School of Computing

Department of Computer Science and Engineering

UNIT 1- ARDUINO PROGRAMMING FOR IoT BOARDS - SCSA1407
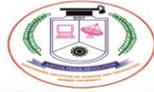
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

COURSE OBJECTIVES

- To program Arduino to control lights, motors, and other devices.
- To learn Arduino's architecture, including inputs and connectors for add-on devices.
- To add third-party components such as LCDs, accelerometers, gyroscopes, and GPS trackers to extend Arduino's functionality.
- To understand various options in programming languages, from C to drag-and-drop languages.
- To test, debug, and deploy the Arduino to solve real world problems.

COURSE OUTCOMES:

CO1: Recall the basics of sensors, its functioning.

CO2: Execute basic and advanced assembly language programs.

CO3: Learn the ways to interface I/O devices with processor for task sharing.

CO4: Recall the basics of co-processor and its ways to handle float values by its instruction set.

CO5: Recognize the functionality of micro controller, latest version processors and its applications.

CO6: Acquire design thinking capability, ability to design a component with realistic constraints, to solve real world engineering problems and analyse the results.

UNIT 1: Introduction to sensors   9 hours

Transducers, Classification, Roles of sensors in IOT, Various types of sensors, Design of sensors, sensor architecture, special requirements for IOT sensors, Role of actuators, types of actuators.

UNIT 2: Hardware   9 hours

Physical device – Arduino Interfaces, Hardware requirement for Arduino, Connecting remotely over the network using VNC, GPIO Basics, Controlling GPIO Outputs Using a Web Interface, – Programming, APIs / Packages- Quark SOC processor, programming, Arduino Boards using GPIO (LED, LCD, Keypad, Motor control and sensor)
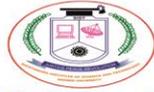
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

UNIT 3: Platforms 9 hours

History - Creative Coding Platforms - Open Source Platforms – PIC - Arduino, Sketch, Iterative coding methodology – Python Programming - Mobile phones and similar devices - Arm Devices - Basic Electronics (circuit theory, measurements, parts identification) Sensors and Software: Understanding Processing Code Structure, variables and flow control, Interfacing to the Real World

Unit 4 Programming an Arduino IoT Device 9 hours

Preparing the development environment (Arduino IDE), Exploring the Arduino language (C/C++) syntax, Coding, compiling, and uploading to the microcontroller, Working with Arduino Communication Modules: Bluetooth Modules, WiFi Modules and I2C and SPI, Interfacing arduino and Blynk via USB : LED Blinking, Controlling a Servomotor.

Unit 5 Programming ESP 8266 Module 9 hours

ESP8266 WiFi Serial Module: Overview, Setting Up the Hardware, Interfacing with Arduino, Creating an IoT Temperature and Humidity Sensor System, Overview of DHT-22 Sensor, Interfacing the Hardware: Arduino, ESP8266 WiFi Module, and DHT-22 Sensor, Checking Your Data via ThingSpeak, Connecting Your Arduino Set-up to Blynk via WiFi

Max 45 hours

SCSA1407- ARDUINO PROGRAMMING FOR IoT BOARDS

UNIT 1: Introduction to sensors

Transducers, Classification, Roles of sensors in IOT, Various types of sensors, Design of sensors, sensor architecture, special requirements for IOT sensors, Role of actuators, types of actuators.

1. TRANSDUCER

The transducer changes the physical quantity into an electrical signal. It is an electronic device which has two main functions, i.e., sensing and transduction. It senses the physical quantity and then converts it into mechanical works or electrical signals.

The transducer is of many types, and they can be classified by the following criteria.

1. **By transduction used.**

2. **as a primary and secondary transducer**

3. **as a passive and active transducer**

4. **as analogue and digital transducer**

5. **as the transducer and inverse transducer**

The transducer receives the measurand and gives a proportional amount of output signal. The output signal is sent to the conditioning device where the signal is attenuated, filtered, and modulated.
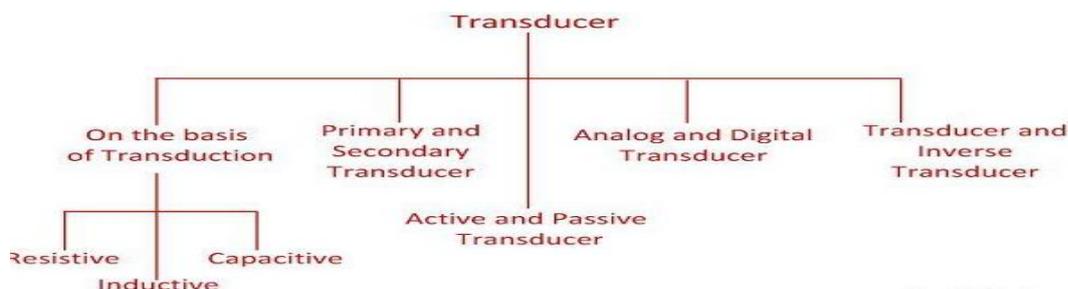
Fig. 1 Types of Transducer
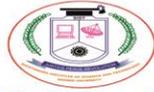
The input quantity is the non-electrical quantity, and the output electrical signal is in the form of the current, voltage or frequency. The transducer is classified by the transduction medium. The transduction medium may be resistive, inductive or capacitive depends on the conversion process that how input transducer converts the input signal into resistance, inductance and capacitance respectively.

1.1 Primary and Secondary Transducer

Primary Transducer – The transducer consists the mechanical as well as the electrical devices. The mechanical devices of the transducer change the physical input quantities into a mechanical signal. This mechanical device is known as the primary transducers.

Secondary Transducer – The secondary transducer converts the mechanical signal into an electrical signal. The magnitude of the output signal depends on the input mechanical signal.

Example of Primary and Secondary Transducer

Consider the Bourdon's Tube shown in the figure below. The tube act as a primary transducer. It detects the pressure and converts it into a displacement from its free end. The displacement of the free ends moves the core of the linear variable displacement transformer. The movement of the core induces the output voltage which is directly proportional to the displacement of the tube free end.

Thus, the two type of transduction occurs in the Bourdon's tube. First, the pressure is converted into a displacement and then it is converted into the voltage by the help of the L.V.D.T.

The Bourdon's Tube is the primary transducer, and the L.V.D.T is called the secondary transducer.

1.2 Passive and Active Transducer

The transducer is classified as the active and passive transducer.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

Passive Transducer – The transducer which requires the power from an external supply source is known as the passive transducer. They are also known as the external power transducer. The capacitive, resistive and inductive transducers are the example of the passive transducer.

Active Transducer – The transducer which does not require the external power source is known as the active transducer. Such type of transducer develops theirs owns voltage or current, hence known as a self-generating transducer. The output signal is obtained from the physical input quantity.

The physical quantity like velocity, temperature, force and the intensity of light is induced with the help of the transducer. The piezoelectric crystal, photo-voltaic cell, tacho generator, thermocouples, photovoltaic cell are the examples of the active transducers.

Examples – Consider the examples of a piezoelectric crystal. The crystal is sandwiched between the two metallic electrodes, and the entire sandwiched is fastened to the base. The mass is placed on the top of the sandwiched.

The piezo crystal has the special property because of which when the force is applied to the crystal, they induce the voltage. The base provides the acceleration due to which the voltage is generated. The mass applies on the crystals induces an output voltage. The output voltage is proportional to the acceleration. The above mention transducer is known as the accelerometer which converts the acceleration into an electric voltage. This transducer does not require any auxiliary power source for the conversion of physical quantity into an electrical signal.

1.3 Analog and Digital Transducer

The transducer can also be classified by their output signals. The output signal of the transducer may be continuous or discrete.

Analog Transducer – The Analog transducer changes the input quantity into a continuous function. The strain gauge, L.V.D.T, thermocouple, thermistor are the examples of the analogue transducer.
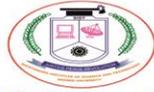
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

Digital Transducer – These transducers convert an input quantity into a digital signal or in the form of the pulse. The digital signals work on high or low power.

1.4 Transducer and Inverse Transducer

Transducer – The device which converts the non-electrical quantity into an electric quantity is known as the transducer.

Inverse Transducer – The transducer which converts the electric quantity into a physical quantity, such type of transducers is known as the inverse transducer. The transducer has high electrical input and low non-electrical output.

Examples of transducer

1. A mechanical force or displacement being converted into an electrical signal.
2. A thermistor reacts to temperature variations.
3. A photo cell changes in light intensity.
4. Measurement of electrical noise.
5. Telemetering system—when input and output are in electrical form.

Characteristics of Transducer

Sensitivity. It can be defined as the ratio of the incremental output and the incremental input. While defining the sensitivity, we assume that the input-output characteristic of the instrument is approximately linear in that range.

Range. The range of the sensor is the maximum and minimum values of applied parameters that can be measured.

Precision. The concept of precision refers to the degree of reproducibility of a measurement. In other words, if exactly the same value were measured a number of times, an ideal sensor

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

would output exactly the same value every time. But real sensors output a range of values distributed in some manner relative to the actual correct value.

Resolution. The smallest difference between measured values that can be discriminated. For example, it corresponds to the last stable figure on a digital display. This specification is the smallest detectable incremental change of the input parameter that can be detected in the output signal. Resolution can be expressed either as a proportion of the reading (or the full-scale reading) or in absolute terms.

Accuracy. The accuracy of the sensor is the maximum difference that will exist between the actual value and the indicated value at the output of the sensor. Again, the accuracy can be expressed either as a percentage of full scale or in absolute terms.

Linearity. The linearity of the transducer is an expression of the extent to which the actual measured curve of a sensor departs from the ideal curve.

Examples of transducer

1. A mechanical force or displacement being converted into an electrical signal.
2. A thermistor reacts to temperature variations.
3. A photo cell changes in light intensity.
4. Measurement of electrical noise.
5. Telemetering system—when input and output are in electrical form.

1.5 Characteristics of Transducer

Sensitivity. It can be defined as the ratio of the incremental output and the incremental input. While defining the sensitivity, we assume that the input-output characteristic of the instrument is approximately linear in that range.

Range. The range of the sensor is the maximum and minimum values of applied parameters that can be measured.
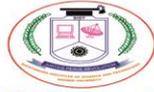
Precision. The concept of precision refers to the degree of reproducibility of a measurement. In other words, if exactly the same value were measured a number of times, an ideal sensor would output exactly the same value every time. But real sensors output a range of values distributed in some manner relative to the actual correct value.

Resolution. The smallest difference between measured values that can be discriminated. For example, it corresponds to the last stable figure on a digital display. This specification is the smallest detectable incremental change of the input parameter that can be detected in the output signal. Resolution can be expressed either as a proportion of the reading (or the full-scale reading) or in absolute terms.

Accuracy. The accuracy of the sensor is the maximum difference that will exist between the actual value and the indicated value at the output of the sensor. Again, the accuracy can be expressed either as a percentage of full scale or in absolute terms.

Linearity. The linearity of the transducer is an expression of the extent to which the actual measured curve of a sensor departs from the ideal curve.

1.6 Transducer Types Characteristics

The characteristics of a transducer are given below that are determined by examining the o/p response of a transducer to a variety of i/p signals. Test conditions create definite operating conditions as closely as possible. The methods of computational and standard statistical can be applied to the test data.

The characteristics of the transducer play a key role while selecting the appropriate transducer, especially for a specific design. So knowing its characteristics is essential for suitable selection. So transducer characteristics are categorized into two types like static and dynamic.

- Precision
- Resolution

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

- Sensitivity

- Drift

- Linearity

- Conformance

- Span

- Hysteresis

- Distortion

- Noise

- Linearity

- Sensitivity

- Resolution

- Threshold

- Span & Range

- Accuracy

- Stability

- Drift

- Repeatability

- Responsiveness

- Threshold

- Input & O/P Impedances

1.6.1 Static Characteristics

The transducer's static characteristics are a set of act criteria that are recognized throughout static calibration which means the explanation of the value of measurement through fundamentally maintaining the calculated quantities because constant values change very slowly.
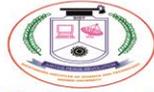
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

For instruments, the set of criteria can be defined to calculate the quantities which are gradually changing with time otherwise mostly constant that does not differ through time is known as static characteristics. The characteristics include the following.

1.6.2 Dynamic Characteristics

The transducer's dynamic characteristics relay toward its performance once the measured capacity is a function of time which changes quickly with respect to time. Once these characteristics rely on the transducer's performance, then the measured quantity is basically stable.

So these characteristics rely on dynamic inputs because they are reliant on their own parameters & the character of the input signal. The dynamic characteristics of the transducer include the following.

- Fidelity
- Speed of Response
- Bandwidth
- Dynamic Error

In general, both the characteristics of a transducer like static & dynamic will verify its performance & specify how efficiently it can recognize preferred input signals as well as refuse unnecessary inputs.

1.7 Transducer Types Applications

The applications of transducer types are discussed below.

- The transducer types are used in electromagnetic applications like antennas, magnetic cartridges, hall-effect sensors, disk read & writes heads.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

- The transducer types are used in electromechanical applications like accelerometers, LVDT, galvanometers, pressure sensors, load cells, MEMS, potentiometers, airflow sensors, linear & rotary motors.

- The transducer types are used in electrochemical applications like oxygen sensors, hydrogen sensors, pH meters,

- The transducer types are used in electroacoustic applications like speakers, piezoelectric crystals, microphones, ultrasonic transceivers, sonar, etc

- The transducer types are used in photoelectric applications like LED, photodiodes, laser diodes, photoelectric cells, LDRs, fluorescent, incandescent lamps, and phototransistor

- The transducer types are used in thermoelectric applications like thermistors, thermocouples, Resistance Temperature Detectors (RTD)

- The transducer types are used in radio acoustic applications like Geiger-Muller Tube, radio transmitters & receivers.

2. ROLE OF SENSORS

Different types of applications require different types of sensors to collect data from the environment. This article takes a look at some common IoT sensors. In an Internet of Things (IoT) ecosystem, two things are very important: the Internet and physical devices like sensors and actuators. As shown in Fig. 1, the bottom layer of the IoT system consists of sensor connectivity and network to collect information. This layer is an essential part of the IoT system and has network connectivity to the next layer, which is the gateway and network layer.
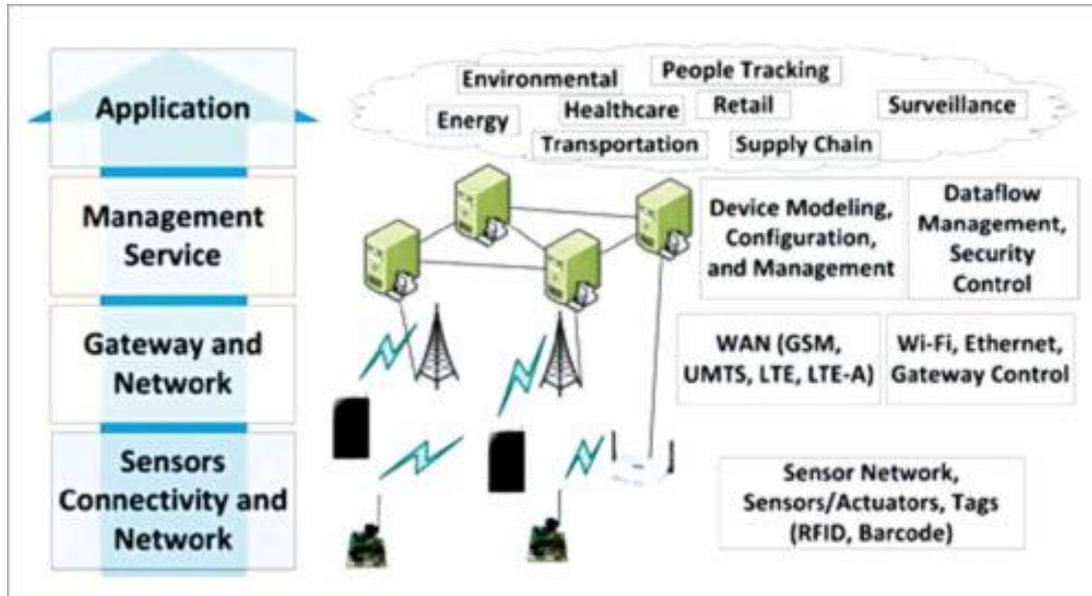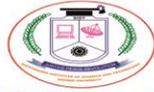
Fig. 2 IoT Architecture Layers

The main purpose of sensors is to collect data from the surrounding environment. Sensors, or 'things' of the IoT system, form the front end. These are connected directly or indirectly to IoT networks after signal conversion and processing. But all sensors are not the same and different IoT applications require different types of sensors. For instance, digital sensors are straightforward and easy to interface with a microcontroller using Serial Peripheral Interface (SPI) bus. But for analogue sensors, either analogue-to-digital converter (ADC) or Sigma-Delta modulator is used to convert the data into SPI output.

2.1 Types of sensors

All the parameters i.e. the Sensors (which give inputs to the Computers), the Computers (the brains of the system) and the mechanics (the outputs of the system like engines and motors) are equally important in building a successful automated system. Sensor as an input device which provides an output (signal) with respect to a specific physical quantity (input). Sensor means that it is part of a bigger system which provides input to a main control system (like a Processor or a Microcontroller).

| S.No	Sensor	Applications	Technology
1.	Inertial sensors	Industrial machinery, automotive, human activity	MEMS and Gyroscope
2.	Speed Measuring Sensor	Industrial machinery, automotive, human activity	Magnetic, light
3.	Proximity sensor	Industrial machinery, automotive, human activity	Capacitive, Inductive, Magnetic, Light, Ultrasound
4.	Occupancy sensor	Home/office monitoring	PassiveIR, Ultrasound most common
5.	Temperature/humidity sensor	Home/office HVAC control, automotive, industrial	Solid state, thermocouple
6.	Light sensor	Home/office/industrial lighting control	Solid state, photocell, Photo resistor, photodiode
7.	Power (current) sensor	Home/office/industrial powermonitoring/control Technology	Coil (Faraday's law), Hall effect
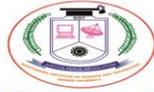
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

8.	Air/fluid pressure sensor	Industrial monitoring/control, automotive, agriculture	Capacitive, Resistive
9.	Acoustic sensor	Industrial monitoring/control, human interface	Diaphragm condenser
10.	Strain sensor	Industrial monitoring/control, civil infrastructure	Resistive thin films

In the first classification of the sensors, they are divided in to Active and Passive. Active Sensors are those which require an external excitation signal or a power signal. Passive Sensors, on the other hand, do not require any external power signal and directly generates output response. The other type of classification is based on the means of detection used in the sensor. Some of the means of detection are Electric, Biological, Chemical, Radioactive etc.

The next classification is based on conversion phenomenon i.e. the input and the output. Some of the common conversion phenomena are Photoelectric, Thermoelectric, Electrochemical, Electromagnetic, Thermo-optic, etc. The final classification of the sensors are Analog and Digital Sensors. Analog Sensors produce an analog output i.e. a continuous output signal with respect to the quantity being measured.

Digital Sensors, in contrast to Analog Sensors, work with discrete or digital data. The data in digital sensors, which is used for conversion and transmission, is digital in nature.
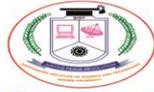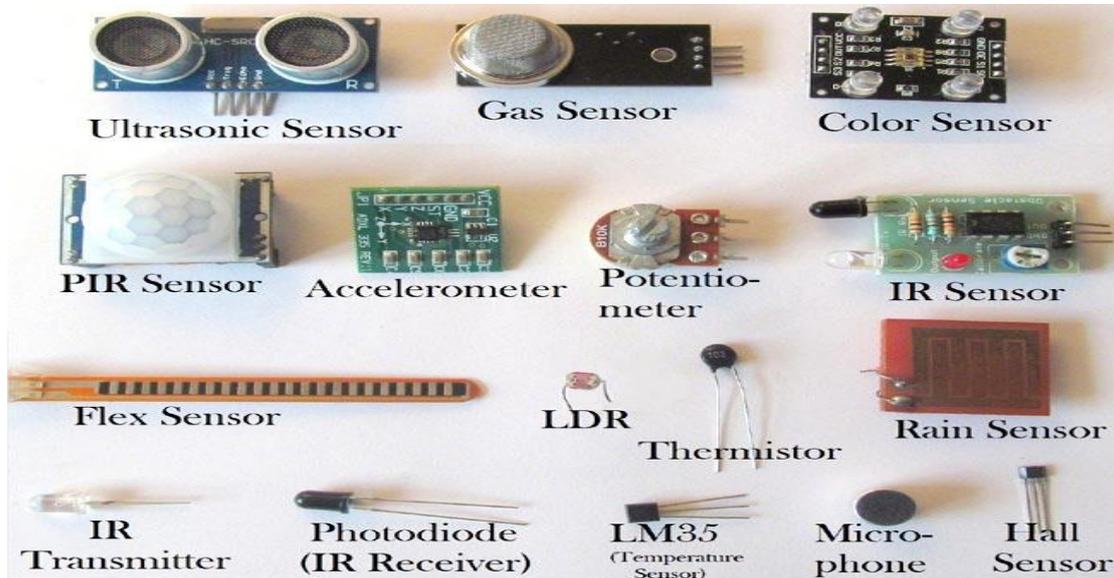
**Fig 3.Examples of Sensors**

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.**
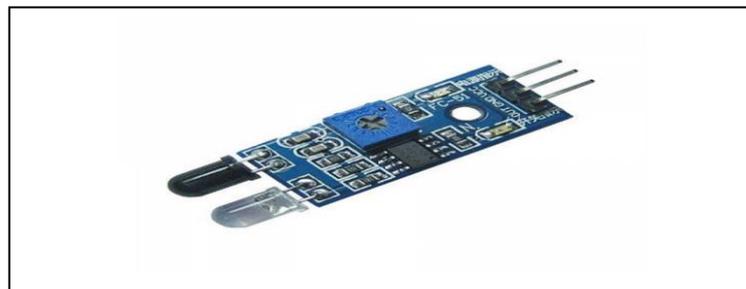
Fig 4. LED sensor

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

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. Check here the working of Photodiode with IR sensor.
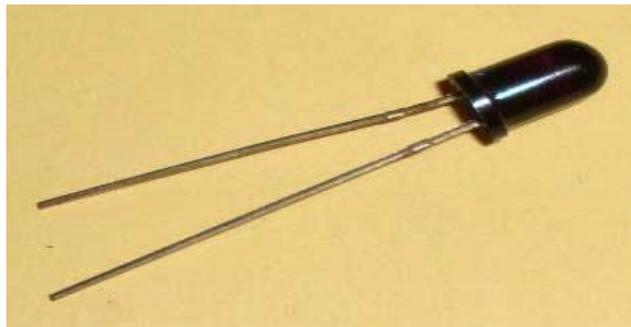
Fig. 5 Photo diode

3.Proximity Sensor

A Proximity Sensor is a non-contact type sensor that detects the presence of an object. Proximity Sensors can be implemented using different techniques like Optical (like Infrared or Laser), Ultrasonic, Hall Effect, Capacitive, etc.
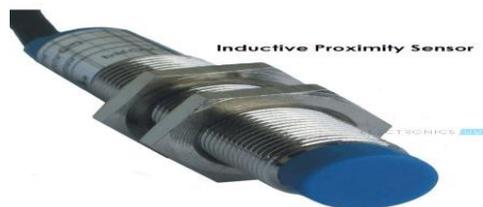

Fig .6 Proximity sensor
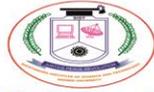
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

Some of the applications of Proximity Sensors are Mobile Phones, Cars (Parking Sensors), industries (object alignment), **Ground Proximity in Aircrafts, etc. Proximity Sensor in Reverse Parking is implemented in this Project: Reverse Parking Sensor Circuit.**

4.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.
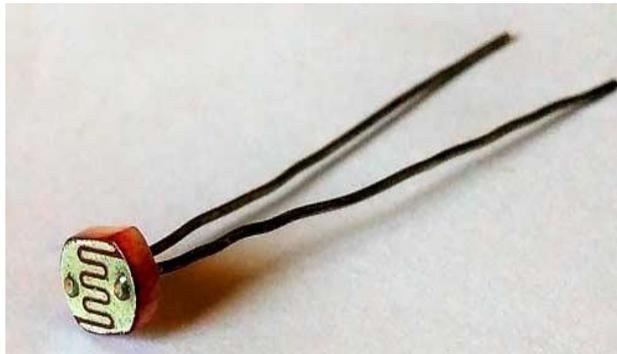

Fig. 7 LDR

With a proper circuitry using a transistor it can be used to detect the availability of light. A voltage divider biased transistor with R2 (resistor between base and emitter) replaced with an LDR can work as a light detector.

5.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

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

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. Here is one simple application for Thermistor to control the DC fan automatically.

Fig. 8 Thermistor

6.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. 9 Thermo couple

7.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 ($R=\rho l/a$). The same principle can be used

![Sathyabama Institute of Science and Technology logo]

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

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 .10 Strain Guage

8.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. Here is the project on how to measure weight using Load cell.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

Fig 11.Load Cell

9.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 12.Potentiometer

10.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

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in
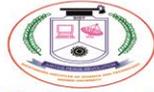
pass through it only a few light rays get detected. Further, these rays are encoded into a digital data (in terms of binary) which represents the specific position.

Fig 13.Encoder

11 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.


Fig 14.Hall sensor

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. Here is one practical application of Hall Effect sensor in Door Alarm.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in
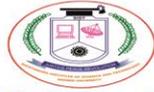
12. Flex Sensor

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. 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 15. Flex sensor

13.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 16.Microphone

14.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 17.Utrasonic sensor

There are many applications with the ultrasonic sensor. We can make use of it avoid obstacles for the automated cars, moving robots etc. The same principle will be used in the RADAR for

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

detecting the intruder missiles and airplanes. A mosquito can sense the ultrasonic sounds. So, ultrasonic waves can be used as mosquito repellent.

15.Touch Sensor

In this generation, we can say that almost all are using smartphones which have widescreen that too a screen which can sense our touch. So, let's see how this touchscreen works. Basically, 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 touch screen* 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.
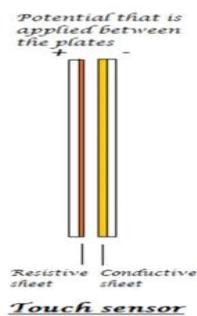
**Fig 18.Touch sensor**

16.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. Here is the complete working or PIR sensor.
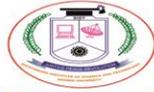
Fig 19.PIR Sensor

17.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.
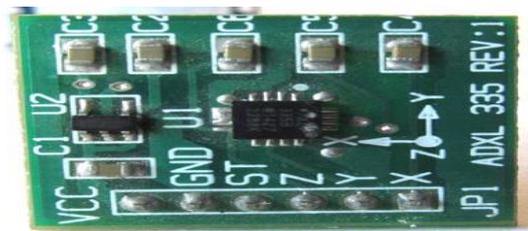
Fig 20.Accelerometer

These are commonly seen in mobiles and laptops in order to avoid breakage of processors leads. When the device falls the accelerometer detects the falling condition and does respective action based on the software.

18.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
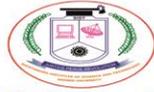
how this sensor works. Underneath a metal sheet there exists a sensing element which is connected to the terminals where a current is applied to it. When the gas particles hit the sensing element, it leads to a chemical reaction such that the resistance of the elements varies and current through it also alters which finally can detect the gas.


Fig 21.Gas Sensor

So finally, we can conclude that sensors are not only used to make our work simple to measure the physical quantities, making the devices automated but also used to help living beings with disasters.

19. Resistive Sensors

Resistive sensors, such as the potentiometer, have three terminals: power input, grounding terminal, and variable voltage output. These mechanical devices have varied resistance that can be changed through movable contact with its fixed resistor. Output from the sensor varies depending on whether the movable contact is near the resistor's supple end or ground end. Thermistors are also variable resistors, although the resistance of the sensor varies with temperature

Fig 22 Resistive Sensors

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in
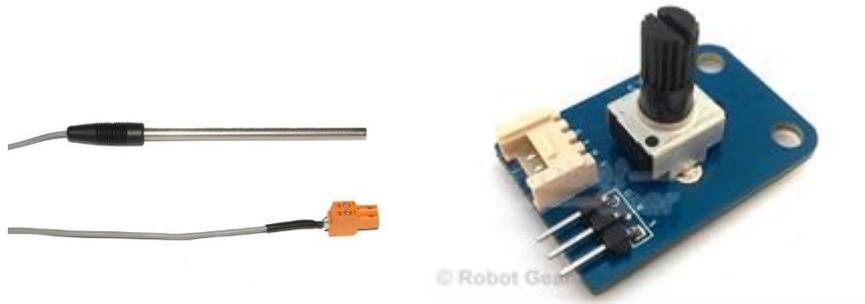
20.Voltage generating sensors

Voltage-generating sensors, such as piezo electrics, generate electricity by pressure with types of crystals like quartz. As the crystal flexes or vibrates, AC voltage is produced. Knock sensors utilize this technology by sending a signal to an automobile's on-board computer that engine knock is happening. The signal is generated through crystal vibration within the sensor, which is caused by cylinder block vibration. The computer, in turn, reduces the ignition timing to stop the engine knock.
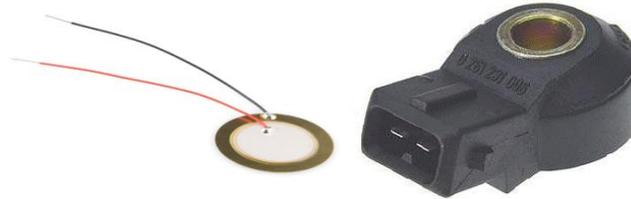
Fig 23.Voltage Generating Sensors

21.Switch Sensors

Switch sensors are composed of a set of contacts that open when close to a magnet. A reed switch is a common example of a switch sensor and is most commonly used as a speed or position sensor. As a speed sensor, a magnet is attached to the speedometer cable and spins along with it. Each time one of the magnet's poles passes the reed switch, it opens and then closes. How fast the magnet passes allows the sensor to read the vehicle's speed.

Fig 24.Switch Sensors
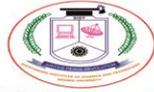
3. WIRELESS SENSOR NETWORK ARCHITECTURE AND ITS APPLICATIONS

Currently, WSN (Wireless Sensor Network) is the most standard services employed in commercial and industrial applications, because of its technical development in a processor, communication, and low-power usage of embedded computing devices. The wireless sensor network architecture is built with nodes that are used to observe the surroundings like temperature, humidity, pressure, position, vibration, sound, etc. These nodes can be used in various real-time applications to perform various tasks like smart detecting, a discovery of neighbor nodes, data processing and storage, data collection, target tracking, monitor and controlling, synchronization, node localization, and effective routing between the base station and nodes. Presently, WSNs are beginning to be organized in an enhanced step. It is not awkward to expect that in 10 to 15 years that the world will be protected with WSNs with entree to them via the Internet. This can be measured as the Internet becoming a physical n/w. This technology is thrilling with infinite potential for many application areas like medical, environmental, transportation, military, entertainment, homeland defense, crisis management, and also smart spaces.

A Wireless Sensor Network is one kind of wireless network that includes a large number of circulating, self-directed, minute, low powered devices named sensor nodes called motes. These networks certainly cover a huge number of spatially distributed, little, battery-operated, embedded devices that are networked to caringly collect, process, and transfer data to the operators, and it has controlled the capabilities of computing & processing. Nodes are tiny computers, which work jointly to form networks.
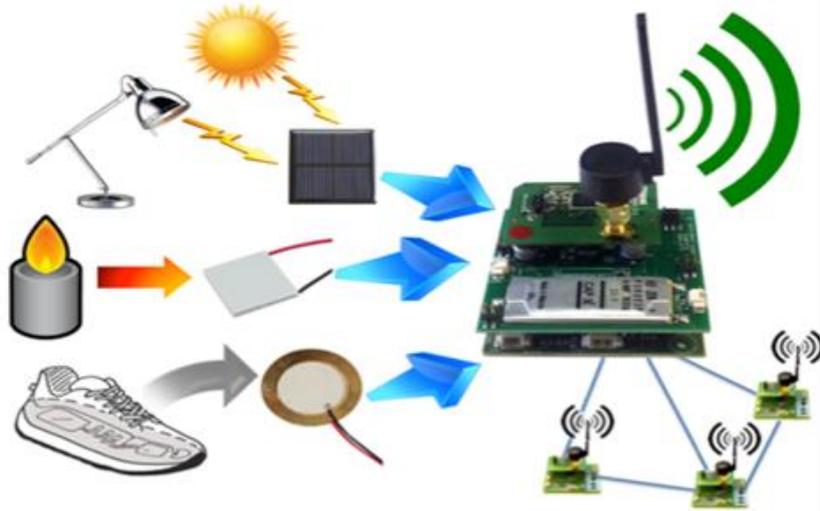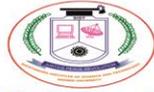
Fig. 25 Wireless Sensor Network

The sensor node is a multi-functional, energy-efficient wireless device. The applications of motes in industrial are widespread. A collection of sensor nodes collects the data from the surroundings to achieve specific application objectives. The communication between motes can be done with each other using transceivers. In a wireless sensor network, the number of motes can be in the order of hundreds/ even thousands. In contrast with sensor n/ws, Ad Hoc networks will have fewer nodes without any structure.

The most common wireless sensor network architecture follows the OSI architecture Model. The architecture of the WSN includes five layers and three cross layers. Mostly in sensor n/w, we require five layers, namely application, transport, n/w, data link & physical layer. The three cross planes are namely power management, mobility management, and task management. These layers of the WSN are used to accomplish the n/w and make the sensors work together in order to raise the complete efficiency of the network. The architecture used in WSN is sensor network architecture. This kind of architecture is applicable in different places such as hospitals, schools, roads, buildings as well as it is used in different applications such as security management, disaster management & crisis management, etc. There are two types of architectures used in wireless sensor networks which include the following. There are 2 types of wireless sensor architectures: Layered Network Architecture, and Clustered Architecture.

These are explained as following below.

- Layered Network Architecture
- Clustered Network Architecture

Layered Network Architecture

This kind of network uses hundreds of sensor nodes as well as a base station. Here the arrangement of network nodes can be done into concentric layers. It comprises five layers as well as 3 cross layers which include the following.

The five layers in the architecture are:

- Application Layer
- Transport Layer
- Network Layer
- Data Link Layer
- Physical Layer

The three cross layers include the following:

- Power Management Plane
- Mobility Management Plane
- Task Management Plane

These three cross layers are mainly used for controlling the network as well as to make the sensors function as one in order to enhance the overall network efficiency. The above mentioned five layers of WSN are discussed below.
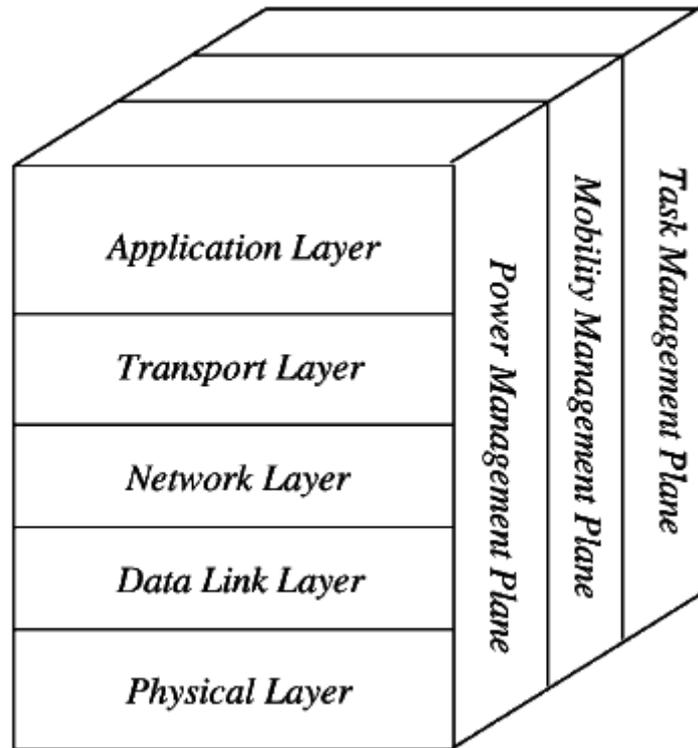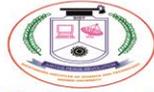
Fig. 27 Wireless Sensor Network Architecture

3.1 Application Layer

The application layer is liable for traffic management and offers software for numerous applications that convert the data in a clear form to find positive information. Sensor networks arranged in numerous applications in different fields such as agricultural, military, environment, medical, etc.

3.2 Transport Layer

The function of the transport layer is to deliver congestion avoidance and reliability where a lot of protocols intended to offer this function are either practical on the upstream. These protocols use dissimilar mechanisms for loss recognition and loss recovery. The transport layer is exactly needed when a system is planned to contact other networks.
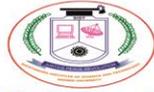
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

Providing a reliable loss recovery is more energy-efficient and that is one of the main reasons why TCP is not fit for WSN. In general, Transport layers can be separated into Packet driven, Event-driven. There are some popular protocols in the transport layer namely STCP (Sensor Transmission Control Protocol), PORT (Price-Oriented Reliable Transport Protocol and PSFQ (pump slow fetch quick).

3.3 Network Layer

The main function of the network layer is routing, it has a lot of tasks based on the application, but actually, the main tasks are in the power conserving, partial memory, buffers, and sensor don't have a universal ID and have to be self-organized.

The simple idea of the routing protocol is to explain a reliable lane and redundant lanes, according to a convincing scale called a metric, which varies from protocol to protocol. There are a lot of existing protocols for this network layer, they can be separated into; flat routing and hierarchal routing or can be separated into time-driven, query-driven & event-driven.

3.4 Data Link Layer

The data link layer is liable for multiplexing data frame detection, data streams, MAC, & error control, confirm the reliability of point–point (or) point– multipoint.

3.5 Physical Layer

The physical layer provides an edge for transferring a stream of bits above the physical medium. This layer is responsible for the selection of frequency, generation of a carrier frequency, signal detection, Modulation & data encryption. IEEE 802.15.4 is suggested as typical for low rate particular areas & wireless sensor networks with low cost, power consumption, density, the range of communication to improve the battery life. CSMA/CA is used to support star & peer to peer topology. There are several versions of IEEE 802.15.4.V.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in
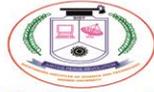
The main benefits of using this kind of architecture in WSN is that every node involves simply in less-distance, low- power transmissions to the neighboring nodes due to which power utilization is low as compared with other kinds of sensor network architecture. This kind of network is scalable as well as includes a high fault tolerance.

3.6 Clustered Network Architecture

In this kind of architecture, separately sensor nodes add into groups known as clusters which depend on the "Leach Protocol" because it uses clusters. The term 'Leach Protocol' stands for "Low Energy Adaptive Clustering Hierarchy". The main properties of this protocol mainly include the following.
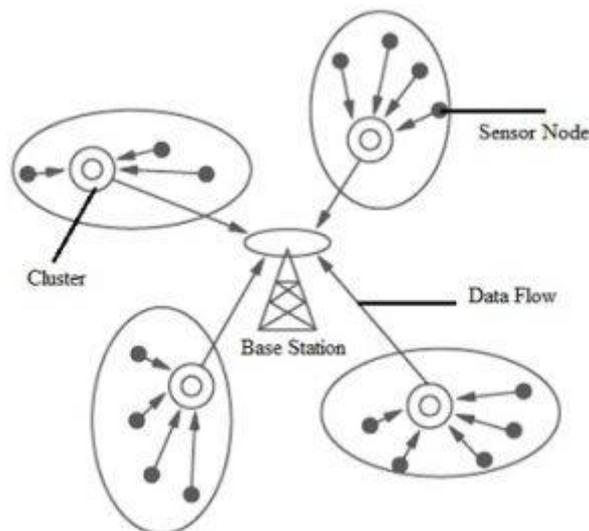
Fig. 28 Clustered Network Architecture

- This is a two-tier hierarchy clustering architecture.
- This distributed algorithm is used to arrange the sensor nodes into groups, known as clusters.
- In every cluster which is formed separately, the head nodes of the cluster will create the TDMA (Time-division multiple access) plans.
- It uses the Data Fusion concept so that it will make the network energy efficient.
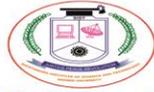
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

This kind of network architecture is extremely used due to the data fusion property. In every cluster, every node can interact through the head of the cluster to get the data. All the clusters will share their collected data toward the base station. The formation of a cluster, as well as its head selection in each cluster, is an independent as well as autonomous distributed method.

3.7 Design Issues of Wireless Sensor Network Architecture

The design issues of wireless sensor network architecture mainly include the following.

- Energy Consumption
- Localization
- Coverage
- Clocks
- Computation
- Cost of Production
- Design of Hardware
- Quality of Service

3.7.1 Energy Consumption

In WSN, power consumption is one of the main issues. As an energy source, the battery is used by equipping with sensor nodes. The sensor network is arranged within dangerous situations so it turns complicated for changing otherwise recharging batteries. The energy consumption mainly depends on the sensor nodes' operations like communication, sensing & data processing. Throughout communication, the energy consumption is very high. So, energy consumption can be avoided at every layer by using efficient routing protocols.

3.7.2 Localization

For the operation of the network, the basic, as well as critical problem, is sensor localization. So sensor nodes are arranged in an ad-hoc manner so they don't know about their location. The
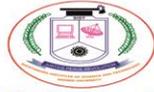
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

difficulty of determining the sensor's physical location once they have been arranged is known as localization. This difficulty can be resolved through GPS, beacon nodes, localization based on proximity.

3.7.3 Coverage

The sensor nodes in the wireless sensor network utilize a coverage algorithm for detecting data as well as transmit them to sink through the routing algorithm. To cover the whole network, the sensor nodes should be chosen. There efficient methods like least and highest exposure path algorithms as well as coverage design protocol are recommended.

3.7.4 Clocks

In WSN, clock synchronization is a serious service. The main function of this synchronization is to offer an ordinary timescale for the nodes of local clocks within sensor networks. These clocks must be synchronized within some applications like monitoring as well as tracking.

3.7.5 Computation

The computation can be defined as the sum of data that continues through each node. The main issue within computation is that it must reduce the utilization of resources. If the life span of the base station is more dangerous, then data processing will be completed at each node before data transmitting toward the base station. At every node, if we have some resources then the whole computation should be done at the sink.

3.7.6 Production Cost

In WSN, the large number of sensor nodes is arranged. So if the single node price is very high then the overall network price will also be high. Ultimately, the price of each sensor node has to be kept less. So the price of every sensor node within the wireless sensor network is a demanding problem.
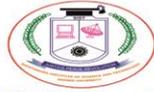
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

3.7.7 Hardware Design

When designing any sensor network's hardware like power control, micro-controller & communication unit must be energy-efficient. Its design can be done in such a way that it uses low-energy.

3.7.8 Quality of Service

The quality of service or QoS is nothing but, the data must be distributed in time. Because some of the real-time sensor-based applications mainly depend on time. So if the data is not distributed on time toward the receiver then the data will turn useless. In WSNs, there are different types of QoS issues like network topology that may modify frequently as well as the accessible state of information used for routing can be imprecise.

3.8 Structure of a Wireless Sensor Network

The structure of WSN mainly comprises various topologies used for radio communications networks like a star, mesh, and hybrid star. These topologies are discussed below in brief.

3.8.1 Star Network

The communication topology like a star network is used wherever only the base station can transmit or receive a message toward remote nodes. There is a number of nodes are available which are not allowed to transmit messages to each other. The benefits of this network mainly comprise simplicity, capable of keeping the power utilization of remote nodes to a minimum.

It also lets communications with less latency among the base station as well as a remote node. The main drawback of this network is that the base station should be in the range of radio for all the separate nodes. It is not robust like other networks because it depends on a single node to handle the network.
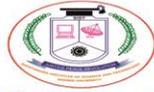
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

3.8.2 Mesh Network

This kind of network permits to the transmission of the data from one node to another within the network that is in the range of radio transmission. If a node needs to transmit a message to another node and that is out of radio communications range, then it can utilize a node like an intermediate to send the message toward the preferred node.

The main benefit of a mesh network is scalability as well as redundancy. When an individual node stops working, a remote node can converse to any other type of node within the range, then forwards the message toward the preferred location. Additionally, the network range is not automatically restricted through the range among single nodes; it can extend simply by adding a number of nodes to the system.

The main drawback of this kind of network is power utilization for the network nodes that execute the communications like multi-hop are usually higher than other nodes that don't have this capacity of limiting the life of battery frequently. Moreover, when the number of communication hops increases toward a destination, then the time taken to send the message will also increase, particularly if the low power process of the nodes is a necessity.

3.8.3 Hybrid Star – Mesh Network

A hybrid among the two networks like star and mesh provides a strong and flexible communications network while maintaining the power consumption of wireless sensor nodes to a minimum. In this kind of network topology, the sensor nodes with less power are not allowed to transmit the messages. This permits to maintenance least power utilization. But, other network nodes are allo wed with the capability of multi-hop by allowing them to transmit messages from one node to another on the network. Usually, the nodes with the multi-hop capacity have high power and are frequently plugged into the mains line. This is the implemented topology through the upcoming standard mesh networking called ZigBee.
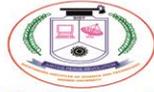
3.9 Structure of a Wireless Sensor Node

The components used to make a wireless sensor node are different units like sensing, processing, transceiver & power. It also includes additional components that depend on an application like a power generator, a location finding system & a mobilizer. Generally, sensing units include two subunits namely ADCs as well as sensors. Here sensors generate analog signals which can be changed to digital signals with the help of ADC, after that it transmits to the processing unit.

Generally, this unit can be associated through a tiny storage unit to handle the actions to make the sensor node work with the other nodes to achieve the allocated sensing tasks. The sensor node can be connected to the network with the help of a transceiver unit. In the sensor node, one of the essential components is a sensor node. The power-units are supported through power scavenge units like solar cells whereas the other subunits depend on the application.

A wireless sensing nodes functional block diagram is shown above. These modules give a versatile platform to deal with the requirements of wide applications. For instance, based on the sensors to be arranged, the replacement of signal conditioning block can be done. This permits to use of different sensors along with the wireless sensing node. Likewise, the radio link can be exchanged for a specified application.

3.10 Characteristics of Wireless Sensor Network

The characteristics of WSN include the following.

- The consumption of Power limits for nodes with batteries
- Capacity to handle node failures
- Some mobility of nodes and Heterogeneity of nodes
- Scalability to a large scale of distribution
- Capability to ensure strict environmental conditions
- Simple to use

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

- Cross-layer design

3.11 Advantages of Wireless Sensor Networks

The advantages of WSN include the following

- Network arrangements can be carried out without immovable infrastructure.
- Apt for the non-reachable places like mountains, over the sea, rural areas, and deep forests.
- Flexible if there is a casual situation when an additional workstation is required.
- Execution pricing is inexpensive.
- It avoids plenty of wiring.
- It might provide accommodations for the new devices at any time.
- It can be opened by using centralized monitoring.

**3.12 Wireless Sensor Network Applications**

Wireless sensor networks may comprise numerous different types of sensors like low sampling rate, seismic, magnetic, thermal, visual, infrared, radar, and acoustic, which are clever to monitor a wide range of ambient situations. Sensor nodes are used for constant sensing, event ID, event detection & local control of actuators. The applications of wireless sensor networks mainly include health, military, environmental, home, & other commercial areas.
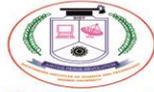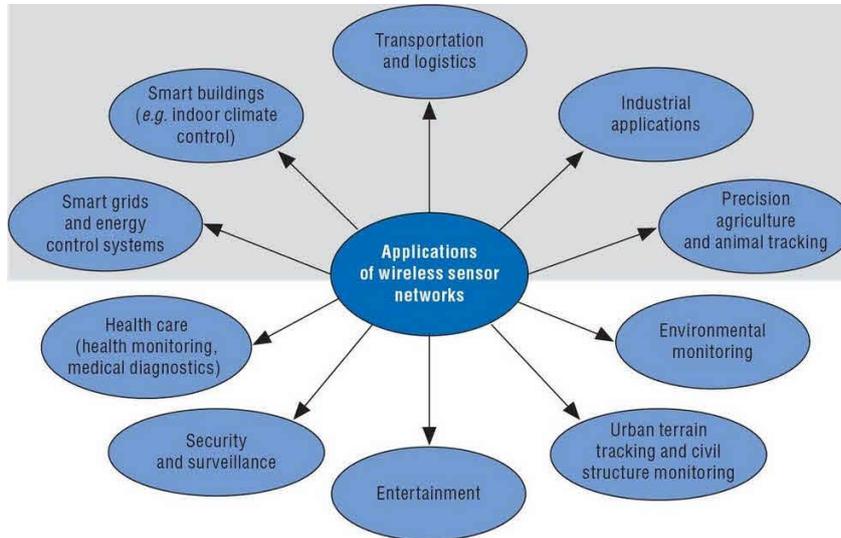
- Military Applications

- Health Applications

- Environmental Applications

- Home Applications

- Commercial Applications

- Area monitoring

- Health care monitoring

- Environmental/Earth sensing

- Air pollution monitoring

- Forest fire detection

- Landslide detection

- Water quality monitoring

- Industrial monitoring

4. REQUIREMENTS OF IOT SENSOR

Smart devices, powered by the hyper-connected Internet of Things (IoT), are becoming ever more prevalent and pervasive in our lives, and the trend will only continue. Every industry is seeking ways to use device-enabled insights to improve the lives of their customers, and the

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

health of machines. With a growing number of devices, the opportunities to use IoT to reshape industries and societies are also increasing.

Yet many organisations are facing challenges in their IoT journey. In reaping the benefits of IoT, enterprises face many challenges, including integration of the IoT infrastructure with existing systems, understanding unfamiliar data formats, and communication protocols as well as implementing new technologies across the IoT continuum. Navigating these challenges requires careful planning, domain knowledge, and rigorous implementation. In order to make the IoT initiatives a su4ccess, there are five essential requirements for processes and practices that organisations should consider:

4.1 Edge computing/analytics

Edge computing, a technology that is expected to grow at a high by 2023, captures and analyses data on distributed devices positioned at the edge of a network. It involves both local sensors that gather data and edge gateways that process it. Edge computing enables data analysis close to where it is captured, resulting in faster response to changing conditions. In fact, an edge-processing system can respond in a few milliseconds, compared with a cloud system, which could take more than 100 milliseconds.

Before considering edge computing, organisations should, firstly, fully assess lifetime device costs at the planning stage, factoring in the operational overhead expenses, such as monitoring, upgrades, and power requirement. Secondly, they need to create policies to secure devices with appropriate firewalls and hardened operating systems, and encrypt data at rest and in transit. Lastly, organisations should assess which analyses are most time-critical for their business and perform them at the edge to allow immediate action.

4.2 Data ingestion and stream processing

Six out of 10 IT executives say collecting, storing, integrating and analysing real-time data from endpoint devices is a key barrier to a successful IoT implementation. Organisations should put processes in place to gather data from multiple devices and sensors, and transform
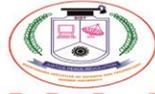
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

it for use by cloud-based analytic platforms. Data ingestion refers to device telemetry data being imported and converted into a format usable by cloud-based IoT services. It helps to normalise the data into a common data model that is easier to analyse by business applications and users. Data ingestion also comes handy when organisations have to ensure that ingested data is stored in compliance with government or industry regulations, such as European Union's General Data Protection Regulation or Personal Data Protection Act in Singapore.

### 4.3 Security and device management

With rapid proliferation of IoT sensors, and growing complexity and volume of data exchanges, it is imperative for organisations to strengthen their adoption and enforcement of highly evolved security practices and procedures. The scale of investments, talent as well as thought leadership around security would need to dramatically increase as IoT implementations grow in scale and start becoming the backbone of day-to-day operations in organisations.

Businesses need to ensure their IoT devices are provisioned securely, communicate efficiently, and can be updated with accelerated and agile approaches. Device management covers the hardware, software, and the processes that ensure devices are properly registered, managed, secured, and upgraded.

Required functions include device configuration, security, command dispatching, operational control, remote monitoring, and troubleshooting. The organisation will need to account for these functions, even if the cloud provider doesn't offer the required device management components. Comprehensive device management enables connected devices to easily and securely communicate with other devices and cloud platforms, while helping the enterprise reliably scale to billions of connected devices and trillions of messages.

4.4 Cold path and advanced analytics

Currently, large-scale processing can include loads greater than 100,000 events per second. With the adoption of cold path processing, large amounts of data are analysed by advanced algorithms after the data is stored on the cloud platform.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

Such analysis can uncover trends or corrective actions needed to improve the business or customer experience. Unlike streaming analytics (hot path) that apply relatively simple rules to data in real time for short-term actions (detecting fraud, security breaches, or critical component failures), cold path processing involves more sophisticated big data analytics, such as machine learning and AI, being applied to provide deeper insights.

To drive the most insights from data, organisations should consider using a complex event processing framework that combines data from multiple sources, such as enterprise applications and IoT devices, to dynamically define and process analytical rules by inferring meaning from complex situations. It is also important to aggregate data before than during analysis to improve processing speed. Usage of data lakes, which store data in their native format, can also help consolidate data and allow easier access. Organisations should also consider creating dedicated data services to make it easier for users to access data on demand.

4.5 Enterprise integration with business systems

IoT insights need to be delivered to enterprise systems and receive reference metadata in order to interpret device data. Integration with business applications and enterprise systems enables the sharing of raw and processed data, as well as analysis-driven insights. With deep enterprise integration, the IoT architecture can deliver benefits such as improved efficiencies, reduced costs, increased sales, heightened customer satisfaction, and the ability to create and lead new markets. To share data and insights, businesses need mechanisms such as application programming interface (API) gateways, service buses and custom connectors.

Every IoT implementation will be distinct, depending on each business's requirements, expected outcomes, levels of IoT and data skills, and technology infrastructure maturity. In all cases, however, these five requirements are essential to ensuring a successful IoT implementation, with minimal cost and delay. Each enterprise must conduct a rigorous needs assessment, and carefully plan its roadmap to deliver a flexible, secure, and scalable IoT solution. To help guide the implementation, organisations should also consider using pre-built

solutions, reference architectures, and blueprints from experienced technology service providers.

5. ACTUATOR

Actuators use energy from a source upon the receipt of a signal so as to bring about a mechanical motion. This blog tells you about how they function and the many types of actuators used today. Actuators are mechanical or electro-mechanical devices that, upon being operated electrically, manually, or by various fluids, allow controlled and sometimes limited movements or positioning. They refer to that component of a machine that helps carry out the moving and controlling of a mechanism or system; take for instance opening a valve. To put it simply, they can be called movers.

Actuators basically need a control signal and a source of energy. Upon receiving a control signal, the actuator uses energy from the source to bring about a mechanical motion. The control system can be a human, a fixed mechanical or electronic system, or even software-based, say a printer driver, or a robot control system. Examples of actuators include electric motors, stepper motors, electroactive polymers, screw jacks, servomechanism, solenoids and hydraulic cylinders.

5.1 Types of Actuators

Actuator types also vary depending on motions, power configurations, styles and sizes depending on the application.

5.1.1 Mechanical actuators

Mechanical actuators create movement by converting one kind of motion, such as rotary motion, into another kind, such as linear motion. Say for instance, a rack and a pinion. Another example is that of a chain block hoisting weight where the mechanical motion of the chain is used to lift a load.The functioning of mechanical actuators relies on the combinations of their structural components, such as gears and rails, or pulleys and chains. High reliability, simplicity

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

of utilisation, easier maintenance and greater precision of positioning are some of the advantages. They can be categorised into hydraulic, pneumatic and electric actuators.

5.1.2 Hydraulic actuators

Hydraulic actuators have a cylinder or fluid motor that uses hydraulic power to generate mechanical motion, which in turn leads to linear, rotatory or oscillatory motion. Given the fact that liquids are nearly impossible to compress, a hydraulic actuator can exert a large force. When the fluid enters the lower chamber of the actuator's hydraulic cylinder, pressure inside increases and exerts a force on the bottom of the piston, also inside the cylinder. The pressure causes the sliding piston to move in a direction opposite to the force caused by the spring in the upper chamber, making the piston move upward and opening the valve. The downside with these actuators is the need for many complementary parts and possibility of fluid leakage.

5.1.3 Pneumatic actuators

Pneumatic actuators convert energy in the form of compressed air into mechanical motion. Here pressurised gas or compressed air enters a chamber thus building up the pressure inside. Once this pressure goes above the required pressure levels in contrast to the atmospheric pressure outside the chamber, it makes the piston or gear move kinetically in a controlled manner, thus leading to a straight or circular mechanical motion. Examples include pneumatic cylinders, air cylinders, and air actuators. Cheaper and often more powerful than other actuators, they can quickly start or stop as no power source has to be stored in reserve for operation. Often used with valves to control the flow of air through the valve, these actuators generate considerable force through relatively small pressure changes. Examples of maker projects using pneumatic actuators include lifting devices and humanoid robots with arms and limbs, typically used for lifting.

5.1.4 Electric Linear actuators

Taking off from the two basic motions of linear and rotary, actuators can be classified into these two categories: linear and rotary. Electric linear actuators take electrical energy and
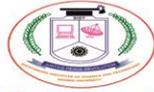
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

turn it into straight line motions, usually for positioning applications, and they have a push and pull function. They convert energy from the power source into linear motion using mechanical transmission, electro-magnetism, or thermal expansion; they are typically used whenever tilting, lifting, pulling and pushing are needed. They are also known for offering precision and smooth motion control; this is why they are used in industrial machinery, in computer peripherals such as disk drives and printers, opening and closing dampers, locking doors and for braking machine motions. They are also used in 3d printers and for controlling valves. Some of them are unpowered and manually operated with a rotating knob or handwheel. Electric linear actuators

5.1.5 Electric Rotary actuators

Consisting of motors and output shaft mechanisms with limited rotary travel, electric rotary actuators convert electrical energy into rotary motion. Used in a wide range of industries where positioning is needed, and driven by various motor types, voice coils, these actuators work as per specifications such as the intended application, drive method, number of positions, output configuration, mounting configuration, physical dimensions and electrical characteristics. A common use is for controlling valves such as ball or butterfly valves. Other applications include automation applications where a gate, door or valve needs controlled movement to certain rotational positions.

5.1.6 Electromechanical actuators

Electromechanical actuators are mechanical actuators where there's an electric motor in place of the control knob or handle. The rotary motion of the motor leads to linear displacement. The inclined plane concept is what drives most electromechanical actuators; the lead screw's threads work like a ramp converting the small rotational force by magnifying it over a long distance, thus allowing a big load to be moved over a small distance. While there are many design variations among electromechanical actuators available today, most have the lead screw and the nut incorporated into the motion. The biggest advantages are their greater accuracy in relation to pneumatics, their longer lifecycle and low maintenance effort required. On the other hand, they do not boast the highest speed.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

5.1.7 Electrohydraulic actuators

Instead of hydraulic systems, electrohydraulic actuators have self-contained actuators functioning solely on electrical power. They are basically used to actuate equipment such as multi-turn valves, or electric-powered construction and excavation equipment. In case of controlling the flow of fluid through a valve, a brake is typically installed above the motor to prevent the fluid pressure from forcing open the valve. The main advantage here is that these actuators help do away with the need for separate hydraulic pumps and tubing, simplifying system architectures and enhancing reliability and safety. Originally developed for the aerospace industry, today they are found in many other industries where hydraulic power is used.

5.1.8 Thermal actuators

A thermal actuator is a non-electric motor that generates linear motion in response to temperature changes. Its main components are a piston and a thermal sensitive material. When there is a rise in temperature, the thermal-sensitive materials begin to expand in response, driving the piston out of the actuator. Similarly, upon detecting a drop in the temperature, the thermal-sensitive materials inside contract, making the piston retract. Thus these actuators can be used for carrying out tasks such as releasing latches, working switches and opening or closing valves. They have many applications, particularly in the aerospace, automotive, agricultural and solar industries.

5.1.9 Magnetic actuators

Magnetic actuators are those that use magnetic effects to produce motion of a part in the actuator. They usually come in the following categories: moving coil actuator, moving magnet actuator, moving iron actuator and electromagnetic actuator.

In case of the first kind (moving coil actuator), a mobile coil driven by a current is placed in a static magnetic field, where it is subject to the Lorentz force. This force is proportional to the applied current.

![Sathyabama Institute of Science and Technology logo]

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

Moving magnet actuators work differently; here mobile permanent magnet is placed between two magnet poles and is switched from one pole to the other using coils. Such actuators can generate high forces but are not easily controlled.

In moving iron actuators, a soft magnetic part placed into a coil system moves in a fashion that keeps the system magnetic energy to a minimum.

Lastly, electromagnetic actuators are the ones comprising electric motors such as Brushless DC motors (BLDC) and stepper motors. These magnetic actuators are used for various purposes such as valve control, pump and compressor actuation, locking mechanisms, aerospace engineering, vibration generation, fast positioning etc. Advantages include reduced system cost, improved robustness, and reduced control complexity.

School of Computing

Department of Computer Science and Engineering

UNIT- II ARDUINO PROGRAMMING FOR IoT BOARDS - SCSA1407

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in
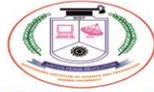
Unit II

Physical device – Arduino Interfaces, Hardware requirement for Arduino, Connecting remotely over the network using VNC, GPIO Basics, Controlling GPIO Outputs Using a Web Interface, – Programming, APIs / Packages- Quark SOC processor, programming, Arduino Boards using GPIO (LED, LCD, Keypad, Motor control and sensor)

1. **PHYSICAL DEVICE- INTRODUCTION TO AUDUINO**

Arduino is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board. Accepts analog and digital signals as input and gives desired output.

ARDUIN0 UN0	
Feature	Value
OperatingVoltage	5V
ClockSpeed	16MHz
Digital I/O	14
AnalogInput	6
PWM	6
UART	1
Interface	USB via ATMega16U2

BOARD DETAILS:

- **Power Supply:**
- **USB or power barrel jack**
- **Voltage Regulator**
- **LED Power Indicator**
- **Tx-Rx LED Indicator**
- **Output power,**
- **Ground**
- **Analog Input Pins**
- **Digital I/O Pin**

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SET UP:

- Power the board by connecting it to a PC via USB cable

- Launch the Arduino IDE

- Set the board type and the port for the board

- TOOLS -> BOARD -> select your board

- TOOLS -> PORT -> select your port

TYPES:

1. Arduino Uno (R3)

2. LilyPad Arduino

3. RedBoard

4. Arduino Mega (R3)

5. Arduino Leonardo
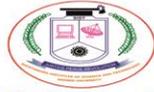
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in
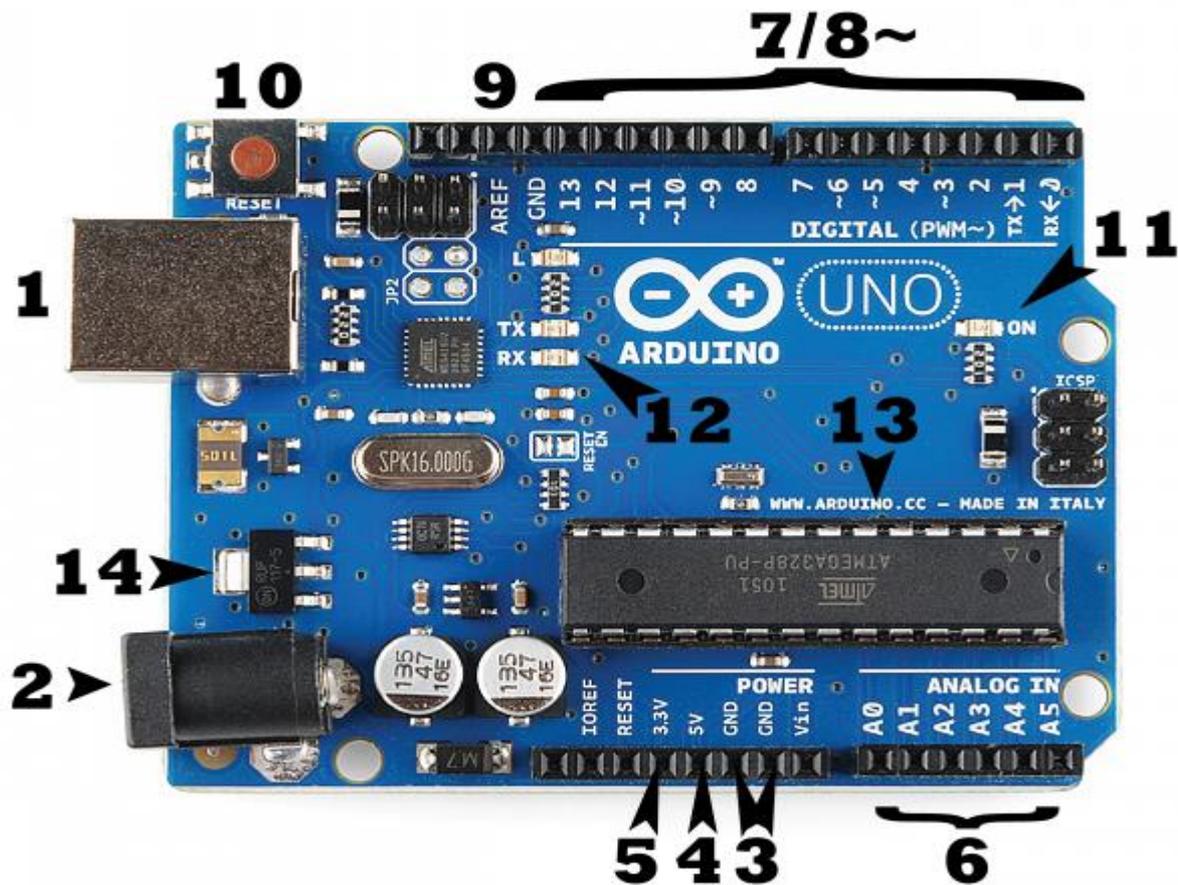
Fig. 1 Arduino Board

Power (USB / Barrel Jack):

Every Arduino board needs a way to be connected to a power source. The Arduino UNO can be powered from a USB cable coming from your computer or a wall power supply (like this) that is terminated in a barrel jack. In the picture above the USB connection is labeled (1) and the barrel jack is labeled (2). The USB connection is also how you will load code onto your Arduino board.
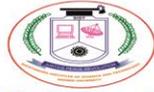
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

NOTE: Do NOT use a power supply greater than 20 Volts as you will overpower (and thereby destroy) Arduino. The recommended voltage for most Arduino models is between 6 and 12 Volts.

Pins (5V, 3.3V, GND, Analog, Digital, PWM, AREF):

The pins on your Arduino are the places where you connect wires to construct a circuit (probably in conjunction with a breadboard and some wire. They usually have black plastic 'headers' that allow you to just plug a wire right into the board. The Arduino has several different kinds of pins, each of which is labeled on the board and used for different functions.

GND (3): Short for 'Ground'. There are several GND pins on the Arduino, any of which can be used to ground your circuit.

5V (4) & 3.3V (5): As you might guess, the 5V pin supplies 5 volts of power, and the 3.3V pin supplies 3.3 volts of power. Most of the simple components used with the Arduino run happily off of 5 or 3.3 volts.

Analog (6): The area of pins under the 'Analog In' label (A0 through A5 on the UNO) are Analog In pins. These pins can read the signal from an analog sensor (like a temperature sensor) and convert it into a digital value that we can read.

Digital (7): Across from the analog pins are the digital pins (0 through 13 on the UNO). These pins can be used for both digital input (like telling if a button is pushed) and digital output (like powering an LED).

PWM (8): You may have noticed the tilde (~) next to some of the digital pins (3, 5, 6, 9, 10, and 11 on the UNO). These pins act as normal digital pins, but can also be used for something called Pulse-Width Modulation (PWM). We have a tutorial on PWM, but for now, think of these pins as being able to simulate analog output (like fading an LED in and out).
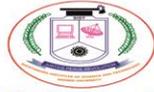
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

AREF (9): Stands for Analog Reference. Most of the time you can leave this pin alone. It is sometimes used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

Reset Button

Just like the original Nintendo, the Arduino has a reset button **(10)**. Pushing it will temporarily connect the reset pin to ground and restart any code that is loaded on the Arduino. This can be very useful if your code doesn't repeat, but you want to test it multiple times. Unlike the original Nintendo however, blowing on the Arduino doesn't usually fix any problems.

Power LED Indicator

Just beneath and to the right of the word "UNO" on your circuit board, there's a tiny LED next to the word 'ON' **(11)**. This LED should light up whenever you plug your Arduino into a power source. If this light doesn't turn on, there's a good chance something is wrong. Time to re-check your circuit!

TX RX LEDs

TX is short for transmit, RX is short for receive. These markings appear quite a bit in electronics to indicate the pins responsible for serial communication. In our case, there are two places on the Arduino UNO where TX and RX appear – once by digital pins 0 and 1, and a second time next to the TX and RX indicator LEDs (12). These LEDs will give us some nice visual indications whenever our Arduino is receiving or transmitting data (like when we're loading a new program onto the board).

Main IC

The black thing with all the metal legs is an IC, or Integrated Circuit **(13)**. Think of it as the brains of our Arduino. The main IC on the Arduino is slightly different from board type to board type, but is usually from the ATmega line of IC's from the ATMEL company. This can be important, as you may need to know the IC type (along with your board type) before loading up a new program from the Arduino software. This information can usually be found in writing

on the top side of the IC. If you want to know more about the difference between various IC's, reading the datasheets is often a good idea.p

Voltage Regulator

The voltage regulator **(14)** is not actually something you can (or should) interact with on the Arduino. But it is potentially useful to know that it is there and what it's for. The voltage regulator does exactly what it says – it controls the amount of voltage that is let into the Arduino board. Think of it as a kind of gatekeeper; it will turn away an extra voltage that might harm the circuit. Of course, it has its limits, so don't hook up your Arduino to anything greater than 20 volts.

ARDINO IDE OVERVIEW:

Program coded in Arduino IDE is called a SKETCH

1. To create a new sketchFile -> New

   To open an existing sketch File -> open ->

   There are some basic ready-to-use sketches available in the EXAMPLES section

   File -> Examples -> select any program
2. Verify: Checks the code for compilation errors
3. Upload: Uploads the final code to the controller board
4. New: Creates a new blank sketch with basic structure
5. Open: Opens an existing sketch
6. Save: Saves the current sketch

Fig. 2 Compilation and Execution

Serial Monitor: Opens the serial console

☐ All the data printed to the console are displayed here

Fig. 3 Structure of SKETCH

A sketch can be divided into two parts:

☐ Setup ()

☐ Loop()

☐ The function setup() is the point where the code starts, just like the main() function in C and C++

☐ I/O Variables, pin modes are initialized in the Setup() function ☐ Loop() function, as the name suggests, iterates the specified task in the program

DATA TYPES:

Void ,Long, Int ,Char ,Boolean, Unsigned char ,Byte, Unsigned int, Word ,Unsigned long ,Float,   Double,   Array   ,String-char   array,   String-object,   Short

Arduino Function libraries
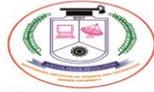
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

Input/Output Functions:

The arduino pins can be configured to act as input or output pins using the pinMode() function

Void setup ()

{

pinMode (pin , mode);

}

Pin- pin number on the Arduino board Mode- INPUT/OUTPUT

digitalWrite() : Writes a HIGH or LOW value to a digital pin

analogRead() : Reads from the analog input pin i.e., voltage applied across the pin

Character functions such as isdigit(), isalpha(), isalnum(), isxdigit(), islower(), isupper(), isspace() return 1(true) or 0(false)

Delay() function is one of the most common time manipulation function used to provide a delay of specified time. It accepts integer value (time in miliseconds)

EXAMPLE BLINKING LED:

Requirement:

☐ Arduino controller board, USB connector, Bread board, LED, 1.4Kohm resistor, connecting wires, Arduino IDE

☐ Connect the LED to the Arduino using the Bread board and the connecting wires

☐ Connect the Arduino board to the PC using the USB connector

☐ Select the board type and port ☐ Write the sketch in the editor, verify and upload

Connect the positive terminal of the LED to digital pin 12 and the negative terminal to the ground pin (GND) of Arduino Board


```
void setup()
{
 pinMode(12, OUTPUT); // set the pin mode
} void loop()
{
```

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

```
digitalWrite(12, HIGH); // Turn on the LED delay(1000);
digitalWrite(12, LOW); //Turn of the LED delay(1000);
}
```

Set the pin mode as output which is connected to the led, pin 12 in this case.

Use digitalWrite() function to set the output as HIGH and LOW

Delay() function is used to specify the delay between HIGH-LOW transition of the output

Connect he board to the PC

□ Set the port and board type

□ Verify the code and upload,

notice the TX – RX led in the board starts flashing as the code is uploaded.

2. RASPBERRY PI:

Raspberry Pi is a credit card sized micro processor available in different models with different processing speed starting from 700 MHz. Whether you have a model B or model B+, or the very old version, the installation process remains the same. People who have checked out the official Raspberry Pi website, But using the Pi is very easy and from being a beginner, one will turn pro in no time. So, it's better to go with the more powerful and more efficient OS, the Raspbian. The main reason why Raspbian is extremely popular is that it has thousands of pre built libraries to perform many tasks and optimize the OS. This forms a huge advantage while building applications.
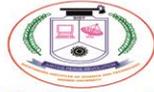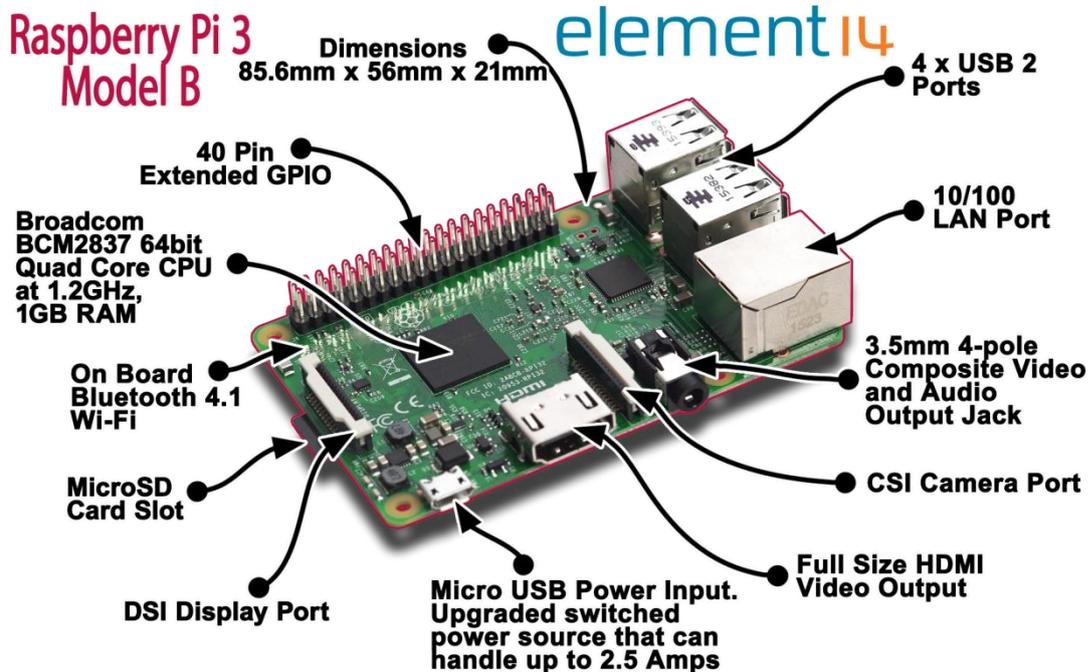
Fig. 4 Raspberry Pi Elements

As for the specifications, the Raspberry Pi is a credit card-sized computer powered by the Broadcom BCM2835 system-on-a-chip (SoC). This SoC includes a 32-bit ARM1176JZFS processor, clocked at 700MHz, and a Videocore IV GPU. It also has 256MB of RAM in a POP package above the SoC. The Raspberry Pi is powered by a 5V micro USB AC charger or at least 4 AA batteries (with a bit of hacking). While the ARM CPU delivers real-world performance similar to that of a 300MHz Pentium 2, the Broadcom GPU is a very capable graphics core capable of hardware decoding several high definition video formats. The Raspberry Pi model available for purchase at the time of writing — the Model B — features HDMI and composite video outputs, two USB 2.0 ports, a 10/100 Ethernet port, SD card slot, GPIO (General Purpose I/O Expansion Board) connector, and analog audio output (3.5mm headphone jack). The less expensive Model A strips out the Ethernet port and one of the USB ports but otherwise has the same hardware. Raspberry Pi Basics: installing Raspbian and getting it up and running.
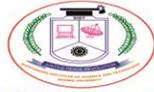
1 Downloading Raspbian and Image writer.

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

2 Writing the image

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

After that, click on the "Write" button in the bottom. As an example, see the image below, where the SD card (or micro SD) drive is represented by the letter "G:\"
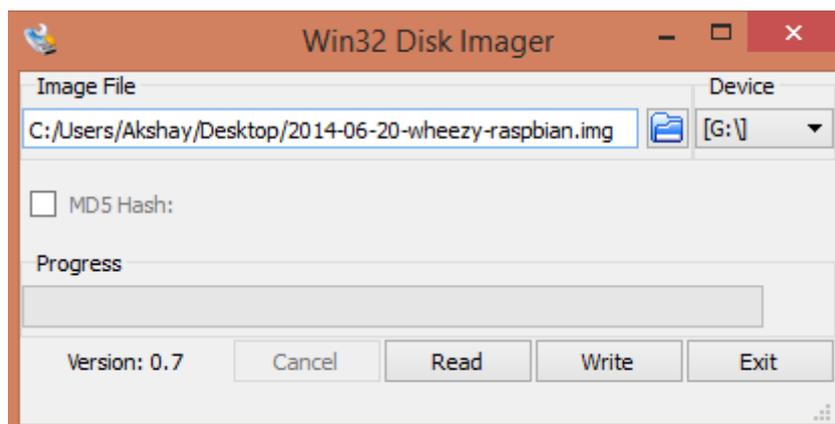
Fig. 5 OS Installation

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

3 Setting up the Pi
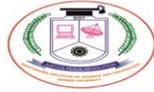
![Sathyabama Institute of Science and Technology logo]
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

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

login: pi

password: raspberry

When the Pi has been booted for the first time, a configuration screen called the "Setup Options" should appear and it will look like the image below.
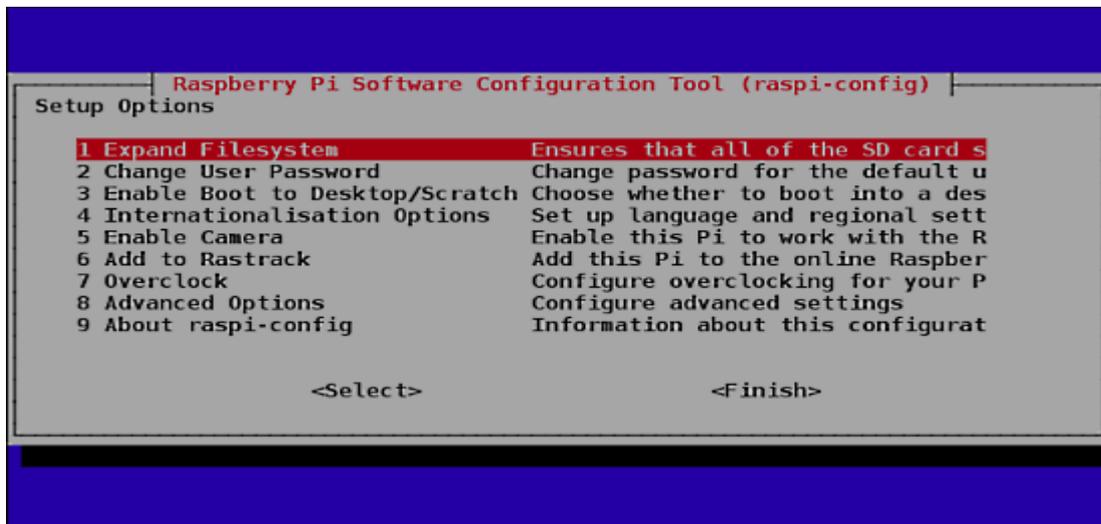
Fig. 6 Raspberry Configuration

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

sudo raspi-config

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

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

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in
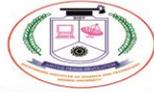
- The first thing to do:

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

- The second thing to do:

  Select the third option in the list of the setup options window, that is select the "**Enable Boot To Desktop/Scratch" option** and hit the enter key. It will take you to another window called the "**choose boot option**" window that looks like the image below.
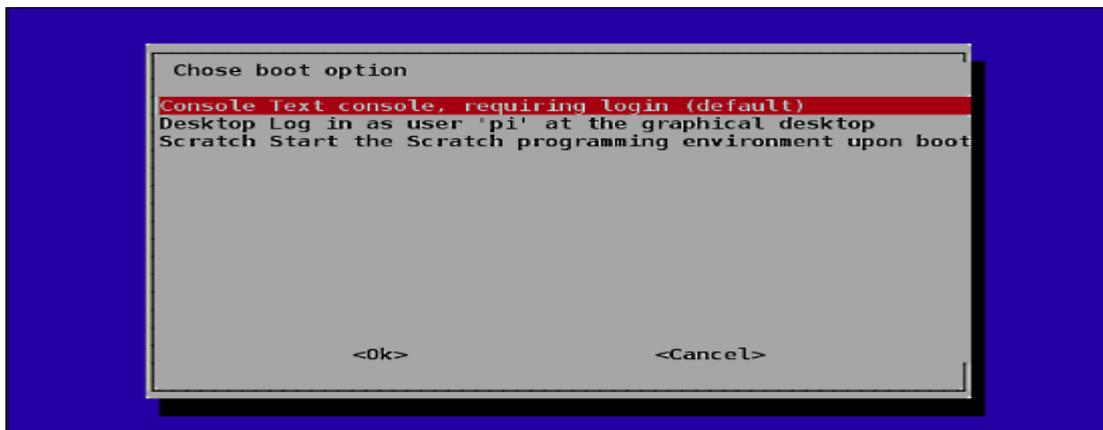
Fig. 7 Boot Options

In the "choose **boot option window**", select the second option, that is, "**Desktop Log in as user 'pi' at the graphical desktop**" and hit the enter button. Once done you will be taken back to the "**Setup Options**" page, if not select the "OK" button at the bottom of this window and you will be taken back to the previous window. We do this because we want to boot into the desktop environment which we are familiar with. If we don't do this step then the Raspberry Pi boots into a terminal each time with no GUI options. Once, both the steps are done, select the **"finish"** button at the bottom of the page and it should reboot automatically. If it doesn't, then use the following command in the terminal to reboot.
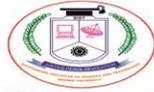
sudo reboot

Updating the firmware

After the reboot from the previous step, if everything went right, then you will end up on the desktop which looks like the image below.
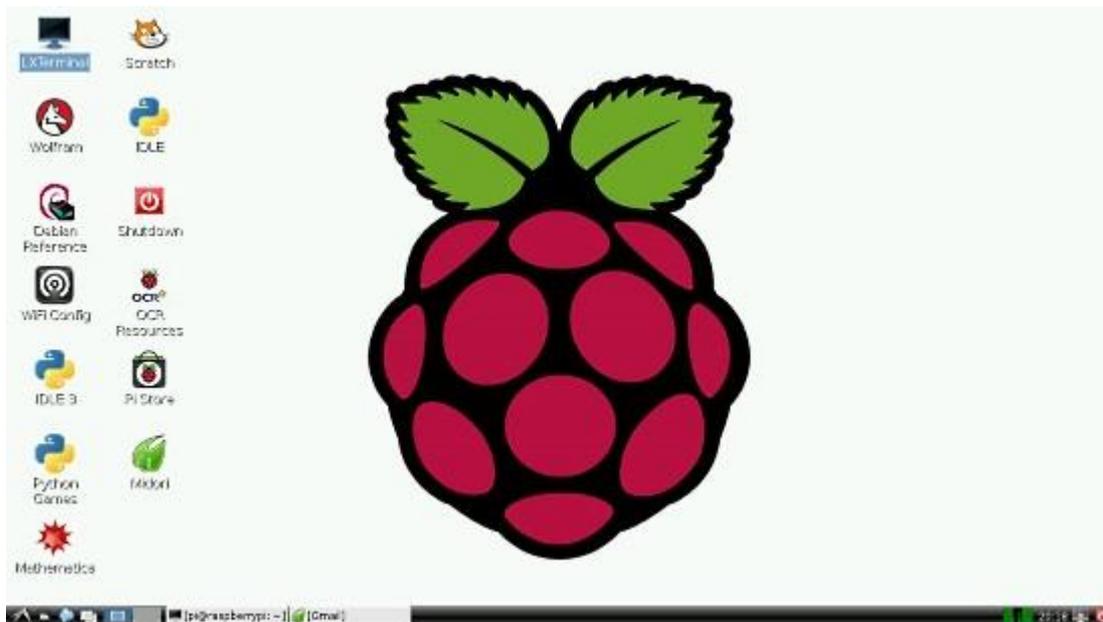
Fig. 8 Raspberry Desktop

Once you are on the desktop, open a terminal and enter the following command to update the firmware of the Pi.

sudo rpi-update

Updating the firmware is necessary because certain models of the Pi might not have all the required dependencies to run smoothly or it may have some bug. The latest firmware might have the fix to those bugs, thus its very important to update it in the beginning itself.

5 Conclusion

So, we have covered the steps to get the Pi up and running. This method works on all the different models of Raspberry Pi (model A, B, B+ and also RPi 2) as Raspbain was made to be supported on all models. However, while installing other software or libraries, the procedure

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

might change a bit while installing depending on the model of the Pi or the version of Raspbian itself. The concept of Raspberry is to keep trying till you get the result or build that you want. This might involve a lot of trial and error but spending the time will be worth it. The actual usage doesn't end here. This is just the beginning. It is up to you to go ahead to build something amazing out of it.
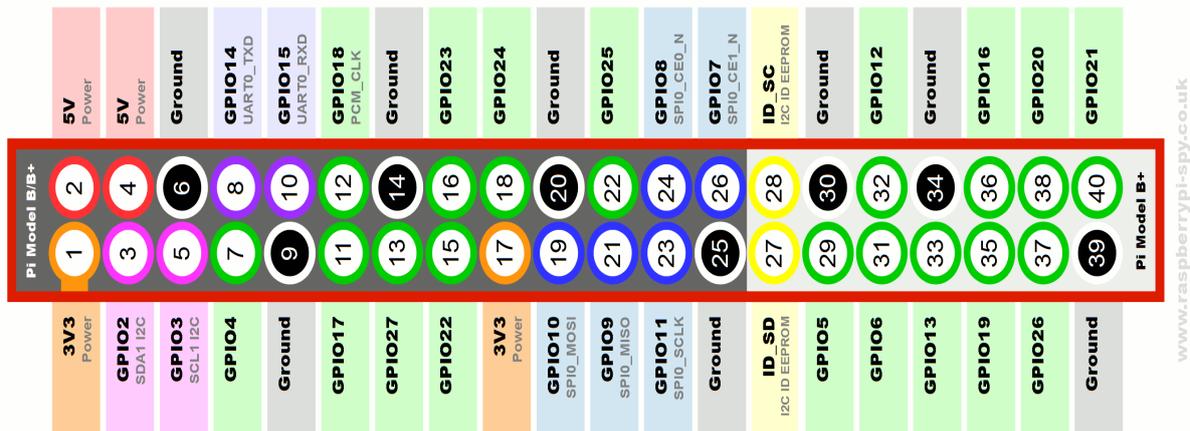
Fig. 9 GPIO Pins

GPIO:

Act as both digital output and digital input.

Output: turn a GPIO pin high or low.

Input: detect a GPIO pin high or low

Installing GPIO library:

Open terminal

Enter the command "sudoapt-**get install python-dev"** to install python development

Enter the command **"sudoapt-get install python-rpi.gpio"** to install GPIO library.

Basic python coding:
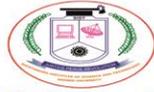
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

Open terminal enter the command

sudo nano filename.py

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

 Ctrl+O : Writes the code to the file

 Ctrl+X : Exits the editor

Blinking LED Code:

```
import RPi.GPIO as GPIO #GPIO library import time

GPIO.setmode(GPIO.BOARD) # Set the type of board for pin numbering

GPIO.setup(11, GPIO.OUT) # Set GPIO pin 11as output pin

for i in range (0,5): GPIO.output(11,True) # Turn on GPIO pin 11

time.sleep(1)

GPIO.output(11,False)

 time.sleep(2)

GPIO.output(11,True)

GPIO.cleanup()
```

Power Pins

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

 Basic GPIO

The header provides 17 Pins that can be configured as inputs and outputs. By default they are all configured as inputs except GPIO 14 & 15.
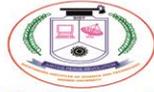
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

In order to use these pins you must tell the system whether they are inputs or outputs. This can be achieved a number of ways and it depends on how you intend to control them. I intend on using Python.

SDA & SCL: The 'DA' in SDA stands for data, the 'CL' in SCL stands for clock; the S stands for serial. You can do more reading about the significance of the clock line for various types of computer bus, You will probably find I^2C devices that come with their own userspace drivers and the linux kernel includes some as well. Most computers have an I^2C bus, presumably for some of the purposes listed by wikipedia, such as interfacing with the RTC (real time clock) and configuring memory. However, it is not exposed, meaning you can't attach anything else to it, and there are a lot of interesting things that could be attached -- pretty much any kind of common sensor (barometers, accelerometers, gyroscopes, luminometers, etc.) as well as output devices and displays. You can buy a USB to I^2C adapter for a normal computer, but they cost a few hundred dollars. You can attach multiple devices to the exposed bus on the pi.

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

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

SPI, MOSI/MISO/CE0/CE1: SPI is a serial bus protocol serving many of the same purposes as I^2C, but because there are more wires, it can operate in full duplex which makes it faster and more flexible.

Raspberry Pi Terminal Commands

[sudo apt-get update] - Update Package Lists
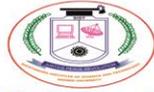
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

[sudo apt-get upgrade] - Download and Install Updated Packages

[sudo raspi-config] - The Raspberry Pi Configuration Tool

[sudo apt-get clean] - Clean Old Package Files

[sudo reboot] - Restart your Raspberry Pi

[sudo halt] - Shut Down your Raspberry Pi

3. **REMOTE DESKTOP ON THE RASPBERRY Pi WITH VNC**

- VNC: stands for Virtual Network Computing, allows you to access your Raspberry Pi

- RealVNC: a company which originated VNC (there are many other implementations). A RealVNC server is included with the Raspberry Pi, so that's the implementation we're going to use.

- VNC Server: an application which runs on the Raspberry Pi, and allows the VNC client to connect, view and control your Raspberry Pi desktop.

- VNC Client: an application which you can install on your desktop computer (Windows / Linux / Mac / …) or smartphone / tablet, to connect to the Raspberry Pi running the VNC server also called **VNC viewer**
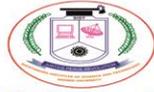
Fig. 10 Using RealVNC to access the Raspberry Pi's graphical desktop

Enable VNC

You will need to interact with your Pi in order to turn on the VNC server. To do this, you have several options:

- Connect a keyboard, mouse, and monitor. Click the **Terminal** icon on the top left of the desktop to open a terminal window.
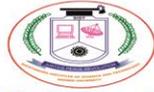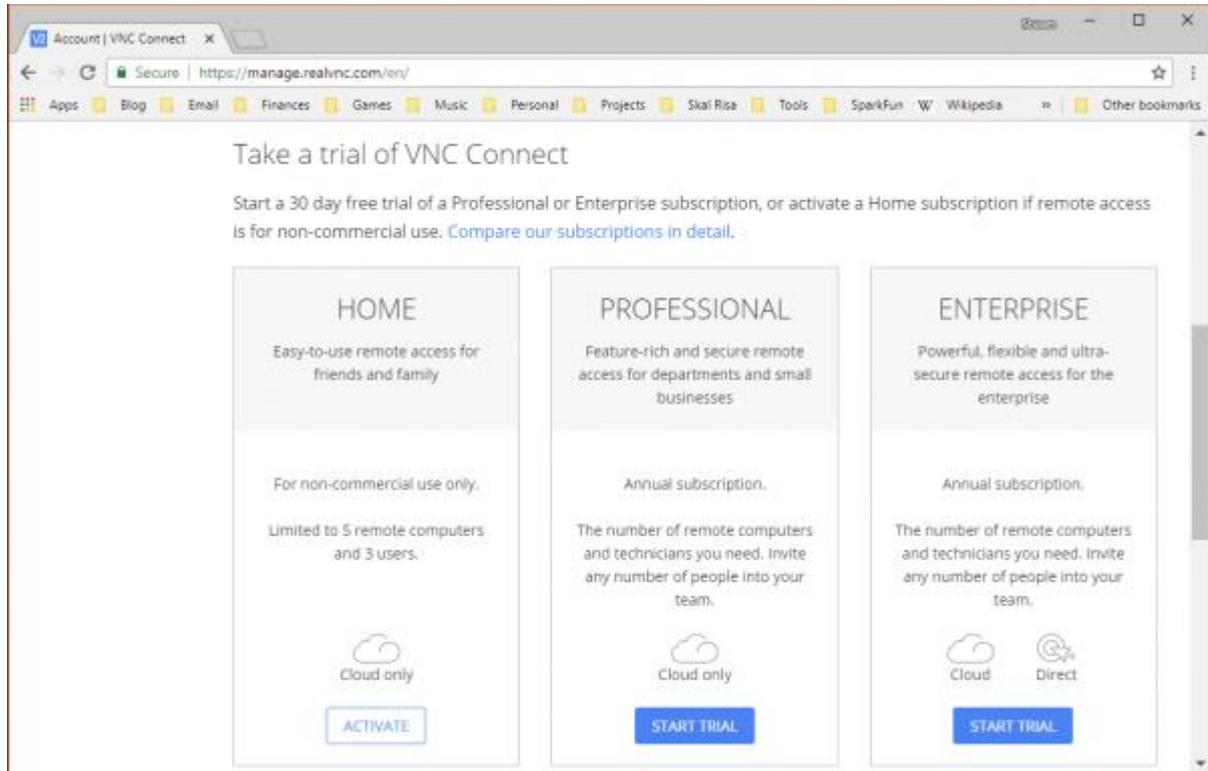
Fig .11 VNC Connect

Enable VNC Cloud Connection on the Pi

Enable RealVNC using the Raspberry Pi Configuration tool

Click on the Raspberry Pi OS menu, select **Preferences**, and in the submenu **Raspberry Pi Configuration**.
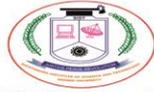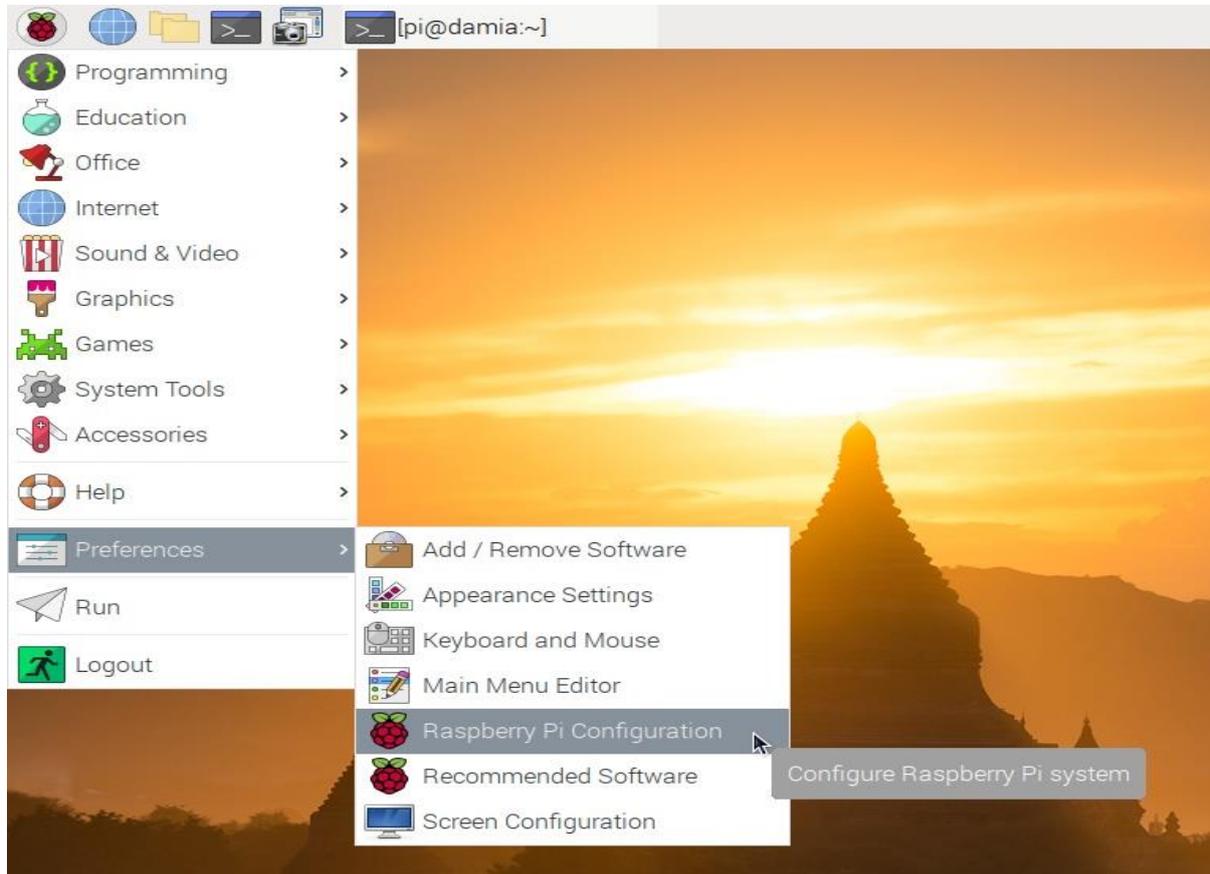
Fig. 12 Raspberry Pi Configuration

Configure your Raspberry Pi system using the Raspberry Pi Configuration tool. The tool will have several tabs, click on the tab "Interfaces", to see the available options there. Note that VNC is disabled by default: Raspberry Pi Configuration tool, showing VNC as disabled ("Enable remote access to this Pi using RealVNC") Click on the enable radio button, and then on ok button. VNC is now enabled in the Interfaces tab of the Raspberry Pi Configuration tool. Watch the task bar at the top of the screen. A new icon with a V2 symbol will appear. Raspberry Pi taskbar top right corner, before enabling VNC

![Sathyabama Institute of Science and Technology logo]

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

Second step: obtain your Raspberry Pi RealVNC IP address and credentials

To be able to control your Raspberry Pi remotely, you need to know the IP address of the Raspberry Pi running the RealVNC server to connect to it.

Click on the new icon (V2) in the taskbar (using the left mouse button, single click). A window will appear, showing you all you need to know to connect:
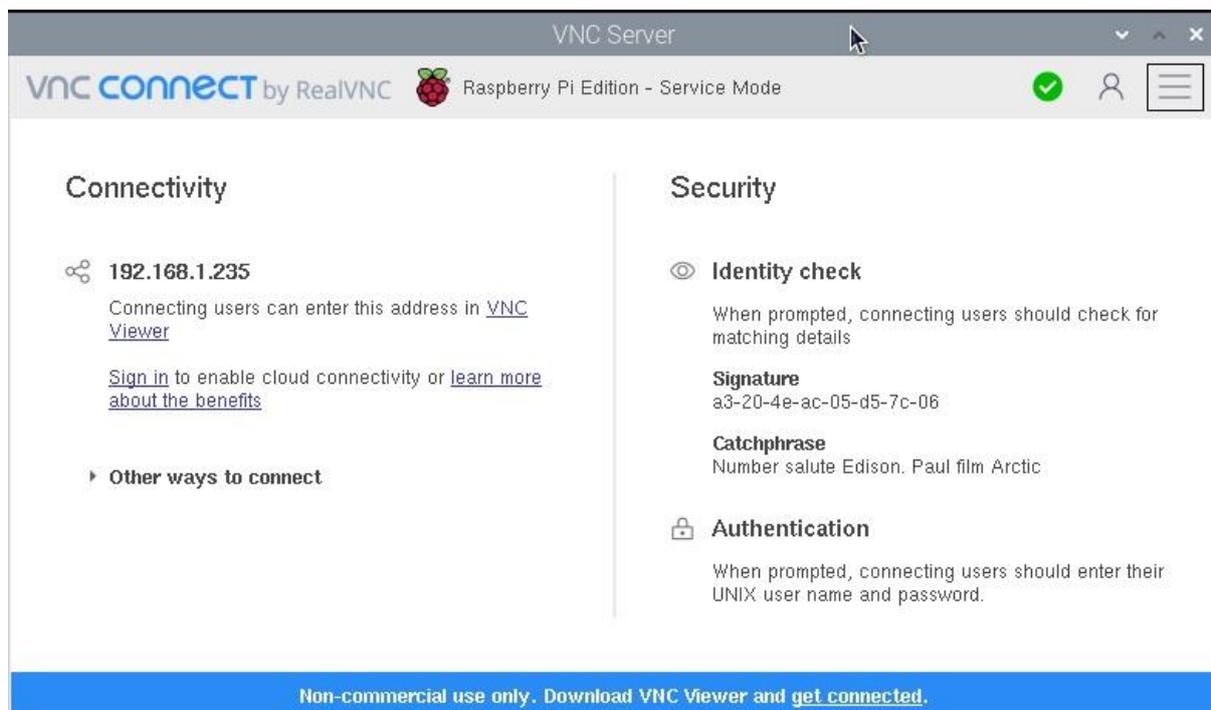
Fig. 13 Real VNC

VNC Server VNC connect by RealVNC Raspberry Pi Edition. Showing the IP address, and the identity check signature and Catchphrase. Authentication is with your UNIX user name and password. Download and install the RealVNC viewer.

The following steps are run on your main system, from which you desire to control the Raspberry Pi remotely using VNC. Be sure to select the appropriate operating system

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

(Windows / macOS / Linux / Raspberry Pi / iOS / Android / Chrome / Solaris / HP-UX / AIX). Install the VNC viewer according to the default procedure on your operating system. We can also use other VNC viewers, but using the RealVNC VNC viewer is recommended, since they have the best interoperability.

Here you can enter the Raspberry Pi IP address we have identified in step 2. Type in the address, and click on the "connect to address or hostname" area (or simply press enter).Now the VNC Viewer will show you an authentication screen, asking you to sign in with your credentials (password and username).
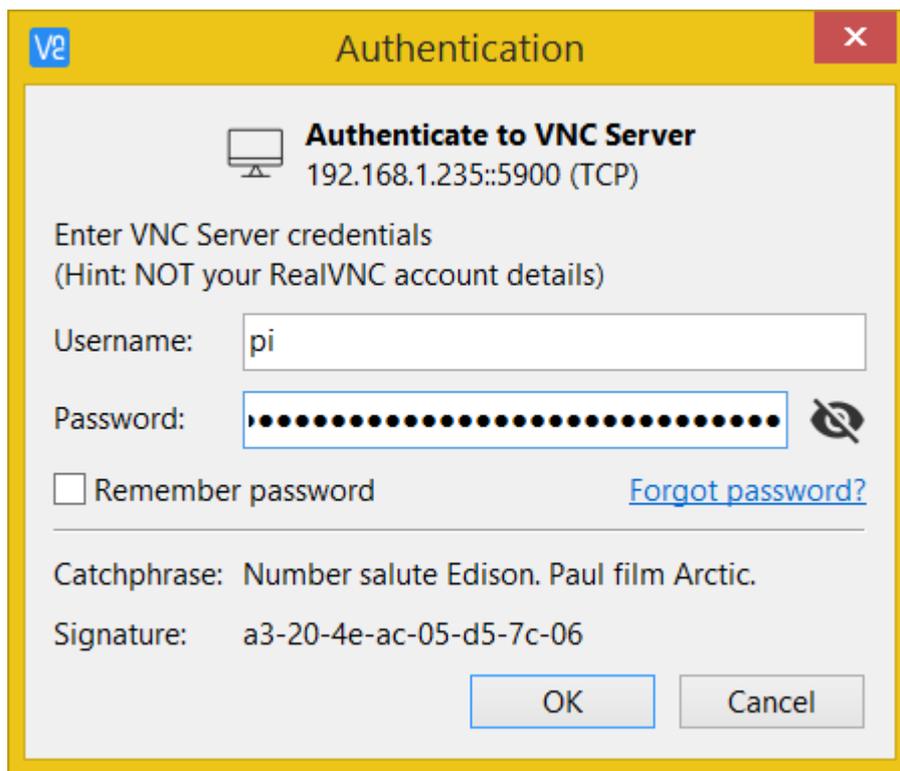
Fig 14 Authentication
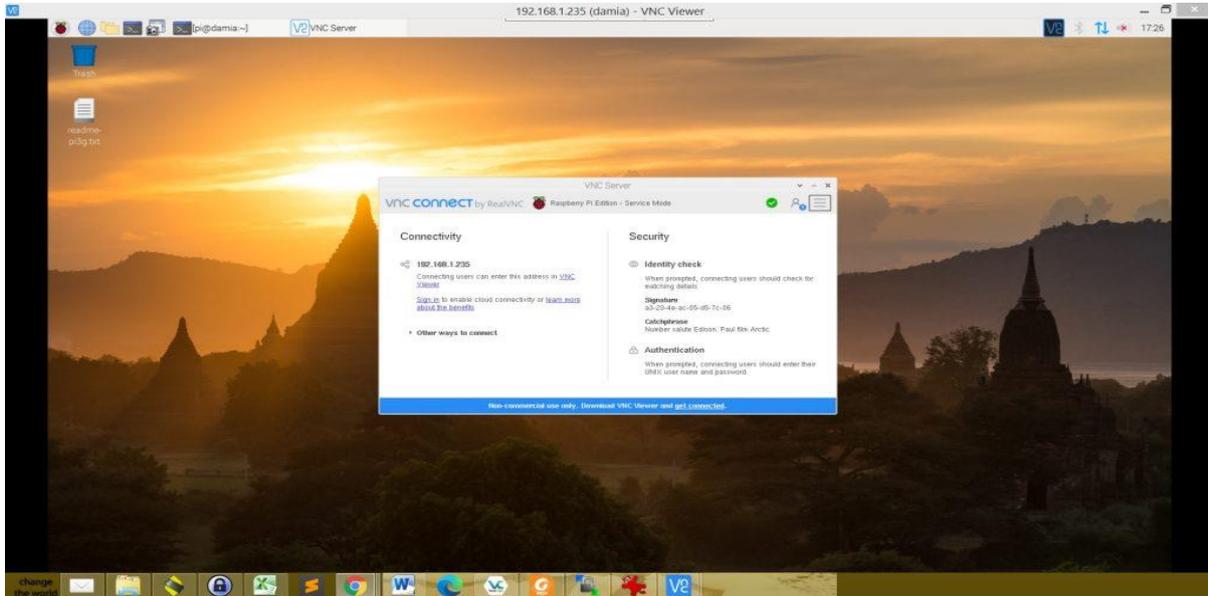
Fig 15 View Raspberry using VNC Viewer

On your host machine, download and install the RealVNC viewer. Open the application, and click the **Sign in** button in the top-right. Enter your email and password, and click **Sign in**.

On the right side, you should see an address book (previously used connections) and something showing your "Team" (computers available for a VNC cloud connection). Click on your **Team**, and you should see your VNC-ready Raspberry Pi listed.

Double-click on your Raspberry Pi to connect to it. You should see a pop-up window explaining that the VNC server on your Raspberry Pi has been verified. Click **Continue**. You should be prompted with an Authentication window. If you did not change the login username and password for your Pi, your default login credentials are:

- **Username:** pi
- **Password:** raspberry

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

Another slick feature is the ability to control your Raspberry Pi from your smartphone or tablet! Download the VNC Viewer app from the iTunes store or Google Play. Open the app, sign in, and connect to your Raspberry Pi!

4.   CONTROLLING GPIO USING WEB INTERFACE

In this part, we will **install Apache web server in** Raspberry Pi **to control the LED** from a webpage that can be accessed from anywhere over the internet. Here we **control an LED connected to Raspberry Pi by using Apache web server**. For this, we create an HTML/php web page which has two buttons - one for turning on the LED and the second for turning off the LED.

 Components Required

1.   Raspberry pi board (With Raspbian operating system)
2.   LED
3.   4. 250-ohm resistor
4.   Jumper Wires

 An SSH client (Putty) is used to connect the Raspberry pi using a Laptop or computer. For this, the raspberry pi needs to be connected to a network via LAN or Wi-Fi. If you have a separate monitor for your raspberry pi, then it's better to connect raspberry pi with the monitor and you don't have to use any SSH client. Python is a very useful programming language that has an easy to read syntax, and allows programmers to use fewer lines of code than would be possible in languages such as assembly, C, or Java.

The Python programming language actually started as a scripting language for Linux. Python programs are similar to shell scripts in that the files contain a series of commands that the computer executes from top to bottom. Compare a "hello world" program written in C to the same program written in Python:

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

```
1  #!/usr/bin/python
2
3  print "Hello, World!";
4
```

"Hello, World!" program
in Python

```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Hello, World! \n");
6      return 0;
7  }
8
```

"Hello, World!" program
in C

Unlike C programs, Python programs don't need to be compiled before running them. However, you will need to install the Python interpreter on your computer to run them. The Python interpreter is a program that reads Python files and executes the code.

It is possible to run Python programs without the Python interpreter installed though. Like shell scripts, Python can automate tasks like batch renaming and moving large amounts of files. It can be used just like a command line with IDLE, Python's REPL (read, eval, print, loop) function. However, there are more useful things you can do with Python. For example, you can use Python to program things like:

- Web applications

- Desktop applications and utilities

- Special GUIs

- Small databases

- 2D games

Python also has a large collection of libraries, which speeds up the development process. There are libraries for everything you can think of – game programming, rendering graphics, GUI interfaces, web frameworks, and scientific computing.

Many (but not all) of the things you can do in C can be done in Python. Python is generally slower at computations than C, but its ease of use makes Python an ideal language for prototyping programs and designing applications that aren't computationally intensive.
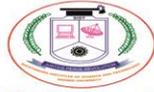
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

4.1 INSTALLING AND UPDATING PYTHON

Python 2 and Python 3 come pre-installed on Raspbian operating systems, but to install Python on another Linux OS or to update it, simply run one of these commands at the command prompt:

sudo apt-get install python3

Installs or updates Python 3.

sudo apt-get install python

Installs or updates Python 3

4.2 WRITING A PYTHON PROGRAM

To demonstrate creating and executing a Python program, we'll make a simple "hello world" program. To begin, open the Nano text editor and create a new file named hello-world.py by entering this at the command prompt:

sudo nano hello-world.py

Enter this code into Nano, then press Ctrl-X and Y to exit and save the file:

```
#!/usr/bin/python

print "Hello, World!";
```

All Python program files will need to be saved with a ".py" extension. You can write the program in any text editor such as Notepad or Notepad++, just be sure to save the file with a ".py" extension. To run the program without making it executable, navigate to the location where you saved your file, and enter this at the command prompt:

python hello-world.py
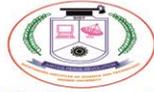
CONNECT THE LED TO THE RASPBERRY PI

Components:

- Raspberry Pi
- One LED
- One 330 Ohm resistor
- Jumper wires
- Breadboard

Connect the components as shown in the wiring diagram below.
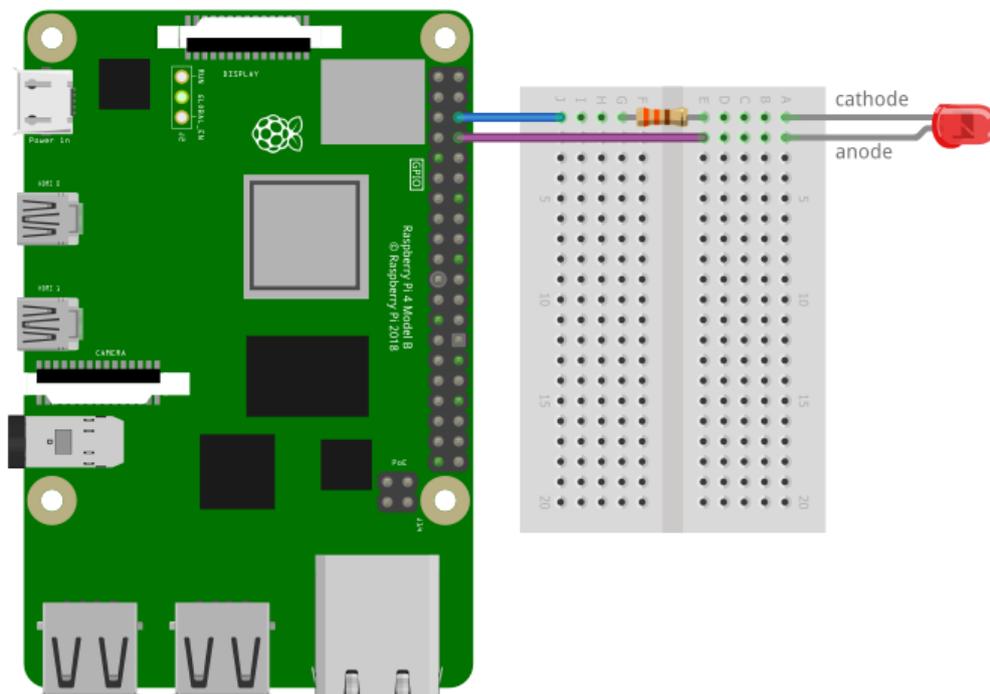
Fig. 16 Raspberry pi with LED

The 330 Ohm resistor is a current limiting resistor. Current limiting resistors should always be used when connecting LEDs to the GPIO pins. If an LED is connected to a GPIO pin without a resistor, the LED will draw too much current, which can damage the Raspberry Pi or burn out the LED. Here is a nice calculator that will give you the value of a current

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

limiting resistor to use for different LEDs. After connecting the hardware components, the next step is to create a Python program to switch on and off the LED. This program will make the LED turn on and off once every second and output the status of the LED to the terminal. The first step is to create a Python file. To do this, open the Raspberry Pi terminal and type nano LED.py. Then press Enter. This will create a file named LED.py and open it in the Nano text editor. Copy and paste the Python code below into Nano and save and close the file.

```
import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)

GPIO.setup(14,GPIO.OUT)

# While loop

while True:

    # set GPIO14 pin to HIGH

    GPIO.output(14,GPIO.HIGH)

    # show message to Terminal

    print "LED is ON"

    # pause for one second

    time.sleep(1)

    # set GPIO14 pin to HIGH
```
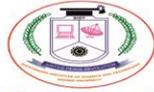
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

```
GPIO.output(14,GPIO.LOW)

# show message to Terminal

print "LED is OFF"

# pause for one second

time.sleep(1)
```

At the top of the program we import the RPi.GPIO and time libraries. The RPi.GPIO library will allow us to control the GPIO pins. The time library contains the sleep() function that we will use to make the LED pause for one second.

Next we initialize the GPIO object with GPIO.setmode(GPIO.BCM). We are using the BCM pin numbering system in this program. We use .GPIO.setwarnings(False) to disable the warnings and GPIO.setup(14,GPIO.OUT) is used to set GPIO14 as an output.

Now we need to change the on/off state of GPIO14 once every second. We do this with the GPIO.output() function. The first parameter of this function is the GPIO pin that will be switched high or low. We have the LED connected to GPIO14 in this circuit, so the first argument is 14.

The second parameter of the GPIO.output() function is the voltage state of the GPIO pin. We can use either GPIO.HIGH or GPIO.LOW as an argument to turn the pin on or off.

Each GPIO.output() function in the code above is followed by a sleep() function that causes the pin to hold its voltage state for the time (in seconds) defined in the parameter of the function. In this program we are switching the LED on and off once every second so the argument is 1. You can change this value to make the LED blink on and off faster or slower. Run the Python program above by entering the following into the Raspberry Pi's terminal:

![Sathyabama Institute of Science and Technology logo]

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in
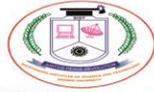
sudo python LED.py

You should see the LED blinking on and off once every second.

You should also see a message in the terminal with "LED is ON" when the LED is turned on, and "LED is OFF" when the LED is turned off.

Fig. 17 output

4.3 Using a PIR sensor

Humans and other animals emit radiation all the time. This is nothing to be concerned about, though, as the type of radiation we emit is infrared radiation (IR), which is pretty harmless at the levels at which it is emitted by humans. In fact, all objects at temperatures above absolute zero (-273.15C) emit infrared radiation. A PIR sensor detects changes in the amount of infrared radiation it receives. When there is a significant change in the amount of infrared radiation it detects, then a pulse is triggered. This means that a PIR sensor can detect when a human (or any animal) moves in front of it. The pulse emitted when a PIR detects motion needs to be amplified, and so it needs to be powered. There are three pins on the PIR: they should be

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in
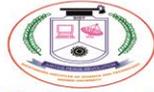
labelled Vcc, Gnd, and Out. These labels are sometimes concealed beneath the Fresnel lens (the white cap), which you can temporarily remove to see the pin labels.
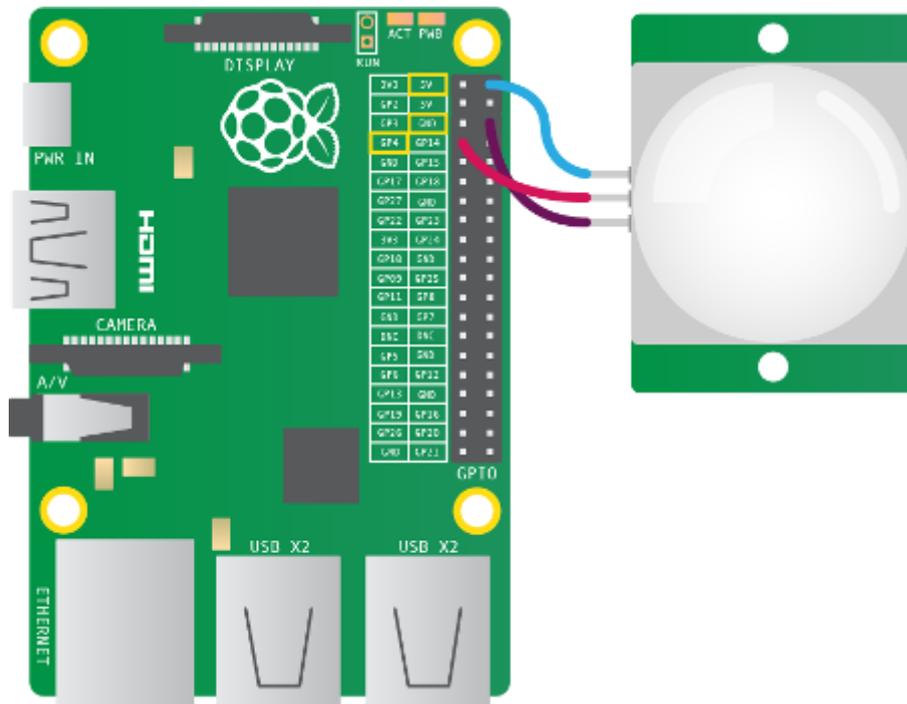
Fig. 18 Raspberry with PIR

1. As shown above, the Vcc pin needs to be attached to a 5V pin on the Raspberry Pi.
2. The Gnd pin on the PIR sensor can be attached to any ground pin on the Raspberry Pi.
3. Lastly, the Out pin needs to be connected to any of the GPIO pins.

Tuning a PIR

Most PIR sensors have two potentiometers on them. These can control the sensitivity of the sensors, and also the period of time for which the PIR will signal when motion is detected.

Program

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
PIR_PIN = 7
GPIO.setup(PIR_PIN, GPIO.IN)

try:
        print "PIR Module Test (CTRL+C to exit)"
        time.sleep(2)
        print "Ready"
        while True:
                if GPIO.input(PIR_PIN):
                        print "Motion Detected!"
                time.sleep(1)
except KeyboardInterrupt:
        print " Quit"
        GPIO.cleanup()
```

4.4 Controlling LED using Raspberry Pi Webserver

Step 1: Connections

The connections in this project are quite simple - the positive pin of LED is connected to GPIO 27 pin and the negative pin to a 270 ohm resistor, the other side of which is connected to GND pin.
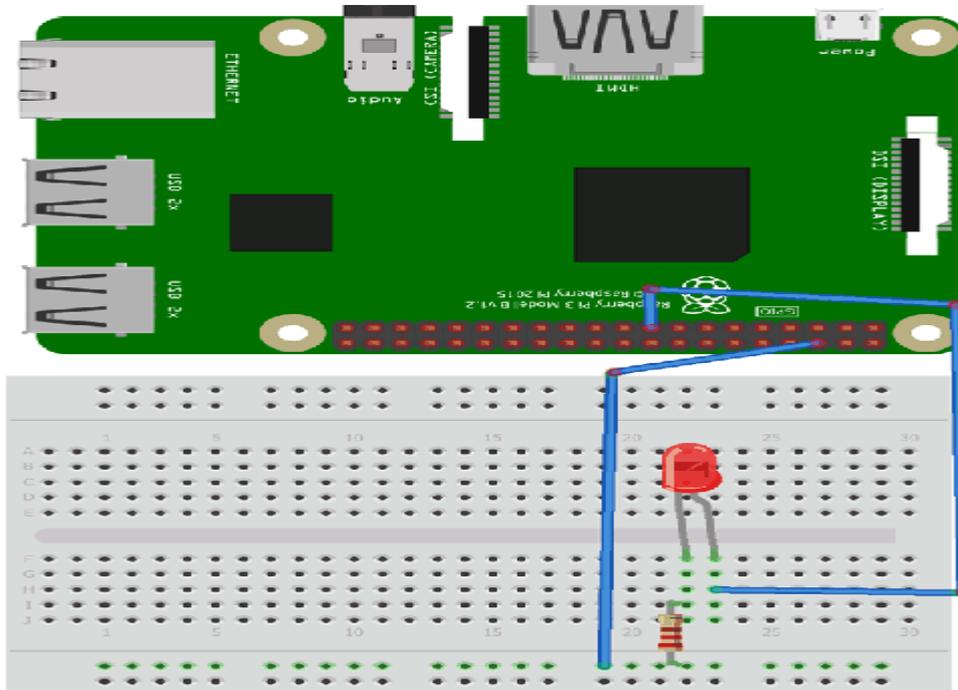
Fig 19 LED with Raspberry pi

Step 2: Installing WiringPi Library

WiringPi is a PIN-based GPIO access library written in C for the BCM2835, BCM2836, and BCM2837 SoC devices used in all Raspberry Pi versions. It's released under the GNU LGPLv3 license and is usable from C, C++, and RTB (BASIC) as well as many other languages with suitable wrappers.

1. First we will update our Pi with the latest versions of Raspbian using the command:

sudo apt-get update

2. Now we will install git by using this command:

sudo apt-get install git-core

3. Now obtain WiringPi using git by this command:

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

git clone git://git.drogon.net/wiringPi

4. Then install WiringPi library using:

cd wiringP./build

Step 3: Installing a Web Server

Apache is a very popular webserver, designed to create web servers that have the ability to host one or more HTTP-based websites. Apache Web Server can be enhanced by manipulating the code base or adding multiple extensions/add-ons. In our project, we are using an HTTP server and its PHP extension.

To Install Apache web server, we will use the following commands:

First, update the available packages:

sudo apt-get update

 Now, install the apache2 package by using this command in the terminal:

sudo apt-get install apache2 -y

To test the web server whether it is working or not, go to your browser and type the Pi's IP address in the tab. To find the Pi's IP address, type *ifconfig* at the command line. By default, Apache puts a test HTML file in the web folder. This default web page is served when you browse to http://192.168.1.31 (whatever the Pi's IP address is) from another computer on the network. Browse to the default web page either on the Pi or from another computer on the network and you will see the following:
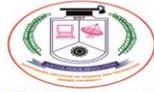
Fig. 20 Apache web Server

 Now we will see how to change the default web page with your own HTML page

This default web page is just an HTML file on the filesystem. It is located at *var/www/html/index.html.*

Navigate to this directory in a terminal window and have a look at what's inside:

cd var/www/html

ls -al

This will show you:

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

total 12

drwxr-xr-x 2 root root 4096 Jan 8 01:29 .

drwxr-xr-x 12 root root 4096 Jan 8 01:28 ..

-rw-r--r-- 1 root root 177 Jan 8 01:29 index.html

This shows that by default there is one file in /var/www/html/ called index.html and it is owned by the root user. To edit the file, you need to change its ownership to your own username. Change the owner of the file using:

Sudo chown pi: index.html.

You can now try editing this file and then refresh the browser to see the web page change.

Install PHP in Raspberry Pi

Now if we want to use PHP code along with HTML, then we have to further install the PHP extension in Raspberry pi. Using PHP code, we can create shell commands to control the LED from the PHP script.

To allow the Apache server to edit PHP files, we will install the latest version of PHP and the PHP module for Apache. Use the following command in terminal to install these:

sudo apt-get install php libapache2-mod-php -y

Now remove the default index.html file:

sudo rm index.html

And create your own index.php file:

sudo nano index.php
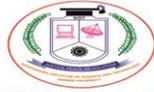
Now enter the below code in index.php to test the PHP installation.

```php
<?php phpinfo(); ?>
```

Save it by pressing CTRL + X and the 'y' and enter. Now refresh the webpage in your browser, you will see a long page with lots of information about PHP. This shows that the PHP extension is installed properly. If you have any problem with the pages or if the pages do not appear, try reinstalling the apache server and its PHP extension.

Step 5: Start Coding for controlling GPIO pin using this Raspberry Pi Webserver

Now delete the previous code in index.php (<?php phpinfo(); ?>) file and insert below PHP code to control GPIO pins inside body of HTML code.

Below is the complete code for creating two buttons to turn on and off the LED connected to Raspberry Pi.

```html
<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Raspberry Pi WiFi Controlled LED</title>

</head>

    <body>

    <center><h1>Control LED using Raspberry Pi Webserver</h1>

     <form method="get" action="index.php">

        <input type="submit" style = "font-size: 14 pt" value="OFF" name="off">
```

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

```
        <input type="submit" style = "font-size: 14 pt" value="ON" name="on">

    </form>

            </center>

<?php

  shell_exec("/usr/local/bin/gpio -g mode 27 out");

  if(isset($_GET['off']))

    {

            echo "LED is off";

            shell_exec("/usr/local/bin/gpio -g write 27 0");

    }

      else if(isset($_GET['on']))

      {

            echo "LED is on";

            shell_exec("/usr/local/bin/gpio -g write 27 1");

      }

?>

  </body>

</html>
```

← → C ⊙ http://192.168.1.31   ⓐ ⌗ | Ⓜ :

Control LED using Raspberry Pi Webserver

OFF | ON

In the above code there is a PHP script which checks which button is pressed by using below code and then turns on and off the LED accordingly.

```php
<?php

    shell_exec("/usr/local/bin/gpio -g mode 27 out");

    if(isset($_GET['off']))

        {

                echo "LED is off";

                shell_exec("/usr/local/bin/gpio -g write 27 0");

        }

        else if(isset($_GET['on']))

        {

                echo "LED is on";

                shell_exec("/usr/local/bin/gpio -g write 27 1");

        }

?>
```
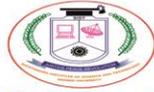
Here we have used **shell_exec()** command in php code, this command is used to run the shell command from the PHP script. Learn more about _shell_exec here_. If you run the command inside _shell_exec_ directly form the terminal of Raspberry pi, you can directly make GPIO pin 27 low or high. Below are two commands to test the LED directly from terminal.

/usr/local/bin/gpio -g write 27 0

/usr/local/bin/gpio -g write 27 1

After completing this, run the code in your browser by typing the IP address of raspberry pi in the browser. You will see 2 buttons - ON, OFF to control your LED by clicking these buttons.

5.LIQUID CRYSTAL DISPLAYS (LCD) WITH ARDUINO

The Liquid Crystal library allows you to control LCD displays that are compatible with the Hitachi HD44780 driver. There are many of them out there, and you can usually tell them by the 16-pin interface.

Fig. 21 Output of the sketch in 16x2 LCD

The LCDs have a parallel interface, meaning that the microcontroller has to manipulate several interface pins at once to control the display. The interface consists of the following pins:

- A **register select (RS) pin** that controls where in the LCD's memory you're writing data to. You can select either the data register, which holds what goes on the screen, or an instruction register, which is where the LCD's controller looks for instructions on what to do next.
- A **Read/Write (R/W) pin** that selects reading mode or writing mode
- An **Enable pin** that enables writing to the registers
- 8 **data pins (D0 -D7)**. The states of these pins (high or low) are the bits that you're writing to a register when you write, or the values you're reading when you read.

There's also a **display contrast pin (Vo)**, **power supply pins (+5V and GND)** and **LED Backlight (Bklt+ and BKlt-)** pins that you can use to power the LCD, control the display contrast, and turn on and off the LED backlight, respectively.

The process of controlling the display involves putting the data that form the image of what you want to display into the data registers, then putting instructions in the instruction register. The LiquidCrystal Library simplifies this for you so you don't need to know the low-level instructions.

The Hitachi-compatible LCDs can be controlled in two modes: 4-bit or 8-bit. The 4-bit mode requires seven I/O pins from the Arduino, while the 8-bit mode requires 11 pins. For displaying text on the screen, you can do most everything in 4-bit mode, so example shows how to control a 16x2 LCD in 4-bit mode.

Hardware Required

- Arduino Board
- LCD Screen (compatible with Hitachi HD44780 driver)
- pin headers to solder to the LCD display pins

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

- 10k ohm potentiometer
- 220 ohm resistor
- hook-up wires
- breadboard

Before wiring the LCD screen to your Arduino board we suggest to solder a pin header strip to the 14 (or 16) pin count connector of the LCD screen, as you can see in the image further up.

To wire your LCD screen to your board, connect the following pins:

- LCD RS pin to digital pin 12
- LCD Enable pin to digital pin 11
- LCD D4 pin to digital pin 5
- LCD D5 pin to digital pin 4
- LCD D6 pin to digital pin 3
- LCD D7 pin to digital pin 2
- LCD R/W pin to GND
- LCD VSS pin to GND
- LCD VCC pin to 5V
- LCD LED+ to 5V through a 220 ohm resistor
- LCD LED- to GND

Additionally, wire a 10k potentiometer to +5V and GND, with it's wiper (output) to LCD screens VO pin (pin3).
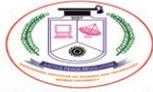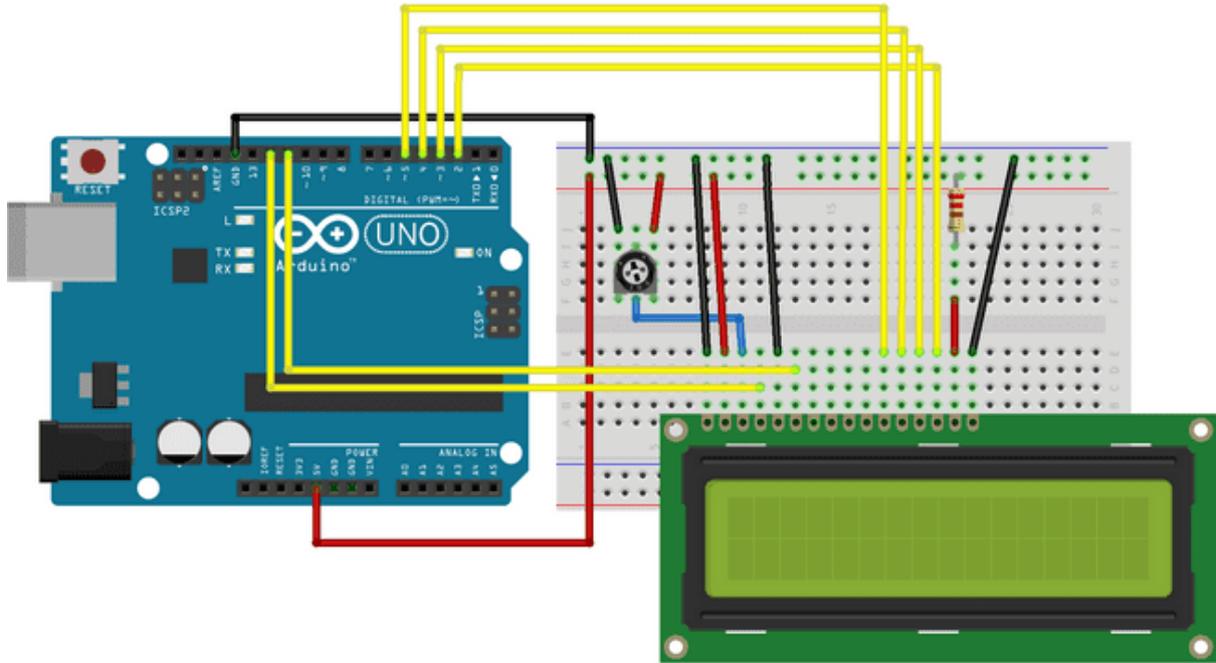
Fig. 22 Arduino with LCD

The circuit (made using Fritzing).

Schematic

The schematic (made using Fritzing).

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

Hello world example

/*

 LiquidCrystal Library - Hello World

Demonstrates the use a 16x2 LCD display. The LiquidCrystal

library works with all LCD displays that are compatible with the

Hitachi HD44780 driver. There are many of them out there, and you

can usually tell them by the 16-pin interface.

This sketch prints "Hello World!" to the LCD

and shows the time.

 The circuit:

* LCD RS pin to digital pin 12

* LCD Enable pin to digital pin 11

* LCD D4 pin to digital pin 5

* LCD D5 pin to digital pin 4

* LCD D6 pin to digital pin 3

* LCD D7 pin to digital pin 2

* LCD R/W pin to ground

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

* LCD VSS pin to ground

* LCD VCC pin to 5V

* 10K resistor:

* ends to +5V and ground

* wiper to LCD VO pin (pin 3)

```
// include the library code:

#include <LiquidCrystal.h>


// initialize the library by associating any needed LCD interface pin

// with the arduino pin number it is connected to

const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);


void setup() {

  // set up the LCD's number of columns and rows:

  lcd.begin(16, 2);

  // Print a message to the LCD.

  lcd.print("hello, world!");
```

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

```
}

void loop() {

  // set the cursor to column 0, line 1

  // (note: line 1 is the second row, since counting begins with 0):

  lcd.setCursor(0, 1);

  // print the number of seconds since reset:

  lcd.print(millis() / 1000);

}
```

Autoscroll Example

This example sketch shows how to use the autoscroll() and noAutoscroll() methods to move all the text on the display left or right.

autoscroll() moves all the text one space to the left each time a letter is added

noAutoscroll() turns scrolling off

This sketch prints the characters 0 to 9 with autoscroll off, then moves the cursor to the bottom right, turns autoscroll on, and prints them again.

```
/*

  LiquidCrystal Library - Autoscroll

 Demonstrates the use a 16x2 LCD display.  The LiquidCrystal

 library works with all LCD displays that are compatible with the
```

Hitachi HD44780 driver. There are many of them out there, and you

can usually tell them by the 16-pin interface.

This sketch demonstrates the use of the autoscroll()

and noAutoscroll() functions to make new text scroll or not.

The circuit:

* LCD RS pin to digital pin 12

* LCD Enable pin to digital pin 11

* LCD D4 pin to digital pin 5

* LCD D5 pin to digital pin 4

* LCD D6 pin to digital pin 3

* LCD D7 pin to digital pin 2

* LCD R/W pin to ground

* 10K resistor:

* ends to +5V and ground

* wiper to LCD VO pin (pin 3)

// include the library code:

#include <LiquidCrystal.h>

// initialize the library by associating any needed LCD interface pin

// with the arduino pin number it is connected to

```
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {

  // set up the LCD's number of columns and rows:

  lcd.begin(16, 2);

}

void loop() {

  // set the cursor to (0,0):

  lcd.setCursor(0, 0);

  // print from 0 to 9:

  for (int thisChar = 0; thisChar < 10; thisChar++) {

    lcd.print(thisChar);

    delay(500);

  }

  // set the cursor to (16,1):

  lcd.setCursor(16, 1);

  // set the display to automatically scroll:

  lcd.autoscroll();

  // print from 0 to 9:
```

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

```
for (int thisChar = 0; thisChar < 10; thisChar++) {

  lcd.print(thisChar);

  delay(500);

 }

 // turn off automatic scrolling

 lcd.noAutoscroll();

 // clear screen for the next loop:

 lcd.clear();

}
```

6.ARDUINO WITH KEYPAD

The buttons on a keypad are arranged in rows and columns. A 3X4 keypad has 4 rows and 3 columns, and a 4X4 keypad has 4 rows and 4 columns:

Fig. 23 Keypad

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in
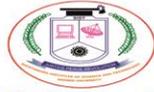
Beneath each key is a membrane switch. Each switch in a row is connected to the other switches in the row by a conductive trace underneath the pad. Each switch in a column is connected the same way – one side of the switch is connected to all of the other switches in that column by a conductive trace. Each row and column is brought out to a single pin, for a total of 8 pins on a 4X4 keypad:
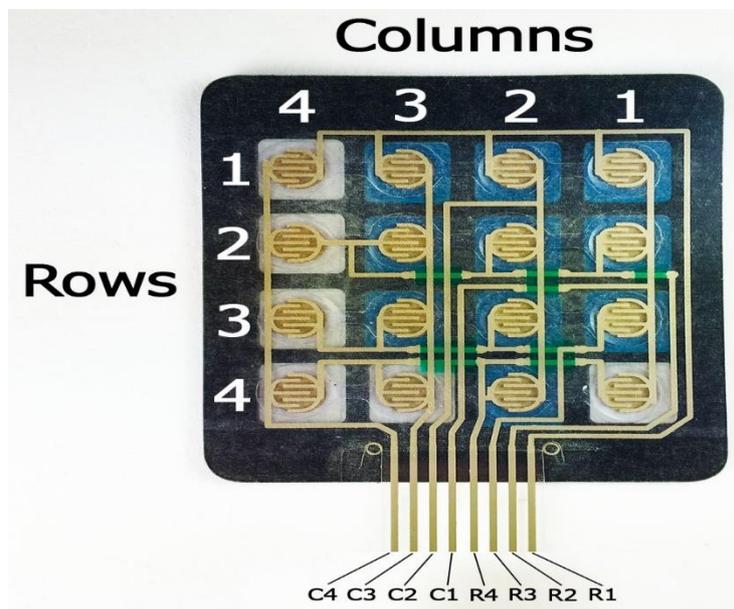
Fig. 24 keypad Structure

Pressing a button closes the switch between a column and a row trace, allowing current to flow between a column pin and a row pin. The Arduino detects which button is pressed by detecting the row and column pin that's connected to the button.

This happens in four steps:

1. First, when no buttons are pressed, all of the column pins are held HIGH, and all of the row pins are held LOW:

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

2. When a button is pressed, the column pin is pulled LOW since the current from the HIGH column flows to the LOW row pin:

3. The Arduino now knows which column the button is in, so now it just needs to find the row the button is in. It does this by switching each one of the row pins HIGH, and at the same time reading all of the column pins to detect which column pin returns to HIGH:

4. When the column pin goes HIGH again, the Arduino has found the row pin that is connected to the button:

The pin layout for most membrane keypads will look like this:
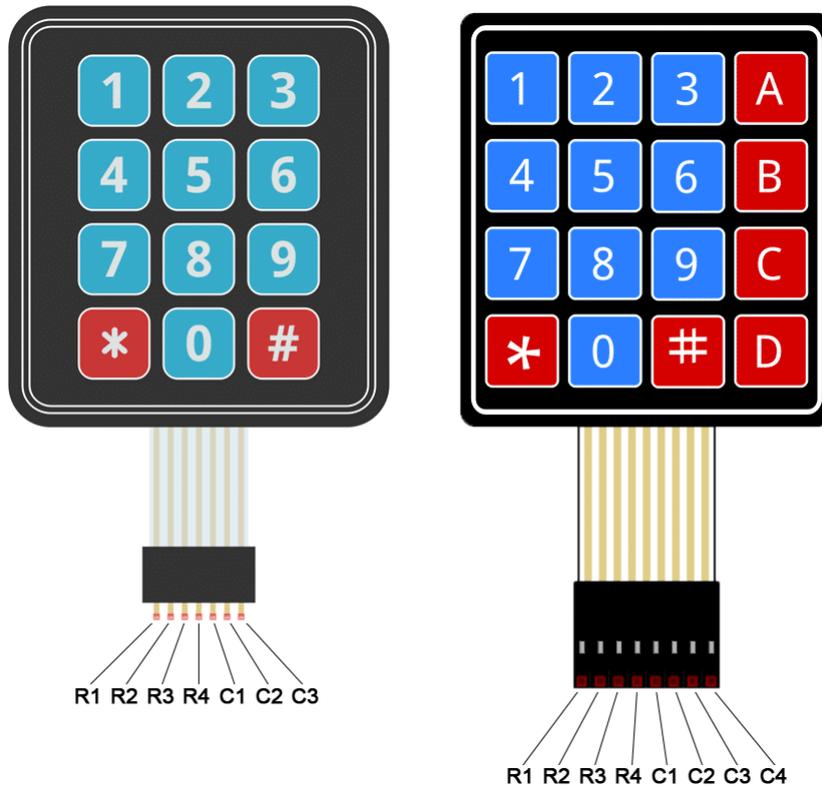
Fig 25 Pin Layout

Follow the diagrams below to connect the keypad to an Arduino Uno, depending on whether you have a 3X4 or 4X4 keypad: First, find out which keypad pins are connected to the button rows. Insert the ground (black) wire into the first pin on the left. Press any button in row 1 and hold it down. Now insert the positive (red) wire into each one of the other pins. If the LED lights up at one of the pins, press and hold another button in row 1, then insert the positive wire into each one of the other pins again. If the LED lights up on a different pin, it means the ground wire is inserted into the row 1 pin. If none of the buttons in row 1 make the LED light up, the ground wire is not connected to row 1. Now move the ground wire over to the next pin, press a button in a different row, and repeat the process above until you've found the pin for each row.

To figure out which pins the columns are connected to, insert the ground wire into the pin you know is row 1. Now press and hold any one of the buttons in that row. Now insert the positive wire into each one of the remaining pins. The pin that makes the LED light up is the pin that's connected to that button's column. Now press down another button in the same row, and insert the positive wire into each one of the other pins. Repeat this process for each one of the other columns until you have each one mapped out.

PROGRAMMING THE KEYPAD

For a basic demonstration of how to setup the keypad, I'll show you how to print each key press to the serial monitor. We'll use the Keypad library by Mark Stanley and Alexander Brevig. This library takes care of setting up the pins and polling the different columns and rows. To install the Keypad library, go to Sketch > Include Library > Manage Libraries and search for "keypad". Click on the library, then click install. Once the Keypad library is installed, you can upload this code to the Arduino if you're using a 4X4 keypad:

THE CODE FOR A 3X4 KEYPAD
If you're using a 3X4 keypad, you can use this code:

```cpp
#include <Keypad.h>

const byte ROWS = 4;

const byte COLS = 4;

char hexaKeys[ROWS][COLS] = {

  {'1', '2', '3', 'A'},

  {'4', '5', '6', 'B'},

  {'7', '8', '9', 'C'},

  {'*', '0', '#', 'D'}

};

byte rowPins[ROWS] = {9, 8, 7, 6};

byte colPins[COLS] = {5, 4, 3, 2};

Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

void setup(){

  Serial.begin(9600);

}

  void loop(){

  char customKey = customKeypad.getKey();
```
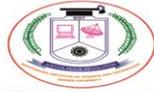
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

```
if (customKey){

Serial.println(customKey);

 }

}
```

7.ARDUINO WITH SENSORS

7.1 Ultrasonic sensor

It works by sending sound waves from the transmitter, which then bounce off of an object and then return to the receiver. You can determine how far away something is by the time it takes for the sound waves to get back to the sensor. Let's get right to it!. Connections

The connections are very simple:

- VCC to 5V
- GND to GND
- Trig to pin 9
- Echo to pin 10

You can actually connect Trig and Echo to whichever pins you want, 9 and 10 are just the ones I'm using.
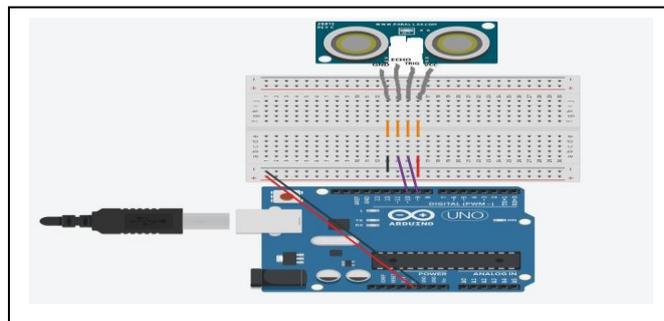
Fig 26. Ultra Sonic Sensor
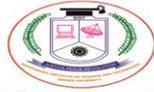
SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

Code:

we define the pins that Trig and Echo are connected to.

```
const int trigPin = 9;
const int echoPin = 10;
```

Then we declare 2 floats, duration and distance, which will hold the length of the sound wave and how far away the object is.

```
float duration, distance;
```

Next, in the setup, we declare the Trig pin as an output, the Echo pin as an input, and start Serial communications.

```
void setup() {
 pinMode(trigPin, OUTPUT);
 pinMode(echoPin, INPUT);
 Serial.begin(9600);
}
```

Now, in the loop, what we do is first set the trigPin low for 2 microseconds just to make sure that the pin in low first. Then, we set it high for 10 microseconds, which sends out an 8 cycle sonic burst from the transmitter, which then bounces of an object and hits the receiver(Which is connected to the Echo Pin).

```
void loop() {
 digitalWrite(trigPin, LOW);
 delayMicroseconds(2);
 digitalWrite(trigPin, HIGH);
 delayMicroseconds(10);
 digitalWrite(trigPin, LOW);
```

When the sound waves hit the receiver, it turns the Echo pin high for however long the waves were traveling for. To get that, we can use a handy Arduino function called pulseIn(). It takes 2 arguments, the pin you are listening to(In our case, the Echo pin), and a state(HIGH or LOW). What the function does is waits for the pin to go whichever sate you put in, starts timing, and then stops timing when it switches to the other state. In our case we would put HIGH since we want to start timing when the Echo pin goes high. We will store the time in the duration variable. (It returns the time in microseconds)

duration = pulseIn(echoPin, HIGH);

Now that we have the time, we can use the equation speed = distance/time, but we will make it time x speed = distance because we have the speed. What speed do we have? The speed of sound, of course! The speed of sound is approximately 340 meters per second, but since the pulseIn() function returns the time in microseconds, we will need to have a speed in microseconds also, which is easy to get. A quick Google search for "speed of sound in centimeters per microsecond" will say that it is .0343 c/μS. You could do the math, but searching it is easier. Anyway, with that information, we can calculate the distance! Just multiply the duration by .0343 and the divide it by 2(Because the sound waves travel to the object AND back). We will store that in the distance variable.

distance = (duration*.0343)/2;

The rest is just printing out the results to the Serial Monitor.

Serial.print("Distance: ");
 Serial.println(distance);
 delay(100);
}

7.2 Temperature Sensor
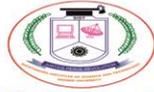
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

These sensors have little chips in them and while they're not that delicate, they do need to be handled properly. Be careful of static electricity when handling them and make sure the power supply is connected up correctly and is between 2.7 and 5.5V DC.They come in a "TO-92" package which means the chip is housed in a plastic hemi-cylinder with three legs. The legs can be bent easily to allow the sensor to be plugged into a breadboard. You can also solder to the pins to connect long wires.

Reading the Analog Temperature Data

Unlike the FSR or photocell sensors we have looked at, the TMP36 and friends doesn't act like a resistor. Because of that, there is really only one way to read the temperature value from the sensor, and that is plugging the output pin directly into an Analog (ADC) input. Remember that you can use anywhere between 2.7V and 5.5V as the power supply. For this example I'm showing it with a 5V supply but note that you can use this with a 3.3v supply just as easily. No matter what supply you use, the analog voltage reading will range from about 0V (ground) to about 1.75V.

If you're using a 5V Arduino, and connecting the sensor directly into an Analog pin, you can use these formulas to turn the 10-bit analog reading into a temperature:

Voltage at pin in milliVolts = (*reading from ADC*) * (5000/1024)

This formula converts the number 0-1023 from the ADC into 0-5000mV (= 5V) If you're using a 3.3V Arduino, you'll want to use this:

Voltage at pin in milliVolts = (*reading from ADC*) * (3300/1024)

This formula converts the number 0-1023 from the ADC into 0-3300mV (= 3.3V) Then, to convert millivolts into temperature, use this formula:

**Centigrade temperature = [(analog voltage in mV) - 500] / 10**
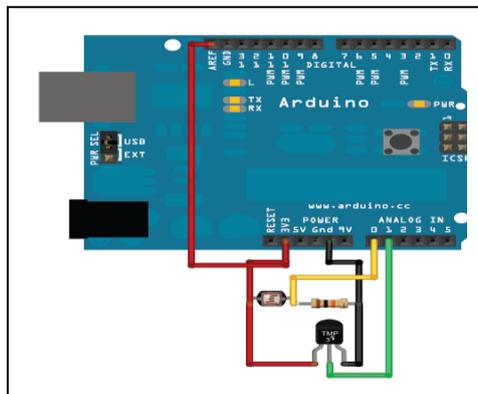

Fig 27 Temperature Sensor

This example code for Arduino shows a quick way to create a temperature sensor, it simply prints to the serial port what the current temperature is in both Celsius and Fahrenheit. For better results, using the 3.3v reference voltage as ARef instead of the 5V will be more precise and less noisy.

int sensorPin = 0;
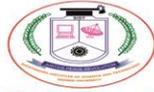
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

```
 * setup() - this function runs once when you turn your Arduino on
void setup()
{
  Serial.begin(9600);  //Start the serial connection with the computer
                //to view the result open the serial monitor
}
void loop()                // run over and over again
{
 //getting the voltage reading from the temperature sensor
 int reading = analogRead(sensorPin);
 // converting that reading to voltage, for 3.3v arduino use 3.3
 float voltage = reading * 5.0;
 voltage /= 1024.0;
 // print out the voltage
 Serial.print(voltage); Serial.println(" volts");
 // now print out the temperature
 float temperatureC = (voltage - 0.5) * 100 ;  //converting from 10 mv per degree wit 500
mV offset
                                //to degrees ((voltage - 500mV) times 100)
 Serial.print(temperatureC); Serial.println(" degrees C");
 // now convert to Fahrenheit
 float temperatureF = (temperatureC * 9.0 / 5.0) + 32.0;
 Serial.print(temperatureF); Serial.println(" degrees F");
 delay(1000);                                //waiting a second
}
```

School of Computing

Department of Computer Science and Engineering

UNIT III - ARDUINO PROGRAMMING FOR IoT BOARDS - SCSA1407
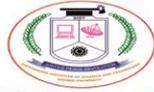
UNIT 3: Platforms

History - Creative Coding Platforms - Open Source Platforms – PIC - Arduino, Sketch, Iterative coding methodology – Python Programming - Mobile phones and similar devices - Arm Devices - Basic Electronics (circuit theory, measurements, parts identification) Sensors and Software: Understanding Processing Code Structure, variables and flow control, Interfacing to the Real World

1. CREATIVE CODING PLATFORMS

There's no reason why they can't choose the language in the same way they do for a desktop project. If a Raspberry Pi is running Linux, it's behaviour is not that different from a desktop. By the time the debate on languages makes its way to the servers, there's no difference there, either. They speak with the hubs and sensors—usually with some kind of microservice architecture—then the data is pushed into standard databases. While the usual suspects of popular languages dominate the IoT space already, the Eclipse survey found at least 14 different languages that were mentioned by 5 percent or more of the developers. Here are some of the top choices that are being used to build the foundations of the next generation of things connected to the Internet.

1.1 Java

The top choice of the Eclipse survey and another survey by embedded-computing.com was Java, a result that's not surprising for a language still known for being "write once, run anywhere." The original project was aimed at set-top boxes, one of the first domains for non-desktop computing. Java's advantages are well known. Developers can create and debug code on their desktop and then move it to any chip with a Java Virtual Machine. That means the code can run not just on places where JVMs are common (servers and smartphones), but also on the smallest machines.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

Java ME, or micro edition, has been available on small phone handsets and other embedded devices since the specification was approved in 2000. It saved space with a very limited collection of class libraries and other tools. Today, most of the focus is on Java SE Embedded, which is much closer in capability to the Standard Edition. Developers can use the latest features of the Java 8 platform and then move their code to a smaller, embedded device. Most of the computing resource savings with Java SE Embedded comes from stripping out the classes needed to display information when the machines can be configured to run headlessly, without a monitor or keyboard. All of the communication goes through the network. There are multiple open-source projects, such as Pi4J and BlueJ, that show how the embedded version of Java runs well, even on chips that seem limited.

1.2 C programming

The syntax is cluttered with punctuation, and there are a million different little mistakes you can make, but the language is still the first choice for many programmers who write for the lowest layer of software, the one closest to the hardware. The language hides nothing from you, and that means you can fiddle with every part of the code to squeeze out the best performance from an underpowered device. Every bit can be flipped. Every value on the stack is available. Just don't make a mistake, because there are few safety nets.

More advanced or bigger devices with full operating systems still use plenty of C code, he added, but he said that other languages such as Java are starting to be used just as frequently. When a smartphone comes from Apple, much of the programming is still done in Objective C, but this will probably be gradually replaced by Swift.

1.3 Python

It started as a scripting language to glue together real code, but it's increasingly used as the main language for many developers. When small devices have enough memory and computational power, the developers are free to choose the language that makes their life easier
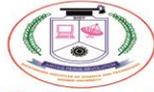
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

and that is more and more often turning out to be Python. The syntax is clean and simple, attracting a greater range of programmers. The language is often the first choice for social scientists and biologists, for instance. When they need a smart device in the lab, they're happy to use a language they know, Python. "Python is the language of choice for one of the most popular microcontrollers on the market, the Raspberry Pi," said Covey. Much of the training literature is written in Python, and many schools use the platform to teach computer programming. If the project is relatively simple and there are no great computational demands, it's possible to build effective tools from the same boards and libraries that are used in elementary schools. There are also versions designed to be even smaller. The MicroPython board and software package is a small microcontroller optimized to run Python on a small board that's only a few square inches.

1.4 JavaScript

While many still think of JavaScript as a language for popping up alert boxes on web pages, the language's relatively newfound popularity on the server makes it a surprisingly popular choice for IoT applications. A full 41.8 percent of the developers in the Eclipse survey chose JavaScript, and 31.5 percent indicated that they were using Node.js in their projects. Much of this work is focused on the servers and gateways or hubs that gather the information and then store it. The smaller smart hubs and sensors that run Linux can usually run Node.js. But even if most of the Node.js code runs on larger machines, there are some efforts designed to bring it to smaller ones. Espruino and Tessel are two examples of microcontrollers that run JavaScript from the beginning. Tessel, for instance, is built around Node.js, making it easy for web developers to move into the IoT without learning a new language.

1.5 Swift

While Swift is still mainly used to build applications for Apple's iOS and macOS devices, the preponderance of these machines means that it's often part of the IoT stack. If you want your things to interact with an iPhone or an iPad, you're probably going to want to build the app in
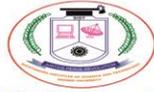
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

Swift (or perhaps its predecessor, Objective C). There are other good reasons to work in this space. Apple wants to make its iOS devices the center of the home network of sensors, so it's been creating libraries and infrastructure that handle much of the work. These libraries are the foundation of its HomeKit platform, which provides support for integrating the data feeds from a network of compatible devices. This means you can concentrate on the details of your task and leave much of the integration overhead to HomeKit.

1.6 PHP

This language may be the first choice of bloggers and website prototypers, but it is also surprisingly popular in the IoT. After the big languages and their cousins such as C#, PHP is the one language that is mentioned the most often by developers in the Eclipse survey; 11.2 percent say they are including PHP code in their stack. While the code's role on the server to juggle microservices is an obvious application, it is also finding some traction at the lowest level. A number of Raspberry Pi developers are talking about starting up a full LAMP stack with Apache, MySQL, and PHP running on top of Linux. They are effectively inverting the paradigm and turning the lowliest thing on the Internet into a full web server.

    2. Open Source Platforms

IoT Platforms

1. Zetta
2. Arduino
3. OpenRemote
4. Node-RED
5. Flutter
6. M2MLabs Mainspring
7. ThingsBoard
8. Kinoma

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

9. Kaa IoT Platform
10. SiteWhere
11. DSA
12. Thinger

IoT platforms and tools are considered as the most significant component of the IoT ecosystem. Any IoT device permits to connect to other IoT devices and applications to pass on information using standard Internet protocols. IoT platforms fill the gap between the device sensors and data networks. It connects the data to the sensor system and gives insights using back-end applications to create a sense of the plenty of data developed by the many sensors. The Internet of Things (IoT) is the future of technology that helps the Artificial intelligence (AI) to regulate and understand the things in a considerably stronger way. We have picked up a mix of best known IoT platforms and tools that help you to develop the IoT projects in an organized way.

2.1 Zetta

Zetta is API based IoT platform based on Node.js. It is considered as a complete toolkit to make HTTP APIs for devices. Zetta combines REST APIs, WebSockets to make data-intensive and real-time applications. The following are some notable features.

- It can run on the cloud, or a PC, or even modest development boards.
- Easy interface and necessary programming to control sensors, actuators, and controllers.
- Allows developers to assemble smartphone apps, device apps, and cloud apps.
- It is developed for data-intensive and real-time applications.
- Turns any machine into an API.

2.2 Arduino

If you are seeking to make a computer that can perceive and exercise stronger control over the real world when related to your ordinary stand-alone computer, then Arduino can be your wise preference. Offering an appropriate blend of IoT hardware and software, Arduino is a simple-

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

to-use IoT platform. It operates through an array of hardware specifications that can be given to interactive electronics. The software of Arduino comes in the plan of the Arduino programming language and Integrated Development Environment (IDE).

2.3 OpenRemote

OpenRemote has introduced a new open-source IoT platform to create professional energy management, crowd management, or more generic asset management applications. Summing up the most important features:

- Generic asset and attribute model with different asset types
- Protocol agents like HTTP REST or MQTT to connect your IoT devices, gateways, or data services or build a missing vendor-specific API.
- Flow editor for data processing, and a WHEN-THEN and a Groovy UI for event-based rules.
- Standard Dashboard for provisioning, automating, controlling, and monitoring your application as well as Web UI components to build project-specific apps.
- Android and iOS consoles which allow you to connect to your phone services, e.g., geofences, and push notifications.
- Edge Gateway solution to connect multiple instances with a central management instance.
- Multi-realms multi-tenant solution, combined with account management and identity service.

2.4 Node-RED

Node-RED is a visual tool for lining the Internet of Things, i.e., wiring together hardware devices, APIs, and online services in new ways. Built on Node.js, Node-RED describes itself as "a visual means for wiring the Internet of Things."
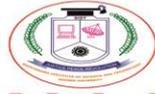
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

It provides developers to connect devices, services, and APIs using a browser-based flow editor. It can run on Raspberry Pi, and further 60,000 modules are accessible to increase its facilities.

2.5 Flutter

Flutter is a programmable processor core for electronics projects, designed for students, and engineers. Flutter's take to glory is it's long-range. This Arduino-based board includes a wireless transmitter that can show up to more than a half-mile. Plus, you don't require a router; flutter boards can interact with each other quickly.

It consists of 256-bit AES encryption, and it's simple to use. Some of the other features are below.

- Fast Performance
- Expressive and Flexible UI
- Native Performance
- Visual finish and functionality of existing widgets.

2.6 M2MLabs Mainspring

M2MLabs Mainspring is an application framework for developing a machine to machines (M2M) applications such as remote control, fleet administration, or smart terminal. Its facilities include flexible design of devices, device structure, connection between machines and applications, validation and normalization of data, long-term data repository, and data retrieval functions.

It's based on Java and the Apache Cassandra NoSQL database. M2M applications can be modeled in hours rather than weeks and subsequently passed on to a high-performance execution environment made on top of a standard J2EE server and the highly-scalable Apache Cassandra database.
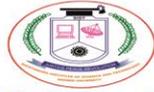
2.7 ThingsBoard

ThingsBoard is for data collection, processing, visualization, and device management. It upholds all standard IoT protocols like CoAP, MQTT, and HTTP as quickly as cloud and on-premise deployments. It builds workflows based on design life cycle events, REST API events, RPC requests.

Let's take a look at the following ThigsBoard features.

- A stable platform that is combining scalability, production, and fault-tolerance.
- Easy control of all connected devices in an exceptionally secure system
- Transforms and normalizes device inputs and facilitates alarms for generating alerts on all telemetry events, restores, and inactivity.
- Enables use-state specific features using customizable rule groups.
- Handles millions of devices at the same time.
- No single moment of failure, as every node in the bundle is exact.
- Multi-tenant installations out-of-the-wrap.
- Thirty highly customized dashboard widgets for successful user access.

2.8 Kinoma

Kinoma, a Marvell Semiconductor hardware prototyping platform, involves three different open source projects. Kimona Create is a DIY construction kit for prototyping electronic devices. Kimona Studio is the development environment that functions with Set up and the Kinoma Platform Runtime. Kimona Connect is a free iOS and Android app that links smartphones and stands with IoT devices.

2.9 Kaa IoT Platform

Kaa is a production-ready, flexible, multi-purpose middleware platform for establishing end-to-end IoT solutions, connected applications, and smart devices. It gives a comprehensive way
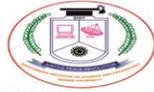
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

of carrying out effective communication, deals with, and interoperation capabilities in connected and intelligent devices.

It mounts from tiny startups to a great enterprise and holds advanced deployment models for multi-cloud IoT solutions. It is primarily based on flexible microservices and readily conforms to virtually any need and application — some other features as below.

- Facilitates cross-device interoperability.
- Performs real-time device control, remote device provisioning, and structure.
- Create cloud services for smart products
- Consists of topic-based warning systems to provide end-users to deliver messages of any predefined format to subscribed endpoints.
- Perform real-time device monitoring
- Manage an infinite quantity of connected devices
- Collect and analyze sensor data

2.10 SiteWhere

SiteWhere platform offers the ingestion, repository, processing, and assimilation of device inputs. It runs on Apache Tomcat and provides highly tuned MongoDB and HBase implementations. You can deploy SiteWhere to cloud platforms like AWS, Azure, GCP, or on-premises. It also supports Kubernetes cluster provisioning.

The following are some of the other features.

- Run any estimate of IoT applications on a single SiteWhere instance
- Spring brings the root configuration framework.
- Add widgets through self-registration, REST services, or in batches.
- InfluxDB for event data storage
- Connect devices with MQTT, Stomp, AMQP and other protocols
- Integrates third-party integration frameworks
- 3Eclipse Californium for CoAP messaging

- HBase for the non-relational datastore
- Grafana to visualize SiteWhere data

2.11 DSA

Distributed Services Architecture (DSA) is for implementing inter-device communication, logic, and efforts at every turn of the IoT infrastructure. It allows cooperation between devices in a distributed manner and sets up a network engineer to share functionality between discrete computing systems. You can manage node attributes, permission, and links from DSLinks.

2.12 Thinger

Thinger.io provides a scalable cloud base for connecting devices. You can deal with them quickly by running the admin console or combine them into your project logic using their REST API. It supports all types of hackers boards such as Raspberry Pi, Intel Edison, ESP8266. Thinger can be integrated with IFTT, and it provides real-time data on a beautiful dashboard.

3. ARDUINO SKETCHES

A sketch is the name that Arduino uses for a program. It's the unit of code that is uploaded to and run on an Arduino board.

3.1 Comments

The first few lines of the Blink sketch are a comment:

At the start of each line is only there to make the comment look pretty, and isn't required). It's there for people reading the code: to explain what the program does, how it works, or why it's written the way it is. It's a good practice to comment your sketches, and to keep the comments

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

up-to-date when you modify the code. This helps other people to learn from or modify your code.

There's another style for short, single-line comments. These start with // and continue to the end of the line. For example, in the line:

COPY

1int ledPin = 13;   // LED connected to digital pin 13

the message "LED connected to digital pin 13" is a comment.

3.2 Variables

A variable is a place for storing a piece of data. It has a name, a type, and a value. For example, the line from the Blink sketch above declares a variable with the name

ledPin

, the type

int

, and an initial value of 13. It's being used to indicate which Arduino pin the LED is connected to. Every time the name ledPin appears in the code, its value will be retrieved. In this case, the person writing the program could have chosen not to bother creating the ledPin variable and instead have simply written 13 everywhere they needed to specify a pin number. The advantage of using a variable is that it's easier to move the LED to a different pin: you only need to edit the one line that assigns the initial value to the variable. Often, however, the value of a variable will change while the sketch runs. For example, you could store the value read from an input into a variable.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in
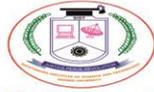
3.3 Functions

A function (otherwise known as a procedure or sub-routine) is a named piece of code that can be used from elsewhere in a sketch. For example, here's the definition of the setup() function from the Blink example:

```
void setup()

{

  pinMode(ledPin, OUTPUT);     // sets the digital pin as output

}
```

The first line provides information about the function, like its name, "setup". The text before and after the name specify its return type and parameters: these will be explained later. The code between the { and } is called the body of the function: what the function does. You can call a function that's already been defined (either in your sketch or as part of the Arduino language).

For example, the line pinMode(ledPin, OUTPUT) and calls the pinMode() function, passing it two parameters: ledPin and OUTPUT. . These parameters are used by the pinMode() function to decide which pin and mode to set. pinMode(), digitalWrite(), and delay(). The pinMode() function configures a pin as either an input or an output. To use it, you pass it the number of the pin to configure and the constant INPUT or OUTPUT. When configured as an input, a pin can detect the state of a sensor like a pushbutton; this is discussed in the Digital Read Serial tutorial. As an output, it can drive an actuator like an LED. The digitalWrite() functions outputs a value on a pin. For example, the line:

digitalWrite(ledPin, HIGH) set the ledPin (pin 13) to HIGH, or 5 volts. Writing a LOW to pin connects it to ground, or 0 volts. The delay() causes the Arduino to wait for the specified number of milliseconds before continuing on to the next line. There are 1000 milliseconds in a second, so the line delay(1000) creates a delay of one second.

setup() and loop() are two special functions that are a part of every Arduino sketch:
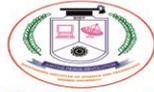
setup() and loop(). The setup() is called once, when the sketch starts. It's a good place to do setup tasks like setting pin modes or initializing libraries. The loop() function is called over and over and is heart of most sketches. You need to include both functions in your sketch, even if you don't need them for anything.

3.4 For Loop Iteration

For instance, this example blinks 6 LEDs attached to the Arduino by using a **for()** loop to cycle back and forth through digital pins 2-7. The LEDS are turned on and off, in sequence, by using both the digitalWrite() and delay() functions. We also call this example "Knight Rider" in memory of a TV-series from the 80's where David Hasselhoff had an AI machine named KITT driving his Pontiac. The car had been augmented with plenty of LEDs in all possible sizes performing flashy effects. In particular, it had a display that scanned back and forth across a line, as shown in this exciting fight between KITT and KARR. This example duplicates the KITT display.

Hardware Required

- Arduino Board

- 6 220 ohm resistors

- 6 LEDs

- hook-up wires

- breadboard

Circuit

Connect six LEDS, with 220 ohm resistors in series, to digital pins 2-7 on your Arduino.
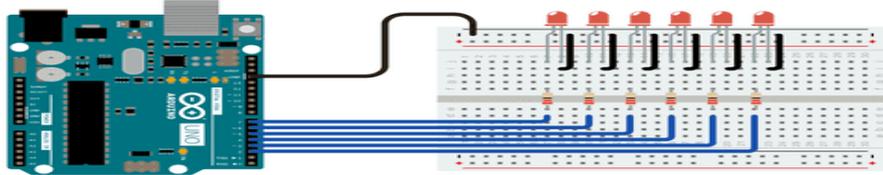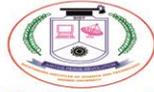
Fig. 1 Arduino with LED

Schematic:

The code below begins by utilizing a

```
for()
```

loop to assign digital pins 2-7 as outputs for the 6 LEDs used.
In the main loop of the code, two

```
for()
```

loops are used to loop incrementally, stepping through the LEDs, one by one, from pin 2 to pin seven. Once pin 7 is lit, the process reverses, stepping back down through each LED.

```
/*

  For Loop Iteration

*/

int timer = 100;          // The higher the number, the slower the timing.

void setup() {

  // use a for loop to initialize each pin as an output:

  for (int thisPin = 2; thisPin < 8; thisPin++) {

    pinMode(thisPin, OUTPUT);
```

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

```
  }

}

void loop() {

  // loop from the lowest pin to the highest:

  for (int thisPin = 2; thisPin < 8; thisPin++) {

    // turn the pin on:

    digitalWrite(thisPin, HIGH);

    delay(timer);

    // turn the pin off:

    digitalWrite(thisPin, LOW);

  }


  // loop from the highest pin to the lowest:

  for (int thisPin = 7; thisPin >= 2; thisPin--) {

    // turn the pin on:

    digitalWrite(thisPin, HIGH);

    delay(timer);

    // turn the pin off:

    digitalWrite(thisPin, LOW);

  }

}
```

![Sathyabama Institute of Science and Technology]
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

4. PYTHON IN INTERNET OF THINGS

The rapidly changing automotive industry has allowed <u>IoT</u> to revolutionize the automotive industry. The IoT makes driving safe and efficient. It has unleashed a range of benefits in agriculture from improving productivity to crop failure risks. The capability IoT to diagnose a problem and avoid failure of the system is helping in preventing the breakdown scenario. The usage of IoT devices has increased year to year. More than 8 billion IoT devices have registered from 2016 to 2018. As per the IoT expert's analysis, at the end of 2020, the count of IoT devices will reach more than 30 billion. And the market value of IoT will reach $7 trillion. As the internet of things (IoT) is evolving continuously, it can be difficult to analyze which tools are more useful in IoT development. Many programming languages are used to develop IoT devices. But which programming languages are most efficient in IoT development. Python language is one among the most popular programming languages for IoT. The coding flexibility & dynamic nature of python helps developers in creating intelligent IoT devices.

IoT definition says that IoT is a network of electronic devices that consist of software, sensors, actuators, and connectivity which allows these things to connect, interact and exchange data. The IoT is like an ecosystem in which physical objects are connected with each other and can be accessed through the internet.

The IoT revolution is going to change the human living style in the future. It will change the way of communication between human being. The primary use of IoT is not only an automatic coffee maker, door opening and switching on lights. IoT collects data from the connected devices which will help in making a fruitful decision. The IoT applications in the hospital will help to collect patient data then analyze with different IoT monitoring machines. This will help doctors for treating patients even from remote areas. The IoT devices allows consumers, businesses and other connected devices to run efficiently.
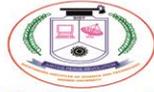
Python is a very popular high-level programming language that focuses on code readability. It is a dynamic and interpreted programming language. Python supports multiple programming paradigm. Generally, Python has fewer steps as compared to Java and C. Python language is also called a general-purpose programming language. Python can be used for software development, mathematics, web development, and system scripting. In Python web development, developers use python on the server-side to develop web applications. Python can handle big and complex data easily. It can work on different platforms Windows, Mac, Linux, and Raspberry Pi. Python is an efficient and fast programming language because it runs on the interpreter. Python can be treated as procedural, functional and object-oriented, etc. With the scripting language, you can develop desktop applications and web applications. It also translated into binary language like java.

4.1 Python in IoT Development

A database is a no-brainer when it comes to most IoT applications. All the IoT devices send data to the internet. Then there should be a database required which can store generated data. MySQL is the go-to relational database for most developers. In this regard, mysqldb is a very convenient little tool which circumvents the need to execute shell commands within a Python script to read and write to a database. And comes Python into the picture. You can also use other programming languages along with Python such as:

- Assembly
- C
- C++
- Java
- Javascript
- PHP
- Python and many more

Before developers were using a java programming language in IoT development. Nowadays Python is quite a favorite language of developers. The reason behind using Python in IoT development is the specific feature that Python provides:

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

Easy to code: With the clear syntax developers get an idea on code identification instead of {};

Simple Syntax: Python has a simple syntax similar to the English Language

Interpreted Language: Python runs on interpreter system. The code can be executed as soon as it is written. Prototyping can be very quick.

Embeddable: Python allows integration with other languages. It is possible to put our code in other programming languages like C++ etc.

Extensible: Python is extensible language. It allows developers to write programs with fewer lines than some other programming languages.

Portable: Python code is portable there is no need to change the code for different machines. You can run one code in many machines

Free and open-source: Python is open-source language. Its source code is freely available to the public you can download it, change it and distribute it.

Community supports: Python has already got its huge response in the market with the above-mentioned features thus provides many users grouped into community to support the advancements further.

Easy to learn: Learning and implementation of python is relatively simple and easy when compared to other native languages like C++ and java.

Easy to debug: Python scripting language is one of the better language to debug than C++ and C. In this source code is executed line by line.

Library support: Python supports large standard libraries. Installation of the libraries is easy, and it saves time.

Python is very easy to learn. You can learn Python by yourself without taking any coaching classes from outside. The software is available on Google at free of cost. You can download Python 3.7.2, the latest version of python. Free Python documentation is available on python.org created by the python community. Across industries, IoT is being increasingly used to streamline processes and make them more efficient. For example, manufacturing production lines and agriculture are great examples of different industries taking advantage of the many benefits of IoT. In the specific case of agriculture, IoT helps coordinate harvesters with trucks that have elevators to efficiently handle grains. For many developers,
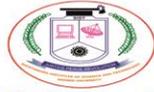
Python is the language of choice in the market. It's easy to learn, has clean syntax, and it is supported by a large community online. In IoT, Python is a great choice for the backend side of development as well as the software development of devices. In addition, Python is available to run on Linux devices and you can use Micro Python for microcontrollers. Some of the many advantages of working with Python for IoT devices are the speed at which you can develop code and a large number of libraries for all kinds of platforms. Python is a great ally to develop device prototypes. Even if you rewrite some of your code during production to C, C++, or Java to improve performance, in general, the system will function perfectly in Python. In this case, you can control the I/O ports on the Raspberry Pi expansion bar. Fortunately, the board supports wireless communication (Bluetooth and WiFi), Ethernet, and you can also connect a monitor to the HDMI outputs, a specialized 3.2" 320x240 TFT LCD, or a low energy consumption E-Ink 2.13" 250x122 display for Raspberry Pi. The controllers, available in a wide range of computing power and budgets, can be chosen for your IoT system - from the fast Raspberry Pi 4 Model B 8GB to the smallest Raspberry Pi Zero, all supporting Python. If necessary, you can install the previous version of Python 2.7 for past compatibility.

To control the I/O ports, you can easily write a couple of lines in Python using the GPIO Zero library: This example shows how easy it is to receive and process signals by pressing the button on the second pin at the moment of pressing and at the moment of releasing. The advantages of using this approach are the availability of a wide range of development tools, libraries, and communications for the most complex devices based on Raspberry Pi, including video processing from cameras.

```
from gpiozero import Button

from time import sleep

button = Button(2)

while True:
```
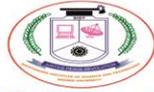
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

```
if button.is_pressed:

    print("Button Press")

else:

    print("Button Release")

sleep(1)
```

4.2 Python on PyBoard

The next great solution for Python in IoT devices is the PyBoard with an STM32F405RG microcontroller. The PyBoard is a compact and powerful electronics development board that runs Micro Python. It connects to your PC via USB, giving you a USB flash drive to save your Python scripts and a serial Python prompt (a REPL) for instant programming. This works with Windows, Mac, and Linux. Micro Python runs bare-metal on the PyBoard, essentially giving you a Python operating system. The built-in pyb module contains functions and classes to control the peripherals available on the board, such as UART, I2C, SPI, ADC, and DAC. ESP8266, ESP32 with Micro python . After installing Python on your computer, you can enter the pip install esptool from the command line. The MicroPython installation process is quite simple: download the firmware from the website and install it using esptool, not forgetting to format the board before installing it. There are already quite a few IDEs for developing with Micropython. The entire development process is carried out on a working computer, then it is compiled and stored in the memory of an ESP8266 or ESP32 microcontroller. See how simple this code might look like: MicroPython will impose a lot of restrictions compared to regular Python but, in general, you can easily write the necessary functionality on the client-side and run it efficiently on ESP microcontrollers. This solution is more cost-effective than purchasing PyBoard.

```
from machine import Pin

import time
```

```
led_pin = Pin(2,Pin.OUT)

while True:

    led_pin.on()

    time.sleep(1)

    led_pin.off()

    time.sleep(1)
```

4.3 MQTT protocol with Python

The most popular connection method for IoT devices is MQTT, a protocol that is effectively implemented with Python. The MQTT protocol is a machine-to-machine (M2M)/Internet of Things connectivity protocol that is designed as an extremely lightweight publish/subscribe messaging transport. It is useful for connections with remote locations where a small code footprint is required and/or network bandwidth is at a premium.

The Eclipse Paho MQTT Python client library implements versions 5.0, 3.1.1, and 3.1 of the MQTT protocol. The Poho library code provides a client class that enables applications to connect to an MQTT broker to publish messages, subscribe to topics, and receive published messages. It also provides some helper functions to make the publishing of one-off messages to MQTT servers straightforward.

4.4 IoT backend on Flask in Python

For a quick, and hassle-free tool to write the backend for your IoT systems and easily set up server-side input/output information, the Flask microframework is here to rescue you and is packed with lots of functionalities. To get started, decide on the requests you need to serve from your IoT devices, set up the Flask microframework, and write a couple of lines of code. The GET method
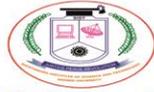
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

will now return information upon request from the client's side. In many cases, you are best off targeting the RESTful protocol when working with your IoT devices. This simplifies the exchange between the system components and allows you to expand the information exchange system in the future. Let's go further. Let's say the following task has arisen: display information from IoT devices on a web page. The Flask microframework will rescue you again with its built-in template mechanism where you can build the needed web page with the data display, including graphics.

The disadvantage of using this approach is the potential lack of initiating the transfer of data from the server to the device. That is, the IoT must independently and periodically pull from the server. Rest easy, as there are solutions to address this risk. You can use web sockets or a Python library for Pushsafer. With PushSafer, you can easily and safely send and receive push notifications in real-time to your iOS, Android, and Windows devices (mobile and desktop), as well as browsers like Chrome, Firefox, Opera, etc.

5. ARM Devices

System-on-chip solutions based on ARM embedded processors address many different market segments including enterprise applications, automotive systems, home networking and wireless technologies. The ARM Cortex™ family of processors provides a standard architecture to address the broad performance spectrum required by these diverse technologies. The ARM Cortex family includes processors based on the three distinct profiles of the ARMv7 architecture; the A profile for sophisticated, high-end applications running open and complex operating systems; the R profile for real-time systems; and the M profile optimized for cost-sensitive and microcontroller applications. The Cortex-M3 processor is the first ARM processor based on the ARMv7-M architecture and has been specifically designed to achieve high system performance in power- and cost-sensitive embedded applications, such as microcontrollers, automotive body systems, industrial control systems and wireless networking, while significantly simplifying programmability to make the ARM architecture an option for even the simplest applications.
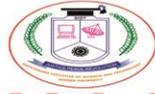
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

There have been several generations of the ARM design. The original ARM1 used a 32-bit internal structure but had a 26-bit address space that limited it to 64 MB of main memory. This limitation was removed in the ARMv3 series, which has a 32-bit address space, and several additional generations up to ARMv7 remained 32-bit. Released in 2011, the ARMv8-A architecture added support for a 64-bit address space and 64-bit arithmetic with its new 32-bit fixed-length instruction set.Arm Ltd. has also released a series of additional instruction sets for different rules; the "Thumb" extension adds both 32- and 16-bit instructions for improved code density, while Jazelle added instructions for directly handling Java bytecodes, and more recently, JavaScript. More recent changes include the addition of simultaneous multithreading (SMT) for improved performance or fault tolerance.

Due to their low costs, minimal power consumption, and lower heat generation than their competitors, ARM processors are desirable for light, portable, battery-powered devices including smartphones, laptops and tablet computers, as well as other embedded systems. However, ARM processors are also used for desktops and servers, including the world's fastest supercomputer. With over 180 billion ARM chips produced, as of 2021, ARM is the most widely used instruction set architecture (ISA) and the ISA produced in the largest quantity. Currently, the widely used Cortex cores, older "classic" cores, and specialised SecurCore cores variants are available for each of these to include or exclude optional capabilities.

5.1 ARM1

Acorn chose VLSI Technology as the "silicon partner",as they were a source of ROMs and custom chips for Acorn. Acorn provided the design and VLSI provided the layout and production. The first samples of ARM silicon worked properly when first received and tested on 26 April 1985. Known as ARM1, these versions ran at 6 MHz. The first ARM application was as a second processor for the BBC Micro, where it helped in developing simulation software to finish development of the support chips (VIDC, IOC, MEMC), and sped up the CAD software used in ARM2 development. Wilson subsequently rewrote BBC BASIC in ARM assembly language. The in-depth knowledge gained from designing the instruction set enabled the code to be very dense, making ARM BBC BASIC an extremely good test for any ARM emulator.
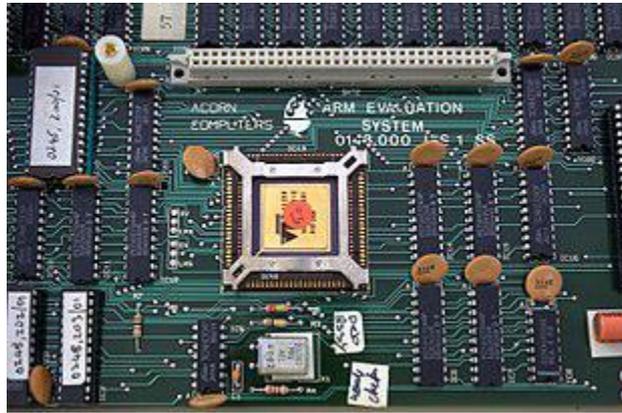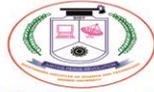
Fig. 2 ARM1 processor

5.2 ARM2

The result of the simulations on the ARM1 boards led to the late 1986 introduction of the ARM2 design running at 8 MHz, and the early 1987 speed-bumped version at 10 to 12 MHz.[b] A significant change in the underlying architecture was the addition of a Booth multiplier, whereas previously multiplication had to be carried out in software.[37] Additionally, a new Fast Interrupt reQuest mode, FIQ for short, allowed registers 8 through 14 to be replaced as part of the interrupt itself. This meant FIQ requests did not have to save out their registers, further speeding interrupts. The ARM2 was roughly seven times the performance of a typical 7 MHz 68000-based system like the Commodore Amiga or Macintosh SE. It was twice as fast as a Intel 80386 running at 16 MHz, and about the same speed as a multi-processor VAX-11/784 supermini. The only systems that beat it were the Sun SPARC and MIPS R2000 RISC-based workstations.[39] Further, as the CPU was designed for high-speed I/O, it dispensed with many of the support chips seen in these machines, notably, it lacked any dedicated direct memory access (DMA) controller which was often found on workstations. The graphics system was also simplified based on the same set of underlying assumptions about memory and timing. The result was a dramatically simplified design, offering performance on par with expensive workstations but at a price point similar to contemporary desktops. The
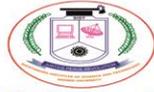
SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

ARM2 featured a 32-bit data bus, 26-bit address space and 27 32-bit registers. The ARM2 had a transistor count of just 30,000, compared to Motorola's six-year-older 68000 model with around 40,000.

This simplicity enabled low power consumption, yet better performance than the Intel 80286. A successor, ARM3, was produced with a 4 KB cache, which further improved performance. The address bus was extended to 32 bits in the ARM6, but program code still had to lie within the first 64 MB of memory in 26-bit compatibility mode, due to the reserved bits for the status flags.

In order to achieve higher performance, processors can either work hard or work smart. Pushing higher clock frequencies may increase performance but is also accompanied by higher power consumption and design complexity. On the other hand, higher compute efficiency at slower clock speeds results in simpler and lower power designs that can perform the same tasks. At the heart of the Cortex-M3 processor is an advanced 3-stage pipeline core, based on the Harvard architecture, that incorporates many new powerful features such as branch speculation, single cycle multiply and hardware divide to deliver an exceptional Dhrystone benchmark performance of 1.25 DMIPS/MHz. The Cortex-M3 processor also implements the new Thumb®-2 instruction set architecture, helping it to be 70% more efficient per MHz than an ARM7TDMI-S® processor executing Thumb instructions.

The Cortex-M3 processor has been designed to be fast and easy to program, with the users not required to write any assembler code or have deep knowledge of the architecture to create simple applications. The processor has a simplified stack-based programmer's model which still maintains compatibility with the traditional ARM architecture but is analogous to the systems employed by legacy 8- and 16-bit architectures, making the transition to 32-bit easier. Additionally a hardware based interrupt scheme means that writing interrupt service routines (handlers) becomes trivial, and that start-up code is now significantly simplified as no assembler code register manipulation is required. Key new features in the underlying Thumb-
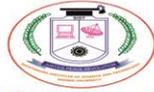
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

2 Instruction Set Architecture (ISA) implement C code more naturally, with native bitfield manipulation, hardware division and If/Then instructions. Further, from a development perspective, Thumb-2 instructions speed up development and simplify long term maintenance and support of compiled objects through automatic optimization for both performance and code density, without the need for complex interworking between code compiled for ARM or Thumb modes.

ARM6

In the late 1980s, Apple Computer and VLSI Technology started working with Acorn on newer versions of the ARM core. In 1990, Acorn spun off the design team into a new company named Advanced RISC Machines Ltd.,which became ARM Ltd. when its parent company, Arm Holdings plc, floated on the London Stock Exchange and NASDAQ in 1998.[46] The new Apple- ARM work would eventually evolve into the ARM6, first released in early 1992. Apple used the ARM6-based ARM610 as the basis for their Apple Newton PDA.
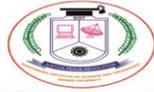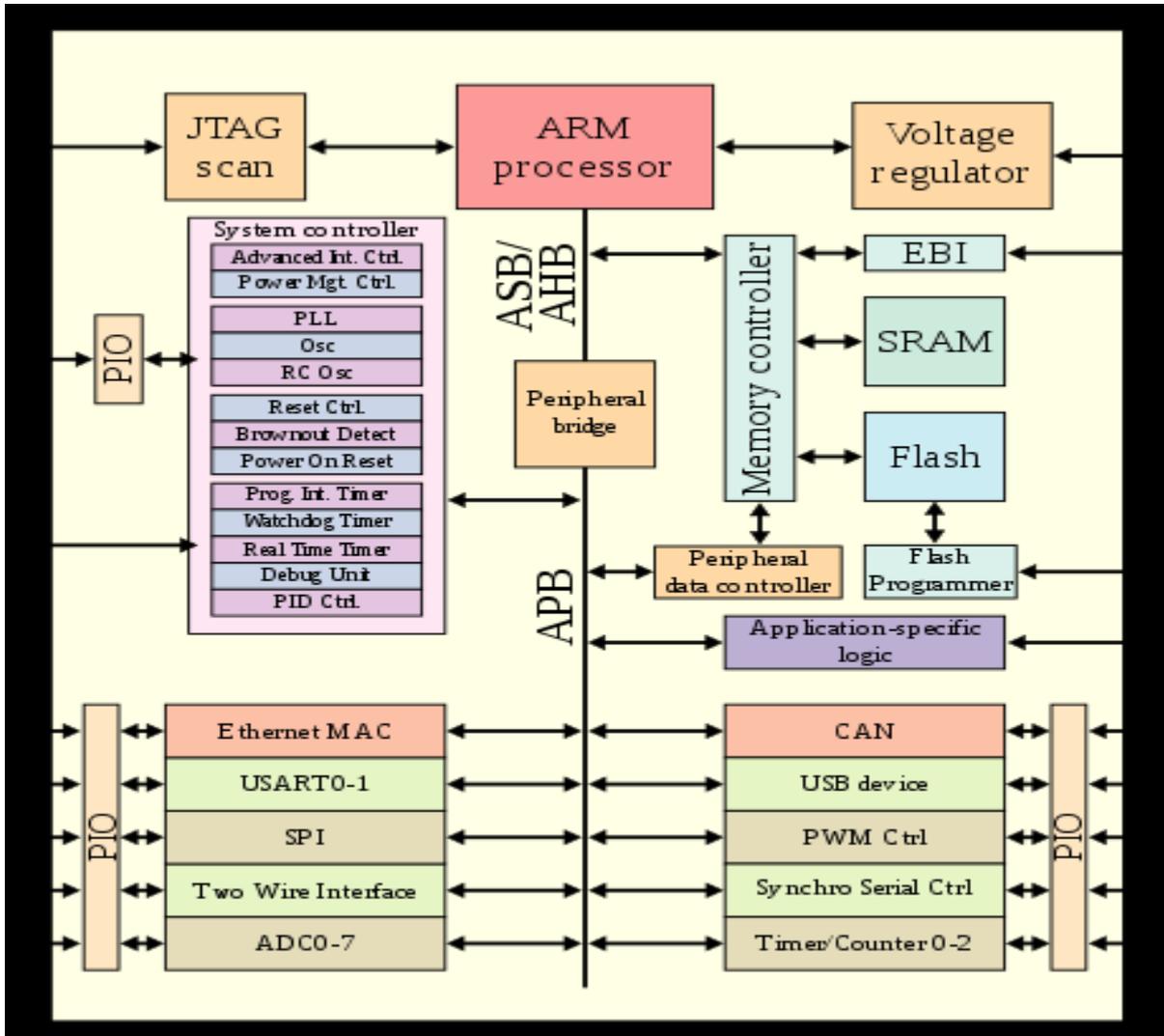
Fig. 3 Microprocessor-based system on a chip

Embedded systems typically have no graphical user interface making software debug a special challenge for programmers. In-circuit Emulator (ICE) units have traditionally been used as plug-in devices to provide a window into the system through a familiar PC interface. As systems get smaller and more complex, physically attaching such debug units is no longer a viable solution. The Cortex-M3 processor implements debug technology in the hardware itself with several integrated components that facilitate quicker debug with trace & profiling, breakpoints, watchpoints and code patching, significantly reducing time to market.
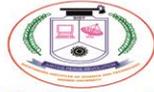
6. SENSORS AND SOFTWARE

Sensors are devices that detect and respond to changes in an environment. Inputs can come from a variety of sources such as light, temperature, motion and pressure. Sensors output valuable information and if they are connected to a network, they can share data with other connected devices and management systems. Sensors are crucial to the operation of many of today's businesses. They can warn you of potential problems before they become big problems, allowing businesses to perform predictive maintenance and avoid costly downtime. The data from sensors can also be analyzed for trends allowing business owners to gain insight into crucial trends and make informed evidence-based decisions. Sensors come in many shapes and sizes. Some are purpose-built containing many built-in individual sensors, allowing you to monitor and measure many sources of data. In brownfield environments, it's key for sensors to include digital and analog inputs so that they can read data from legacy sensors.

6.1 Temperature Sensors

In the past, IoT temperature sensors have been used for heat, ventilation, and air conditioning systems (HVAC), refrigerators, and other similar devices used for environmental control. However, the emergence of IoT has seen its role expand. Nowadays, you can find temperature sensors throughout industries such as manufacturing and agriculture. How are they used within these industries? In the manufacturing process, there are many machines that need both specific environment temperatures and a certain device temperature. Using an IoT temperature sensor ensures both of these remain optimal.

When it comes to agriculture, the soil temperature is crucial for crop growth. Previously monitoring this would have been a tiresome, manual process. Now temperature sensors make it easier to monitor and control remotely via an IoT application. This helps enable the mass production of plants.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

6.2. Pressure Sensors

An IoT pressure sensor is any device that senses pressure and converts it into an electric signal. The level of voltage given out by the sensor depends on the level of pressure applied. These sensors enable IoT systems that monitor systems and devices that are pressure propelled. If there's any deviation from standard pressure ranges, the device notifies the administrator of the problem.

These are commonly found and used in the maintenance of whole water and heating systems. This is because they can easily detect any fluctuations or drops in pressure. However, pressure sensors are also used within the manufacturing industry too.

6.3 Accelerometers

An IoT accelerometer detects – or senses – an object's acceleration. In other words, the rate of change of an object's velocity with respect to time. On top of acceleration, the accelerometer can also detect changes in gravity. Typical uses of accelerometers in IoT include smart pedometers and monitoring driving fleets. However, they're present in millions of devices now, including smartphones. They can also be used for anti-theft protection. Accelerometers alert the system they're used in if an object that should be stationary moves.

6.4. Proximity Sensors

IoT proximity sensors provide non-contact detection of objects that are in close proximity to the sensor. They usually do this by emitting electromagnetic fields or beams of radiation such as infrared. These are frequently found within the retail industry. This is because they can detect motion and the correlation between customers and the product they might be interested in. In turn, the user can be notified of any discount, special offers, or similar products located near the sensor. Proximity sensors may also be found in parking lots of malls, stadiums, or airports to indicate available parking spots.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

6.5. Humidity Sensors

IoT humidity sensors measure the amount of water vapor in the air. In scientific terms, they measure Relative Humidity (RH). This kind of sensor is usually used in addition to IoT temperature sensors when a manufacturing process requires absolute perfect working conditions. They're usually found in heating, ventilation, and air conditioning (HVAC) systems – in both the home and business settings. However, they are also used by meteorological centers to report and predict the weather.

6.6. Image Sensors

IoT image sensors are used to convert images into electronic signals. These are then either displayed or become electronically stored files. The most common use of image sensors is in digital cameras and IoT WiFi modules. Popular IoT image sensor manufacturers include:

7. Level Sensors

Level sensors are used to detect the levels of certain types of objects. These include liquids, granular materials, and powders.

As you can imagine, this kind of sensor is useful and used in many different industries and applications. These include:

- Oil manufacturing
- Beverage manufacturing
- Food manufacturing
- Water treatment

8. Gas Sensors

Gas sensors monitor and detect changes in the air. These sensors are vital to our safety as they're able to detect the presence of potentially harmful or even toxic gases.
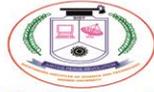
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

Gas sensors are most commonly used within the mining, oil and gas, and chemical research. However, gas sensors are also prominent in most homes via carbon dioxide detectors.

6.9. Infrared Sensors

An Infrared Sensor – otherwise known as an IR sensor – scan and sense characteristics within their surroundings. They do this by either emitting or detecting infrared radiation. In addition, these IoT sensors are also able to measure the heat coming off of objects. Infrared sensors can be adapted for several different IoT applications. However, their most common usage has been within the healthcare industry. For instance, infrared sensors can be used to monitor or blood flow or blood pressure.

10. Motion Detector Sensors

Not to be confused with proximity sensors, motion detector sensors are used to detect physical movement in a given area. In turn, this then sets off an electronic signal.

The most obvious use of this is within the security industry. Businesses use motion detector sensors in areas where there should be no movement. The sensor kicks in and alerts the systems administrator – or security guard – when there is any form of movement.

Besides security, these IoT sensors can also be found in many devices within modern commercial buildings. These include:

- Boom barriers
- Automated sinks, hand dryers, and towel dispensers
- Energy management systems

7. PROCESSING OVERFLOW

Processing is a simple programming environment that was created to make it easier to develop visually oriented applications with an emphasis on animation and providing users with instant feedback through interaction. The developers wanted a means to "sketch" ideas in code. As its capabilities have expanded over the past decade, Processing has come to be used for
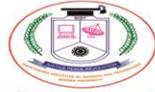
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

more advanced production-level work in addition to its sketching role. Originally built as a domain-specific extension to Java targeted towards artists and designers, Processing has evolved into a full-blown design and prototyping tool used for large-scale installation work, motion graphics, and complex data visualization.

Processing is based on Java, but because program elements in Processing are fairly simple, you can learn to use it even if you don't know any Java. If you're familiar with Java, it's best to forget that Processing has anything to do with Java for a while, until you get the hang of how the API works.

An important goal for the project was to make this type of programming accessible to a wider audience. For this reason, Processing is free to download, free to use, and open source. But projects developed using the Processing environment and core libraries can be used for any purpose. This model is identical to GCC, the GNU Compiler Collection. GCC and its associated libraries (e.g. libc) are open source under the GNU Public License (GPL), which stipulates that changes to the code must be made available. However, programs created with GCC (examples too numerous to mention) are not themselves required to be open source. Processing consists of:

- The Processing Development Environment (PDE). This is the software that runs when you double-click the Processing icon. The PDE is an Integrated Development Environment (IDE) with a minimalist set of features designed as a simple introduction to programming or for testing one-off ideas.
- A collection of functions (also referred to as commands or methods) that make up the "core" programming interface, or API, as well as several libraries that support more advanced features such as sending data over a network, reading live images from a webcam, and saving complex imagery in PDF format.
- A language syntax, identical to Java but with a few modifications.
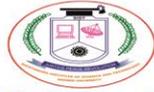
For this reason, references to "Processing" can be somewhat ambiguous. Are we talking about the API, the development environment, or the web site? We'll be careful in this text when referring to each.

7.1 Sketching with Processing

A Processing program is called a *sketch*. The idea is to make Java-style programming feel more like scripting, and adopt the process of scripting to quickly write code. Sketches are stored in the *sketchbook*, a folder that's used as the default location for saving all of your projects. Sketches that are stored in the sketchbook can be accessed from File → Sketchbook. Alternatively, File → Open... can be used to open a sketch from elsewhere on the system.

Advanced programmers need not use the PDE, and may instead choose to use its libraries with the Java environment of choice. However, if you're just getting started, it's recommended that you use the PDE for your first few projects to gain familiarity with the way things are done. While Processing is based on Java, it was never meant to be a Java IDE with training wheels. To better address our target audience, the conceptual model (how programs work, how interfaces are built, and how files are handled) is somewhat different from Java. The Processing equivalent of a "Hello World" program is simply to draw a line:

line(15, 25, 70, 90);

Enter this example and press the Run button, which is an icon that looks like the Play button from any audio or video device. Your code will appear in a new window, with a gray background and a black line from coordinate (15, 25) to (70, 90). The (0, 0) coordinate is the upper left-hand corner of the display window. Building on this program to change the size of the display window and set the background color, type in the code below:

size(400, 400);
background(192, 64, 0);
stroke(255);
line(150, 25, 270, 350);
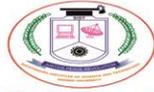
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

This version sets the window size to 400 x 400 pixels, sets the background to an orange-red, and draws the line in white, by setting the stroke color to 255. By default, colors are specified in the range 0 to 255. Other variations of the parameters to the stroke() function provide alternate results.

stroke(255);   // sets the stroke color to white

stroke(255, 255, 255);   // identical to the line above

stroke(255, 128, 0);   // bright orange (red 255,green 128, blue 0)

stroke(#FF8000);   // bright orange as a web color

stroke(255, 128, 0, 128);   // bright orange with 50% transparency

The same alternatives work for the fill() function, which sets the fill color, and the background() function, which clears the display window. Like all Processing functions that affect drawing properties, the fill and stroke colors affect all geometry drawn to the screen until the next fill and stroke functions.

A program written as a list of statements (like the previous examples) is called a *static* sketch. In a static sketch, a series of functions are used to perform tasks or create a single image without any animation or interaction. Interactive programs are drawn as a series of frames, which you can create by adding functions titled setup() and draw() as shown in the code below. These are built-in functions that are called automatically.

```
void setup() {
  size(400, 400);
  stroke(255);
  background(192, 64, 0);
}
void draw() {
  line(150, 25, mouseX, mouseY);
}
```

The setup() block runs once, and the draw() block runs repeatedly. As such, setup() can be used for any initialization; in this case, setting the screen size, making the background orange,
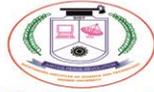
and setting the stroke color to white. The draw() block is used to handle animation. The size() function must always be the first line inside setup().

Because the background() function is used only once, the screen will fill with lines as the mouse is moved. To draw just a single line that follows the mouse, move the background() function to the draw() function, which will clear the display window (filling it with orange) each time draw() runs.

```
void setup() {
  size(400, 400);
  stroke(255);
}


void draw() {
  background(192, 64, 0);
  line(150, 25, mouseX, mouseY);
}
```

Static programs are most commonly used for extremely simple examples, or for scripts that run in a linear fashion and then exit. For instance, a static program might start, draw a page to a PDF file, and exit. Most programs will use the setup() and draw() blocks. More advanced mouse handling can also be introduced; for instance, the mousePressed() function will be called whenever the mouse is pressed. In the following example, when the mouse is pressed, the screen is cleared via the background() function:

```
void setup() {
  size(400, 400);
  stroke(255);
}
void draw() {
  line(150, 25, mouseX, mouseY);
}
```
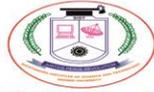
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

```
void mousePressed() {
  background(192, 64, 0);
}
```

Exporting and distributing your work

One of the most significant features of the Processing environment is its ability to bundle your sketch into an application with just one click. Select File → Export Application to package your current sketch as an application. This will bundle your sketch as an application for Windows, Mac OS X, or Linux depending on which operating system you're exporting from. The application folders are overwritten whenever you export—make a copy or remove them from the sketch folder before making changes to the contents of the folder. Alternatively, you can turn off the automatic file erasure in the Preferences.

Creating images from your work

If you don't want to distribute the actual project, you might want to create images of its output instead. Images are saved with the saveFrame() function. Adding saveFrame() at the end of draw() will produce a numbered sequence of TIFF-format images of the program's output, named *screen-0001.tif*, *screen-0002.tif*, and so on. A new file will be saved each time draw() runs — watch out, this can quickly fill your sketch folder with hundreds of files. You can also specify your own name and file type for the file to be saved with a function like:

saveFrame("output.png")

To do the same for a numbered sequence, use # (hash marks) where the numbers should be placed:

saveFrame("output-####.png");

For high quality output, you can write geometry to PDF files instead of the screen, as described in the later section about the size() function.
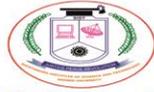
Examples and reference

While many programmers learn to code in school, others teach themselves and learn on their own. Learning on your own involves looking at lots of other code: running, altering, breaking, and enhancing it until you can reshape it into something new. With this learning model in mind, the Processing software download includes hundreds of examples that demonstrate different features of the environment and API.

The examples can be accessed from the File → Examples menu. They're grouped into categories based on their function (such as Motion, Typography, and Image) or the libraries they use (PDF, Network, and Video).

Find an interesting topic in the list and try an example. You'll see functions that are familiar, e.g. stroke(), line(), and background(), as well as others that have not yet been covered. To see how a function works, select its name, and then right-click and choose Find in Reference from the pop-up menu (Find in Reference can also be found beneath the Help menu). This will open the reference for that function in your default web browser.

In addition to a description of the function's syntax, each reference page includes an example that uses the function. The reference examples are much shorter (usually four or five lines apiece) and easier to follow than the longer code examples.

The size() function sets the global variables width and height. For objects whose size is dependent on the screen, always use the width and height variables instead of a number. This prevents problems when the size() line is altered.

```
size(400, 400);
```

```
// The wrong way to specify the middle of the screen
ellipse(200, 200, 50, 50);
```

```
// Always the middle, no matter how the size() line changes
```
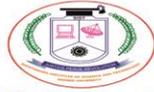
ellipse(width/2, height/2, 50, 50);

In the earlier examples, the size() function specified only a width and height for the window to be created. An optional parameter to the size() function specifies how graphics are rendered. A renderer handles how the Processing API is implemented for a particular output function (whether the screen, or a screen driven by a high-end graphics card, or a PDF file). The default renderer does an excellent job with high-quality 2D vector graphics, but at the expense of speed. In particular, working with pixels directly is slow. Several other renderers are included with Processing, each having a unique function. At the risk of getting too far into the specifics, here's a description of the other possible drawing modes to use with Processing.

size(400, 400, P2D);

The P2D renderer uses OpenGL for faster rendering of two-dimensional graphics, while using Processing's simpler graphics APIs and the Processing development environment's easy application export.

size(400, 400, P3D);

The P3D renderer also uses OpenGL for faster rendering. It can draw three-dimensional objects and two-dimensional object in space as well as lighting, texture, and materials.

size(400, 400, PDF, "output.pdf");

The PDF renderer draws all geometry to a file instead of the screen. To use PDF, in addition to altering your size() function, you must select Import Library, then PDF from the Sketch menu. This is a cousin of the default renderer, but instead writes directly to PDF files.

Loading and displaying data

One of the unique aspects of the Processing API is the way files are handled. The loadImage() and loadStrings() functions each expect to find a file inside a folder named *data*, which is a subdirectory of the sketch folder.
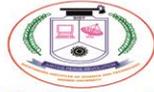
8. ARDUINO PROGRAM CONTROL FLOW, STRUCTURE AND STATEMENTS

Single threaded means lines of codes are executed one code at a time, while sequential means onecode line is executed after another. This method of program execution can be referred to as "**program control flow**" as the name implies, we actually 'mean how to control the flow of the arduino program.' As stated above, the arduino code executes sequentially one after the other down the code lines, however, situations may arise in a program where the program execution is expected to jump steps or return backwards even without getting to the last line of the code, even in situations like this, the rule of thumb is that the program flow must remain logical without errors. To tackle such situations, we need to structure the program in such a way that the flow will be logical and error free using structural logical statements to control the program flow.

8.1 Arduino control structure and control statements

The control structure tells how the codes are organized to enable the program take actions based on certain conditions. Take for example; you want your program to turn on an LED only when a sensor recorded 50°C, you have to structure your code in such a way to execute such command accurately. The method of ensuring such actions in a program can be seen as program control structuring, also known as "**control structure**". This is seen especially when certain conditions are present in a program, just as the one stated above. In other to realize a smart and effective control structure in arduino programming we use control statements.

Control statements are elements (functions) in a source code that control the flow of program execution, either to wait, jump, repeat, etc.

Arduino control statements include:

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

- If statement

- Else statement

- Else if statement

- For statement

- While statement

- Do while statement

- Switch case

- Continue

8.1.1 IF Statement

IF statement is basically the simplest form conditional control statements, it is a conditional statement. An "**if statement**" code evaluates a unique condition, and executes a series of instructions or just an instruction if the condition is true. An if control statement is basically the type of control statement used to program dark activated street lights, the statement evaluates if the environment is dark or not, if the environment is dark, the if statement code instructs the microcontroller to execute a code instruction that will turn on the street light, and if the environment is not dark, the if statement code instructs the microcontroller to execute a code instruction that will not turn off the street light and keep it off until the environment gets dark. The arduino IF statement flow chart and syntax are shown below:

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in
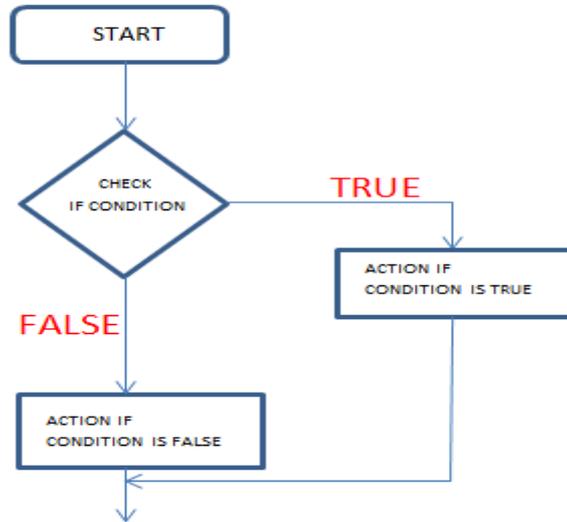
Fig. 4 IF statement flow chart

Fig. 5 Simple IF statement code

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in
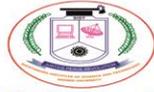
8.1.2 Else Statement

Most time, an IF statement is immediately followed by an **ELSE statement**, the ELSE statement tells the alternate instruction that should be executed when the IF statement is false. Check the sketch below.

```
sketch_jun17a

int sensorPin = A1;
  int ledPin = 2;
void setup()
{
 pinMode (sensorPin, INPUT);
  pinMode (ledPin, OUTPUT);
   }

void loop()
{
 if (analogRead(sensorPin) >= 122)
  {
   digitalWrite (ledPin, HIGH);
}

else
{
   digitalWrite (ledPin, LOW);
}
 }
```

Fig. 6 IF and ELSE statements combine

The sketch in the image above is better than the sketch that has only if statement for designing the dark activated street light.

8.1.3 ELSE IF STATEMENT

"**Else if statement**" is used when we want to check for three different conditions. It includes an IF statement, ELSE IF statement and ELSE statement all in same sketch.

8.1.4 For statement

For statement is also a conditional statement for arduino control structure used for repetitive operation. As the name implies, it is used to carry out a repetitive operation for a true condition. Take for example, when a telecommunication company warns you that "if you try recharging

your phone number with a wrong recharge pin **FOR** 5 times you will be barred" the condition there is, you can try recharging that number repetitively with a wrong recharge pin for up to four times without being barred, however, at the fifth trail, the condition breaks and you are barred.

For statement gives a condition and checks if the condition still holds, if it does, an action is taking and a repetition can take place, this will keep happening until the condition no longer holds the action that is tied to the condition stops being executed. This can be used to increment an event. For instance, we can use a **For statement** to fade an LED up and down. See image below.

```
sketch_jun20a §
 1  int PWMpin =6;
 2  void setup()
 3  {
 4    pinMode (PWMpin, OUTPUT);
 5  }
 6  void loop()
 7  {
 8    int y = 1;
 9    for (int i = 0; i>-1; i=i+y)
10    {
11      analogWrite(PWMpin,  i  );
12      if ( i==225)
13      {
14        y = -1;
15      }
16      delay (10);
17    }
18  }
```

Fig. 7 Fading an LED up and down with for loop

8.1.5 While Statement

A **while statement** is just like an "if statement" except it continues to repeat block of code (a block of code is what is within the curly braces) as long as the condition is true. See example below
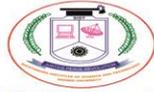
```
1   int sensorPin = A1;
2   int light_1= 2; int light_2 = 3;
3   void setup()
4 { pinMode (sensorPin, INPUT); pinMode (light_1, OUTPUT); pinMode (light_2, OUTPUT); }
5   void loop()
6   {
7 while (analogRead (sensorPin)>100)
8 {
9   digitalWrite (light_1, HIGH);
10  delay (500);
11  digitalWrite(light_1, HIGH);
12  delay(500);
13 }
14 while (analogRead (sensorPin)<100)
15 { digitalWrite (light_2, HIGH);
16  delay (500);
17  digitalWrite(light_2, HIGH);
18  delay(500);}
19  }
```

Done compiling.

Fig. 8 A While Statement used to blink two LEDs for two different conditions

8.1.6 Do While Statement

A do while statement is like the else if statement but works in the same manner as the while loop, except that the condition is tested at the end of the loop, hence, the do statement will always run at least once. See the sketch below:

```
1   int sensorPin = A1;
2   int light_1= 2;
3   void setup()
4 {
5 pinMode (sensorPin, INPUT);
6 pinMode (light_1, OUTPUT);
7 }
8   void loop()
9   {
10 do
11 {
12  digitalWrite (light_1, HIGH);
13  delay (500);
14  digitalWrite(light_1, HIGH);
15  delay(500);
16 }
17 while (analogRead (sensorPin)>100);
18  }
```

Done compiling.

Fig. 9 Do while statement used to blink an LED
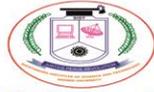
8.1.7 Switch Case Statement

There comes a time in a design, when we wish to have an action taking with respect to a specific result, in a wide range of results. Take for example, let's say you are trying to monitor the level of water in a tank using an ultrasonic sensor, you wish to turn on an LED forvarious levels of the water in the tank. Let's say we are looking at 10 levels. In our arduino code, we would have a variable that records the distance of the water from the ultrasonic sensor, with this distance; we can pick the levels we want. To program the arduino to light LEDs at the various levels we have chosen, we can use the "if statement". With the "if statement", we tell the microcontroller to light an LED if the distance recorded by the ultrasonic sensor is such and such. Using an "if statement" will do a good job, but to make the program elegant, we use the "**switch case**" statement.

Using the "if statement" will require that we repeat the statement for every level we wish to execute. See image below:

```
if (distance ==50)
{
digitalWrite (LED1, HIGH);
}
if (distance ==100)
{
digitalWrite (LED2, HIGH);
}
if (distance ==150)
{
digitalWrite (LED3, HIGH);
}
if (distance ==200)
{
digitalWrite (LED4, HIGH);
}
```

Fig.10 Use of if statement to determine so many conditions
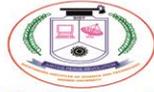
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

```
1  switch(distance)
2  {
3    case 50:
4    digitalWrite (LED1, HIGH);
5    break;
6     case 100:
7    digitalWrite (LED2, HIGH);
8    break;
9     case 150:
10   digitalWrite (LED3, HIGH);
11   break;
12    case 200:
13   digitalWrite (LED4, HIGH);
14   break;
15  }
```

Fig. 11 Using switch case statement to execute conditional codes

Line 1 code initiates the switch statement to begin checking for the conditions. Line 3 code checks for the condition when the variable distance is 50. Line 4 code carries out an action if the case 50 is true, i.e. if distance is equal to 50 Line 5 code breaks the loop for the case condition distance equals 50. Line 6 code checks for case distance equal to 100. The process repeats to check for all the conditions and breaks after the break function is called. A switch case statement makes a sketch look elegant and smart. Continue can be used to jump steps in an iteration process. Let's say you are counting numbers from 1 to 20 and at the same time printing them on the screen, you can use the continue statement to skip printing some numbers in the iteration. See code below.

```
1  for (i = 0; i<=20; i++)
2  {
3    if (i >= 5 && <=8)
4    {
5      continue;
6    }
7    Serial.println (i);
8    delay (1000);
9  }
10
```

Fig. 12 Using the continue statement

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

line 1 code iterates 1 to 20, Line 3 code creates figures to skip (5 to 8), Line 5 code calls skip function Line 7 code prints all numbers from 1 to 20 other than the ones that should be skipped.

9. INTERFACING THE REAL WORLD

IoT applications promise to bring immense value into our lives. With newer wireless networks, superior sensors and revolutionary computing capabilities, the **Internet of Things** could be the next frontier in the race for its share of the wallet. IoT applications are expected to equip billions of everyday objects with connectivity and intelligence. It is already being deployed extensively, the outline of the article is as follows:

Imagine an intelligent device such as a traffic camera. The camera can monitor the streets for traffic congestion, accidents, weather conditions, and communicate this data to a common gateway. This gateway also receives data from other such cameras and relays the information further to a city-wide traffic monitoring system.
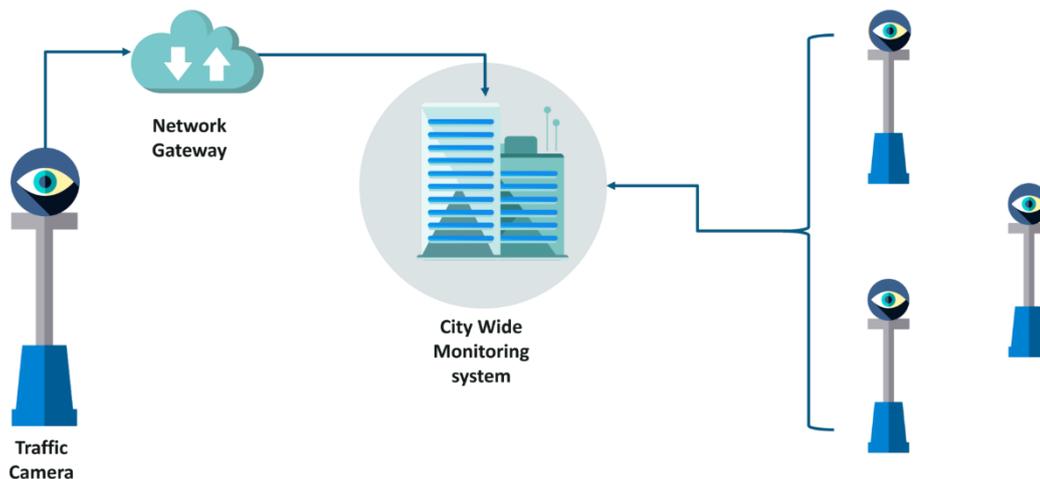
Fig. 12 IoT Applications

Now, take, for instance, the Municipal Corporation decides to repair a certain road. This may cause a traffic congestion on the way to a national highway. This insight is sent to the city-

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

wide traffic monitoring system. Now, considering this is a smart traffic system, it quickly learns and predicts patterns in traffic, with the use of Machine Learning. The smart system can, thus, analyze the situation, predict its impact and relay the information to other cities that connect to the same highway via their own respective smart systems. The Traffic Management System can analyze data acquired and derive routes around the project to avoid bottlenecks. The system could also convey live instructions to drivers through smart devices and radio channels.

The bottom line is a big motivation for starting, investing in, and operating any business, without a sound and solid business models for IoT we will have another bubble, this model must satisfy all the requirements for all kinds of e-commerce; vertical markets, horizontal markets, and consumer markets. One key element is to bundle service with the product, for example, devices like Amazon's Alexa will be considered just another wireless speaker without the services provided like voice recognition, music streaming, and booking Uber service to mention few.

The IoT can find its applications in almost every aspect of our daily life. Below are some of the examples.

1) Prediction of natural disasters: The combination of sensors and their autonomous coordination and simulation will help to predict the occurrence of land-slides or other natural disasters and to take appropriate actions in advance.

2) Industry applications: The IoT can find applications in industry e.g., managing a fleet of cars for an organization. The IoT helps to monitor their environmental performance and process the data to determine and pick the one that need maintenance.

3) Water Scarcity monitoring: The IoT can help to detect the water scarcity at different places. The networks of sensors, tied together with the relevant simulation activities might not only monitor long term water interventions such as catchment area management, but may even be used to alert users of a stream, for instance, if an upstream event, such as the accidental release of sewage into the stream, might have dangerous implications.

4) Design of smart homes: The IoT can help in the design of smart homes e.g., energy consumption management, interaction with appliances, detecting emergencies, home safety and finding things easily, home security etc.
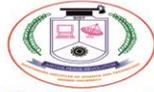
5) Medical applications: The IoT can also find applications in medical sector for saving lives or improving the quality of life e.g., monitoring health parameters, monitoring activities, support for independent living, monitoring medicines intake etc.

6) Agriculture application: A network of different sensors can sense data, perform data processing and inform the farmer through communication infrastructure e.g., mobile phone text message about the portion of land that need particular attention. This may include smart packaging of seeds, fertilizer and pest control mechanisms that respond to specific local conditions and indicate actions. Intelligent farming system will help agronomists to have better understanding of the plant growth models and to have efficient farming practices by having the knowledge of land conditions and climate variability. This will significantly increase the agricultural productivity by avoiding the inappropriate farming conditions.

7) Intelligent transport system design: The Intelligent transportation system will provide efficient transportation control and management using advanced technology of sensors, information and network. The intelligent transportation can have many interesting features such as non-stop electronic highway toll, mobile emergency command and scheduling, transportation law enforcement, vehicle rules violation monitoring, reducing environmental pollution, anti-theft system, avoiding traffic jams, reporting traffic incidents, smart beaconing, minimizing arrival delays etc.

8) Design of smart cities: The IoT can help to design smart cities e.g., monitoring air quality, discovering emergency routes, efficient lighting up of the city, watering gardens etc.

9) Smart metering and monitoring: The IoT design for smart metering and monitoring will help to get accurate automated meter reading and issuance of invoice to the customers. The IoT can be used to design such scheme for wind turbine maintenance and remote monitoring, gas, water as well as environmental metering and monitoring.

10) Smart Security: The IoT can also find applications in the field of security and surveillance e.g., surveillance of spaces, tracking of people and assets, infrastructure and equipment maintenance, alarming etc.

School of Computing

Department of Computer Science and Engineering

UNIT IV - ARDUINO PROGRAMMING FOR
IoT BOARDS - SCSA1407
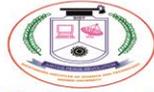
UNIT 4 Programming an Arduino IoT Device

Preparing the development environment (Arduino IDE), Exploring the Arduino language (C/C++) syntax, Coding, compiling, and uploading to the microcontroller, Working with Arduino Communication Modules: Bluetooth Modules, WiFi Modules and I2C and SPI, Interfacing arduino and Blynk via USB : LED Blinking, Controlling a Servomotor.

1. PREPARING THE DEVELOPMENT ENVIRONMENT (ARDUINO IDE)

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, 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. Programs written using Arduino Software (IDE) are called **sketches**. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom righthand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

Upload Compiles your code and uploads it to the configured board. See uploading below for details. Open Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window overwriting its content. Save Saves your sketch. Serial Monitor Opens the serial monitor.

Additional commands are found within the five menus: **File**, **Edit**, **Sketch**, **Tools**, **Help**. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.
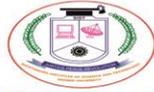
1.1 File

New Creates a new instance of the editor, with the bare minimum structure of a sketch already in place. Open Allows to load a sketch file browsing through the computer drives and folders. Open Recent Provides a short list of the most recent sketches, ready to be opened. Sketchbook Shows the current sketches within the sketchbook folder structure; clicking on any name opens the corresponding sketch in a new editor instance. Examples Any example provided by the Arduino Software (IDE) or library shows up in this menu item. All the examples are structured in a tree that allows easy access by topic or library. Close Closes the instance of the Arduino Software from which it is clicked.

Save Saves the sketch with the current name. If the file hasn't been named before, a name will be provided in a "Save as.." window. Save as... Allows to save the current sketch with a different name. Page Setup It shows the Page Setup window for printing. Print Sends the current sketch to the printer according to the settings defined in Page Setup. Preferences Opens the Preferences window where some settings of the IDE may be customized, as the language of the IDE interface. Quit Closes all IDE windows. The same sketches open when Quit was chosen will be automatically reopened the next time you start the IDE.

1.2 Edit

Undo/Redo Goes back of one or more steps you did while editing; when you go back, you may go forward with Redo. Cut Removes the selected text from the editor and places it into the clipboard. Copy Duplicates the selected text in the editor and places it into the clipboard. Copy for Forum Copies the code of your sketch to the clipboard in a form suitable for posting to the forum, complete with syntax coloring. Copy as HTML Copies the code of your sketch to the clipboard as HTML, suitable for embedding in web pages. Paste Puts the contents of the clipboard at the cursor position, in the editor. Select All Selects and highlights the whole content of the editor. Comment/Uncomment Puts or removes the // comment marker at the beginning of each selected line. Increase/Decrease Indent Adds or subtracts a space at the
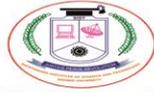
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

beginning of each selected line, moving the text one space on the right or eliminating a space at the beginning. Find Opens the Find and Replace window where you can specify text to search inside the current sketch according to several options. Find Next Highlights the next occurrence - if any - of the string specified as the search item in the Find window, relative to the cursor position. Find Previous Highlights the previous occurrence - if any - of the string specified as the search item in the Find window relative to the cursor position.

Verify/Compile Checks your sketch for errors compiling it; it will report memory usage for code and variables in the console area. Upload Compiles and loads the binary file onto the configured board through the configured Port. Upload Using Programmer This will overwrite the bootloader on the board; you will need to use Tools > Burn Bootloader to restore it and be able to Upload to USB serial port again. However, it allows you to use the full capacity of the Flash memory for your sketch. Please note that this command will NOT burn the fuses. Export Compiled Binary Saves a .hex file that may be kept as archive or sent to the board using other tools. Show Sketch Folder Opens the current sketch folder. Include Library Adds a library to your sketch by inserting #include statements at the start of your code. For more details, see libraries below. Additionally, from this menu item you can access the Library Manager and import new libraries from .zip files.

1.3 ARDINO IDE OVERVIEW:

Program coded in Arduino IDE is called a SKETCH

1. To create a new sketchFile -> New

 To open an existing sketch File -> open ->
There are some basic ready-to-use sketches available in the EXAMPLES section
File -> Examples -> select any program
2. Verify: Checks the code for compilation errors
3. Upload: Uploads the final code to the controller board
4. New: Creates a new blank sketch with basic structure
5. Open: Opens an existing sketch
6. Save: Saves the current sketch

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in
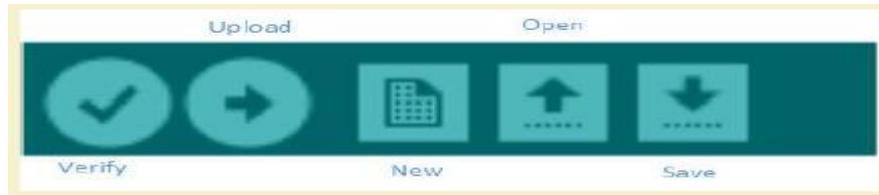
Fig 1.Compilation and Execution

Serial Monitor: Opens the serial console

☐ All the data printed to the console are displayed here

1.4 SKETCH STRUCTURE

Fig. 2 Structure of SKETCH

A sketch can be divided into two parts:

☐ Setup ()

☐ Loop()

The function setup() is the point where the code starts, just like the main() function in C and C++

 I/O Variables, pin modes are initialized in the Setup() function ☐ Loop() function, as the name suggests, iterates the specified task in the program.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

1.5 Arduino Function

Input/Output Functions:

The arduino pins can be configured to act as input or output pins using the pinMode() function

Void setup ()

{

pinMode (pin , mode);

}

 Pin- pin number on the Arduino board Mode- INPUT/OUTPUT

digitalWrite() : Writes a HIGH or LOW value to a digital pin

analogRead() : Reads from the analog input pin i.e., voltage applied across the pin

Character functions such as isdigit(), isalpha(), isalnum(), isxdigit(), islower(), isupper(), isspace() return 1(true) or 0(false)

Delay() function is one of the most common time manipulation function used to provide a delay of specified time. It accepts integer value (time in miliseconds)

1.6 Example Blinking LED

 Arduino controller board, USB connector, Bread board, LED, 1.4Kohm resistor, connecting wires, Arduino IDE

 Connect the LED to the Arduino using the Bread board and the connecting wires

 Connect the Arduino board to the PC using the USB connector

 Select the board type and port Write the sketch in the editor, verify and upload

Connect the positive terminal of the LED to digital pin 12 and the negative terminal to the ground pin (GND) of Arduino Board


```
void setup()
{
pinMode(12, OUTPUT); // set the pin mode
} void loop()
{
```

![Sathyabama Institute of Science and Technology logo]

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

digitalWrite(12, HIGH); // Turn on the LED delay(1000);

digitalWrite(12, LOW); //Turn of the LED delay(1000);

}

Set the pin mode as output which is connected to the led, pin 12 in this case.

Use digitalWrite() function to set the output as HIGH and LOW

Delay() function is used to specify the delay between HIGH-LOW transition of the output

1.7 Preferences

Some preferences can be set in the preferences dialog (found under the **Arduino** menu on the Mac, or **File** on Windows and Linux). The rest can be found in the preferences file, whose location is shown in the preference dialog.

1.8 Boards

The board selection has two effects: it sets the parameters (e.g. CPU speed and baud rate) used when compiling and uploading sketches; and sets and the file and fuse settings used by the burn bootloader command. Some of the board definitions differ only in the latter, so even if you've been uploading successfully with a particular selection you'll want to check it before burning the bootloader. You can find a comparison table between the various boards here.

Arduino Software (IDE) includes the built in support for the boards in the following list, all based on the AVR Core. The Boards Manager included in the standard installation allows to add support for the growing number of new boards based on different cores like Arduino Due, Arduino Zero, Edison, Galileo and so on.

- Arduino Yún An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.

- Arduino Uno An ATmega328P running at 16 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.

![Sathyabama Institute of Science and Technology logo]

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

- Arduino Diecimila or Duemilanove w/ ATmega168 An ATmega168 running at 16 MHz with auto-reset.

- Arduino Nano w/ ATmega328P An ATmega328P running at 16 MHz with auto-reset. Has eight analog inputs.

- Arduino Mega 2560 An ATmega2560 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.

- Arduino Mega An ATmega1280 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.

- Arduino Mega ADK An ATmega2560 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.

- Arduino Leonardo An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.

- Arduino Micro An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.

- Arduino Esplora An ATmega32u4 running at 16 MHz with auto-reset.

- Arduino Mini w/ ATmega328P An ATmega328P running at 16 MHz with auto-reset, 8 Analog In, 14 Digital I/O and 6 PWM.

- Arduino Ethernet Equivalent to Arduino UNO with an Ethernet shield: An ATmega328P running at 16 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.

- Arduino Fio An ATmega328P running at 8 MHz with auto-reset. Equivalent to Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328P, 6 Analog In, 14 Digital I/O and 6 PWM.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

2. EXPLORING THE ARDUINO LANGUAGE (C/C++) SYNTAX

Arduino programs can be divided in three main parts: **Structure, Values** (variables and constants), and **Functions**. In this tutorial, we will learn about the Arduino software program, step by step, and how we can write the program without any syntax or compilation error.

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

- Setup( ) function
- Loop( ) function
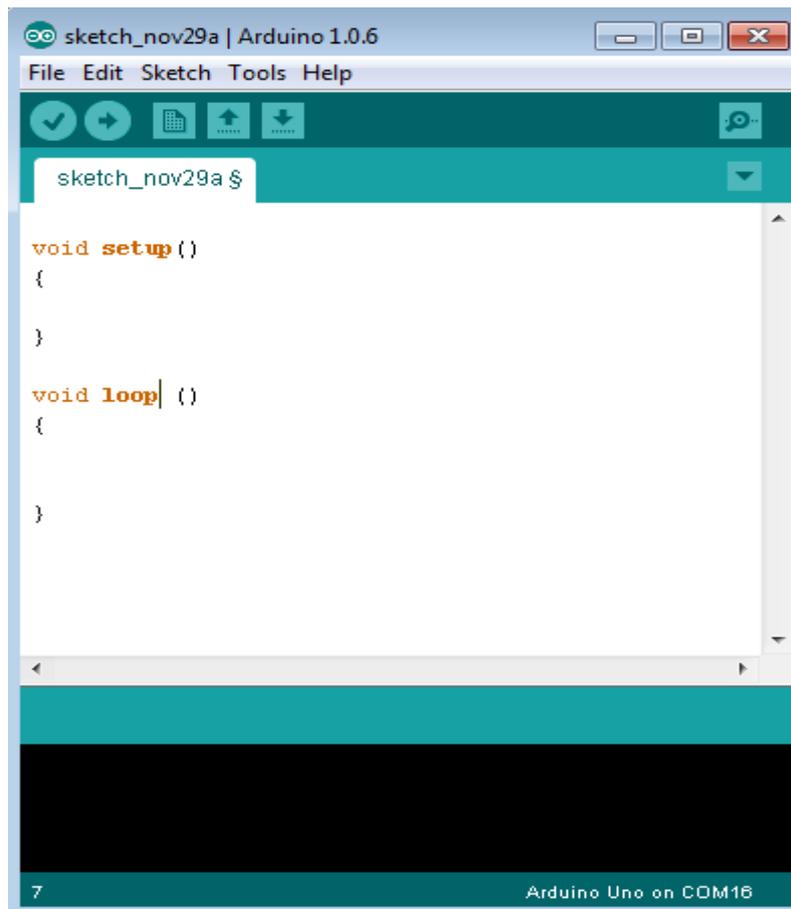
Fig. 3 setup() Function
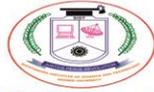
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

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. Arduino, natively, supports a language that we call the Arduino Programming Language, or Arduino Language. This language is based upon the Wiring development platform, which in turn is based upon Processing, which if you are not familiar with, is what p5.js is based upon. It's a long history of projects building upon other projects, in a very Open Source way. The Arduino IDE is based upon the Processing IDE, and the Wiring IDE which builds on top of it.

When we work with Arduino we commonly use the Arduino IDE (Integrated Development Environment), a software available for all the major desktop platforms (macOS, Linux, Windows), which gives us 2 things: a programming editor with integrated libraries support, and a way to easily compile and load our Arduino programs to a board connected to the computer. The Arduino Programming Language is basically a framework built on top of C++.

```
#define LED_PIN 13

void setup() {
  // Configure pin 13 to be a digital output
  pinMode(LED_PIN, OUTPUT);
}

void loop() {
  // Turn on the LED
  digitalWrite(LED_PIN, HIGH);
  // Wait 1 second (1000 milliseconds)
  delay(1000);
  // Turn off the LED
  digitalWrite(LED_PIN, LOW);
```
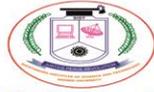
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

```
// Wait 1 second

delay(1000);

}
```

2.1 Handling I/O

The following functions help with handling input and output from your Arduino device.

2.1.1 Digital I/O

- digitalRead() reads the value from a digital pin. Accepts a pin number as a parameter, and returns the HIGH or LOW constant.
- digitalWrite() writes a HIGH or LOW value to a digital output pin. You pass the pin number and HIGH or LOW as parameters.
- pinMode() sets a pin to be an input, or an output. You pass the pin number and the INPUT or OUTPUT value as parameters.
- pulseIn() reads a digital pulse from LOW to HIGH and then to LOW again, or from HIGH to LOW and to HIGH again on a pin. The program will block until the pulse is detected. You specify the pin number and the kind of pulse you want to detect (LHL or HLH). You can specify an optional timeout to stop waiting for that pulse.
- pulseInLong() is same as pulseIn(), except it is implemented differently and it can't be used if interrupts are turned off. Interrupts are commonly turned off to get a more accurate result.
- shiftIn() reads a byte of data one bit at a time from a pin.
- shiftOut() writes a byte of data one bit at a time to a pin.
- tone() sends a square wave on a pin, used for buzzers/speakers to play tones. You can specify the pin, and the frequency. It works on both digital and analog pins.
- noTone() stops the tone() generated wave on a pin.

2.1.2 Analog I/O

- analogRead() reads the value from an analog pin.
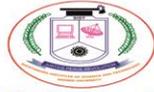
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

- analogReference() configures the value used for the top input range in the analog input, by default 5V in 5V boards and 3.3V in 3.3V boards.

- analogWrite() writes an analog value to a pin

- analogReadResolution() lets you change the default analog bits resolution for analogRead(), by default 10 bits. Only works on specific devices (Arduino Due, Zero and MKR)

- analogWriteResolution() lets you change the default analog bits resolution for analogWrite(), by default 10 bits. Only works on specific devices (Arduino Due, Zero and MKR)

- Time functions

- delay() pauses the program for a number of milliseconds specified as parameter

- delayMicroseconds() pauses the program for a number of microseconds specified as parameter

- micros() the number of microseconds since the start of the program. Resets after ~70 minutes due to overflow

- millis() the number of milliseconds since the start of the program. Resets after ~50 days due to overflow

2.1.3 Math functions

- abs() the absolute value of a number

- constrain() constrains a number to be within a range, see usage

- map() re-maps a number from one range to another, see usage

- max() the maximum of two numbers

- min() the minimum of two numbers

- pow() the value of a number raised to a power

- sq() the square of a number

- sqrt() the square root of a number

- cos() the cosine of an angle

- sin() the sine of an angle

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in
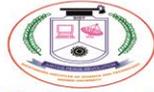
- tan() the tangent of an angle

2.1.4 Alphanumeric characters

- isAlpha() checks if a char is alpha (a letter)

- isAlphaNumeric() checks if a char is alphanumeric (a letter or number)

- isAscii() checks if a char is an ASCII character

- isControl() checks if a char is a control character

- isDigit() checks if a char is a number

- isGraph() checks if a char is a printable ASCII character, and contains content (it is not a space, for example)

- isHexadecimalDigit() checks if a char is an hexadecimal digit (A-F 0-9)

- isLowerCase() checks if a char is a letter in lower case

- isPrintable() checks if a char is a printable ASCII character

- isPunct() checks if a char is a punctuation (a comma, a semicolon, an exclamation mark etc)

- isSpace() checks if a char is a space, form feed \f, newline \n, carriage return \r, horizontal tab \t, or vertical tab \v.

- isUpperCase() checks if a char is a letter in upper case

- isWhitespace() checks if a char is a space character or an horizontal tab \t

- Random numbers generation

- random() generate a pseudo-random number

- randomSeed() initialize the pseudo-random number generator with an arbitrary initial number

3. BLUETOOTH MODULES

Bluetooth Low Energy Modules available at a reasonable cost, most of these modules are not compatible with existing devices that support the classic Bluetooth. The HC-05 is an expensive

module that is compatible with wide range of devices including smartphone, laptops and tablets. Adding a Bluetooth to Arduino can take your project to the next level. It opens up lots of possibilities for user interface (UI) and communication.
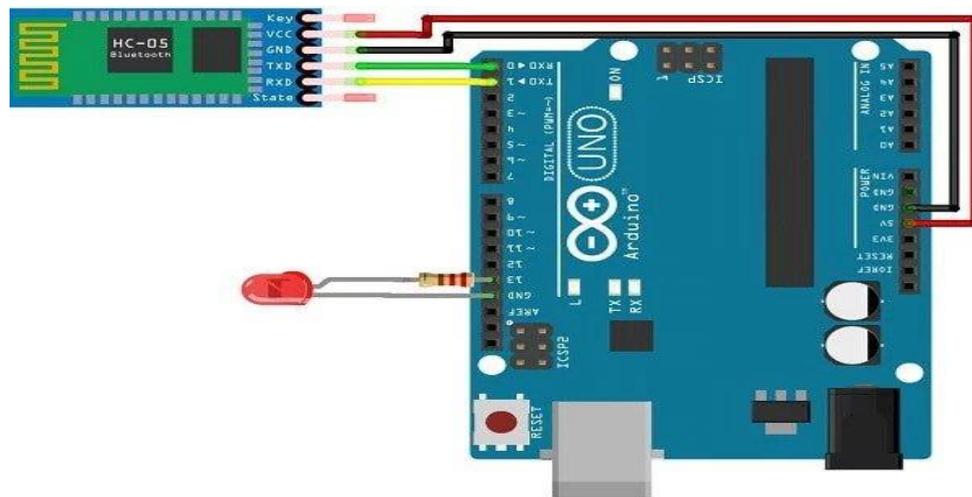
Fig. 5 Arduino with Bluetooth Connection

There are three main parts to this module. An Android smartphone, a Bluetooth transceiver, and an Arduino. HC 05/06 works on serial communication. The Android app is designed to send serial data to the Arduino Bluetooth module when a button is pressed on the app. The Arduino Bluetooth module at the other end receives the data and sends it to the Arduino through the TX pin of the Bluetooth module (connected to RX pin of Arduino). The code uploaded to the Arduino checks the received data and compares it. If the received data is 1, the LED turns ON. The LED turns OFF when the received data is 0. You can open the serial monitor and watch the received data while connecting.
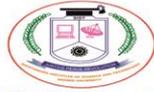
```
 char data = 0;                //Variable for storing received data
void setup()
{
Serial.begin(9600);       //Sets the data rate in bits per second (baud) for serial data transmission
pinMode(13, OUTPUT);        //Sets digital pin 13 as output pin
}
void loop()
{
if(Serial.available() > 0)  // Send data only when you receive data:
{
data = Serial.read();     //Read the incoming data and store it into variable data
Serial.print(data);        //Print Value inside data in Serial monitor
Serial.print("\n");       //New line
if(data == '1')           //Checks whether value of data is equal to 1
digitalWrite(13, HIGH);
else if(data == '0')
digitalWrite(13, LOW);
}
}
```

4. WiFi MODULES

ESP8266WiFi library

ESP8266 is all about Wi-Fi. If you are eager to connect your new ESP8266 module to a Wi-Fi network to start sending and receiving data, this is a good place to start. If you are looking for more in depth details of how to program specific Wi-Fi networking functionality, you are also in the right place. The Wi-Fi library for ESP8266 has been developed based on ESP8266 SDK, using the naming conventions and overall functionality philosophy of the Arduino WiFi library.
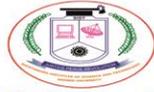
In order to get our ESP8266 to work properly with our Arduino, we need to do some initial programming. Specifically, we will be changing the ESP8266 to work as an access point and a client and changing the baud rate. Since most code samples out there are communicating with the ESP module with a baud rate of 9600, that's what we will use. We will also verify that the ESP8266 module can connect to our router.

With your Arduino Uno connected to your computer, open the serial monitor via the Arduino IDE (ctrl + shift + m). On the bottom of the serial monitor there are dropdowns for line endings and baud rate. Set line endings to "Both NL & CR" and change the baud rate to "115200". Then send the following commands:

1. Verify that the ESP8266 is connected properly.

Command to send: AT

Expected response: OK

2. Change the mode.

Command to send: AT+CWMODE=3

Expected response: OK

3. Connect to your router (Make sure to replace YOUR_SSID and YOUR_WIFI_PASSWORD).

Command to send: AT+CWJAP="YOUR_SSID","YOUR_WIFI_PASSWORD"

Expected response:

WIFI CONNECTED

WIFI GOT IP

OK

4. Set baud rate to 9600.

Command to send: AT+UART=9600,8,1,0,0

Expected response: OK

5. Verify that the ESP8266 is communicating with baud rate of 9600.

Command to send: AT

Expected response: OK

```
#include "WiFiEsp.h"
#include <ArduinoJson.h>

#ifndef HAVE_HWSERIAL1
#include "SoftwareSerial.h"

// set up software serial to allow serial communication to our TX and RX pins
SoftwareSerial Serial1(10, 11);
#endif

// Set  baud rate of so we can monitor output from esp.
#define ESP8266_BAUD 9600

// CHANGE THIS TO MATCH YOUR SETTINGS
char ssid[] = "MY_SSID";
char pass[] = "MY_WIFI_PASSWORD";
i
nt status = WL_IDLE_STATUS;

// Define an esp server that will listen on port 80
WiFiEspServer server(80);

void setup()
{
  // Open up communications for arduino serial and esp serial at same rate
  Serial.begin(9600);
  Serial1.begin(9600);

  // Initialize the esp module
  WiFi.init(&Serial1);
```

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

```
    // Start connecting to wifi network and wait for connection to complete
    while (status != WL_CONNECTED)
    {
        Serial.print("Conecting to wifi network: ");
        Serial.println(ssid);

        status = WiFi.begin(ssid, pass);
    }

    // Once we are connected log the IP address of the ESP module
    Serial.print("IP Address of ESP8266 Module is: ");
    Serial.println(WiFi.localIP());
    Serial.println("You're connected to the network");

    // Start the server
    server.begin();
}

//  Continually check for new clients
void loop()
{
    WiFiEspClient client = server.available();

    // If a client has connected...
    if (client)
    {
        String json = "";
        Serial.println("A client has connected");
```
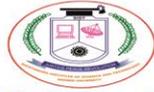
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

```
while (client.connected())
{
    // Read in json from connected client
    if (client.available())
    {
        // ignore headers and read to first json bracket
        client.readStringUntil('{');

        // get json body (everything inside of the main brackets)
        String jsonStrWithoutBrackets = client.readStringUntil('}');

        // Append brackets to make the string parseable as json
        String jsonStr = "{" + jsonStrWithoutBrackets + "}";

        // if we managed to properly form jsonStr...
        if (jsonStr.indexOf('{', 0) >= 0)
        {
            // parse string into json, bufferSize calculated by
https://arduinojson.org/v5/assistant/
            const size_t bufferSize = JSON_OBJECT_SIZE(1) + 20;
            DynamicJsonBuffer jsonBuffer(bufferSize);

            JsonObject &root = jsonBuffer.parseObject(jsonStr);

            // get and print the value of the action key in our json object
            const char *value = root["action"];
            Serial.println(value);

            if (strcmp(value, "on") == 0)
            {
```
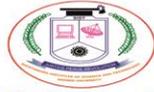
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

```
        // Do something when we receive the on command
        Serial.println("Received on command from client");
      }
    else if (strcmp(value, "off") == 0)
    {
        // Do something when we receive the off command
        Serial.println("Received off command from client");
    }


    // send response and close connection
    client.print(
        "HTTP/1.1 200 OK\r\n"
        "Connection: close\r\n" // the connection will be closed after completion of the
response
        "\r\n");


    client.stop();
  }
  else
  {
    // we were unable to parse json, send http error status and close connection
    client.print(
        "HTTP/1.1 500 ERROR\r\n"
        "Connection: close\r\n"
        "\r\n");


    Serial.println("Error, bad or missing json");
    client.stop();
  }
}
```
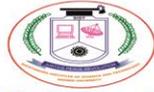
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

```
    }

    delay(10);
    client.stop();
    Serial.println("Client disconnected");
  }
}.
```

Devices that connect to Wi-Fi networks are called stations (STA). Connection to Wi-Fi is provided by an access point (AP), that acts as a hub for one or more stations. The access point on the other end is connected to a wired network. An access point is usually integrated with a router to provide access from a Wi-Fi network to the internet. Each access point is recognized by a SSID (**S**ervice **S**et **ID**entifier), that essentially is the name of network you select when connecting a device (station) to the Wi-Fi.

ESP8266 modules can operate as a station, so we can connect it to the Wi-Fi network. It can also operate as a soft access point (soft-AP), to establish its own Wi-Fi network. When the ESP8266 module is operating as a soft access point, we can connect other stations to the ESP module. ESP8266 is also able to operate as both a station and a soft access point mode. This provides the possibility of building e.g. mesh networks.
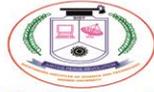
ESP8266 operating in the **Station + Soft Access Point Mode** mode

Fig. 6 ESP8266 Module

5. I2C and SPI

UART, I2C and SPI are one of the most common and basic hardware communication peripherals that makers and electricians use in microcontroller development. Similarly, for the Arduino, they contain UART, I2C and SPI peripheral too. The Arduino Uno Rev 3 is a microcontroller board based on the ATmega328, an 8-bit microcontroller with 32KB of Flash memory and 2KB of RAM.
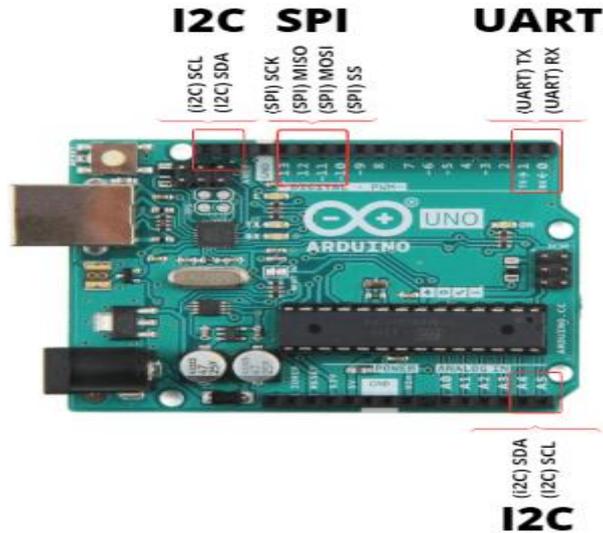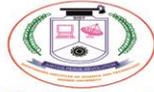
Fig.7 Arduino with I2C and SPI

- It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button

5.1 UART

UART stands for Universal Asynchronous Reception and Transmission and is a simple communication protocol that allows the Arduino to communicate with serial devices. The UART system communicates with digital pin 0 (RX), digital pin 1 (TX), and with another computer via the USB port. This peripheral, found on all Arduino boards, allows the Arduino to directly communicate with a computer thanks to the fact that the Arduino has an onboard USB-to-Serial converter. Therefore, programs written on a Windows, Mac, or Linux OS can be used with an Arduino connected to a USB port as if it was a serial port (serial port communication is trivial compared to USB communication). UART, which stands for Universal Asynchronous Reception and Transmission, is a simple serial communication protocol that allows the host (Arduino) to communicate with serial devices. UART supports bidirectional, asynchronous and serial data transmission. It uses 2 data lines to communicate with each other which are: TX (Pin 1) and RX (Pin 0).
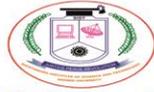
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

- TX – Used for transmitting
- RX – Used for receiving
- They are connected between two devices (eg. USB on Arduino and computer)

UART is found on all types of Arduino boards which allows the Arduino to communicate with a computer due to its onboard USB to Serial converter. If your program is written on a Windows, Mac or Linux OS and wants to use it with your Arduino, just connect them together via their USB port as if it was a serial port.

Advantages and Disadvantages of using UART with Arduino

Advantages of using UART with Arduino

- Simple to operate and use with the Arduino. It is well documented online as it is a widely used method by Arduino users with many resources and tutorials online.
- No clock needed

Disadvantages of using UART with Arduino

- Lower speed compared to I2C and SPI
- Baud rates of each UART must be within 10% of each other to prevent data loss.
- Cannot use multiple master systems like the Arduino and slaves.

5.2 I2C

I2C, which stands for inter-integrated-circuit, is a serial communications protocol specially designed for microcontrollers. While this peripheral is almost never used for PC-device communication, it is incredibly popular with modules and sensors, making it useful for projects that require many parts working together. In fact, I2C allows you to potentially connect up to 128 devices to your main board!. When connecting two circuits to one another, think of the main device as the "master" and the connected devices—such as sensors, pin expansions, and

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

drivers—as "slaves". I2C makes it possible to connect multiple masters and slaves to your board while maintaining a clear communication pathway.

Maintaining a clear communication pathway is possible because I2C uses an address system and a shared bus, meaning many devices can be connected to the exact same wires. However, the Arduino must first select a specific device by transmitting a unique address before sending data. This provides each slave device with what it needs while also supporting multiple masters. I2C uses fewer wires and all data is transmitted on a single wire, keeping your pin count low. The tradeoff for this simplified wiring is slower speeds than SPI.

5.3 SPI

SPI stands for Serial Peripheral Interface. Like I2C, SPI is a different form of serial-communications protocol specially designed for microcontrollers to talk to each other. However, it has some key differences from its I2C counterpart. The most notable difference right off the bat is that, while you can use multiples masters and slaves with I2C, SPI allows a single master device with a maximum of four slave devices.

SPI is typically much faster than I2C due to the simple protocol and, while data/clock lines are shared between devices, each device requires a unique address wire. SPI is commonly found in places where speed is important such as with SD cards and display modules, or when information updates and changes quickly, like with temperature sensors.

6. Interfacing arduino and Blynk

IoT based Temperature and Humidity Monitoring using BLYNK Application

The ESP8266 Integrates 802.11b/g/n HT40 a Wi-Fi transceiver, so it cannot only connect with a Wi-Fi network and interact with the Internet. It can also set up a network of its own, allowing other devices to connect directly to it. There's an on-board voltage regulator that ensures the cleanest possible power to the NodeMCU itself, as well as a push-button reset and a USB connection for easy interface with your computer.
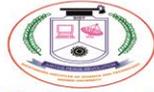
DHT11 is a low-cost digital sensor for sensing temperature and humidity. This sensor can easily interfaced with any microcontroller such as Arduino, Raspberry Pi, etc… to measure humidity and temperature instantaneously. DHT11 humidity and temperature sensor are available as a sensor and as a module. The difference between this sensor and module is the pull-up resistor and a power-on LED. DHT11 is a relative humidity sensor.

The working of the DHT sensor is pretty simple. DHT11 sensor consists of a capacitive humidity sensing element and a thermistor for sensing temperature. The humidity sensing capacitor has two electrodes with a moisture-holding substrate as a dielectric between them. Change in the capacitance value occurs with the change in humidity levels. The IC measure, process this changed resistance values and change them into digital form.

For measuring temperature this sensor uses a Negative Temperature coefficient thermistor, which causes a decrease in its resistance value with an increase in temperature. To get a larger resistance value even for the smallest change in temperature, this sensor is usually made up of semiconductor ceramics or polymers.
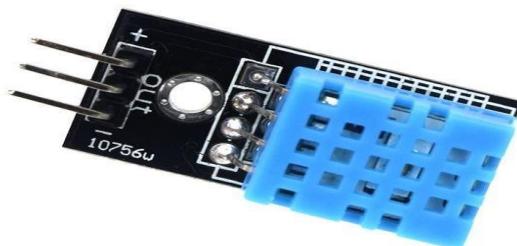
Fig. 8 DHT11 Sensor

The temperature range of DHT11 is from 0 to 50 degrees Celsius with a 2-degree accuracy. The humidity range of this sensor is from 20 to 80% with 5% accuracy. The sampling rate of this

sensor is 1Hz .i.e. it gives one reading for every second. DHT11 is small in size with an operating voltage from 3 to 5 volts. The maximum current used while measuring is 2.5mA.

Blynk was designed for the Internet of Things. It can control hardware remotely, it can display sensor data, it can store data, visualize it, and do many other cool things. Every time you press a Button in the Blynk app, the message travels to the Blynk Cloud, where it magically finds its way to your hardware. It works the same in the opposite direction and everything happens in a blynk of an eye.

There are three major components in the platform:

- **Blynk App** - allows to you create amazing interfaces for your projects using various widgets we provide.
- **Blynk Server** - responsible for all the communications between the smartphone and hardware. You can use our Blynk Cloud or run your private Blynk server locally. It's open-source, could easily handle thousands of devices and can even be launched on a Raspberry Pi.
- **Blynk Libraries** - for all the popular hardware platforms - enable communication with the server and process all the incoming and outcoming commands.

For installation of the blynk app, you need to follow these steps Go to play store app store and type blynk, you will find the green color icon for the blynk app, then install it on your device.
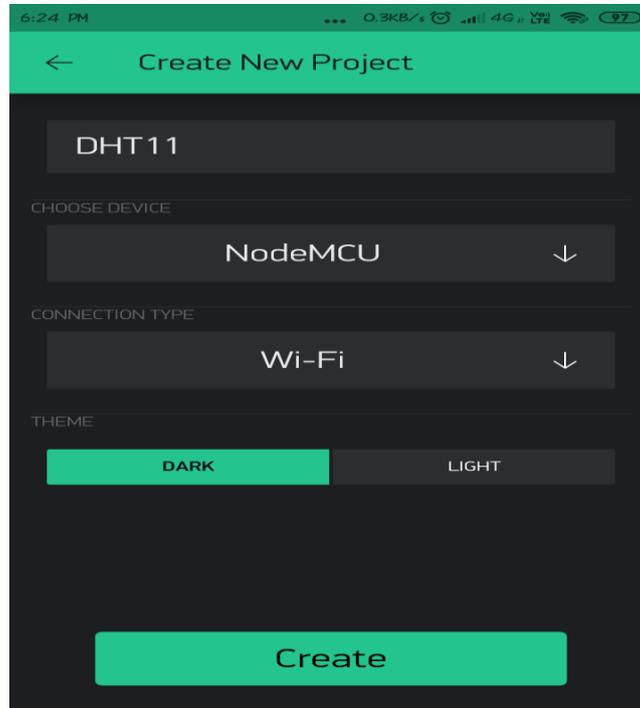
Fig. 9 Blink App

Auth token will be generated this auth token you will get in the settings option of this project and on your **Gmail Account.**

- Once your project is created, you have to insert different types of widget into it, For example I will be adding two buttons from widget box shown below.

- After selecting two Gauge click on the **Settings** Gauge and fill the details like i did in following image.
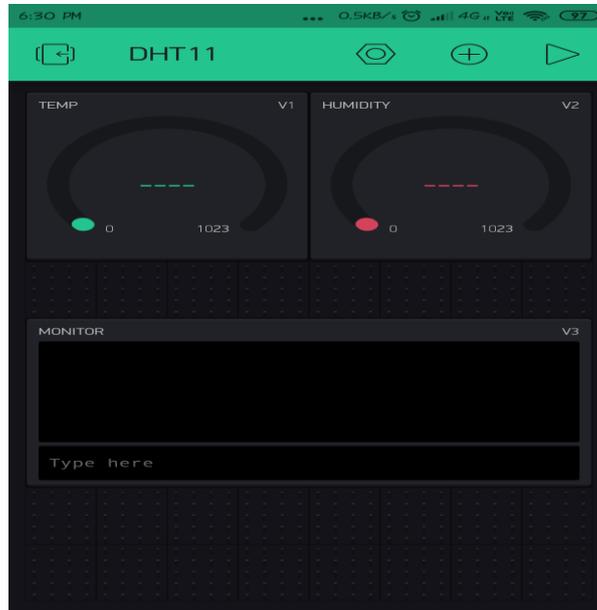
Fig. 10 Dash Board

Program

#define BLYNK_PRINT Serial   // Comment this out to disable prints and save space

#include <SPI.h>

#include <ESP8266WiFi.h>

#include <BlynkSimpleEsp8266.h>

#include <SimpleTimer.h>

#include <DHT.h>

char auth[] = "TB6KiXEXl----------qFo3Bp"; //you can add yout auth token

char ssid[] = "sneha123";   // your Wifi name

char pass[] = "asdfghjkl";   // your wifi password

```
#define DHTPIN D1         // Digital pin D1

float  moisture;

#define DHTTYPE DHT11     // DHT 11

int temp, Humid;

DHT dht(DHTPIN, DHTTYPE);

SimpleTimer timer;

WidgetTerminal terminal(V1);

BLYNK_WRITE(V1)

{

  terminal.write(param.getBuffer(), param.getLength());

  terminal.println();

 // Ensure everything is sent

 terminal.flush();

}

void sendSensor()

{

 float h = dht.readHumidity();

 float t = dht.readTemperature(); // or dht.readTemperature(true) for Fahrenheit

 if (isnan(h) || isnan(t)) {

  Serial.println("Failed to read from DHT sensor!");

  return;

 }
```

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

```
  // You can send any value at any time.

  // Please don't send more that 10 values per second.

  Blynk.virtualWrite(V5, h);  //V5 is for Humidity

  Blynk.virtualWrite(V6, t);  //V6 is for Temperature

}

void setup()

{

  Serial.begin(9600); // See the connection status in Serial Monitor

  Blynk.begin(auth, ssid, pass);

  dht.begin();

  timer.setInterval(1000L, sendSensor);

  terminal.flush();

}

void loop()

{

  temp = dht.readTemperature(); // or dht.readTemperature(true) for Fahrenheit

  Humid = dht.readHumidity();

  Serial.print("temp: ");

  Serial.print(temp);

  Serial.print(" c");

  terminal.print("temp: ");

  terminal.print(temp);
```

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

```
terminal.print(" c");

Serial.print("   Humidity: ");

Serial.print(Humid);

Serial.println(" %");

terminal.print("  Humidity: ");

terminal.print(Humid);

terminal.println(" %");

delay(300);

  terminal.flush();

Blynk.run(); // Initiates Blynk

timer.run(); // Initiates SimpleTimer

}
```

7. Servo meter with Arduino

Servo motors use feedback to determine the position of the shaft, you can control that position very precisely. As a result, servo motors are used to control the position of objects, rotate objects, move legs, arms or hands of robots, move sensors etc. with high precision. Servo motors are small in size, and because they have built-in circuitry to control their movement, they can be connected directly to an Arduino.

Most servo motors have the following three connections:

Black/Brown ground wire.

Red power wire (around 5V).

Yellow or White PWM wire.

In this experiment, we will connect the power and ground pins directly to the Arduino 5V and GND pins. The PWM input will be connected to one of the Arduino's digital output pins.

Experiment 1

Hardware Required

1 x TowerPro SG90 servo motor

1 x Arduino Mega2560

3 x jumper wires

Wiring Diagram

The best thing about a servo motor is that it can be connected directly to an Arduino. Connect to the motor to the Arduino as shown in the table below:

Servo red wire – 5V pin Arduino

Servo brown wire – Ground pin Arduino

Servo yellow wire – PWM(9) pin Arduino
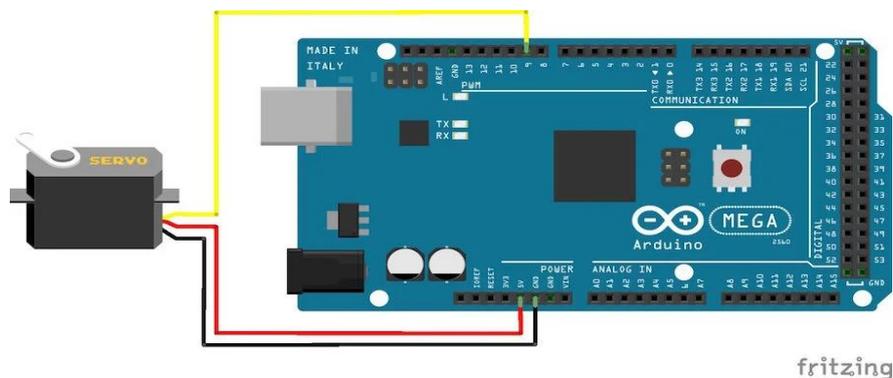
Fig. 11 Arduino with Servo motor
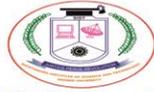
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

When the program starts running, the servo motor will rotate slowly from 0 degrees to 180 degrees, one degree at a time. When the motor has rotated 180 degrees, it will begin to rotate in the other direction until it returns to the home position.

```
#include            //Servo library
Servo servo_test;               //initialize a servo object for the connected servo
int angle = 0;
 void setup()
{
  servo_test.attach(9);          // attach the signal pin of servo to pin9 of arduino
}
  void loop()
{
  for(angle = 0; angle < 180; angle += 1)       // command to move from 0 degrees to 180 degrees
  {
   servo_test.write(angle);               //command to rotate the servo to the specified angle
   delay(15);
  }

  delay(1000);

  for(angle = 180; angle>=1; angle-=5)     // command to move from 180 degrees to 0 degrees
  {
   servo_test.write(angle);            //command to rotate the servo to the specified angle
   delay(5);
  }
   delay(1000);
}
```

School of Computing

Department of Computer Science and Engineering

**UNIT – V ARDUINO PROGRAMMING FOR
IoT BOARDS - SCSA1407**

Unit 5 Programming ESP 8266 Module

ESP8266 WiFi Serial Module: Overview, Setting Up the Hardware, Interfacing with Arduino, Creating an IoT Temperature and Humidity Sensor System, Overview of DHT-22 Sensor, Interfacing the Hardware: Arduino, ESP8266 WiFi Module, and DHT-22 Sensor, Checking Your Data via ThingSpeak, Connecting Your Arduino Set-up to Blynk via WiFi

1. ESP8266 WiFi Serial Module: Overview, Setting Up the Hardware

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

However, as a chip, the ESP8266 is also hard to access and use. You must solder wires, with the appropriate analog voltage, to its pins for the simplest tasks such as powering it on or sending a keystroke to the "computer" on the chip. You also have to program it in low-level machine instructions that can be interpreted by the chip hardware. This level of integration is not a problem using the ESP8266 as an embedded controller chip in mass-produced electronics. It is a huge burden for hobbyists, hackers, or students who want to experiment with it in their own IoT projects.

But, what about Arduino? The Arduino project created an open-source hardware design and software SDK for their versatile IoT controller. Similar to NodeMCU, the Arduino hardware is a microcontroller board with a USB connector, LED lights, and standard data pins. It also defines standard interfaces to interact with sensors or other boards. But unlike NodeMCU, the Arduino board can have different types of CPU chips (typically an ARM or Intel x86 chip) with memory chips, and a variety of programming environments. There is an Arduino reference design for the ESP8266 chip as well. However, the flexibility of Arduino also means significant variations across different vendors. For example, most Arduino boards do not have WiFi capabilities, and some even have a serial data port instead of a USB port.
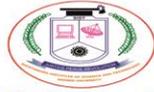
![Sathyabama Institute of Science and Technology logo]

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

1.1 NodeMCU Specifications

The NodeMCU is available in various package styles. Common to all the designs is the base ESP8266 core. Designs based on the architecture have maintained the standard 30-pin layout. Some designs use the more common narrow (0.9″) footprint, while others use a wide (1.1″) footprint – an important consideration to be aware of.

The most common models of the NodeMCU are the Amica (based on the standard narrow pin-spacing) and the LoLin which has the wider pin spacing and larger board. The open-source design of the base ESP8266 enables the market to design new variants of the NodeMCU continually.
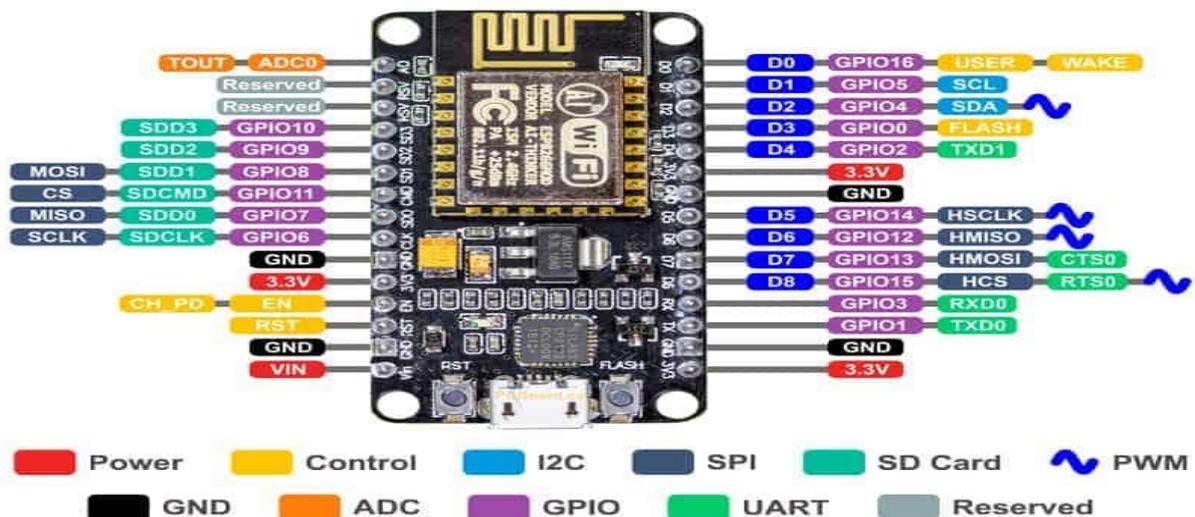
Fig. 1 ESP 8266 Components

- Power Pins There are four power pins. VIN pin and three 3.3V pins.
- VIN can be used to directly supply the NodeMCU/ESP8266 and its peripherals. Power delivered on VIN is regulated through the onboard regulator on the NodeMCU module – you can also supply 5V regulated to the VIN pin
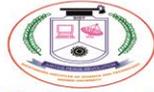
SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

- 3.3V pins are the output of the onboard voltage regulator and can be used to supply power to external components.

- GND are the ground pins of NodeMCU/ESP8266

- I2C Pins are used to connect I2C sensors and peripherals. Both I2C Master and I2C Slave are supported. I2C interface functionality can be realized programmatically, and the clock frequency is 100 kHz at a maximum. It should be noted that I2C clock frequency should be higher than the slowest clock frequency of the slave device.

- GPIO Pins NodeMCU/ESP8266 has 17 GPIO pins which can be assigned to functions such as I2C, I2S, UART, PWM, IR Remote Control, LED Light and Button programmatically. Each digital enabled GPIO can be configured to internal pull-up or pull-down, or set to high impedance. When configured as an input, it can also be set to edge-trigger or level-trigger to generate CPU interrupts.

- ADC Channel The NodeMCU is embedded with a 10-bit precision SAR ADC. The two functions can be implemented using ADC. Testing power supply voltage of VDD3P3 pin and testing input voltage of TOUT pin. However, they cannot be implemented at the same time.

- UART Pins NodeMCU/ESP8266 has 2 UART interfaces (UART0 and UART1) which provide asynchronous communication (RS232 and RS485), and can communicate at up to 4.5 Mbps. UART0 (TXD0, RXD0, RST0 & CTS0 pins) can be used for communication. However, UART1 (TXD1 pin) features only data transmit signal so, it is usually used for printing log.

- SPI Pins NodeMCU/ESP8266 features two SPIs (SPI and HSPI) in slave and master modes. These SPIs also support the following general-purpose SPI features:

- 4 timing modes of the SPI format transfer

- Up to 80 MHz and the divided clocks of 80 MHz

- Up to 64-Byte FIFO

- SDIO Pins NodeMCU/ESP8266 features Secure Digital Input/Output Interface (SDIO) which is used to directly interface SD cards. 4-bit 25 MHz SDIO v1.1 and 4-bit 50 MHz SDIO v2.0 are supported.
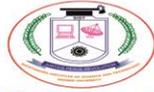
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

- PWM Pins The board has 4 channels of Pulse Width Modulation (PWM). The PWM output can be implemented programmatically and used for driving digital motors and LEDs. PWM frequency range is adjustable from 1000 μs to 10000 μs (100 Hz and 1 kHz).
- Control Pins are used to control the NodeMCU/ESP8266. These pins include Chip Enable pin (EN), Reset pin (RST) and WAKE pin.
- EN: The ESP8266 chip is enabled when EN pin is pulled HIGH. When pulled LOW the chip works at minimum power.
- RST: RST pin is used to reset the ESP8266 chip.
- WAKE: Wake pin is used to wake the chip from deep-sleep.

Control Pins are used to control the NodeMCU/ESP8266. These pins include Chip Enable pin (EN), Reset pin (RST) and WAKE pin.

- **EN:** The ESP8266 chip is enabled when EN pin is pulled HIGH. When pulled LOW the chip works at minimum power.
- **RST:** RST pin is used to reset the ESP8266 chip.
- **WAKE:** Wake pin is used to wake the chip from deep-sleep.

1.2 Install ESP8266 Add-on in Arduino IDE

To install the ESP8266 board in your Arduino IDE, follow these next instructions:

1. In your Arduino IDE, go to **File**> **Preferences**
2. Enter **http://arduino.esp8266.com/stable/package_esp8266com_index.json** into the "Additional Boards Manager URLs" field as shown in the figure below. Then, click the "OK" button:
   Note: if you already have the ESP32 boards URL, you can separate the URLs with a comma as follows:
   https://dl.espressif.com/dl/package_esp32_index.json,
   http://arduino.esp8266.com/stable/package_esp8266com_index.json
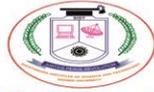
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

3. Open the Boards Manager. Go to **Tools** > **Board** > **Boards Manager…**

4. Search for ESP8266 and press install button for the "ESP8266 by ESP8266 Community":

5. That's it. It should be installed after a few seconds.

2. ESP8266 WITH ARDUINO

The ESP8266 board contain the microcontroller ESP8266EX (32-bit microcontroller) from Espressif Systems, this low cost Wi-Fi module is a very good choice for hobbyists to build IoT projects. IoT: Internet of Things. The ESP8266 module comes with AT firmware which allows us to control it with AT commands through serial interface (RX and TX pins). ESP8266 Programming with Arduino IDE: It's easy to start ESP8266 programming, all what we've to do is adding it to the Arduino IDE software. First, open Arduino IDE and go to File —> Preferences Add the link below to Additional Boards Manager URLs and click on OK: http://arduino.esp8266.com/stable/package_esp8266com_index.json

Fig. 2 Preferences

Now go to Tools —> Board —> Boards Manager …

In the search box write esp8266 and click on Install and the installation of the board should start (the installation may take some time depending on the connection speed):
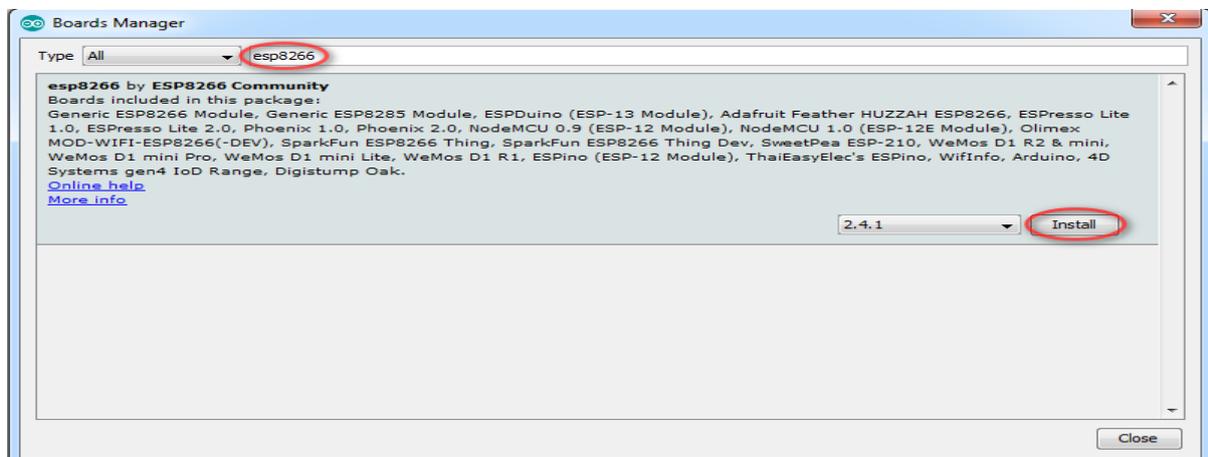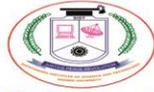
Fig. 3 Installation of Libraries

After the installation select the ESP-01 board by going to: Tools —> Board: —> Generic ESP8266 Module

As known the Arduino UNO board contains Microchip ATmega16U2 microcontroller which is used as USB-to-serial converter. This chip (ATmega16U2) can be used to program (flash) the ESP-01 Wi-Fi module, circuit connections are shown below:
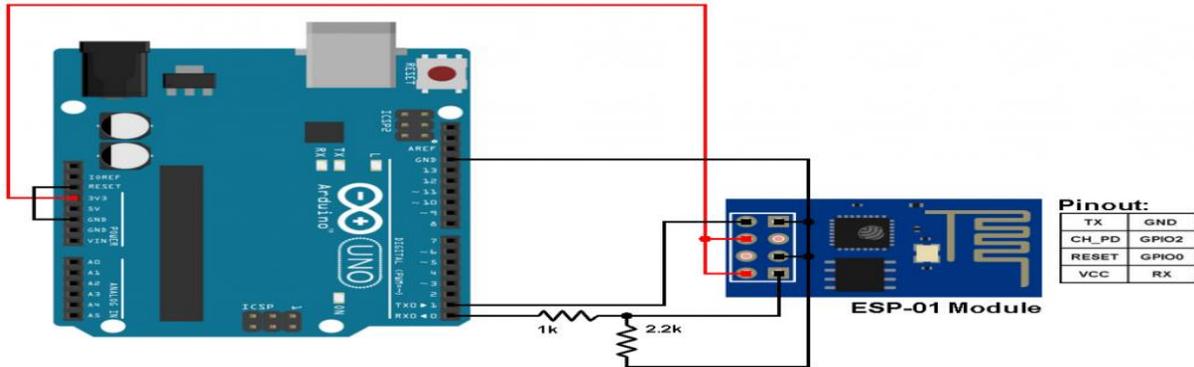
Fig. 4 ESP8266 (ESP-01) Module with Arduino UNO board

In the circuit there are 2 resistor one of 1k ohm and the other one of 2.2k ohm. The two resistors are used to step down the 5V which comes to arduino into about 3.43V which goes to the ESP-01 board (connected to RX pin of the ESP-01) because the ESP8266EX chip works with 3.3V only and applying a 5V directly may damage it.

On the other hand, the TX pin of the ESP-01 is connected directly to the Arduino board without any voltage level converter because here the ESP-01 sends data (at 3.3V) to the Arduino board using this pin. This is a simple example which we should start with, it's the LED blinking example. In this example I'm going to connect one LED to ESP-01 board GPIO2 pin, and make this LED blinks. Circuit diagram is shown below:
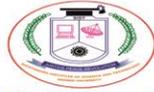
The LED is connected to pin GPIO2 of the ESP-01 module through a 330 ohm resistor. The ESP-01 module needs a 3.3V supply. We can get the 3.3V for example from Arduino UNO board, or using AMS1117 3V3 voltage regulator which steps down 5V into 3.3V, or directly from 3.3V source.

Arduino code for ESP8266 module:

```
#define LED        2              // LED is connected to GPIO2

void setup() {

  pinMode(LED, OUTPUT);          // Configure LED pin as output

}

void loop() {

  digitalWrite(LED, HIGH);       // Turn the LED on
  delay(500);                    // wait 1/2 second
  digitalWrite(LED, LOW);        // turn the LED off
  delay(500);                    // wait 1/2 second
}
```

3. IoT BASED HUMIDITY AND TEMPERATURE MONITORING USING ARDUINO UNO

we can control any electronic equipment in homes and industries. Moreover, you can read a data from any sensor and analyze it graphically from anywhere in the world. Here, we can read temperature and humidity data from DHT11 sensor and upload to a ThingSpeak cloud using Arduino Uno and ESP8266-01 module. Arduino Uno is MCU, it fetch a data of humidity and

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in
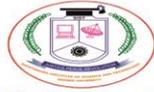
temperature from DHT11 sensor and Process it and give it to a ESP8266 Module.ESP8266 is a WiFi module, it is one of the leading platform for Internet of Things. It can transfer a data to IOT cloud.

Hardware Requirements

- Arduino Uno
- ESP8266-01
- DHT11
- AMS1117-3.3V
- 9V battery

Software Requirements

- Arduino IDE
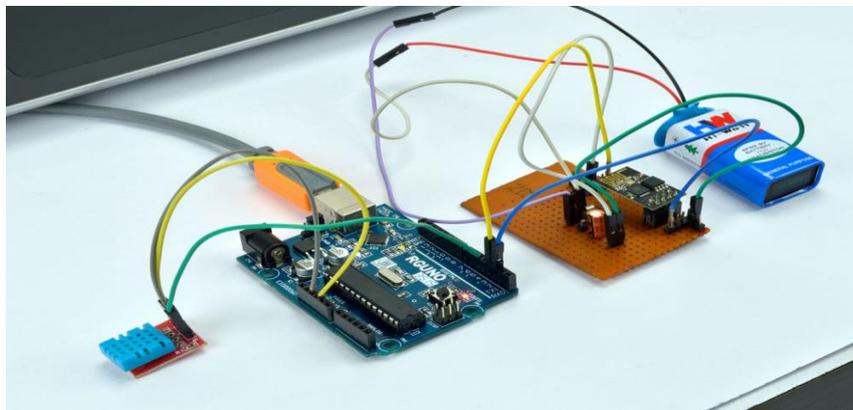
Fig. 5 Arduino with DHT sensor

The 2nd pin is of DHT11 is a data pin, it can send a temperature and humidity value to the 5th pin of Arduino Uno.1st and 4th pin of DHT11 is a Vcc and Gnd and 3rd pin is no connection. The Arduino Uno process a temperature and humidity value and send it to a ESP8266 WiFi module. The Tx and Rx pin of ESP8266 is connected to the 2nd (Rx) and 3rd (Tx) of Arduino
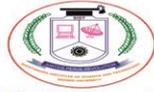
Uno. Make sure that input voltage of ESP8266 must be 3.3V, not a 5V (otherwise it would damage a device).For that, we are using AMS1117 Voltage regulator circuit. It can regulate a voltage from 9V to 3.3V and will give it to Vcc pin of ESP8266.The Ch_Pd is a chip enable pin of ESP8266 and should be pullup to 3.3V through 3.3KΩ resistor. For reset the module pull down the RST pin of ESP8266 to Gnd.ESP8266 have 2 GPIO pins GPIO 0 and GPIO 2.
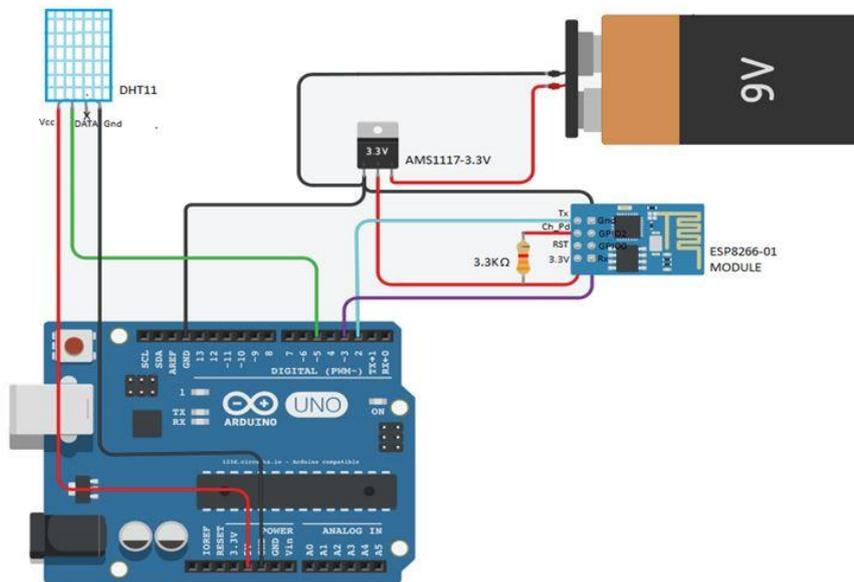
Fig. 6 Circuit diagram for monitoring Humidity and Temperature in IOT cloud

ThingSpeak is an open source platform to store and retrieve a data for Internet of Things application. To use this, you need to register in ThingSpeak cloud and then login to your account. After create a new channel with temperature in one field and humidity in another field as shown in Fig: 1.2. Once you created a new channel, it will generate a two API keys, they are READ API keys and WRITE API keys. First, copy the WRITE API keys from ThingsSpeak and paste it into the line (String apiKey = "OX9T8Y9OL9HD0UBP";) of the program. Next, replace the Host_Name and Password with your WiFi name and WiFi password in the two lines given below in the program. (String Host_Name = "Pantech" and String Password = "pantech123"). The Arduino program Uses DHT library, if it is not presented in your arduino IDE, select Sketch à Include library à Manage libraries à Install DHT Sensor library. Then

compile the program and upload to a Arduino Uno through Arduino IDE. Ensure that WiFi modem and internet connection in your Smartphone or PC are working properly. After uploaded a program, the Temperature and Humidity data is uploaded on ThingSpeak platform.

**Fig.7 Creating new channel on ThingSpeak cloud**

Fig. 8 Graphical representation of Humidity and Temperature data

#include <ESP8266WiFi.h>

#include "DHT.h"

String apiKey = "";

const char *ssid = "";

const char *pass = "";

const char* server = "api.thingspeak.com";

```
DHT dht(D2, DHT11);

WiFiClient client;

void setup() {

  Serial.begin(115200);

  delay(10);

  dht.begin();

  WiFi.begin(ssid, pass);

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

    delay(500);

    Serial.print(".");

  }

  Serial.println("");

  Serial.println("WiFi connected");

}

void loop() {

  float h = dht.readHumidity();

  float t = dht.readTemperature();

  if (isnan(h) || isnan(t)) {

    Serial.println("Failed to read from DHT sensor!");

    return;

  }

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

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

```
String postStr = apiKey;

postStr += "&field1=";

postStr += String(t);

postStr += "&field2=";

postStr += String(h);

postStr += "\r\n\r\n";

client.print("POST /update HTTP/1.1\n");

client.print("Host: api.thingspeak.com\n");

client.print("Connection: close\n");

client.print("X-THINGSPEAKAPIKEY: " + apiKey + "\n");

client.print("Content-Type: application/x-www-form-urlencoded\n");

client.print("Content-Length: ");

client.print(postStr.length());

client.print("\n\n");

client.print(postStr);

Serial.print("Temperature: ");

Serial.print(t);

Serial.print("\t");

Serial.print("Humidity: ");

Serial.println(h);

}

client.stop();
```
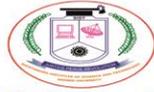
```
  delay(1000);

}
```

4. OVERVIEW OF DHT22 SENSOR

The DHT22 is a very low-cost sensor. It's made up of two components: a capacitive humidity sensor and a thermistor, which measures temperature. Because it's a digital sensor, you can read the sensor data over a GPIO pin.

- Three jump wires

- DHT22

- NodeMCU

- Micro-USB

The left-most pin of the DHT22 is the positive pin. You should connect it to 3V or Vin on the MCU. The second pin of the DHT22 (from the left) is the data pin. You should connect it to D2 on the MCU. The third pin of the DHT22 (from the left) does nothing. The last pin of the DHT22 is the Ground pin. It should be connected GND. In addition to this, you'll have to install two more libraries. To read the sensor, we are going to use Adafruit's DHT22 library. It can be installed using Arduino's library manager. It comes in two components. First, you'll want to download the Adafruit Unified Sensor library and install the DHT sensor library:

Program:

```
#include "DHT.h"

#define DHTPIN 4     // what digital pin the DHT22 is conected to

#define DHTTYPE DHT22   // there are multiple kinds of DHT sensors

DHT dht(DHTPIN, DHTTYPE);

void setup() {
```
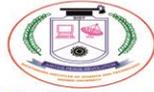
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

```
  Serial.begin(9600);

  Serial.setTimeout(2000);

  // Wait for serial to initialize.

  while(!Serial) { }

  dht.begin();

  Serial.println("Device Started");

  Serial.println("-----------------------------------");

  Serial.println("Running DHT!");

  Serial.println("-----------------------------------");

}

int timeSinceLastRead = 0;

void loop() {

  // Report every 2 seconds.

  if(timeSinceLastRead > 2000) {

    // Reading temperature or humidity takes about 250 milliseconds!

    // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)

    float h = dht.readHumidity();

    // Read temperature as Celsius (the default)

    float t = dht.readTemperature();

    // Read temperature as Fahrenheit (isFahrenheit = true)

    float f = dht.readTemperature(true);

    // Check if any reads failed and exit early (to try again).
```
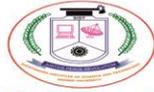
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

```
if (isnan(h) || isnan(t) || isnan(f)) {

  Serial.println("Failed to read from DHT sensor!");

  timeSinceLastRead = 0;

  return;

}

// Compute heat index in Fahrenheit (the default)

float hif = dht.computeHeatIndex(f, h);

// Compute heat index in Celsius (isFahreheit = false)

float hic = dht.computeHeatIndex(t, h, false);

Serial.print("Humidity: ");

Serial.print(h);

Serial.print(" %\t");

Serial.print("Temperature: ");

Serial.print(t);

Serial.print(" *C ");

Serial.print(f);

Serial.print(" *F\t");

Serial.print("Heat index: ");

Serial.print(hic);

Serial.print(" *C ");

Serial.print(hif);

Serial.println(" *F");
```
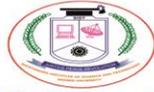
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

```
  timeSinceLastRead = 0;

 }

 delay(100);

 timeSinceLastRead += 100;

}
```

5. CONNECT AND SEND DATA TO THINGSPEAK

ThingSpeak is IoT Cloud platform where you can send sensor data to the cloud. You can also analyze and visualize your data with MATLAB or other software, including making your own applications. The ThingSpeak service is operated by MathWorks. In order to sign up for ThingSpeak, you must create a new MathWorks Account or log in to your existing MathWorks Account. ThingSpeak is free for small non-commercial projects. ThingSpeak includes a Web Service (REST API) that lets you collect and store sensor data in the cloud and develop Internet of Things applications. It works with Arduino, Raspberry Pi and MATLAB (premade libraries and APIs exists) But it should work with all kind of Programming Languages, since it uses a REST API and HTTP.

. 5.1 Creating a channel

To get started you will need to sign up for a free Thingspeak account. Thingspeak is organized in a simple way: you can create channels that contains datafields. A simple example: if you're building a temperature sensor, you probably want to create 1 channel for your device with two fields: a temperature field and a humidity field. To create a channel, go to Channels, My Channels and click on "New Channel".

Fig. 9 Configuring fields in a ThingSpeak channel

5.2 Installing ThingSpeak library

Now we're ready to push data to Thingspeak and you can do this in two ways: by making HTTP calls or by using their MQTT broker. The easiest one is by using HTTP calls, but, you don't need to make these yourself. We can use the Thingspeak Arduino library. I'll install it by adding it to my platformio.ini file, under the section lib_deps. As mentioned before, PlatformIO will take care of downloading and installing this library.

```
#include "ThingSpeak.h"
#define CHANNEL_ID 99999999
#define CHANNEL_API_KEY "XXXXXXXXXXXXX"
WiFiClient client;
int counter = 0;
void setup() {
  Serial.begin(9600);
  connectToWiFi(); // this function comes from a previous video
  ThingSpeak.begin(client);
```

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in
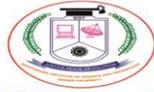
```
}
void loop() {
 counter++;
 ThingSpeak.writeField(CHANNEL_ID, 1, counter, CHANNEL_API_KEY);
 delay(15000); // 15 seconds
}
```

Writing multiple fields

To update multiple fields in 1 go, we have to adapt the code a bit. The example above will make 1 request per field, which isn't efficient if you have multiple fields.

```
void loop()
 {
 counter++;

 ThingSpeak.setField(1, counter);
 ThingSpeak.setField(2, WiFi.RSSI());
 ThingSpeak.writeFields(CHANNEL_ID, CHANNEL_API_KEY);

 delay(15000); // 15 seconds
}
```

6. CONTROL ARDUINO REMOTELY OVER THE INTERNET USING BLYNK APP

Blynk is an IoT platform that allows us to quickly build projects for controlling and monitoring the data using Android and iOS devices. We can create a project dashboard and add widgets like buttons, displays, sliders, etc. for controlling microcontrollers and other peripherals. Using these widgets, we can control the devices and can monitor the sensor data on the phone screen.
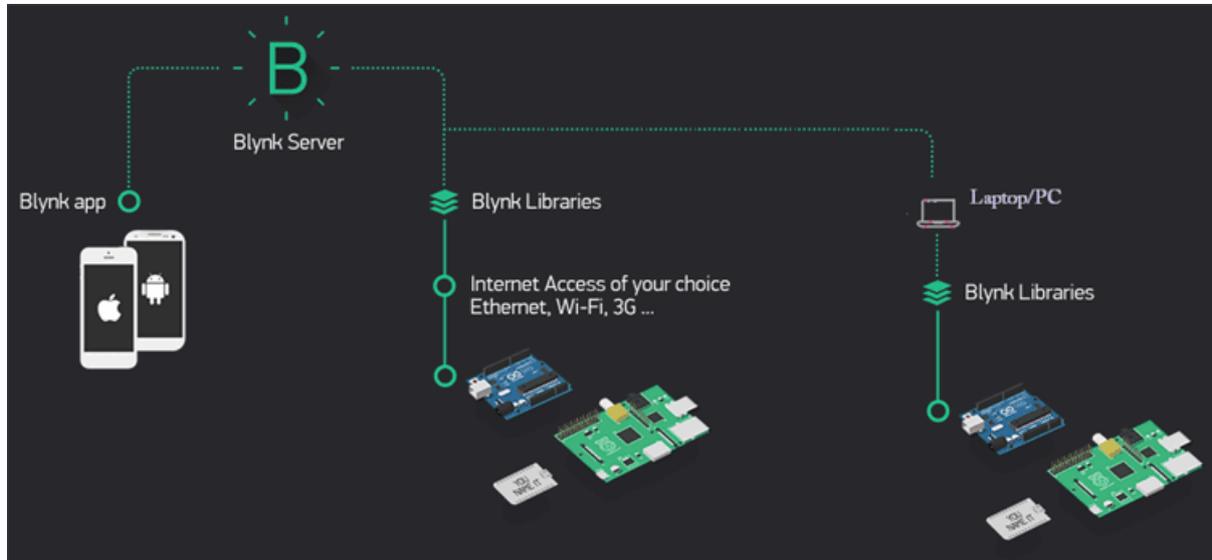
Fig. 10 Blynk Application

6.1 Features of Blynk

1. You can add a notifications service using the Blynk app without using any third-party platform like IFTTT. For example, you can post the data on Twitter and get the e-mail when something reaches its threshold. This can be possible just by configuring the Blynk app.

2. In IoT projects, the hardware part is easy as compare to the software part. But using Blynk, the software part also becomes easier than the hardware. There is very less coding required and all the code is included in its library. Blynk is perfect for building simple projects.

3. Most of the microcontroller available in the market is supported by Blynk and these microcontrollers can be controlled using Blynk app via Wi-fi, BLE, USB, GSM, and Ethernet.

4. You can create your own local **Blynk server** to control the appliances locally just by using few steps and can control easily using the Blynk app.

5. One of the most interesting features of Blynk is the use of virtual pins. Virtual Pin is a concept invented by Blynk to provide the exchange of any data between hardware and the Blynk mobile app. These pins are different from Digital and Analog pins, they don't have any physical properties.

So, if you want any data from the virtual pin, the Blynk app will send the data to a defined virtual pin and then this data can be accessed on MCU pins. Also, the data can be sent from the Blynk app to any virtual pin, and then the data can be easily accessed on the app.

6.2 Circuit Diagram

The circuit diagram is very simple, just connect one LED to PWM pin (5) and the other LED to digital pin 4 of Arduino Uno.
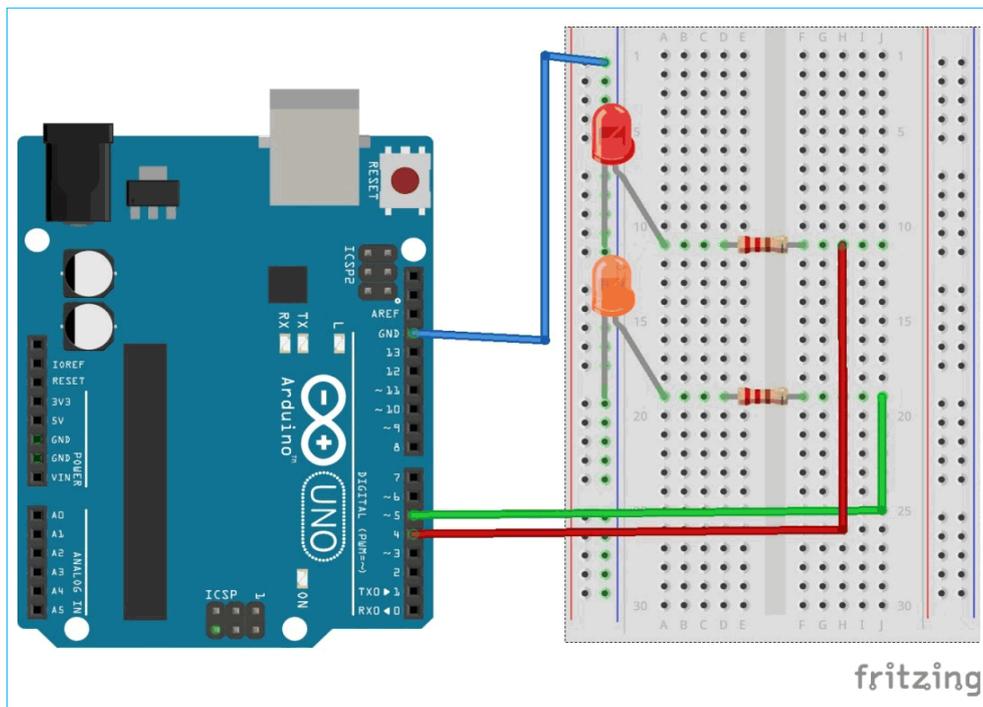

Fig. 11 Arduino with LED

6.3 Installing and Configuring Blynk App to control LED

1. Download the Blynk app from the play store. It is available for both Android and iOS users. Open the app and create an account by entering your e-mail ID and password.

2. Now, we will create a New Project. So, tap on New Project.
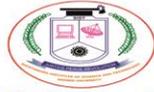
3. Give a **Project name and Choose the Device as Arduino UNO and Connection Type as USB** because we are using serial communication to talk with Arduino and Blynk Server. Now, click on *Create* as shown below.
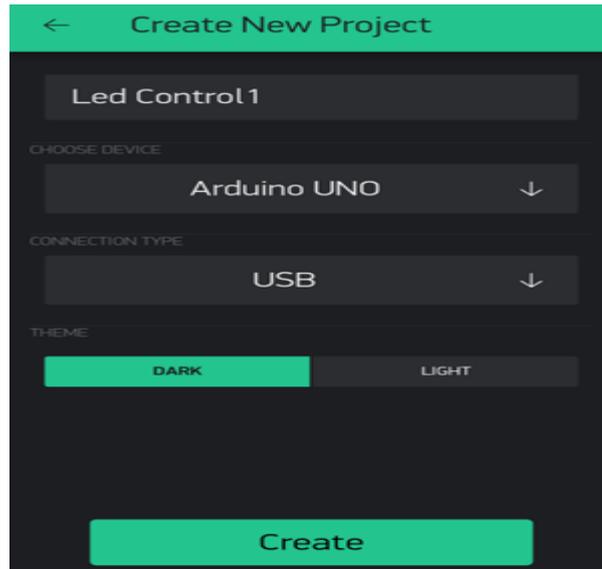

Fig. 12 Blynk App Display

4. After Creating the Project, you will receive an Auth Token on registered mail id. This token will be used in the Code.

5. Now an empty dashboard will be shown where we will place all the required widgets i.e. buttons, displays, sliders, etc. Tap on + sign. All the available widgets are shown here. You can explore all the widgets and can use them according to your requirements.

6. Now, set the properties of both widgets. Tap on the button on the dashboard. Choose Output on D4 and mode as *Switch* then go back to dashboard and tap on the slider. Choose Output pin on Virtual pin V1 and all properties remain the same.

Now we are ready with the Blynk app. Let's Start programming Arduino board for working with the Blynk app, for this there is a library available for Arduino.

![Sathyabama Institute of Science and Technology logo]

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in
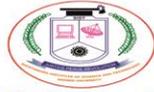
6.4 Installing Blynk Library in Arduino

Before start writing the code for Arduino Uno, we will first install the Blynk Library in Arduino IDE:

To install the library, **Go to** *Sketch -> Include Libraries -> Manage Libraries*. Then search for Blynk and install the latest version as shown below.
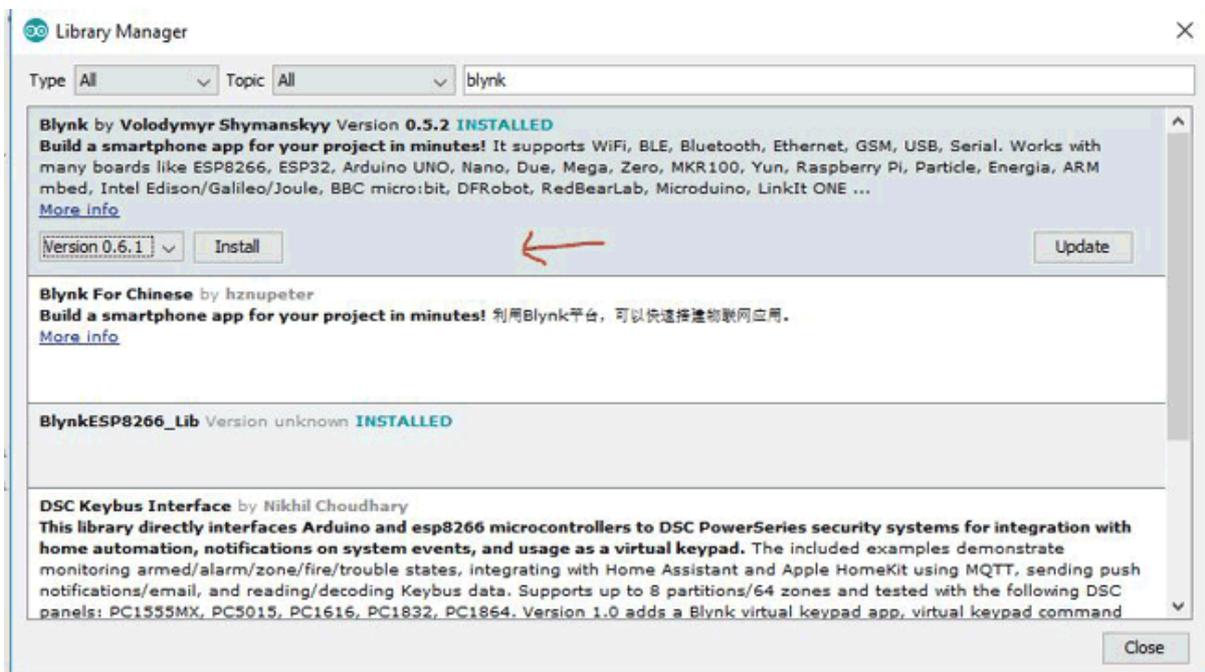
Fig. 13 Installation of Blynk Library

6.5 Programming Arduino for Blynk App

1. First, declare macros using #define as required in the code. Here macro **BLYNK_PRINT is defined as DebugSerial** for printing purpose.

   #define BLYNK_PRINT DebugSerial

2. Now, include header files for software serial and Blynk functions and make a instance for Software serial as *DebugSerial*.

   #include <SoftwareSerial.h>
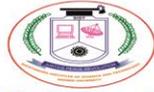
SoftwareSerial DebugSerial(2, 3);   // RX, TX

#include <BlynkSimpleStream.h>

3. Store that auth token in the char array.

 char auth[] = "YourAuthToken";

4. In *setup()* function, initialize software serial, inbuilt serial and Blynk with baud rate 9600. Function *blynk.begin()* takes two arguments namely Serial and auth token.

void setup()
{
DebugSerial.begin(9600);
Serial.begin(9600);
Blynk.begin(Serial, auth);
}

5. In *void loop()* function, there should be very minimum code so that Blynk can work without any interrupt or loss of data. This is because when you put something in void loop function like getting sensor reading from MCU or from the smartphone, it executes million times and this data uploads on the Blynk server which means the Blynk cloud will flooded with tons of messages and server will consider this as Spams so the Blynk cloud will automatically terminate the connection. Also, avoid using the *delay* function in the loop because it completely stops the functioning of MCU and the connection will close in this situation also.

The best choice for getting the sensor data from Arduino is by using the timers. Initialize the timers in the *setup* function and define a function to perform the task.

There will be a bare minimum function required - Blynk.run() and timer function can handle all the tasks of getting the data and sending it to the server. But in this tutorial, we are not sending any data so timers are not required.

void loop()

```
{
  Blynk.run();
}
```

6. The **code for toggling the LED is inbuilt in the** *Blynk.run()* function but we have to make a function for getting the slider value from a smartphone. There are two functions for sending and receiving the data which are *BLYNK_READ()* and *BLYNK_WRITE()*. These functions take virtual pins as an input argument to read and write the data. Therefore we have to use the BLYNK_WRITE function to write the data on virtual pin V1 from the Blynk app.

Now, assign the incoming value from pin V1 to a variable, param.asInt() function returns the received value as integer. If the received value is not integer you can use float, double or string.

param.asFloat(); // get value as a Float

param.asDouble(); // get value as a Double

param.asStr();   // get value as a String

Then put this value in the PWM pin of Arduino using the **analogWrite()** function.

BLYNK_WRITE(V1)

{

int pinValue = param.asInt();

analogWrite(5,pinValue);

}
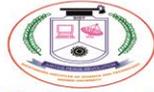
6.6 Script for Internet Connection:

Here we are not using any module with Arduino board but **a working internet connection is needed** to send and receive data over the cloud so there is a script included in the Blynk library that can access our laptop/PC internet connection. Therefore, this script takes the data from the Arduino board through serial communication and uploads the data on Blynk cloud using the laptop internet connection. We have to run this script to start the operation. This script can be found in the Arduino directory which is in the Documents folder, Go to *Libraries -> Blynk -> Scripts*. There is a file named blynk-ser.bat which is the required script. Edit this script with the COM port of the Arduino board and Blynk cloud port number. Open the script using notepad and replace the following things. You have to replace only with your COM port and save the file, all other things remain the same.

set COMM_PORT= COM(port_no.)   //e.g. COM25

set COMM_BAUD=9600

set SERV_ADDR=blynk-cloud.com

set SERV_PORT=8442

6.7 Testing - Controlling the Arduino Remotely using Blynk App

Now, we are all set to **control the Arduino GPIO pin with Blynk app**. Make sure you have connected both the LEDs and have a working internet connection in your laptop and smartphone. For running the project, **double click on the script and it will start executing**. Now, open the app. **Tap on the play button** in the upper right corner.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in
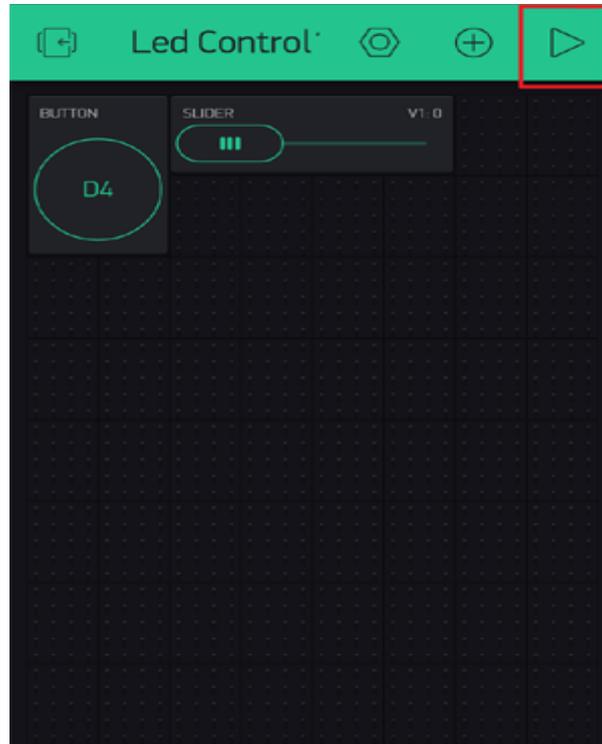
Fig. 14 LED Control

Then **tap on the LED button to turn on the LED and again tap on it to turn the LED off.**

Similarly, move the slider to vary the brightness of the LED.

#define BLYNK_PRINT DebugSerial

#include <SoftwareSerial.h>

SoftwareSerial DebugSerial(2, 3); // RX, TX

#include <BlynkSimpleStream.h>

char auth[] = "YourAuthToken";

BLYNK_WRITE(V1)

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

```
{

  int pinValue = param.asInt();

  analogWrite(5,pinValue);

}

void setup()

{

  DebugSerial.begin(9600);

  Serial.begin(9600);

  Blynk.begin(Serial, auth);

}

void loop()

{

  Blynk.run();

}
```