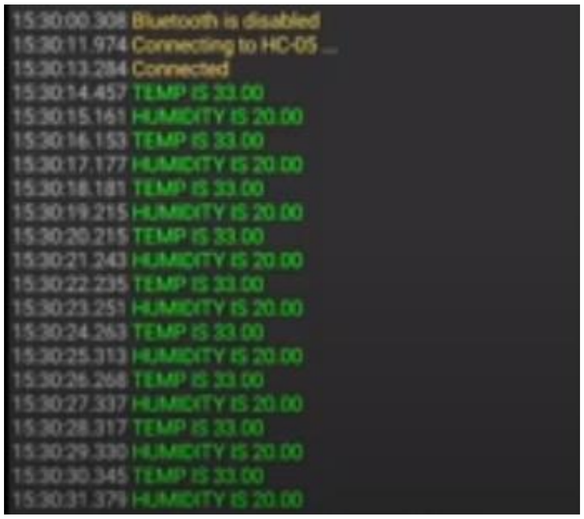
Program:

```
#include <SoftwareSerial.h>
SoftwareSerial BTserial(10, 11); // RX, TX
// Define sensor pin
#define DHTPIN 2
#define DHTTYPE DHT11
#include <DHT.h>
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  // Initialize Serial port
  Serial.begin(9600);
    // Initialize Bluetooth serial port
  BTserial.begin(9600);
  // Initialize DHT sensor
  dht.begin();
}
void loop() {
 // Read sensor data
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();
 // Check if any reads failed
  if (isnan(humidity) || isnan(temperature)) {
  Serial.println("Failed to read from DHT sensor!");
return;
}
// Send sensor data via Bluetooth
BTserial.print("Temperature: ");
BTserial.print(temperature);
BTserial.print(" °C, Humidity: ");
BTserial.print(humidity);
BTserial.println(" %");
// Print sensor data to Serial Monitor
Serial.print("Temperature: ");
Serial.print(temperature);
Serial.print(" °C, Humidity: ");
Serial.print(humidity);
Serial.println(" %");
// Delay
delay(2000);
}
```

Outcomes:

- Successful interfacing of Bluetooth module with Arduino/Raspberry Pi.
- Successful transmission of sensor data to a smartphone using Bluetooth communication.
- The smartphone receives and displays the sensor data sent from Arduino/Raspberry Pi via Bluetooth.

Result:



```
15:30:00.308 Bluetooth is disabled
15:30:11.974 Connecting to HC-05 ...
15:30:13.284 Connected
15:30:14.457 TEMP IS 33.00
15:30:15.161 HUMIDITY IS 20.00
15:30:16.153 TEMP IS 33.00
15:30:17.177 HUMIDITY IS 20.00
15:30:18.181 TEMP IS 33.00
15:30:19.215 HUMIDITY IS 20.00
15:30:20.215 TEMP IS 33.00
15:30:21.243 HUMIDITY IS 20.00
15:30:22.235 TEMP IS 33.00
15:30:23.251 HUMIDITY IS 20.00
15:30:24.263 TEMP IS 33.00
15:30:25.313 HUMIDITY IS 20.00
15:30:26.268 TEMP IS 33.00
15:30:27.337 HUMIDITY IS 20.00
15:30:28.317 TEMP IS 33.00
15:30:29.330 HUMIDITY IS 20.00
15:30:30.345 TEMP IS 33.00
15:30:31.379 HUMIDITY IS 20.00
```

Figure 4.3: Receiving Temperature and Humidity data on Bluetooth terminal of smart phone

Experiment 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.

Objectives:

- **Interfacing Bluetooth with Arduino/Raspberry Pi:**
  Establish communication between Bluetooth module (HC-05/HC-06) and Arduino/Raspberry Pi.
- **Bluetooth Data Reception:**
  Write a C program to receive data ('1' or '0') from a smartphone via Bluetooth.
- **LED Control:**
  Turn ON the LED connected to Arduino/Raspberry Pi when '1' is received.
  Turn OFF the LED connected to Arduino/Raspberry Pi when '0' is received.

Components Required:

- Arduino UNO
- DHT11 sensor
- Resistor(220Ω)
- Connecting cable or USB cable
- Breadboard
- Jumper wires
- HC-05 or HC-06 Bluetooth module
- Smartphone with Bluetooth capability

Circuit Diagram:

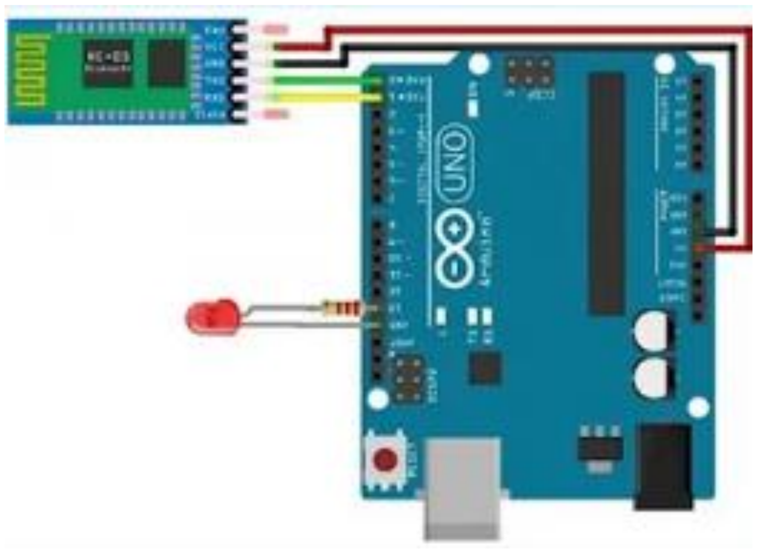


Figure 5.1: Interfacing Bluetooth and LED with Arduino

Pin Connections:

- DHT11 VCC -> Arduino 5V
- DHT11 GND -> Arduino GND
- DHT11 DATA -> Arduino digital pin (e.g., pin 2)
- Resistor -> LED Anode (+)
- LED Cathode (-) -> Arduino GND
- Bluetooth VCC -> Arduino 5V
- Bluetooth GND -> Arduino GND
- Bluetooth TX -> Arduino digital pin (e.g., pin 10)
- Bluetooth RX -> Arduino digital pin (e.g., pin 11)

Procedure:

- **DHT11 Sensor:**
  VCC: Connect to Arduino 5V pin.
  GND: Connect to Arduino GND pin.
  DATA: Connect to Arduino digital pin (e.g., pin 2).
- **Resistor (470Ω):**
  Connect one leg of the resistor to the anode (+) of the LED.

Connect the other leg of the resistor to the anode (+) of the LED.
Connect the cathode (-) of the LED to Arduino GND pin.

- **HC-05/HC-06 Bluetooth Module:**
  VCC: Connect to Arduino 5V pin.
  GND: Connect to Arduino GND pin.
  TX (Transmit): Connect to Arduino digital pin (e.g., pin 10).
  RX (Receive): Connect to Arduino digital pin (e.g., pin 11).

Program:

```
#include <DHT.h>
#include <SoftwareSerial.h>
#define DHTPIN 2     // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11   // DHT 11
#define bluetoothTx 10
#define bluetoothRx 11
DHT dht(DHTPIN, DHTTYPE);
SoftwareSerial bluetooth(bluetoothTx, bluetoothRx);

void setup() {
  Serial.begin(9600);
  bluetooth.begin(9600);  // HC-05 default speed
  dht.begin();
}

void loop() {
  // Reading temperature or humidity takes about 250 milliseconds!
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();

  // Check if any reads failed
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  // Compute heat index in Celsius (isFahreheit = false)
  float heatIndex = dht.computeHeatIndex(temperature, humidity, false);

  // Send sensor data via Bluetooth
  bluetooth.print("Temperature: ");
  bluetooth.print(temperature);
  bluetooth.print(" °C, Humidity: ");
  bluetooth.print(humidity);
  bluetooth.print(" %, Heat index: ");
  bluetooth.print(heatIndex);
  bluetooth.println(" °C");

  // Print sensor data to Serial Monitor
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.print(" °C, Humidity: ");
  Serial.print(humidity);
  Serial.print(" %, Heat index: ");
  Serial.print(heatIndex);
  Serial.println(" °C");

  // Delay
  delay(2000);  // Wait 2 seconds before next reading
}
```

Outcomes:

- Successfully paired the Bluetooth module and established correct pin connections.
- C Programmed the Arduino to interpret Bluetooth data and control the LED.

Result:

Bluetooth	**Smart Phone (1/0)**	**LED Status (ON/OFF)**
Connected		
Not Connected		

Experiment 6

Write a program on Arduino/Raspberry Pi to upload/retrieve temperature and humidity data to thingspeak cloud.

Objectives:

- To interface DHT11 sensor with Arduino/Raspberry Pi and read temperature and humidity data.
- To establish a WiFi connection for Arduino/Raspberry Pi to connect to the internet.
- To upload temperature and humidity data to ThingSpeak cloud using WiFi connectivity.

Components Required:

- Arduino UNO
- DHT11.
- Connecting cable or USB cable.
- Breadboard.
- Jumper wires.

Circuit Diagram

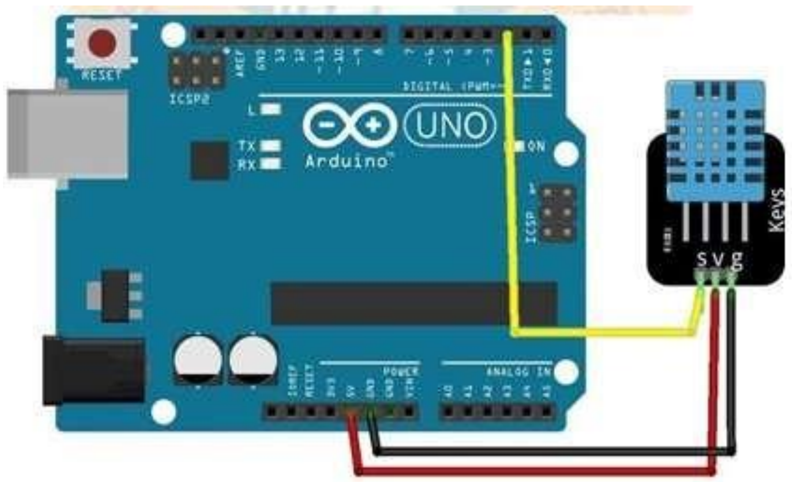


Figure 6.1: Interfacing DHT11 with Arduino

Pin Connections: Arduino UNO Pin Connections:

- DHT11 Sensor
- VCC → 5V
- GND → GND
- DATA → Digital Pin 2

Procedure: Arduino Program:

- Setting ThingSpeak & Getting API Key:
  Go to **https://thingspeak.com/** and set up an account if you do not have one. Login to your account.
- Create a new channel by clicking on the button. Enter the basic details of the channel. Then Scroll down and save the channel. You can follow the video guide below.
- Then go to API keys, copy and paste this key in a separate file. You will require it again while programming.
- **Install DHT Library:**

  Open Arduino IDE.

  Go to Sketch -> Include Library -> Manage Libraries.

  Search for DHT and install the DHT sensor library by Adafruit.
- **Upload the Program:**

  Copy and paste the following Arduino code into the Arduino IDE.

  Replace 'Your_SSID', 'Your_PASSWORD', and 'YOUR_API_KEY' with your WiFi credentials and ThingSpeak API key respectively.
- Select the correct board (Arduino UNO) and port from the Tools menu.
- Click on the Upload button to upload the program to Arduino UNO.
- **Connect the DHT11 Sensor to Arduino UNO:**

  Connect the VCC pin of the DHT11 sensor to the 5V pin on Arduino UNO.

  Connect the GND pin of the DHT11 sensor to the GND pin on Arduino UNO.

  Connect the DATA pin of the DHT11 sensor to Digital Pin 2 on Arduino UNO.
- Connect Arduino UNO to Computer:

  Connect Arduino UNO to the computer using a USB cable.
- Upload the Program:

  Follow the procedure mentioned above to upload the program to Arduino UNO.

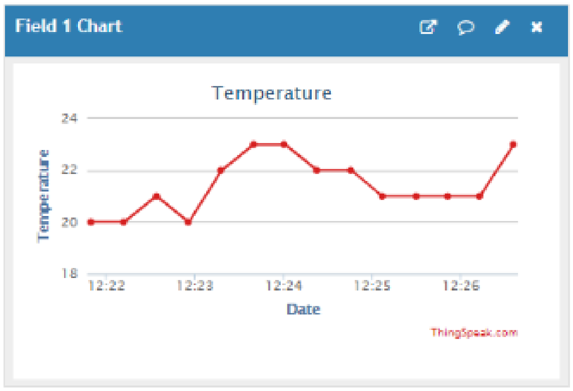
Program:

```
#include <DHT.h>
#include <WiFi.h>
#include <WiFiClient.h>
#define DHTPIN 2          // DHT sensor data pin
#define DHTTYPE DHT11     // DHT sensor type
DHT dht(DHTPIN, DHTTYPE);
const char* ssid = "Your_SSID";         // WiFi SSID
const char* password = "Your_PASSWORD"; // WiFi Password
const char* server = "api.thingspeak.com";
WiFiClient client;
void setup() {
  Serial.begin(115200);
  dht.begin();

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");
}
void loop() {
    float humidity = dht.readHumidity();
    float temperature = dht.readTemperature();
    if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor");
    return;
  }
 Serial.print("Humidity: ");
 Serial.print(humidity);
 Serial.print("% - Temperature: ");
 Serial.print(temperature);
 Serial.println("°C");
 if (client.connect(server, 80)) {
 String postStr = "api_key=YOUR_API_KEY";
    postStr += "&field1=";
    postStr += String(temperature);
    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: YOUR_API_KEY\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);
    Serial.println("Data sent to ThingSpeak");
  } else
  {
    Serial.println("Failed to connect to ThingSpeak");
   }
  client.stop();
  delay(20000); // Wait 20 seconds before sending next data
}
```

Outcomes:

- Successful interfacing of DHT11 sensor with Arduino.
- Establishment of a WiFi connection and internet connectivity for Arduino/Raspberry Pi.
- Successful evaluated the upload of temperature and humidity data to ThingSpeak cloud and visualized the temperature and humidity data on ThingSpeak platform.

Result



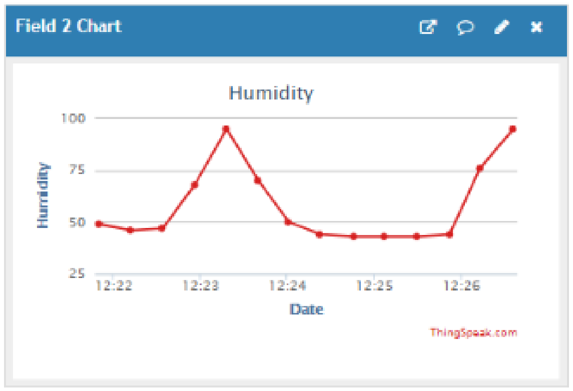


Figure 6.2: Readings in Thingspeak Platform

Experiment 7

Write a program on Arduino/Raspberry Pi to retrieve temperature and humidity data from thingspeak cloud.

Objectives

- To retrieve temperature and humidity data from ThingSpeak cloud using Arduino.
- To establish a WiFi connection for Arduino to connect to the internet.
- To parse and display the retrieved temperature and humidity data from ThingSpeak on the Serial Monitor.

Components Required:

- Arduino UNO
- DHT11.
- Connecting cable or USB cable.
- Breadboard.
- Jumper wires.

Process Diagram:

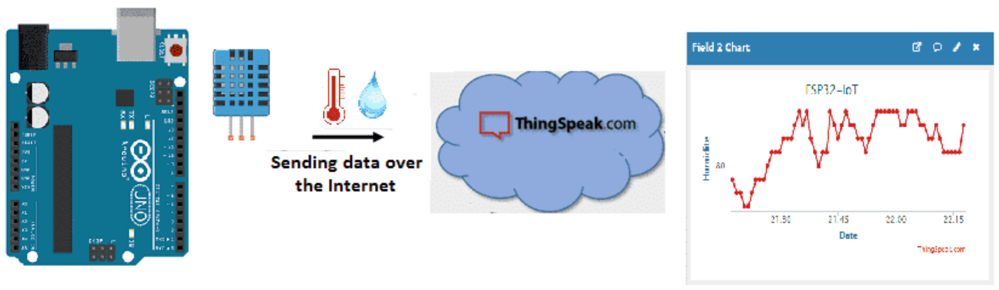


Figure 7.1: Flow of data to ThingSpeak Platform

Circuit Diagram

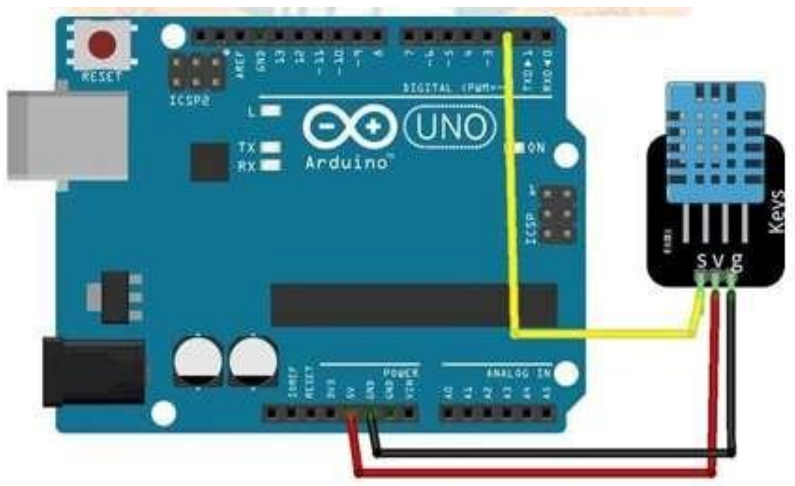


Figure 7.2: Interfacing DHT11 with Arduino

Pin Connection:

- Connect the VCC pin of the DHT11 sensor to the 5V pin on Arduino UNO.
- Connect the GND pin of the DHT11 sensor to the GND pin on Arduino UNO.
- Connect the DATA pin of the DHT11 sensor to Digital Pin 2 on Arduino UNO.

Procedure:

- **Arduino Program:**
  Install WiFi Library:
  Open Arduino IDE.
  Go to Sketch -> Include Library -> Manage Libraries.
  Search for WiFi and install the WiFi library.
- **Upload the Program:**
  Copy and paste the following Arduino code into the Arduino IDE.
- Replace 'Your_SSID', 'Your_PASSWORD', 'YOUR_CHANNEL_ID', and 'YOUR_READ_API_KEY' with your WiFi credentials, ThingSpeak channel ID, and read API key respectively.
- Select the correct board (Arduino UNO) and port from the Tools menu.
- Click on the Upload button to upload the program to Arduino UNO.

- **Connect the DHT11 Sensor to Arduino UNO:**

  Connect the VCC pin of the DHT11 sensor to the 5V pin on Arduino UNO.

  Connect the GND pin of the DHT11 sensor to the GND pin on Arduino UNO.

  Connect the DATA pin of the DHT11 sensor to Digital Pin 2 on Arduino UNO.

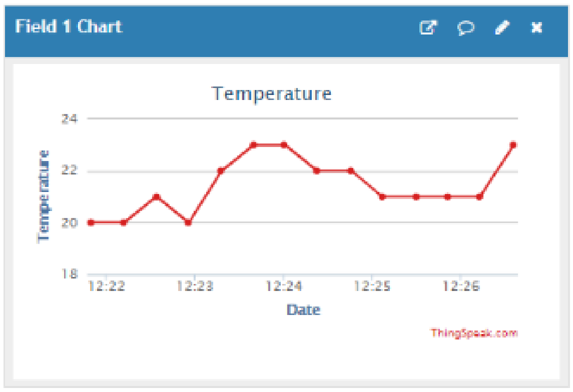
- **Connect Arduino UNO to Computer:**

  Connect Arduino UNO to the computer using a USB cable.

Outcomes:

- Successful interfacing of DHT11 sensor with Arduino.
- Establishment of a WiFi connection and internet connectivity for Arduino and retrieved temperature and humidity data on the Serial Monitor.
- Successful evaluated the upload of temperature and humidity data to ThingSpeak cloud and visualized the temperature and humidity data on ThingSpeak platform.

Result



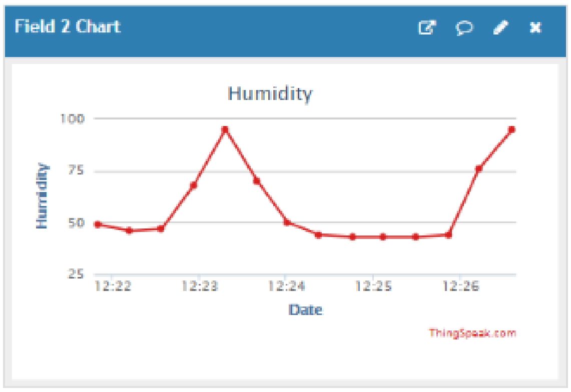
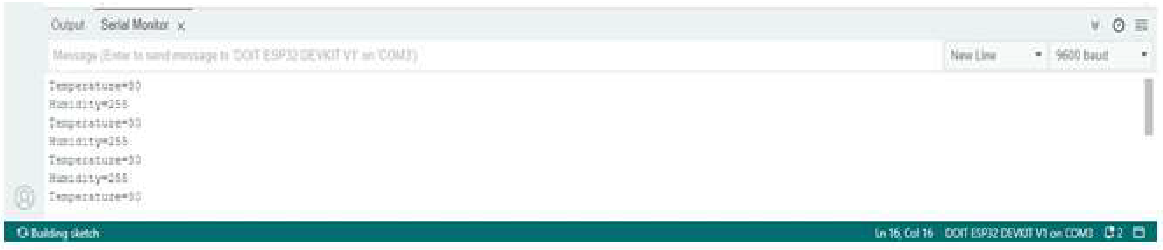


Figure 7.3: Readings in Thingspeak Platform


Output
Serial Monitor
Message (Enter to send message to 'DOIT ESP32 DEVKIT V1' on 'COM3')
New Line
9600 baud
Building sketch
Ln 16, Col 16
DOIT ESP32 DEVKIT V1 on COM3

```
Temperature=30
Humidity=255
Temperature=30
Humidity=255
Temperature=30
Humidity=255
Temperature=30
```

Figure 7.4: Displaying temperature and humidity data on Serial Monitor of Arduino IDE from Thingspeak cloud

Experiment 8

Write a program on Arduino/Raspberry Pi to publish temperature data to MQTT broker

Objectives:

- To publish temperature data from Arduino to an MQTT broker.
- To establish a WiFi connection for Arduino to connect to the internet.
- To connect Arduino to an MQTT broker and publish temperature data to a specific topic.

Method-1: Components Required:

- Arduino UNO
- DHT11.
- Connecting cable or USB cable.
- Breadboard.
- Jumper wires

Circuit Diagram:

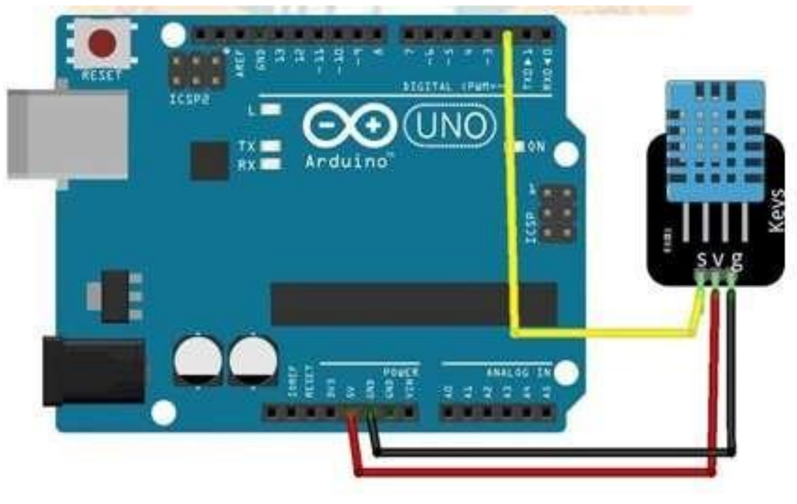


Figure 8.1: Interfacing DHT11 with Arduino

Pin Connection:

- Connect the VCC pin of the DHT11 sensor to the 5V pin on Arduino UNO.
- Connect the GND pin of the DHT11 sensor to the GND pin on Arduino UNO.
- Connect the DATA pin of the DHT11 sensor to Digital Pin 2 on Arduino UNO.

Procedure:

- **Arduino Program:**
  Install WiFi Library:
  Open Arduino IDE.
  Go to Sketch -> Include Library -> Manage Libraries.
  Search for WiFi and install the WiFi library.
- **Upload the Program:**
  Copy and paste the following Arduino code into the Arduino IDE.
- Replace 'Your_SSID', 'Your_PASSWORD', 'YOUR_CHANNEL_ID', and 'YOUR_READ_API_KEY' with your WiFi credentials, ThingSpeak channel ID, and read API key respectively.
- Select the correct board (Arduino UNO) and port from the Tools menu.
- Click on the Upload button to upload the program to Arduino UNO.
- **Connect the DHT11 Sensor to Arduino UNO:**
  Connect the VCC pin of the DHT11 sensor to the 5V pin on Arduino UNO.
  Connect the GND pin of the DHT11 sensor to the GND pin on Arduino UNO.
  Connect the DATA pin of the DHT11 sensor to Digital Pin 2 on Arduino UNO.
- **Connect Arduino UNO to Computer:**
  Connect Arduino UNO to the computer using a USB cable.

Program:

```
#include <WiFi.h>
#include <PubSubClient.h>
const char* ssid = "Your_SSID";         // WiFi SSID
const char* password = "Your_PASSWORD"; // WiFi Password
const char* mqttServer = "broker_address"; // MQTT Broker Address
const int mqttPort = 1883; // MQTT Broker Port
WiFiClient espClient;
PubSubClient client(espClient);
const char* temperatureTopic = "your_topic/temperature"; // MQTT Topic for Temperature
const char* mqttUsername = "your_mqtt_username"; // MQTT Username
const char* mqttPassword = "your_mqtt_password"; // MQTT Password
void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");
  client.setServer(mqttServer, mqttPort);
  client.setCallback(callback);

  while (!client.connected()) {
    Serial.println("Connecting to MQTT...");

    if (client.connect("ArduinoClient", mqttUsername, mqttPassword)) {
      Serial.println("Connected to MQTT");
    } else {
      Serial.print("Failed to connect to MQTT, rc=");
      Serial.println(client.state());
      delay(2000);
    }
  }
}
```

```
void loop() {
  float temperature = readTemperature(); // Replace this with your temperature reading function

  if (!client.connected()) {
    reconnect();
  }

  client.loop();
  char tempStr[10];
  dtostrf(temperature, 4, 2, tempStr);

  client.publish(temperatureTopic, tempStr);
  Serial.print("Temperature published: ");
  Serial.println(tempStr);

  delay(5000); // Wait 5 seconds before publishing next data
}

void callback(char* topic, byte* message, unsigned int length) {
  // Handle incoming messages if needed
}

void reconnect() {
  while (!client.connected()) {
    Serial.println("Attempting MQTT connection...");

    if (client.connect("ArduinoClient", mqttUsername, mqttPassword)) {
      Serial.println("Connected to MQTT");
    } else {
      Serial.print("Failed to connect to MQTT, rc=");
      Serial.println(client.state());
      delay(2000);
    }
  }
}

float readTemperature() {
  // Replace this function with your temperature reading logic return 25.5; // Example temperature
//reading
}
```

Outcomes:

- Successful connection of Arduino to WiFi network.
- Successful connection of Arduino to MQTT broker.
- Evaluated publishing of temperature data from Arduino to the specified MQTT

Result:

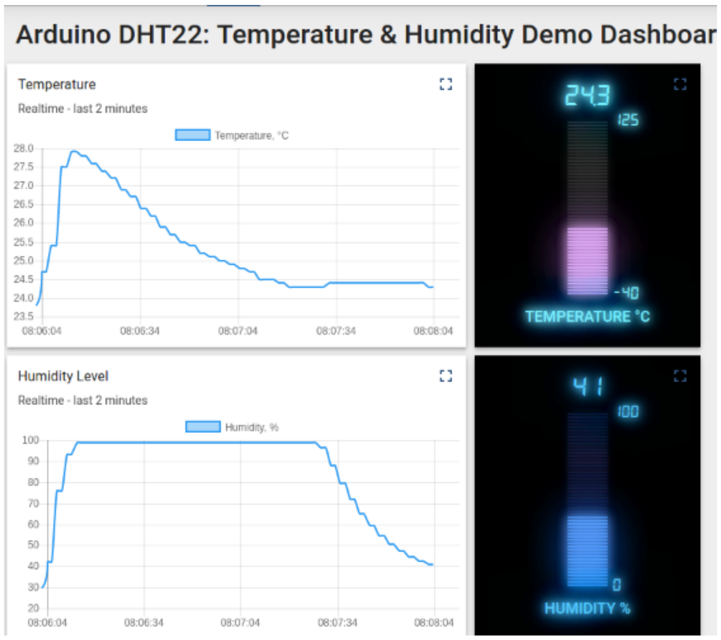


Figure 8.2: Temperature and Humidity Readings

Method-2

Program for Publishing Temperature Data to MQTT Broker on NodeMCU

This experiment involves using a NodeMCU (ESP8266) to publish temperature data from a DHT11 sensor to an MQTT broker. The MQTT broker can either be a public broker like test.mosquitto.org or a private one set up locally.

Theory:

MQTT is a lightweight messaging protocol commonly used in IoT devices to communicate data between clients and brokers. In this example, the NodeMCU will connect to an MQTT broker and send temperature data from the DHT11 sensor.

Required Libraries:

1. **ESP8266WiFi.h**: For Wi-Fi connectivity on NodeMCU.
2. **PubSubClient.h**: For MQTT communication.
3. **DHT.h**: For reading data from the DHT11 sensor.

Circuit Connection:

- **VCC** to **3.3V** on NodeMCU.
- **GND** to **GND** on NodeMCU.
- **Data pin** to **GPIO pin (D2)** on NodeMCU.

Program Code:

```
#include <ESP8266WiFi.h>

#include <PubSubClient.h>

#include <DHT.h>


// Replace these with your network credentials
```

```
const char* ssid = "your_SSID";

const char* password = "your_PASSWORD";



// Local MQTT broker details

const char* mqtt_server = "192.168.1.10"; // Replace with your local IP

const int mqtt_port = 1883;



// MQTT topic

const char* temperature_topic = "home/temperature";



// DHT11 sensor configuration

#define DHTPIN D2 // GPIO pin where the data pin of the DHT11 is connected

#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);



WiFiClient espClient;

PubSubClient client(espClient);



void setup() {

  Serial.begin(115200);

  setup_wifi();

  client.setServer(mqtt_server, mqtt_port);
  dht.begin();

}

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");

  Serial.println("IP address: ");

  Serial.println(WiFi.localIP());

}


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

  float t = dht.readTemperature();
  float h = dht.readHumidity();

  if (isnan(t) || isnan(h)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }


  String temp_str = String(t);

  char temp_data[8];

  temp_str.toCharArray(temp_data, temp_str.length() + 1);

  client.publish(temperature_topic, temp_data);


  Serial.print("Temperature: ");

  Serial.println(temp_str);


  delay(2000); // Delay in milliseconds

}
```

Steps to Upload the Code:

1. **Connect NodeMCU**: Plug the NodeMCU into your computer.
2. **Open Arduino IDE**: Launch the Arduino IDE.
3. **Copy and Paste Code**: Copy the provided code into the Arduino IDE.
4. **Modify Credentials**: Update the ssid and password variables with your Wi-Fi network credentials.
5. **Select Board and Port**:
    - Go to **Tools > Board** and select **NodeMCU 1.0 (ESP-12E Module)**.
    - Go to **Tools > Port** and select the correct COM port for the NodeMCU.
6. **Upload the Code**: Click the **Upload** button (right arrow) to upload the program to the NodeMCU.

MQTT Broker Configuration:

- **Public Broker**: If using a public broker, you can replace the mqtt_server value with test.mosquitto.org.
- **Private Broker**: If you are running your own MQTT broker (e.g., Mosquitto), replace the mqtt_server value with the broker's IP address.

Monitoring with MQTT Explorer:

1. **Download and Install MQTT Explorer**.
2. **Connect to Broker**:
    - Open MQTT Explorer.
    - Click on **'Add new connection'**.
    - Enter **Broker Address** (test.mosquitto.org for public or your local IP for private).
    - Set **Broker Port** to **1883**.
    - Click **'Save'** and **'Connect'**.
3. **Subscribe to Topic**:
    - In MQTT Explorer, subscribe to the **home/temperature** topic.
4. **View Data**: The temperature data from the NodeMCU will appear in the message log.

Result:

- The **NodeMCU** will connect to the specified **Wi-Fi** network and **MQTT broker**.
- It will read the temperature data from the **DHT11 sensor** and **publish** it to the **home/temperature** topic every **2 seconds**.
- You will see the published temperature data in the **MQTT Explorer** or any MQTT client subscribed to the topic.

Experiment-9

Write a program to create UDP server on Arduino/Raspberry Pi and respond with humidity data to UDP client when requested.

Objectives:

- To establish a reliable UDP server on the NodeMCU that connects to a Wi-Fi network and listens for incoming packets on port 4210, providing real-time humidity data from the DHT11 sensor when requested.
- To implement serial communication for monitoring the Wi-Fi connection status and incoming UDP packets on the NodeMCU, ensuring successful packet reception and response with accurate sensor data.

Prerequisites:

1. **Hardware**:
    - NodeMCU (ESP8266)
    - DHT11 Temperature and Humidity Sensor
    - Breadboard and Jumper Wires
2. **Software**:
    - Arduino IDE
    - DHT Sensor Library

Circuit Connections:

1. **NodeMCU to DHT11**:
    - VCC of DHT11 to 3.3V of NodeMCU
    - GND of DHT11 to GND of NodeMCU
    - DATA of DHT11 to D4 (GPIO2) of NodeMCU

Setup Instructions:

1. **Install Arduino IDE**:
    - Download and install the Arduino IDE from Arduino's official website.
2. **Add ESP8266 Board to Arduino IDE**:
    - Open Arduino IDE.
    - Go to **File > Preferences**.
    - In the **Additional Boards Manager URLs** field, add http://arduino.esp8266.com/stable/package_esp8266com_index.json.
    - Go to **Tools > Board > Boards Manager**.
    - Search for **ESP8266** and install it by ESP8266 Community.
3. **Install Libraries**:
    - Go to **Sketch > Include Library > Manage Libraries**.
    - Search for and install the **DHT sensor library**.

Arduino Program:

```
#include <ESP8266WiFi.h>

#include <WiFiUdp.h>

#include <DHT.h>

#define DHTPIN D4  // DHT11 data pin connected to D4 (GPIO2)

#define DHTTYPE DHT11  // DHT11 type

const char* ssid = "YOUR_SSID";  // Replace with your WiFi SSID

const char* password = "YOUR_PASSWORD";  // Replace with your WiFi Password
```

```
WiFiUDP udp;

unsigned int localUdpPort = 4210;  // Local UDP port to listen on

char incomingPacket[255];  // Buffer for incoming packets

char replyPacket[255];  // Buffer for outgoing packets

DHT dht(DHTPIN, DHTTYPE);

void setup() {

  Serial.begin(115200);



  // Connecting to WiFi

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {

    delay(500);

    Serial.print(".");

  }

  Serial.println("WiFi connected");

  Serial.print("IP address: ");

  Serial.println(WiFi.localIP());



  // Start UDP

  udp.begin(localUdpPort);
  Serial.printf("Now listening at IP %s, UDP port %d\n", WiFi.localIP().toString().c_str(),
localUdpPort);
  dht.begin();
}
void loop() {
  int packetSize = udp.parsePacket();
  if (packetSize) {
    // Read the packet into the buffer
    int len = udp.read(incomingPacket, 255);
    if (len > 0) {
      incomingPacket[len] = 0;
    }
    Serial.printf("Received packet: %s\n", incomingPacket);

    // Check if the packet contains the request for humidity
    if (String(incomingPacket) == "GET_HUMIDITY") {
      float h = dht.readHumidity();
      if (isnan(h)) {
        Serial.println("Failed to read from DHT sensor!");
        snprintf(replyPacket, 255, "Failed to read humidity");
      } else {
        snprintf(replyPacket, 255, "Humidity: %.2f%%", h);
      }


      // Send a reply to the IP address and port of the sender
      udp.beginPacket(udp.remoteIP(), udp.remotePort());
      udp.write(replyPacket);
      udp.endPacket();
    }
  }
}
```

Procedure to Upload Program:

1. **Open the Arduino IDE**.
2. **Copy and paste** the provided program into the IDE.
3. **Replace placeholders** with your actual Wi-Fi details:
   - ssid – Your Wi-Fi SSID.
   - password – Your Wi-Fi password.
4. **Select the correct board**:
   - Go to **Tools > Board** and select **NodeMCU 1.0 (ESP-12E Module)**.
5. **Select the correct port**:
   - Go to **Tools > Port** and select the COM port to which your NodeMCU is connected.

6. **Upload the program**:
   - Click on the **Upload** button (right arrow) to upload the program to your NodeMCU.

Monitor the Output:

1. Open the **Serial Monitor** in Arduino IDE by going to **Tools > Serial Monitor** or pressing **Ctrl+Shift+M**.
2. Set the baud rate to **115200**.
3. Observe the output. The Serial Monitor will display Wi-Fi connection status and incoming UDP packets.

Test the UDP Server:

1. **Use a UDP client** (such as **Packet Sender**, **Netcat**, or a custom script) to send a UDP packet with the message GET_HUMIDITY to the NodeMCU's IP address and port 4210.
2. The NodeMCU will respond with the current humidity data from the DHT11 sensor.

Outcome:

- The **NodeMCU** will connect to the specified Wi-Fi network.
- It will listen for incoming UDP packets on **port 4210**.
- When a packet with the message **GET_HUMIDITY** is received, it will respond with the current humidity data from the **DHT11 sensor**.
- The **Serial Monitor** will display the Wi-Fi connection status and incoming UDP packets.

Experiment-10

Write a program to create TCP server on Arduino/Raspberry Pi and respond with humidity data to TCP client when requested

Objectives:

- To establish a TCP server on the NodeMCU that connects to a Wi-Fi network and responds with humidity data from the DHT11 sensor when requested by a TCP client.
- To monitor the connection status of the NodeMCU to the Wi-Fi network and TCP client interactions, and display real-time sensor readings via the Serial Monitor.

Prerequisites:

1. Hardware:
    - NodeMCU (ESP8266)
    - DHT11 Temperature and Humidity Sensor
    - Breadboard and Jumper Wires
2. Software:
    - Arduino IDE
    - DHT Sensor Library

Circuit Connection:

- VCC to 3.3V on NodeMCU.
- GND to GND on NodeMCU.
- Data pin to GPIO2 (D4) on NodeMCU.

Step-by-Step Guide:

Step 1: Gather Components

1. NodeMCU (ESP8266)
2. DHT11 Temperature and Humidity Sensor
3. Breadboard
4. Jumper Wires

Step 2: Circuit Connections

1. Connect the VCC pin of the DHT11 sensor to the 3.3V pin on the NodeMCU.
2. Connect the GND pin of the DHT11 sensor to the GND pin on the NodeMCU.
3. Connect the DATA pin of the DHT11 sensor to GPIO2 (D4) on the NodeMCU.

Step 3: Set up the Software

1. Install Arduino IDE:
   - Download and install the Arduino IDE from the official website.
2. Add ESP8266 Board to Arduino IDE:
   - Open Arduino IDE.
   - Go to File > Preferences.
   - In the Additional Boards Manager URLs field, add: http://arduino.esp8266.com/stable/package_esp8266com_index.json.
   - Go to Tools > Board > Boards Manager.
   - Search for ESP8266 and install it from ESP8266 Community.
3. Install Libraries:
   - Go to Sketch > Include Library > Manage Libraries.
   - Search for and install the DHT sensor library.

Step 4: Write and Upload the Program

1. Open Arduino IDE.
2. Copy and paste the provided program into the IDE.
3. Replace the placeholders with your actual Wi-Fi details:
   - ssid - Your Wi-Fi SSID.
   - password - Your Wi-Fi password.
4. Go to Tools > Board and select NodeMCU 1.0 (ESP-12E Module).
5. Go to Tools > Port and select the COM port to which your NodeMCU is connected.

6. Click on the Upload button (right arrow) to upload the program to your NodeMCU.

Program Code:

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

#define DHTPIN D4    // DHT11 data pin connected to D4 (GPIO2)
#define DHTTYPE DHT11 // DHT11 type
DHT dht(DHTPIN, DHTTYPE);

const char* ssid = "OPPO F21s Pro";  // Replace with your Wi-Fi SSID
const char* password = "Deepashree@04";  // Replace with your Wi-Fi password

WiFiServer server(80); // TCP server on port 80

float humidity, temp_f;
String webString = "";
unsigned long previousMillis = 0; // Will store last time the temp was read
const long interval = 2000; // Interval at which to read sensor

void setup() {
  Serial.begin(115200);
  dht.begin();
  WiFi.begin(ssid, password);
  Serial.print("\n\r \n\rConnecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());

  server.begin();  // Start the TCP server
  Serial.println("TCP server started");
}

void loop() {
  WiFiClient client = server.available();  // Listen for incoming clients

  if (client) {
    Serial.println("New client connected");
    while (client.connected()) {
      if (client.available()) {
        char c = client.read(); // Read the incoming byte
        if (c == '\n') {  // If a new line is received
```

```
          gettemperature(); // Read the sensor data
          webString = "Humidity: " + String((int)humidity) + "%";
          client.println(webString); // Send humidity data to client
          delay(100);
          break;
        }
      }
    }
    client.stop();
    Serial.println("Client disconnected");
  }
}

void gettemperature() {
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    // Save the last time the sensor was read
    previousMillis = currentMillis;
    humidity = dht.readHumidity();
    temp_f = dht.readTemperature(true);

    if (isnan(humidity) || isnan(temp_f)) {
      Serial.println("Failed to read from DHT sensor!");
      return;
    }
  }
}
```

Step 5: Monitor the Output

1. Open the Serial Monitor in the Arduino IDE by going to Tools > Serial Monitor or pressing Ctrl+Shift+M.
2. Set the baud rate to 115200.
3. Observe the output. The Serial Monitor will display Wi-Fi connection status, IP address, and incoming TCP client connections.

Step 6: Test the TCP Server

1. Use a TCP client (such as Telnet, Netcat, or a custom script) to connect to the NodeMCU's IP address on port 80.
2. Send a request to the server (simply pressing Enter should work).
3. The NodeMCU will respond with the current humidity data from the DHT11 sensor.

Outcomes:

- The NodeMCU will connect to the specified Wi-Fi network.
- When a TCP client connects and sends a request, the NodeMCU will respond with the current humidity data from the DHT11 sensor.
- The Serial Monitor will display Wi-Fi connection status, incoming TCP client connections, and sensor readings.

Experiment-11

Write a program on Arduino/Raspberry Pi to subscribe to MQTT broker for temperature data and print it

Objectives

- To establish a reliable connection between the NodeMCU and a Wi-Fi network, enabling the device to subscribe to the MQTT broker and listen for temperature data on the home/temperature topic.
- To continuously read temperature data from the DHT11 sensor, publish it to the MQTT broker at regular intervals, and monitor the connection statuses and incoming messages via the Serial Monitor.

Prerequisites:

1. Hardware:
    - NodeMCU (ESP8266)
    - DHT11 Temperature and Humidity Sensor
    - Breadboard and Jumper Wires
2. Software:
    - Arduino IDE
    - DHT Sensor Library
    - PubSubClient Library (for MQTT)

Circuit Connections:

1. NodeMCU to DHT11:
    - VCC of DHT11 to 3.3V of NodeMCU
    - GND of DHT11 to GND of NodeMCU
    - DATA of DHT11 to D4 (GPIO2) of NodeMCU

Setup Instructions:

1. Install Arduino IDE:
    - Download and install the Arduino IDE from Arduino's official website.
2. Add ESP8266 Board to Arduino IDE:

- Open Arduino IDE.
- Go to File > Preferences.
- In the Additional Board Manager URLs field, add http://arduino.esp8266.com/stable/package_esp8266com_index.json.
- Go to Tools > Board > Boards Manager.
- Search for ESP8266 and install it by ESP8266 Community.

3. Install Libraries:

- Go to Sketch > Include Library > Manage Libraries.
- Search for and install the DHT sensor library.
- Search for and install the PubSubClient library.

Arduino Program:

```
#include <ESP8266WiFi.h>

#include <PubSubClient.h>

#include <DHT.h>


#define DHTPIN D4  // DHT11 data pin connected to D4 (GPIO2)

#define DHTTYPE DHT11  // DHT11 type


const char* ssid = "YOUR_SSID";  // Replace with your WiFi SSID

const char* password = "YOUR_PASSWORD";  // Replace with your WiFi Password

const char* mqttServer = "YOUR_MQTT_BROKER";  // Replace with your MQTT Broker address

const int mqttPort = 1883;

const char* mqttUser = "YOUR_MQTT_USER";  // Replace with your MQTT Broker Username

const char* mqttPassword = "YOUR_MQTT_PASSWORD";  // Replace with your MQTT Broker Password
```

```
WiFiClient espClient;
PubSubClient client(espClient);
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(115200);

  // Connecting to WiFi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("WiFi connected");

  client.setServer(mqttServer, mqttPort);
  client.setCallback(callback);
  dht.begin();
}

void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();
}

void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    if (client.connect("ESP8266Client", mqttUser, mqttPassword)) {
      Serial.println("connected");
      client.subscribe("home/temperature");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}

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

  // Read humidity and temperature data from DHT11
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  // Publish the temperature data to the MQTT topic
  String tempString = String(t);
  client.publish("home/temperature", tempString.c_str());
  delay(2000);
}
```

Procedure to Upload Program:

1. Open the Arduino IDE.
2. Copy and paste the provided program into the IDE.
3. Replace placeholders with your actual Wi-Fi and MQTT broker details:
    - ssid – Your Wi-Fi SSID.
    - password – Your Wi-Fi password.
    - mqttServer – MQTT broker address.

  - mqttUser – MQTT broker username.
  - mqttPassword – MQTT broker password.

4. Select the correct board:
    - Go to Tools > Board and select NodeMCU 1.0 (ESP-12E Module).
5. Select the correct port:
    - Go to Tools > Port and select the COM port to which your NodeMCU is connected.
6. Upload the program:
    - Click on the Upload button (right arrow) to upload the program to your NodeMCU.

Monitor the Output:

1. Open the Serial Monitor in Arduino IDE by going to Tools > Serial Monitor or pressing Ctrl+Shift+M.
2. Set the baud rate to 115200.
3. Observe the output. The Serial Monitor will display Wi-Fi connection status, MQTT connection status, and incoming messages on the subscribed topic.

Verify MQTT Data:

1. Use an MQTT client (such as MQTT.fx, MQTT Explorer, or an online MQTT dashboard) to subscribe to the home/temperature topic.
2. Verify that temperature data is being published to the topic by the NodeMCU.

Outcomes:

- The NodeMCU will connect to the specified Wi-Fi network.
- It will connect to the MQTT broker and subscribe to the home/temperature topic.
- It will read temperature data from the DHT11 sensor and publish it to the home/temperature topic every 2 seconds.
- The Serial Monitor will display connection statuses and incoming messages.