DR. D.Y. PATIL INSTITUTE OF TECHNOLOGY, PUNE
DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

LAB MANUAL

Internet of Things Laboratory
Subject Code: 217531

Prepared By:
ChetanaShravage
Sonam Singh
Jyotsna Barpute
RituDudhmal

DR. D.Y. PATIL INSTITUTE OF TECHNOLOGY, PUNE
DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

Lab Manual

Second Year Engineering

Semester-IV

Internet of Things Laboratory

Subject Code: 217531

Class: SE AI&DS

Academic year 2023-24

Internet of Things Laboratory

Subject Code: 217531

Teaching Scheme	Credit	Examination Scheme
PR: 04 Hours/Week	02	PR: 25 Marks TW: 50 Marks

Guidelines for Instructor's Manual

The instructor‘s manual is to be developed as a reference and hands-on resource. It should include prologue (about University/program/ institute/ department/foreword/ preface), curriculum of the course, conduction and Assessment guidelines, topics under consideration, concept, objectives, outcomes, set of typical applications/assignments/ guidelines, and references

Guidelines for Student Journal

The laboratory assignments are to be submitted by student in the form of journal. Journal consists of Certificate, table of contents, and handwritten write-up of each assignment (Title, Date of Completion, Objectives, Problem Statement, Software and Hardware requirements, Assessment grade/marks and assessor's sign, Theory- Concept in brief, algorithm, flowchart, test cases, Test Data Set(if applicable), mathematical model (if applicable), conclusion/analysis. Program codes with sample output of all performed assignments are to be submitted as softcopy. As a conscious effort and little contribution towards Green IT and environment awareness, attaching printed papers as part of write-ups and program listing to journal must be avoided. Use of DVD containing students programs maintained by Laboratory In-charge is highly encouraged. For reference one or two journals may be maintained with program prints in the Laboratory.

Guidelines for Laboratory /Term Work Assessment

Continuous assessment of laboratory work should be based on overall performance of Laboratory assignments by a student. Each Laboratory assignment assessment will assign grade/marks based on parameters, such as timely completion, performance, innovation, efficient codes, and punctuality.

Guidelines for Practical Examination

Problem statements must be decided jointly by the internal examiner and external examiner. During practical assessment, maximum weightage should be given to satisfactory implementation of the problem statement. Relevant questions may be asked at the time of evaluation to test the student's understanding of the fundamentals, effective and efficient implementation. This will encourage, transparent evaluation and fair approach, and hence will not create any uncertainty or doubt in the minds of the students. So, adhering to these principles will consummate our team efforts to the promising start of student's academics.

Guidelines for Laboratory Conduction

The instructor is expected to frame the assignments by understanding the prerequisites, technological aspects, utility and recent trends related to the topic. The assignment framing policy need to address the average students and inclusive of an element to attract and promote the intelligent students. Use of open source software is encouraged. Based on the concepts learned. Instructor may also set one assignment or mini-project that is suitable to AI & DS branch beyond the scope of the syllabus.

ASSIGNMENT 1

TITLE: STUDY OF RASPBERRY-PI/ BEAGLE BOARD/ ARDUINO AND OTHER MICROCONTROLLER (HISTORY& ELEVATION)

PROBLEM STATEMENT: -

Study of Raspberry-Pi/ Beagle board/ Arduino and other microcontroller (History & Elevation).

OBJECTIVE:

To study the Hardware platforms and operating systems commonly used in IoT systems.

PREREQUISITE: -

Basic of Basic Electronics Engineering

THEORY:

Internet of Things: - IoT is short for Internet of Things. The Internet of Things refers to the ever-growing network of physical objects that feature an IP address for internet connectivity, and the communication that occurs between these objects and other Internet-enabled devices and systems. The Internet of Things (IoT) refers to the use of intelligently connected devices and systems to leverage data gathered by embedded sensors and actuators in machines and other physical objects. In other words, the IoT (Internet of Things) can be called to any of the physical objects connected with the network.

Examples of IoT: -

1) Apple Watch and Home Kit.

2) Smart Refrigerator.

3) Smart cars.

4) Pet Finders

5) Google assistant devices

6) Smart Toothbrush

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

Fig.A.1: - **Raspberry-Pi**

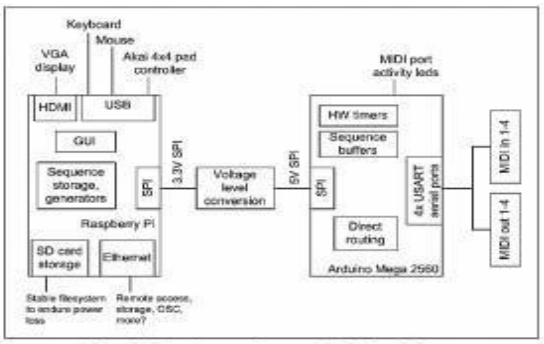


Fig. A.2: -**Raspberry-Pi Architecture**

Here are the various components on the Raspberry Pi board:

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

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

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

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

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

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

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

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

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

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

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

hobbyists. Boot Linux in under 10-seconds and get started on Sitara™ AM335x ARM Cortex-A8 processor development in less than 5 minutes with just a single USB cable.

Here are the various components on the Raspberry Pi board:

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

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

RCA -- An RCA jack allows connection of analog TVs and other similar output devices. · Audio out -- This is a standard 3.55-millimeter jack for connection of audio output devices such as headphones or speakers. There is no audio in.

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

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

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

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

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

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

Fig.B.1: -Beagle Board Black

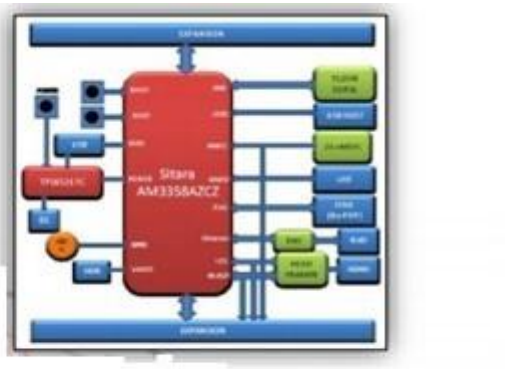


Fig.B.1: - Beagle Board Black architecture

Here are the various components on the Beagle board:

Processor: AM335x 1GHz ARM® Cortex-A8

· 512 MB DDR3 RAM

· 4GB 8-bit eMMC on-board flash storage

· 3D graphics accelerator

· NEON floating-point accelerator

· 2x PRU 32-bit microcontrollers

Connectivity

· USB client for power & communications

· USB host

· Ethernet

· HDMI

· 2x 46 pin headers

Software Compatibility

· Debian

· Android

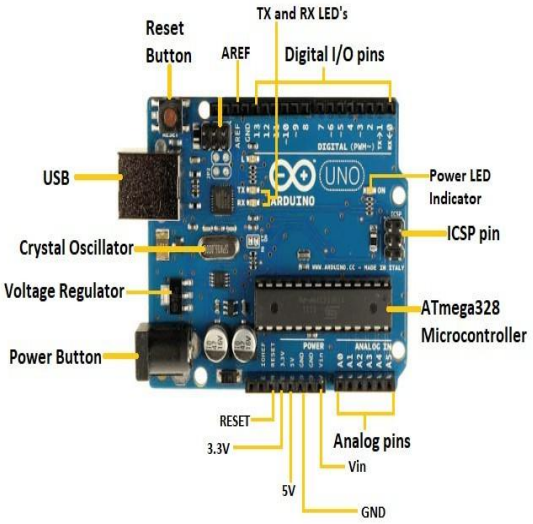
· Ubuntu

· Cloud9 IDE on Node.js w/ BoneScript library

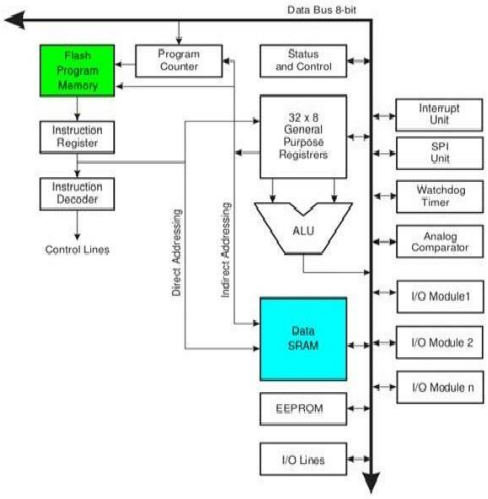
· plus, much more

C) Arduino: -Arduino is an open-source hardware and software company, project and user community that designs and manufactures single-board microcontrollers and microcontroller

kits for building digital devices and interactive objects that can sense and control objects in the physical and digital world. Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards or breadboards (shields) and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers. The microcontrollers are typically programmed using a dialect of features from the programming languages C and C++. In addition to using traditional compiler tool chains, the Arduino project provides an integrated development environment (IDE) based on the Processing language project. Arduino is open-source hardware. The hardware reference designs are distributed under a Creative Commons Attribution Share-Alike 2.5 license and are available on the Arduino website. Layout and production files for some versions of the hardware are also available.



Arduino Board



Arduino Board Architecture

Here are the various components on the Arduino board:

Microcontrollers:

ATmega328P (used on most recent boards)

ATmega168 (used on most ArduinoDiecimila and early Duemilanove)

ATmega8 (used on some older board)

Digital Pins:

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

Analog Pins:

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

Power Pins

VIN (sometimes labelled "9V"). The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin. Note that different boards accept different input voltages ranges, please see the documentation for your board. Also note that the LilyPad has no VIN pin and accepts only a regulated input.

Other Pins

- **AREF**.Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** (Diecimila-only) Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.
- **Analog Reference pin** (orange)
- **Digital Ground** (light green)
- **Digital Pins** 2-13 (green)

·**Digital Pins 0-1/Serial In/Out** - TX/RX (dark green) - These pins cannot be used for digital i/o (digitalRead and digitalWrite) if you are also using serial communication (e.g. Serial.begin).

·**Reset Button** - S1 (dark blue)

·**In-circuit Serial Programmer** (blue-green)

·**Analog In Pins** 0-5 (light blue)

·**Power and Ground Pins** (power: orange, grounds: light orange)

·**External Power Supply In** (9-12VDC) - X1 (pink)

·**Toggles External Power and USB Power** (place jumper on two pins closest to desired supply) SV1 (purple)

·**USB** (used for uploading sketches to the board and for serial communication between the board and the computer; can be used to power the board) (yellow).

CONCLUSION:

In this way we have successfully studied the History & Elevation of Raspberry-Pi/ Beagle board/ Arduino and other microcontroller.

ASSIGNMENT QUESTION

1. What is Raspberry Pi and explain its components.
2. What is Beagle board and explain its components.
3. What is Arduino and explain its components.

ASSIGNMENT 2

TITLE: STUDY OF DIFFERENT OPERATING SYSTEMS FOR RASPBERRY-PI/BEAGLE BOARD/ARDUINO.

PROBLEM STATEMENT: -

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

OBJECTIVE:

1. Hardware platforms and operating systems commonly used in IoT systems.
2. Help the students in providing a good learning environment and also work with real time problems faced in day to day life.

PREREQUISITE: -

Basic of Basic Electronics Engineering

THEORY:

Raspberry-Pi:

The Pi can run the official Raspbian OS, Ubuntu Mate, Snappy Ubuntu Core, the Kodi-based media centers OSMC and LibreElec, the non-Linux based Risc OS (one for fans of 1990s Acorn computers). It can also run Windows 10 IoT Core, which is very different to the desktop version of Windows, as mentioned below. OS which install on Raspberry-Pi: Raspbian, Ubuntu MATE, Snappy Ubuntu, Pidora, Linutop, SARPi, Arch Linux ARM, Gentoo Linux, etc.

How to install Raspbian on Raspberry-Pi:

Step 1: Download Raspbian.

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

are, you need to use certain programs to unzip it.

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

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

Beagle Board Black:

The Beagle Bone Black includes a 2GB or 4GB on-board eMMC flash memory chip. It comes with the Debian distribution factory pre-installed. You can flash new operating systems including Angstrom, Ubuntu, Android, and others.
OS which install on Beagle Bone Black :
Angstrom,Android,Debian,Fedora,Buildroot,Gentoo,NervesErlang/OTP,Sabayon,Ubuntu,Yocto,MINIX 3.

How to install Debian on Beagleboard:

Step 1: Download Debianimg.xz file.

Step 2: Unzip the file.

Step 3: Insert your MicroSD (uSD) card into the proper slot. Most uSD cards come with a full-sized SD card that is really just an adapter. If this is what you have then insert the uSD into the adapter, then into your card reader.

Step 4: Now open Win32 Disk imager, click the blue folder icon, navigate to the debianimg location, and double click the file. Now click Write and let the process

complete. Depending on your processor and available RAM it should be done in around 5 minutes.

Step 5: Alright, once that's done, you'll get a notification pop-up. Now we're ready to get going. Remove the SD adapter from the card slot, remove the uSD card from the adapter. With the USB cable disconnected insert the uSD into the BBB.

Step 6: Now, this next part is pretty straight forward. Plug the USB cable in and wait some more. If everything is going right you will notice that the four (4) leds just above the USB cable are doing the KIT impression. This could take up to 45 minutes; I just did it again in around 5 minutes. Your mileage will vary. Go back and surf reddit some more.

Step 7: If you are not seeing the leds swing back and forth you will need to unplug the USB cable, press and hold down the user button above the uSD card slot (next to the 2 little 10 pin ICs) then plug in the USB cable. Release the button and wait. You should see the LEDs swinging back and forth after a few seconds. Once this happens it's waiting time. When all 4 LEDs next to the USB slot stay lit at the same time the flash process has been completed.

Step 8: Remove the uSD card and reboot your BBB. You can reboot the BBB by removing and reconnecting the USB cable, or hitting the reset button above the USB cable near the edge of the board.

Step 9: Now using putty, or your SSH flavor of choice, connect to the BBB using the IP address 192.168.7.2. You'll be prompted for a username. Type root and press Enter. By default, there is no root password. I recommend changing this ASAP if you plan on putting your BBB on the network. To do this type password, hit enter, then enter your desired password. You will be prompted to enter it again to verify.

Arduino:

The Arduino itself has no real operating system. You develop code for the Arduino using the Arduino IDE which you can download from Arduino - Home. Versions are available for Windows, Mac and Linux. The Arduino is a constrained microcontroller.

Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board. You are literally writing the "firmware" when you write the code and upload it. It's both good and its bad.

CONCLUSION:

In this way we have successfully studied and installed different operating systems for Raspberry-Pi/Beagle board/Arduino

ASSIGNMENT QUESTION:

1. What are different operating systems that are used for Raspberry-Pi?
2. What are different operating systems that are used for Beagle board?
3. What are different operating systems that are used for Arduino?

ASSIGNMENT 3

TITLE: STUDY OF DIFFERENT LOGIC GATES AND SENSORS.

PROBLEM STATEMENT: -

Study of different GATES(AND,OR,XOR), Sensors and basic binary operations.

OBJECTIVE:

1. Hardware platforms and operating systems commonly used in IoT systems.
2. Help the students in providing a good learning environment and also work with real time problems faced in day to day life.

PREREQUISITE: -

Basic of Basic Electronics Engineering

THEORY:

A logic gate is a device that acts as a building block for digital circuits. They perform basic logical functions that are fundamental to digital circuits. Most electronic devices we use today will have some form of logic gates in them. For example, logic gates can be used in technologies such as smartphones, tablets or within memory devices.In a circuit, logic gates will make decisions based on a combination of digital signals coming from its inputs. Most logic gates have two inputs and one output. Logic gates are based on Boolean algebra. At any given moment, every terminal is in one of the two binary conditions, false or true. False represents 0, and true represents 1. Depending on the type of logic gate being used and the combination of inputs, the binary output will differ. A logic gate can be thought of like a light switch, wherein one position the output is off -- 0, and in another, it is on -- 1. Logic gates are commonly used in integrated circuits (IC).

Basic logic gates:

There are seven basic logic gates: AND, OR, XOR, NOT, NAND, NOR, and XNOR.

AND gate:

The AND gate is so named because, if 0 is called "false" and 1 is called "true," the gate acts in the same way as the logical "and" operator. The following illustration and table show the circuit symbol and logic combinations for an AND gate. (In the symbol, the input terminals are at left and the output terminal is at right.) The output is "true" when both inputs are "true." Otherwise, the output is "false." In other words, the output is 1 only when both inputs one AND two are 1.

Input 1	***Input 2***	***Output***
0	*0*	*0*
0	*1*	*0*
1	*0*	*0*
1	*1*	*1*

OR gate:

The OR gate gets its name from the fact that it behaves after the fashion of the logical inclusive "or." The output is "true" if either or both of the inputs are "true." If both inputs are "false," then the output is "false." In other words, for the output to be 1, at least input one OR two must be 1.

Input 1	***Input 2***	***Output***
0	*0*	*0*
0	*1*	*1*
1	*0*	*1*
1	*1*	*1*

XOR gate:

The XOR (exclusive-OR) gate acts in the same way as the logical "either/or." The output is "true" if either, but not both, of the inputs are "true." The output is "false" if both inputs are "false" or if both inputs are "true." Another way of looking at this circuit is to observe that

the output is 1 if the inputs are different, but 0 if the inputs are the same.

Input 1	***Input 2***	***Output***
0	*0*	*0*
0	*1*	*1*
1	*0*	*1*
1	*1*	*0*

Inverter or NOT gate:

A logical inverter, sometimes called a NOT gate to differentiate it from other types of electronic inverter devices, has only one input. It reverses the logic state. If the input is 1,

then the output is 0. If the input is 0, then the output is 1.

Input	***Output***
0	*1*
1	*0*

NAND gate:

The NAND gate operates as an AND gate followed by a NOT gate. It acts in the manner of the logical operation "and" followed by negation. The output is "false" if both inputs are "true." Otherwise, the output is "true."

Input 1	***Input 2***	***Output***
0	*0*	*1*
0	*1*	*1*
1	*0*	*1*
1	*1*	*0*

NOR gate:

The NOR gate is a combination OR gate followed by an inverter. Its output is "true" if both inputs are "false." Otherwise, the output is "false."

Input 1	***Input 2***	***Output***
0	*0*	*1*
0	*1*	*0*
1	*0*	*0*
1	*1*	*0*

Complex operations can be performed using combinations of these logic gates. In theory, there is no limit to the number of gates that can be arrayed together in a single device. But in practice, there is a limit to the number of gates that can be packed into a given physical space. Arrays of logic gates are found in digital ICs. As IC technology advances, the required physical volume for each individual logic gate decreases and digital devices of the same or smaller size become capable of performing ever-more-complicated operations at ever-increasing speeds.

Composition of logic gate. High or low binary conditions are represented by different voltage levels.

The logic state of a terminal can, and generally does, often change as the circuit processes data.

In most logic gates, the low state is approximately zero volts (0 V), while the high state is approximately five volts positive (+5 V).

Logic gates can be made of resistors and transistors or diodes. A resistor can commonly be used as a pull-up or pull-down resistor. Pull-up and pull-down resistors are used when there are any unused logic gate inputs to connect to a logic level 1 or 0. This prevents any false. Switching of the gate. Pull-up resistors are connected to Vcc (+5V), and pull-down resistors are connected to ground (0 V).Commonly used logic gates are TTL and CMOS. TTL, or Transistor-Transistor Logic, ICs will use NPN and PNP type Bipolar Junction Transistors. CMOS, or Complementary Metal-Oxide-Silicon, ICs are constructed from MOSFET or JFET type Field Effect Transistors. TTL IC's may commonly be labeled as the 7400 series of chips, while CMOS ICs may often be marked as a 4000 series of chips.

Types of Sensors:

There are many different types of sensors. Here at Variohm, we offer a full range of sensors for industrial and commercial use.

Sensors are used throughout almost every industry for applications which we come into contact with on a daily basis as well as more industrial and specialist applications.

Sensors can be found in the home, the office, in our cars, buses, trains, trams, computers, medical facilities, labs, power plants, restaurants, food processing factories, production lines etc A Sensor is used to take a measurement, the measurement will be processed and the result of the process, an output will be given. The output will then cause something to change or move. A simple example is the temperature sensor in a thermostat. The temperature sensor is constantly monitoring the temperature, once the measurement taken reaches the desired temperature, the measurement is processed and the output causes the boiler to switch off.

There are many different types of sensors, the main categories are:

- PositionSensors
- PressureSensors
- TemperatureSensors
- ForceSensors
- VibrationSensors
- PiezoSensors
- FluidPropertySensors
- HumiditySensors

- Straingauges
- PhotoOpticSensors
- Flow and Level Switches

These categories can all be split further into subcategories for example, within position sensors there are the following types:

- Contacting
- Non-contacting
- Rotary
- Linear

And these types of sensors can be split even further, within non-contacting you have the following types of sensors:

- Halleffect
- Capacitive
- EddyCurrent
- Ultrasonic
- Laser
- Proximity

By splitting one category – Position Sensors it is clear to see that the number of sensors present in today's world is so vast that one blog post could not cover every type of sensor. However, here is an overview of different types of sensors Variohm can offer.

Types of Sensors – Position Sensors:

As discussed above there are many varieties of position sensor; linear, rotary, contacting, non-contacting and use a variety of different technologies. Position sensors are used to measure and monitor the position or displacement of an object.We have been supplying position sensors for over 40 years and have developed our own range of position sensors which have been added to the comprehensive range from our suppliers and partners. Our own range includes:

- LinearpositionSensors
- VLP
- VXP

- ELPM
- VLPSC
- RotaryPositionSensors
- Euro-XHallEffect
- Euro-XPPuck–2partpuckandmagnetdesign
- Euro–XPD–Dshaft
- CMRS
- CMRT
- CMRK

Types of Sensors – Pressure Sensors:

Pressure sensors are often split into the following two categories; Pressure transducers and pressure switches. The main difference is that pressure transducers give accurate feedback on real-time pressure and pressure switches have a set limit which causes them to switch. Both pressure switches and pressure transducers have mechanisms which use the formula – Pressure = force divided by area to detect pressure.

Pressure sensors can measure the pressure in gases, liquids or solids and are used in a variety of industries. Underwater pressure transducers are referred to as level meters as the pressure they measure is directly related to the level of the water.

Pressure can be gauge, differential, absolute or vacuum and can be measured in Bar or PSI.

Many of our pressure sensors come from our trusted suppliers and we also have our own range of;

Pressure Transducers – EPT range .

Pressure Switches - EPS range.

Types of Sensors – Load Cells and Force Sensors:

Load Cells are available in a wide variety of shapes and sizes. They are used to measure various types of force, the main one being weight. Load cells are used in all types of scales; from bathroom scales to counting scales, industrial scales, truck scales, hopper scales and everything in between.

Most load cells use internal strain gauges to monitor force based on the level of distortion on the strain gauge.

Our load cells come from our trusted suppliers. And can be seen on our website. Further reading on Load Cells

Load Cells for Weighing

Load Cell Types and Applications

Types of Sensors – Temperature Sensors:

Temperature Sensors are another very common type of sensor – they are all around us. Temperature sensors are used to measure and monitor temperature, whether this is the main variable requiring measuring or a secondary variable which requires monitoring as a safety precaution within another application.

Different types of temperature sensors will require different approvals. Medical approvals will be required for temperatures used for patient monitoring or within medical devices. Other certifications will be required for temperature sensors in food and beverage applications.

Temperature sensors come in many different shapes, sizes and types; thermistors, probes, thermocouples, RTDs and temperature transducers to name a few. Temperature Sensors is another type of sensor which we carry our own range of:

ETP – Euro sensor Temperature probes – these are fully customizable to your requirements.

Further reading on temperature sensors

Temperature Sensor Applications

Types of Temperature Sensors

CONCLUSION:

In this way we have successfully studied different GATES (AND,OR,XOR), sensor basic binary operations.

ASSIGNMENT QUESTION:

1. Explain the fundamental principles of AND, OR, and XOR gates, including their

truth tables. How do these gates contribute to digital logic circuits?

2. Investigate the working mechanisms of temperature, light, and proximity sensors. How do these sensors convert physical phenomena into electrical signals, and what are their respective applications in real-world scenarios?

DR. D.Y. PATIL INSTITUTE OF TECHNOLOGY, PUNE
DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

ASSIGNMENT NO: 4

Title:
Study of Connectivity and configuration of Raspberry-Pi /Beagle board/Arduino circuitwith basic peripherals like LEDS. Understanding GPIO and its use in the program.

Aim:
The aim of this lab is to introduce students to the connectivity and configuration of microcontroller boards such as Raspberry Pi, BeagleBoard, or Arduino with basic peripherals like LEDs. Students will gain hands-on experience in establishing connections between the microcontroller boards and peripherals, understanding GPIO (General Purpose Input/Output) pins, and writing programs to control external devices.

Objectives

- To understand the basic components and functionalities of microcontroller boards (Raspberry Pi, BeagleBoard, Arduino).
- To learn the process of connecting LEDs to GPIO pins (for Raspberry Pi / BeagleBoard) or digital pins (for Arduino).
- To comprehend the concept of GPIO and its role in interfacing with external peripherals.
- To gain practical experience in writing programs to control LEDs using GPIO pins on microcontroller boards.
- To familiarize students with the software development environment (IDEs, libraries, etc.) relevant to the chosen microcontroller board.

Hardware Requirement*:* Raspberry-Pi /Beagle board/Arduino, LED etc.

Theory:

Connecting Hardware Peripherals to Raspberry-Pi board. Raspberry-Pi setup through SSH (HeadlessConfiguration of Raspberry-Pi-3 (VNC, Putti))

How to Connect a Raspberry-Pi to the Laptop or PC Display

Required devices: Raspberry Pi, Ethernet Cable, Laptop/ PC, SD Card with Raspbian Micro USB Cable.

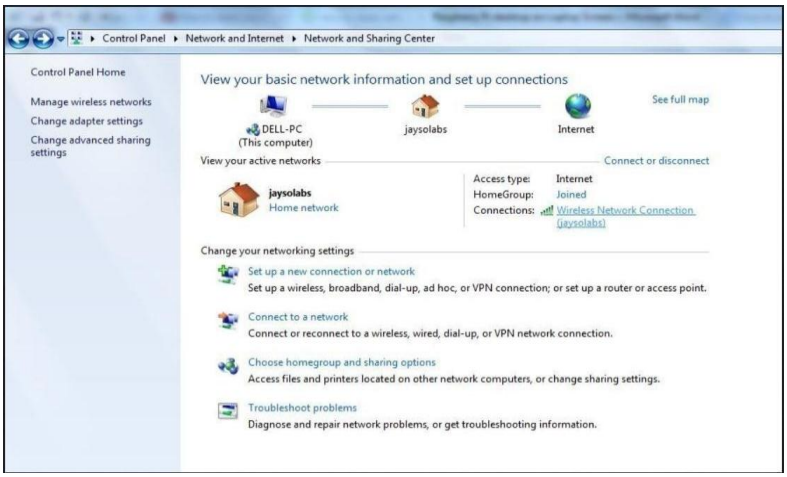
How does it Work?

To connect a Raspberry Pi to a laptop or PC display, you can simply use an Ethernet cable.

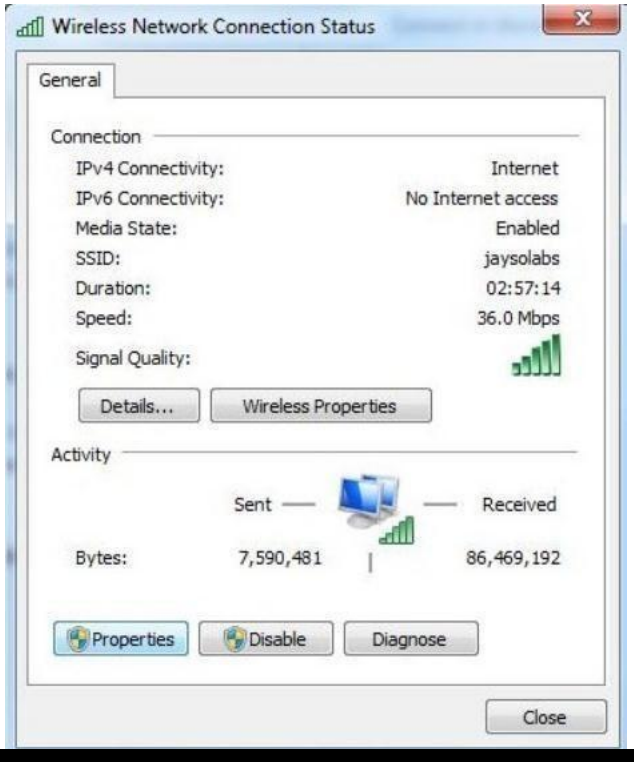
The Raspberry Pi's desktop GUI can be viewed through the laptop or PC display using a 100mbps Ethernet connection between the two. We used VNC server software to connect the Raspberry-Pi to our laptop or PC. Installing the VNC server on your Raspberry-Pi allows you to see the Raspberry Pi's desktop remotely.

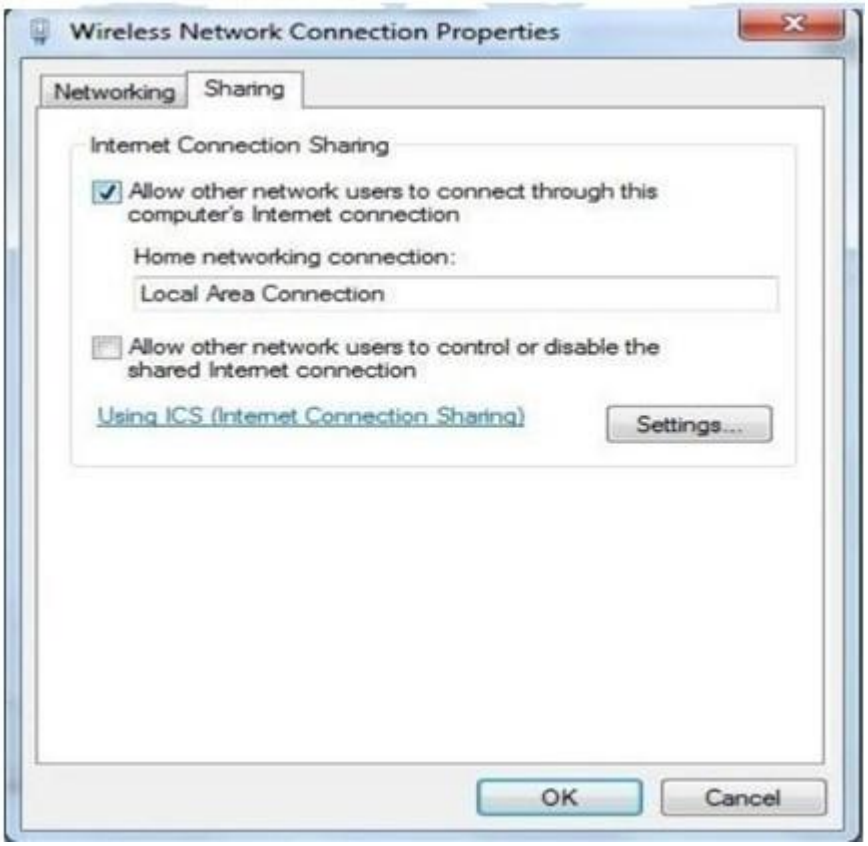
- Setting up your Raspberry Pi
- Install Raspbian OS on blank SD card. Insert this SD card into Raspberry-Pi board.
- Connect micro USB cable to power the Raspberry-Pi. Sharing Internet Over Ethernet in Window OS.

This step explains how you can share your laptop or PC with the Raspberry Pi via Ethernet cable.



To share internet with multiple users over Ethernet, go to Network and Sharing Center. Then click on the WiFi network and Double click on Wireless area connection Click on Properties (shown below)
Go to “Sharing” tab and click on “Allow other network users to connect”.

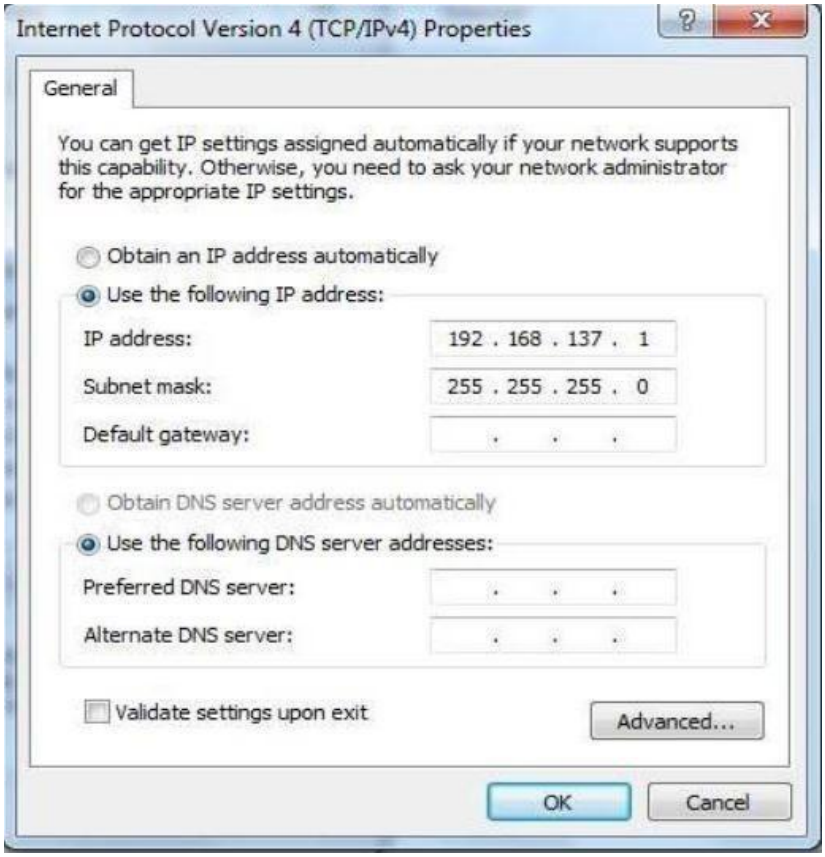




After this, make sure that the networking connection is changed to “Local Area Connection”.

Now, to check the IP assigned to the network established, click on the new local area connection link

created:



Now open command prompt.
Ping the broadcast address of your IP. (Type) E.g. : ping 192.168.137.1
Stop the ping after 5 seconds.
To get the IP address of Raspberry Pi in the established network, use the Software "AdvanceIP Scanner". It is free software.

Setting up the VNC Server to Connect Your Raspberry Pi to the Laptop or PC DisplayFirst install VNC server and Putty on your laptop/ PC

Open Putty Software, and enter login ID: pi and Password: raspberry.

After that, enter commands into Putty i.e,

$ sudo apt-get update

$ sudo apt-get install tightvncserver

$ vncserver :1

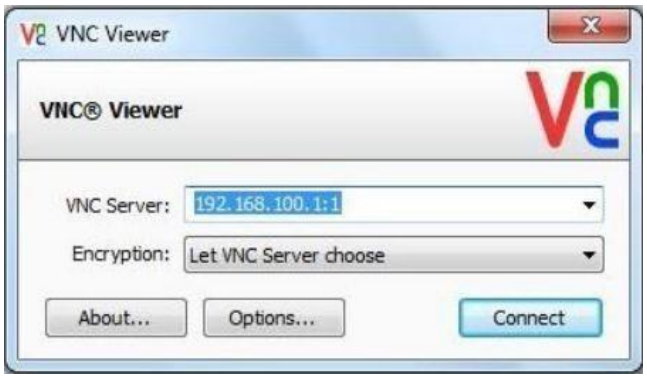
You will be prompted to enter and confirm a password.This will be asked only once, during first time setup.

Enter an 8 digit password.

Note that this is the password you will need to use to connect to your Raspberry Pi remotely.Setting Up the Client Side (Laptop or PC)

Download VNC client and install it.

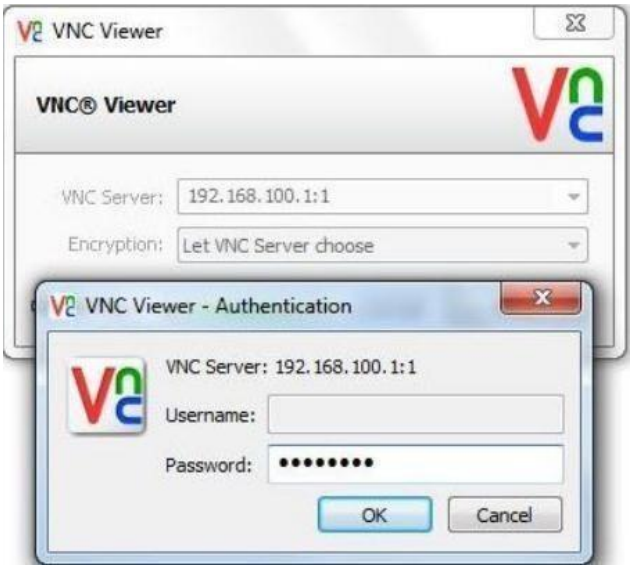
When you first run VNC viewer, you will see following screen



Enter the IP address of your Raspberry Pi given dynamically by your laptop and append with : 1(denoting port number) and press connect.

You will get a warning message, press ‘Continue’:

Enter the 8 digit password which was entered in the VNC server installation on your



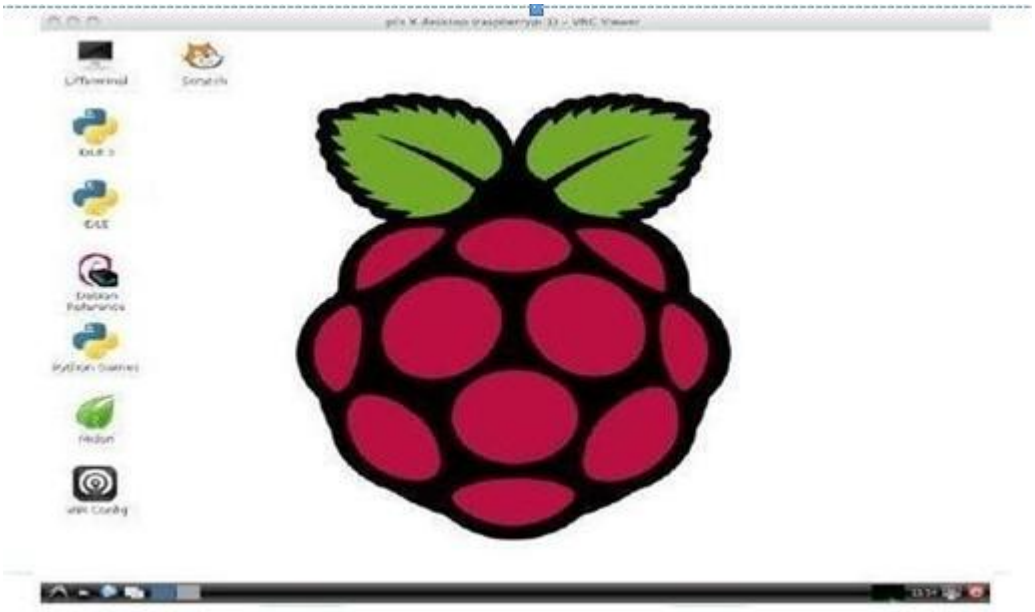
Raspberry Pi:

Finally, the Raspberry Pi desktop should appear as a VNC window.

You will be able to access the GUI and do everything as if you are using the Pi's keyboard, mouse, andmonitor directly.

Raspberry-Pi setup using mouse, keyboard and monitor

To work with Raspberry-Pi, we have to connect some peripherals to it. They are as follows



A monitor with power cable and data cable

A micro USB power supply, A wired keyboard and mouse, or a wireless keyboard

and mouse A micro SD card, Monitors – HDMI

There are several different types of monitor that you can use with the Raspberry Pi:

Most modern television sets and monitors have an HDMI port, and are the easiest to get working withthe Raspberry Pi.

You can use an HDMI cable to connect the Raspberry Pi directly to the television or monitor.VGA

Some old monitors have a VGA port.

These can be trickier to use as you'll need an HDMI-to-VGA converter, which can change digital video toanalogue video.

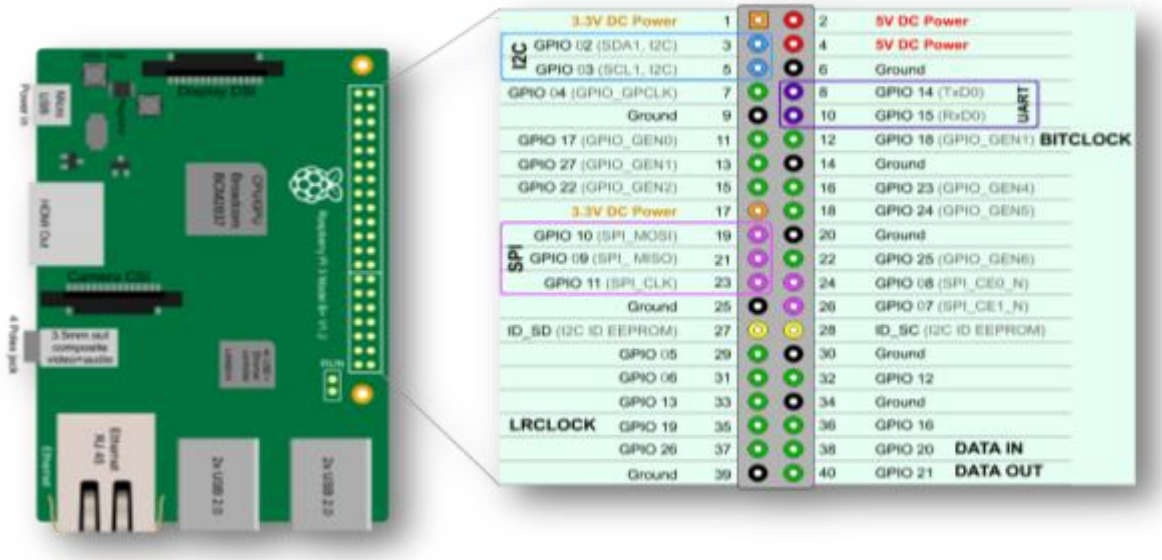
A simple port adapter won't work.

Power supplies

For Raspberry Pi 3, it's recommended to use a 5V,
2.5A power supply.Mobile device charger

Many mobile devices are powered using a 5V micro USB charger.

These can often be used to power the Raspberry Pi, although it's worth checking



that they providesufficient voltage and current (5V / 1.2 - 2.5A).

Keyboard
and
mouse
Wired
keyboard

and
mouse

Any standard USB keyboard and mouse can be used with the Raspberry Pi. These plug and play devices will work without any additional driver. Simply plug them into the Raspberry Pi and they should be recognized when it starts up.

Bluetooth keyboard and mouse

Bluetooth keyboards and mice can work with the Raspberry Pi, but your success rates will vary depending on the model and manufacturer. It's best to consult the manufacturer's documentation to see whether or not a device is compatible with the Raspberry Pi.

SD cards

The latest version of Raspbian, the default operating system recommended for the Raspberry Pi, requires an 8GB (or larger) micro SD card. Not all SD cards are made equal, and some have higher failure rates than others. Any 8GB SD card will work, although you'll need to follow the software setup guide to learn how to load an operating system onto the card.

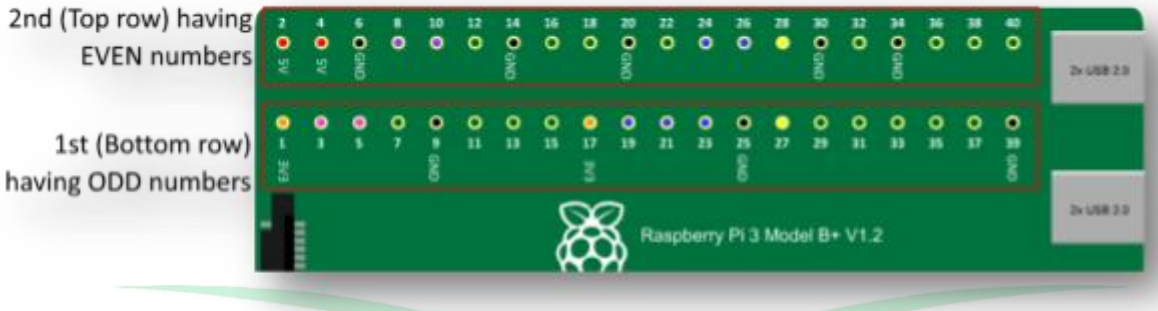
Understanding GPIO pins on Raspberry Pi board and its use in program

Introduction:

Raspberry Pi 3 Model B is the latest version of
raspberry pi board. It is released on 29
February.

The above figure shows the Raspberry Pi 3 Model B and It's GPIO pins

General-purpose input/output (GPIO) is a generic pin on an integrated circuit or computer board whosebehavior—including whether it is an input or output pin—is controllable by the user at run time.



There are 40 pins available on board of Raspberry pi 3 model B.

The Pins are arranged in a 2×20 fashion as shown in the figure aboveOut of these, 26 pins are GPIO pins

As you can observe, the numbers to the pins are given in zigzag manner.

The first (bottom) row starts with number '1'. So the pins in this row have odd numbers i.e. from 1 to 39.

The 2nd (Top) row starts with number '2'. So the pins in this row have even numbers i.e. from 2 to 40. Out of 40 pins, 26 pins are GPIO pins,8 pins are Ground (GND) pins,2 pins are 5V power supply pins 2 pins are 3.3V power supply pins, 2 pins are not used

Now if you're coming to the Raspberry Pi as an Arduino user, you're probably used to referencing pins with a single, unique number.

In Raspberry Pi there are two different numbering schemes for referencing Pi pin numbers:Broadcom chip-specific pin numbers (BCM)

Physical pin numbers (BOARD)

You're free to use either number-system.

The programs require that you declare which scheme you're using at the very beginning of yourprogram.

In a program, at a time, you can use only one number scheme. Broadcom chip-specific pin numbers (BCM)

BCM - Broadcom pin number, commonly called "GPIO", these are the ones you probably want to usewith RPi.GPIO

The parameter used for this system is (GPIO.BCM).

This is a lower level way of working - it refers to the channel numbers on the Broadcom SOC.

To use this system, you have to always work with a diagram describing which channel number goesto which pin on the RPi board.

Your script could break between revisions of Raspberry Pi boards.

In this system 26 GPIO pins are named as GPIO 01 to GPIO 26

Physical Numbering System (BOARD)

This system uses physical - Numbers corresponding to the pin's physical location on the header.The numbers printed on the board are physical numbering system.

The parameter used for this system is (GPIO.BOARD).

The advantage of using this numbering system is that your hardware will always work,regardless of the board revision of the RPi.

You will not need to rewire your connector or change your code. In this system

26 GPIO pins are named between 0 to 40

The below table summarizes the pinout of Raspberry-Pi in both the number systems.

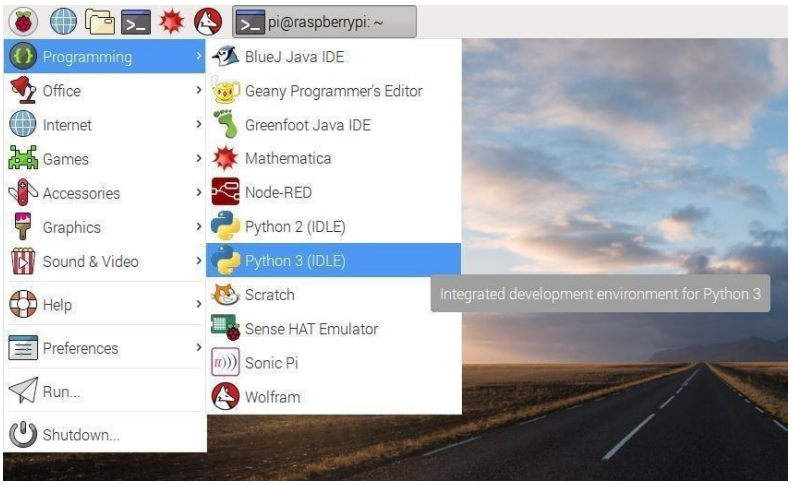
Sr No.	Pins		BOARD Pin No	BCM Pin No
1	3.3V		1, 17	1, 17
2	5V		2, 4	2, 4
3	Ground		6,9,14,20,25,30,34,39	6,9,14,20,25,30,34,39
4	UART	TXD	8	GPIO 14
		RXD	10	GPIO 15
5	I2C1	SDA1	3	GPIO 2
		SCL1	5	GPIO 3
	I2C0	SDA0	27	ID_SD
		SCL0	28	ID_SC
6	SPI0	MOSI 0	19	GPIO 10
		MISO 0	21	GPIO 9
		SCLK 0	23	GPIO 11
		CE 0	24	GPIO 8
	SPI1	MOSI 1	38	GPIO 20
		MISO 1	35	GPIO 19
		SCLK 1	40	GPIO 21
		CE 1	26	GPIO 7

The Python IDLE shell and command line

To use the Python IDLE IDE for programming in Raspberry Pi

follow this

Open Python 3 from the main menu:



Or open terminal window and type the command sudo idle 3.5 and press enterInstall all libraries required for Buzzer as given above.

Write the program as per algorithm given below

Save with Ctrl + S and run with F5.

See output on Python Shell or Terminal Window.Raspberry Pi GPIO programming using Python

The Raspberry Pi is often used in conjunction with other hardware to create interestingelectronic projects.

The Pi 3 comes with 40 GPIO pins that you can use to interface with various hardware devices— for both receiving data from them or for writing data to them.

To do this, we have to program the GPIO pins. To do this, special libraries in Python are used. Toinclude these libraries in the program, the command used is 'import'

This way, we can write applications to both read and also to control devices, i.e., turn them onand off, etc.

The default operating system used in Raspberry-Pi is Raspbian.

The Python package used for Raspberry Pi GPIO programming is RPi.GPIO. It is already installedin Raspbian.

If you are using any other operating system, the package can be installed by using the followingcommand:

$ sudo pip install RPi.GPIO

There are important 8 steps in the programming of Raspberry-Pi using Python as follows Import the RPi.GPIO library using the following command
import RPi.GPIO as GPIO Import the Time library using the following command
import time

Set numbering scheme to be used. The method used for this is GPIO.setmode(). We will usephysical number scheme. So the method is written as

GPIO.setmode(GPIO.BOAD)

Set the pin mode as INPUT or OUTPUT using the commands GPIO.setup(channel, GPIO.IN)GPIO.setup(channel, GPIO.OUT)

Read input using following commandGPIO.input(pin no)

Write output using following comman
GPIO.output(pin no, state)

Give delay using command using following command time.sleep(1) # delay for 1 second Clean up GPIO and exit using following commands GPIO.cleanup()

print("Exiting...")

You must clean up the pin set-ups before your program exits otherwise those pin settings willpersist, and that might cause trouble when you use the same pins in another program.

The Pi 'expresses its displeasure' with a warning.

To clean up the entire set of pins, invoke GPIO.cleanup().

If you want only a few pins to be cleaned up, then the pin numbers should be provided as GPIO.cleanup (channel_list).

Anyway, you can suppress the warning messages by calling GPIO.setwarnings (False).Save the program with proper name. The file is saved with extension '.py'.

The IDE named 'IDLE' used for programming is an interpreter and not a compiler. So to run thepython program, we need to give the super user permission as follows.

Studying Connectivity and Configuration of Raspberry Pi board with basic peripherals (LEDs)

Aim/Objectives:

To understand the concept of Led bar and the common anode & common cathode configuration.To interface LED bar with Raspberry Pi.

Software: Raspbian OS , IDLE editor

Hardware Modules: Raspberry Pi Board, LED bar, Monitor

Introduction to "LED"

LED is a Light Emitting Diode. Light emitting diode is a two lead semiconductor light source. It is a p-n junction diode, which emits light when it is activated. When a suitable voltage is applied to the leads, electrons are able to recombine with electron holes within the device, and the color of light (corresponding to the energy of photon) is determined by the energy band gap of the semiconductor. It has two terminals named as 'anode (+ve)' and 'cathode (-ve)'. Battery is connected to these two terminals. When LED is forward biased, it emits light. In LED bar number of LEDs are connected in series (in our case 8 LEDs are connected) LED bar has two configurations as Common Anode: In this, anode terminal of all the LEDs are made common and connected to the VCC (+5v). By controlling cathode terminal we can make LED ON or OFF (current sourcing). Common Cathode: In this, cathode terminal of all the LEDs are made common and connected to the Ground (0v). By controlling anode terminal we can make LED ON or OFF (current sinking).

Safety precautions:

Raspberry-Pi provides
3.3V and 5V VCC pins
Raspberry-Pi operates on
3.3V.

Various sensors and actuators operate on different voltages.

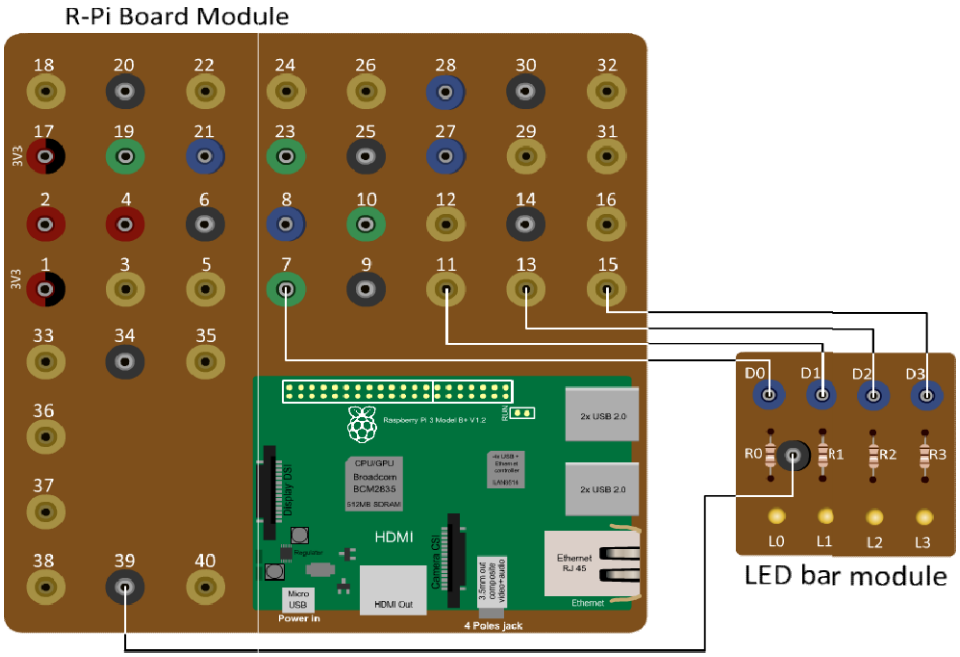
Read datasheet of a given sensor or an actuator and then use appropriate VCC pin to connect asensor or an actuator.

Ensure that signal voltage coming to the Raspberry-Pi from any sensor or actuator does notexceed 3.3V.

If signal/data coming to Raspberry-Pi is greater than 3.3V then use voltage level shifter moduleto decrease the incoming voltage.

The Raspberry-Pi is a costly device, hence you should show the circuit connections to yourinstructor before starting your experiment.

Interface diagram:



Steps for assembling circuit:

Connect led bar module pins from D0- D3 to Raspberry Pi GPIO pins 7, 11, 13, 15 respectively.Connect led bar module pin COM to the GND pin of Raspberry-Pi module.

Procedure:

Write the program as per the algorithm given below.

Save program.

Run code using Run module.

Algorithm:

Import GPIO and Time library

Set mode i.e.
GPIO.BOA
RD

Set GPIO 8
pins as a
Output pin
Print
message
“ON”

After 1 second time delay, Make all the
leds ON one by onePrint message
“OFF”

After 1 second time delay, Make all the leds OFF one by one

Conclusion: -

In this lab, you have learned how to connect LEDs to microcontroller boards such as Raspberry Pi, BeagleBoard, or Arduino and control them using GPIO pins. Understanding GPIO and its usage in programming allows you to interface with various external devices and sensors, opening up a wide range of possibilities for projects and applications.

Assignment No.5

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

Aim: The aim of this lab is to familiarize students with the process of writing a program using Arduino to control LEDs, either by turning them on and off individually or by making them blink at a specific rate. Students will gain practical experience in programming Arduino boards and interfacing with basic peripherals like LEDs.

Objectives:

1. To understand the basic concepts of Arduino programming and the Arduino IDE.
2. To learn how to configure digital pins on an Arduino board for output.
3. To comprehend the logic required to control LEDs, including turning them on and off or making them blink.
4. To gain proficiency in writing Arduino programs to manipulate digital pins and control LED behavior.
5. To develop troubleshooting skills for identifying and rectifying common programming errors.

Materials Required:

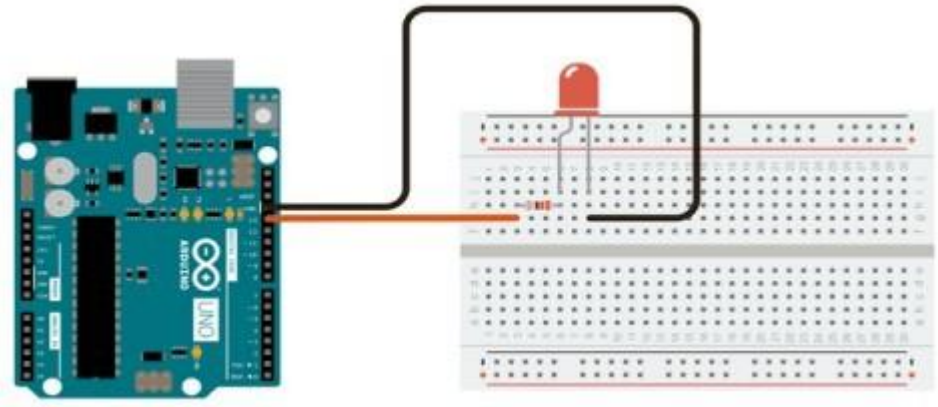
- Arduino board (e.g., Arduino Uno)
- LEDs
- Resistors (220Ω recommended)
- Breadboard
- Jumper wires
- USB cable (for connecting Arduino)
- Computer with Arduino IDE installed

Procedure:

1. **Introduction to Arduino:**

- Provide an overview of Arduino boards, their components, and their capabilities.
- Introduce the Arduino Integrated Development Environment (IDE) and its features.

2. **Circuit Setup:**
   - Explain the connection of LEDs to the Arduino board.
   - Connect the anode (longer leg) of the LED to a digital pin on the Arduino through a resistor, and connect the cathode (shorter leg) to the ground (GND) pin.
   - If using multiple LEDs, repeat the above step for each LED, connecting them to different digital pins.



3. **Writing the Program:**
   - Open the Arduino IDE on the computer.
   - Write a program to control the LEDs, either by turning them on and off individually or by making them blink.
   - For turning LEDs on and off individually:
     - Use the **digitalWrite()** function to set the desired digital pins to **HIGH** or **LOW** to turn the LEDs on and off.
   - For making LEDs blink:
     - Use the **delay()** function to create a blinking effect by alternating between turning the LEDs on and off with a specified delay between each state change.
4. **Uploading the Program:**

- Connect the Arduino board to the computer using a USB cable.
- Select the appropriate board type and port in the Arduino IDE.
- Click the "Upload" button to compile and upload the program to the Arduino board.

5. **Testing and Observation:**
   - Observe the behavior of the LEDs connected to the Arduino board.
   - Verify whether the LEDs are turning on and off individually or blinking as per the programmed logic.
6. **Experimentation and Modification:**
   - Encourage students to experiment with the program by modifying the delay times or adding additional LEDs to control.
   - Guide students in exploring different patterns and sequences for LED control, such as fading, pulsing, or random blinking.
7. **Troubleshooting:**
   - Assist students in troubleshooting any programming errors or circuit connection issues.
   - Encourage students to analyze error messages and debug their code systematically.

Conclusion: Through this lab, students have gained hands-on experience in writing Arduino programs to control LEDs, either by turning them on and off individually or by creating a blinking effect. By understanding the basics of Arduino programming and digital pin manipulation, students are equipped with foundational skills for further exploration and experimentation in the field of embedded systems and electronics.

Assignment No.6

Title:
Create a program that illuminates the green LED if the counter is less than 100, illuminates the yellow LED if the counter is between 101 and 200 and illuminates the red LED if the counter is greater than 200.

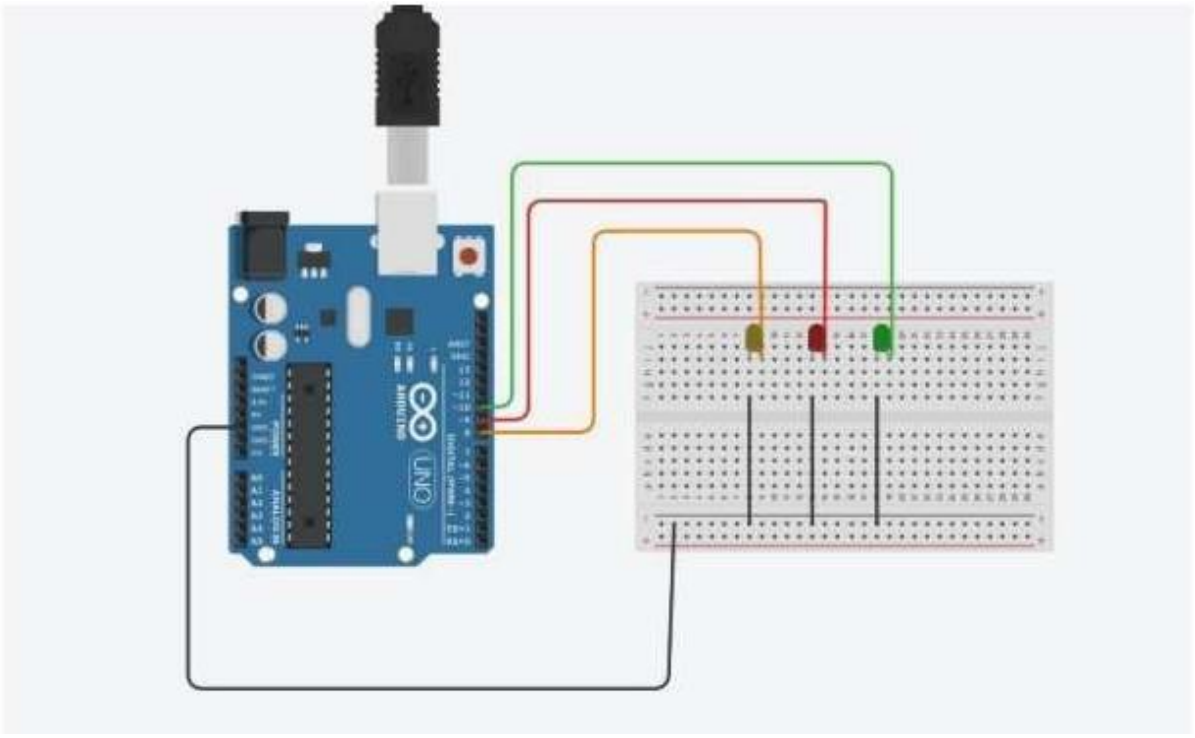
Aim:
The aim of this lab is to develop a program using Arduino that controls the illumination of three LEDs (green, yellow, and red) based on a counter variable. The program will illuminate the green LED if the counter is less than 100, the yellow LED if the counter is between 101 and 200, and the red LED if the counter is greater than 200.

Objectives:
1. To understand the concept of conditional statements and variable comparison in Arduino programming.
2. To learn how to interface multiple LEDs with Arduino and control their illumination based on specific conditions.
3. To develop proficiency in writing Arduino programs that implement complex logic and decision-making processes.
4. To gain practical experience in testing and verifying the functionality of the program using LEDs as visual indicators.
5. To reinforce troubleshooting skills for identifying and resolving programming errors.

- **Hardware Requirement*:*** Arduino board (e.g., Arduino Uno), Green, yellow, and red LEDs, Resistors (220Ω recommended), Breadboard, Jumper wires, USB cable (for connecting Arduino)

Software Requirement: Arduino IDE

Procedure:
1. **Introduction to the Project:**
   - Provide an overview of the project objectives and the behavior expected from the program.
   - Explain the concept of conditional statements and variable comparison, highlighting their importance in achieving the desired functionality.
2. **Circuit Setup:**
   - Connect the green LED to digital pin 2, the yellow LED to digital pin 3, and the red LED to digital pin 4 on the Arduino board.
   - Connect the cathode (shorter leg) of each LED to the ground (GND) pin on the Arduino through a resistor (220Ω recommended).
   - Ensure that the anode (longer leg) of each LED is connected to its respective digital pin.



3. **Writing the Program:**
   - Open the Arduino IDE on the computer.
   - Write a program that reads a counter variable and controls the illumination of the LEDs based on its value.
   - Define a counter variable and initialize it to a starting value (e.g., 0).
   - Use conditional statements (if-else) to check the value of the counter variable and control the illumination of the LEDs accordingly.
   - Illuminate the green LED if the counter is less than 100, the yellow LED if the counter is between 101 and 200, and the red LED if the counter is greater than 200.

     Or

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

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

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

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

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

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

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

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

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

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

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

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

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

Going Deeper: Arduino Pedestrian Crossing

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

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

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

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

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

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

Code for the Arduino Pedestrian Crossing

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

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

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

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

Change the entire loop function to the following instead:

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

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

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

Here's how this looks (sped up):

Arduino Traffic Light with Junction

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

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

and 13.Code for the Arduino Traffic Light with

Junction

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

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

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

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

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

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

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

4. **Uploading the Program:**
   - Connect the Arduino board to the computer using a USB cable.
   - Select the appropriate board type and port in the Arduino IDE.
   - Click the "Verify" button to compile and upload the program to the Arduino board.
5. **Testing and Observation:**
   - Observe the behavior of the LEDs connected to the Arduino board.
   - Verify whether the LEDs illuminate according to the programmed logic based on the counter variable's value.
6. **Experimentation and Modification:**
   - Encourage students to experiment with different counter values and observe how the LEDs respond.
   - Guide students in modifying the program to implement additional features or behaviors, such as adding a reset function for the counter.
7. **Troubleshooting:**
   - Assist students in troubleshooting any programming errors or circuit connection issues.

- Encourage students to analyze the program's behavior and debug any unexpected outcomes systematically.

Conclusion: Through this lab, students have developed a program using Arduino that controls the illumination of three LEDs based on a counter variable's value. By understanding conditional statements and variable comparison, students have learned how to implement complex logic and decision-making processes in Arduino programming. This lab reinforces fundamental programming concepts while providing hands-on experience in interfacing with external peripherals and testing program functionality using visual indicators (LEDs).

DR. D.Y. PATIL INSTITUTE OF TECHNOLOGY, PUNE
DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

ASSIGNMENT NO: 7

Title:
Create a program so that when the user enters "B" the green light blinks, "g" the green light is illuminated "y" the yellow light is illuminated and "r" the red light is illuminated

Aim:
Connectivity, configuration and control of LED using Arduino circuit under different conditions.

Objectives
1. Hardware platforms and operating systems commonly used in IoT systems.
2. Help the students in providing a good learning environment and also work with real time problems faced in day to day life.

Hardware Requirement*:* Arduino, LED, 220-ohm resistor etc.

Software Requirement: Arduino IDE

Theory: Create a simple Arduino traffic light project, a fun endeavor that can be completed in under an hour. Learn how to construct it using an Arduino and explore the circuit for an advanced variation

To build an Arduino Traffic Light Controller, in addition to the basic Arduino, you'll require the following components:

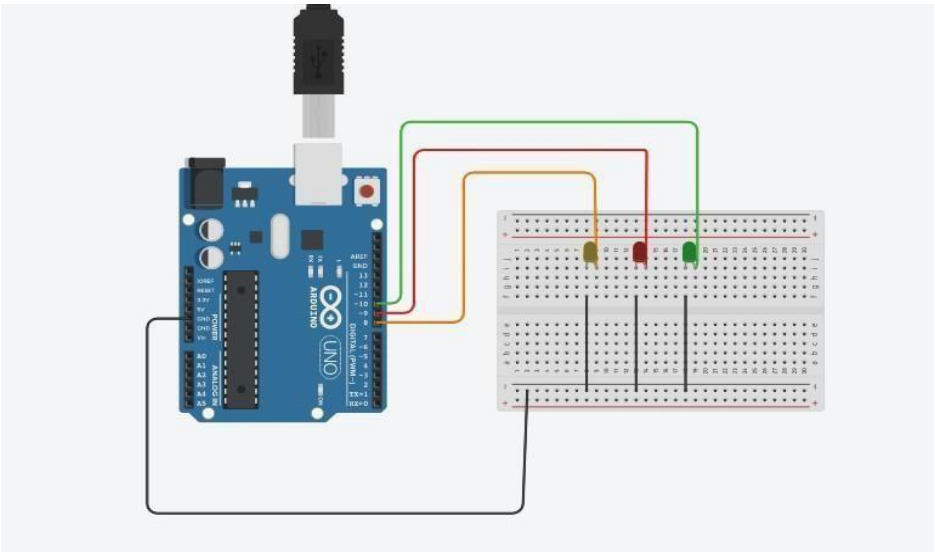
- 1x 10k-ohm resistor
- 1x push-button switch
- 6x 220-ohm resistors
- A breadboard
- Connecting wires
- Red, yellow, and green LEDs

Arduino Traffic Light: Getting Started.
To begin, let's start with a simple single traffic light circuit.

Here's the basic circuit:
1. Connect the anode (long leg) of each LED to digital pins eight, nine, and ten (using a

220-ohm resistor for each).

2. Connect the cathodes (short leg) of the LEDs to the Arduino's ground



Syntax is as follow:

Begin by defining variables to address the lights by name rather than a number. Initiate a new Arduino project and start with these lines

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

Next, let's incorporate the setup function, where you'll configure the red, yellow, and green LEDs as outputs. With the variables representing the pin numbers, you can now refer to the pins by name instead.

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

The pinMode function configures the Arduino to use a specified pin as an output. This is essential for the LEDs to function correctly

Conclusion : Thus we conclude program when the user enters "b" the green light blinks, "g" the green light is illuminated "y" the yellow light is illuminated and "r" the red light is illuminated

ASSIGNMENT NO: 8

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

Aim:
Understand working principle of DHT11, LM35 temperature sensor

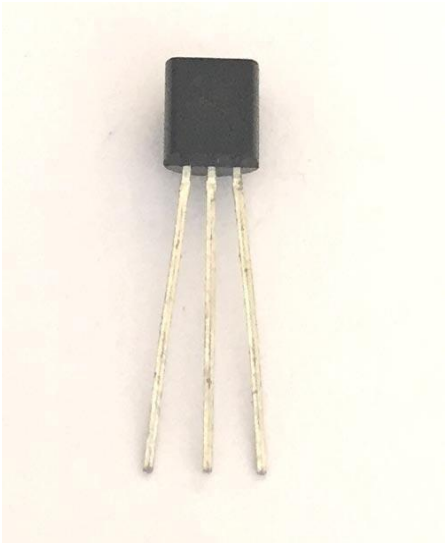
Objectives:.

1. Hardware platforms and operating systems commonly used in IoT systems.
2. Help the students in providing a good learning environment and also work with real time problems faced in day to day life

Hardware Requirement*:* Arduino, LED, LM35, DHT11 etc.

Software Requirement: Arduino IDE

Temperature Sensor:



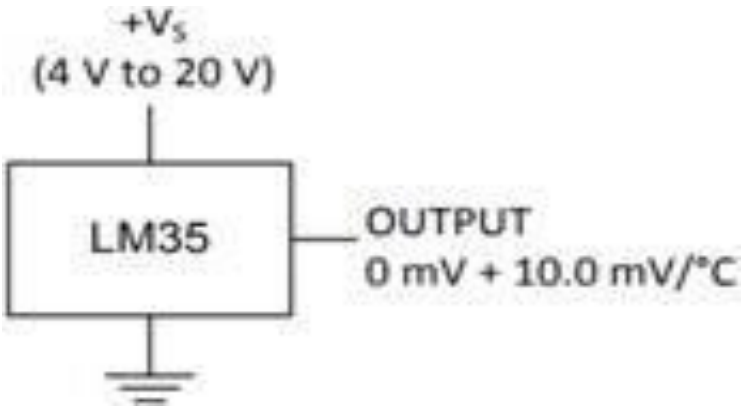
PinNumber	PinName	Description
1	Vcc	Input voltage is+5Vfor typical applications
2	Analog Out	There will be increase in 10mV for raise of every1°C.Can range from-1V(-55°C) to 6V(150°C)
3	Ground	Connected to ground of circuit

LM35 Sensor Features:

- Minimum and Maximum Input Voltage: 2V and 35V, respectively (Typically 5V).
- Temperature Measurement Range: -55°C to 150°C.
- Output Voltage is linearly proportional to temperature, with a rise of 10mV (0.01V) for every 1°C increase.
- Accuracy: ±0.5°C.
- Drain Current: Less than 60uA.
- Cost-effective temperature sensor.
- Compact size, suitable for remote applications.
- Available in TO-92, TO-220, TO-CAN, and SOIC packages.

LM35 Temperature Sensor Equivalents: LM34, DS18B20, DS1620, LM9402

How to use LM35 Temperature Sensor:
LM35 is a precise temperature sensor with an output voltage varying based on the temperature. It operates from -55°C to 150°C and interfaces easily with microcontrollers, requiring +5V power and a ground connection for temperature measurement in voltage



If the temperature is 0°C, the LM35's output voltage is 0V. For every degree Celsius increase, there's a rise of 0.01V (10mV). The temperature can be obtained by converting the voltage using the following formula

$$V_{OUT} = 10\ mv/°C \times T$$

where

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

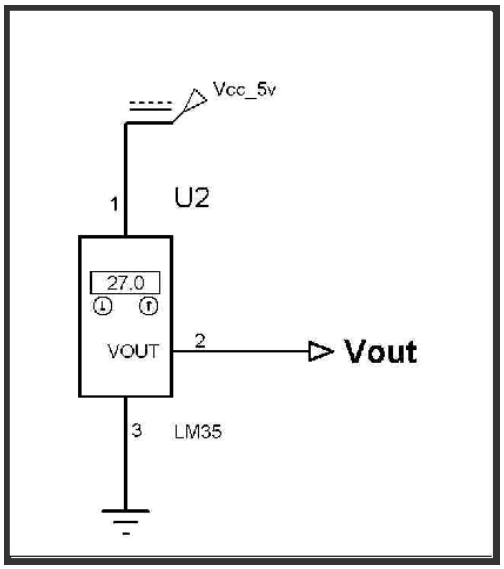
LM35 Temperature Sensor is used for:

- Measuring environmental temperature
- Providing thermal shutdown in circuits/components
- Monitoring battery temperature
- Measuring temperatures in HVAC applications

Working of LMT 35:

The main advantage of LM35 is its linearity, providing an output of 10mV per degree Celsius rise in temperature. For example, if the output is 220mV, the temperature is 22°C. Therefore, if the room temperature is 32°C, the LM35 output will be 320mV (0.32V)

Circuit Diagram **LMT 35**



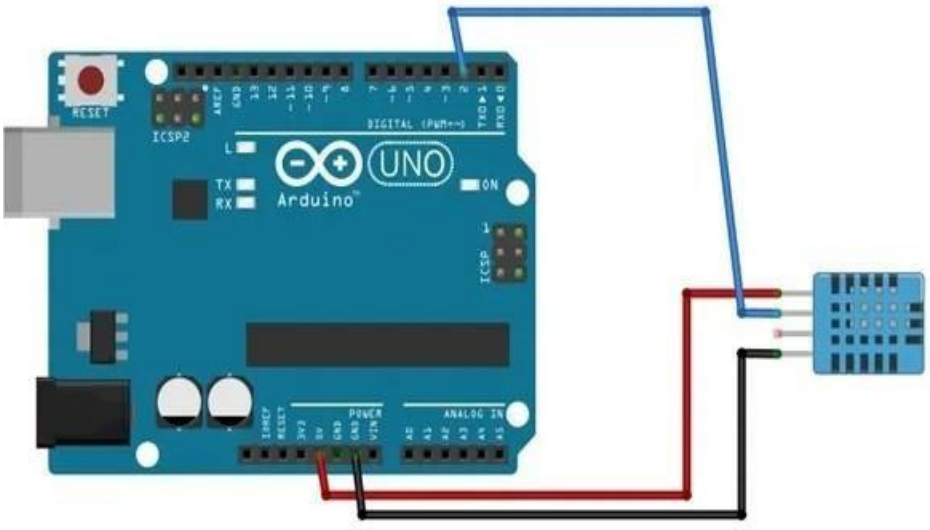
DHT11 Interfacing with Arduino and Weather Station

The DHT11 sensor is employed to measure both temperature and humidity. It incorporates a resistive humidity sensing component with a negative temperature coefficient (NTC) and an 8-bit MCU for rapid response. Despite being very affordable, it provides simultaneous readings of temperature and humidity.

Specifications of DHT11:

Humidity Range: 20% to 90% RH
Temperature Range: 0°C to 50°C
Signal Transmission Range: 20m
Cost-effectiveness
Fast response and durability
In weather stations, the DHT11 is commonly utilized for monitoring and recording both temperature and humidity levels.

The DHT11 sensor is employed to measure both temperature and humidity. It incorporates a resistive humidity sensing component with a negative temperature coefficient (NTC) and an 8-bit MCU for rapid response. Despite being very affordable, it provides simultaneous readings of temperature and humidity



To install the DHT11 library for Arduino, follow these steps:

1. To execute this code in the Arduino IDE, you need to first install the DHT library in your Arduino directory
2. Download the zip file from this location and move it into your Arduino library folder.
3. The path to the Arduino library folder on my computer path : Documents/Arduino/Libraries
4. Extract the downloaded file and move its contents to the specified folder
5. After copying the files, ensure that the Arduino library folder contains a new folder named "DHT" with the files "dht.h" and "dht.cpp." Next, copy the following code into the Arduino IDE and upload it

Temperature Scales

Thermometers help measure temperature, and there are three common scales: Celsius & Fahrenheit and Kelvin.

Celsius Scale & Fahrenheit Scale:

Celsius uses 0ºC for water freezing and 100ºC for boiling. Fahrenheit uses 32ºF for water freezing and 212ºF for boiling. One degree Celsius is 1.8 times larger than one degree Fahrenheit.

Kelvin Scale:

Kelvin is widely used in science, starting at 0 K (absolute zero). Water freezes at 273.15 K and boils at 373.15 K. Kelvin is an absolute scale with no negative temperatures, representing the lowest theoretically achievable temperature

Relationship between Different Temperature Scales

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

Conclusion*:* Thus we conclude program read the temperature sensor and send the values to the serial monitor on the computer

ASSIGNMENT NO: 9

Title:

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

Aim:

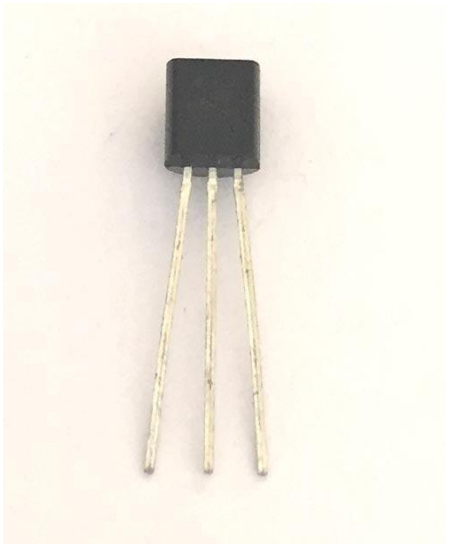
Understand working principle of DHT11, LM35 temperature sensor.

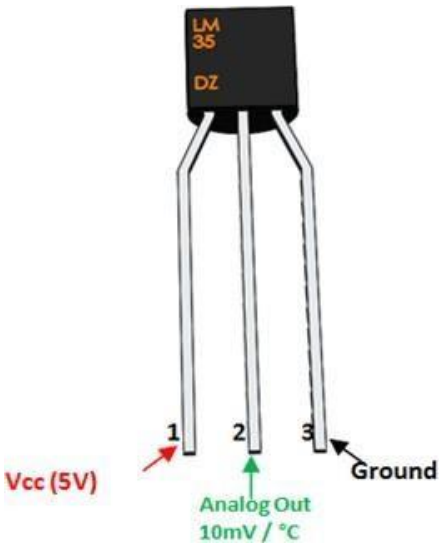
Objectives:.

- Temperature Conversion: Convert the temperature readings from Celsius to Fahrenheit for display.
- Track Maximum and Minimum Temperatures: Keep track of the maximum and minimum temperatures observed.
- Serial Monitor Display: Continuously display the current temperature in Fahrenheit, along with the maximum and minimum values, on the Serial Monitor.
- Delay: Introduce a short delay between readings to make the output more readable and avoid rapid updates

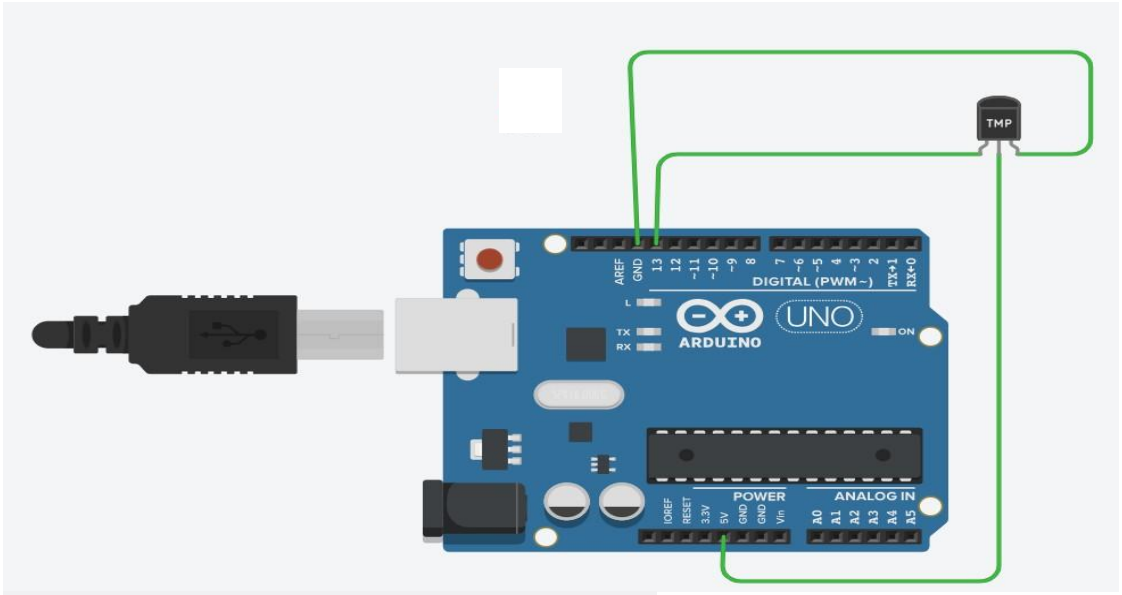
Hardware Requirement*:* Arduino, LED, LM35, DHT11 etc.

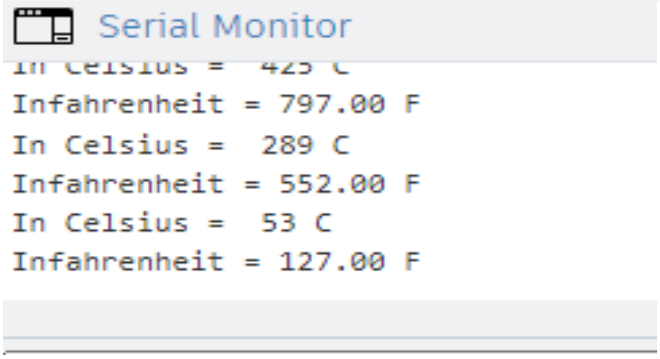
Software Requirement: Arduino IDE





This program provides a practical solution for monitoring temperature using an Arduino and a BME280 sensor. It not only displays real-time temperature in Fahrenheit but also keeps track of the highest and lowest temperatures encountered, enhancing its utility for various applications requiring temperature monitoring





To install the DHT11 library for Arduino, follow these steps:

1. To execute this code in the Arduino IDE, you need to first install the DHT library in your Arduino directory
2. Download the zip file from this location and move it into your Arduino library folder.
3. The path to the Arduino library folder on my computer path :

Documents/Arduino/Libraries

4. Extract the downloaded file and move its contents to the specified folder
5. After copying the files, ensure that the Arduino library folder contains a new folder named "DHT" with the files "dht.h" and "dht.cpp." Next, copy the following code into the Arduino IDE and upload it

Temperature Scales

Thermometers help measure temperature, and there are three common scales: Celsius & Fahrenheit and Kelvin.

Celsius Scale & Fahrenheit Scale:

Celsius uses 0ºC for water freezing and 100ºC for boiling. Fahrenheit uses 32ºF for water freezing and 212ºF for boiling. One degree Celsius is 1.8 times larger than one degree Fahrenheit.

Kelvin Scale:

Kelvin is widely used in science, starting at 0 K (absolute zero). Water freezes at 273.15 K and boils at 373.15 K. Kelvin is an absolute scale with no negative temperatures, representing the lowest theoretically achievable temperature

Relationship between Different Temperature Scales

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

Conclusion: Thus we conclude the temperature in Fahrenheit as well as the maximum and minimum temperatures it has seen.

Assignment No. 10

Title:

Write a program to show the temperature and shows a graph of the recent measurements.

Aim

Understanding working principle of DHT11 temperature sensor.

Objectives

1. Hardware platforms and operating systems commonly used in IoT systems.
2. Help the students in providing a good learning environment and also work with real time problems faced in day to day life.

Hardware Requirement: Arduino, LM35, DHT11, etc.

Software Requirement: Arduino IDE

Theory

Introduction:

Arduino is an open-source electronics platform based on easy-to-use hardware and software. It consists of a programmable circuit board (microcontroller) and a development environment used to write and upload computer code to the board. Arduino boards are widely used in various applications including hobbyist projects, prototyping, and educational purposes.

The DHT11 is a low-cost digital temperature and humidity sensor that is commonly used in Arduino projects and various other applications. An Arduino library for the DHT11 temperature and humidity sensor. This library provides a simple and easy-to-use interface to read temperature and humidity data from a DHT11 sensor.
It provides a simple and reliable way to measure temperature and humidity with reasonable accuracy.

The Temperature Sensor LM35 series are precision integrated-circuit temperature devices with an output voltage linearly proportional to the Centigrade temperature.

The LM35 device has an advantage over linear temperature sensors calibrated in Kelvin, as the user is not required to subtract a large constant voltage from the output to obtain convenient Centigrade scaling. The LM35 device does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full −55°C to 150°C temperature range.

Sensing Element: -The DHT11 sensor contains a sensing element made of a humidity-sensitive resistor and a thermistor for temperature measurement. The humidity-sensitive resistor changes its resistance based on the surrounding humidity level. The thermistor alters its resistance with changes in temperature.

Signal Processing Circuitry: - The sensor has built-in signal processing circuitry responsible for converting the analog signals from the sensing elements into digital signals.
This circuitry includes an analog-to-digital converter (ADC) and a microcontroller.

The DHT11 sensor provides digital output for both temperature and humidity readings. It communicates with the Arduino or other microcontrollers using a single-wire serial interface.
The data is transmitted in the form of a serial signal with a specific timing protocol.

Communication with the DHT11 sensor follows a specific timing protocol.
To request data from the sensor, the Arduino sends a start signal followed by a specific sequence of pulses. The sensor responds by sending data bits in a predefined format.

The data sent by the DHT11 sensor consists of 40 bits organized into 5 bytes.
The first two bytes represent the relative humidity, the next two bytes represent the temperature, and the last byte is a checksum for data integrity verification.
Each bit is transmitted as a pulse with varying duration, where a shorter pulse represents a logical 0 and a longer pulse represents a logical 1.

The DHT11 sensor is suitable for measuring temperature and humidity in environments with moderate accuracy requirements.
It operates within a specified range of temperature and humidity, typically between 0°C to 50°C for temperature and 20% to 80% for relative humidity.
The sensor may not be suitable for applications requiring high precision or extreme environmental conditions.

Arduino IDE Serial Plotter:
Arduino IDE (Integrated Development Environment) provides a built-in serial plotter tool that allows real-time plotting of data sent from the Arduino board via the serial port. The serial plotter can graphically display numeric data such as sensor readings, allowing users to visualize trends and patterns.

Output:

After Uploading the Ardunio code IDE, graph is shown on Arduino IDE Serial Plotter.

Conclusion:

Successfully studied the working principle of DHT11 temperature sensor.

Assignment No. 11

Title:

Write a program using piezo element and use it to play a tune after someone knocks.

Aim

To Detect a Knock.

Objectives

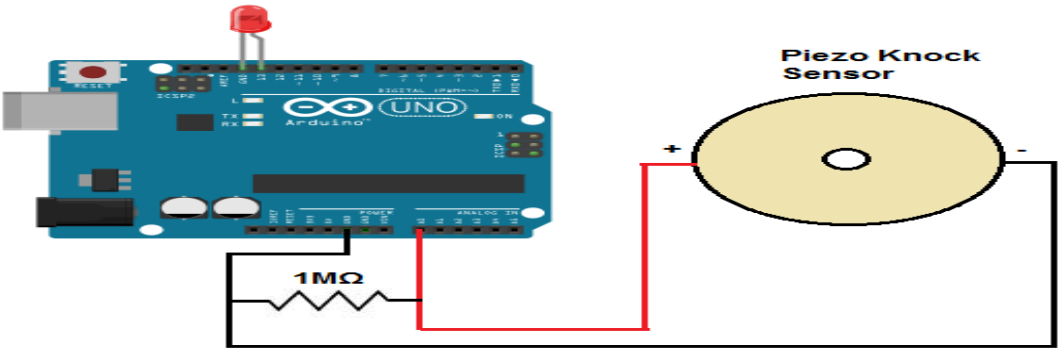
1. Hardware platforms and operating systems commonly used in IoT systems.
2. Help the students in providing a good learning environment and also work with real time problems faced in day to day life.

Hardware Requirement: Arduino, Piezo electric dis, solid surface & 1 Megohm resistor.

Software Requirement: Arduino IDE

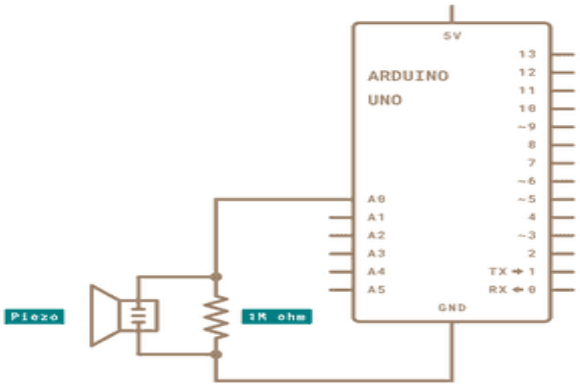
Theory

A piezo is an electronic device that generates a voltage when it's physically deformed by a vibration, sound wave, or mechanical strain. Similarly, when you put a voltage across a piezo, it vibrates and creates a tone. Piezo can be used both to play tones and to detect tones.The sketch reads the piezo output using theanalogRead() command, encoding the voltage range from 0 to 5 volts to a numerical range from 0 to 1023 in a process referred to as analog-to-digital conversion, or ADC.If the sensors output is stronger than a certain threshold, your board will send the string "Knock!" to the computer over the serial port.Piezos are polarized, meaning that voltage passes through them (or out of them) in a specific direction. Connect the black wire (the lower voltage) to ground and the red wire (the higher voltage) to analog pin 0. Additionally, connect a 1-megohm resistor in parallel to the Piezo element to limit the voltage and current produced by the piezo and to protect the analog input.It is possible to acquire piezo elements without a plastic housing. These will look like a metallic disc, and are easier to use as input sensors. Piezo sensors work best when firmly pressed against, taped, or glued their sensing surface.



Schematic:-

The piezo is attached to analog pin 0 with a 1 Megohm resistor in between the two legs. The placement of the resistor is used to save the piezo from damage from extra current. Without it, the analog pin might not be capable of reading the piezo's signal.



Conclusion:

Successfully studied piezo element and used it to play a tune after someone knocks.

Assignment No.12

Title:

Understanding the connectivity of Raspberry-Pi /Beagle board circuit / Arduino with IR sensor. Write an application to detect obstacle and notify user using LEDs.

Aim

To detect obstacle and notify user using LEDs.

Objectives

1. Hardware platforms and operating systems commonly used in IoT systems.
2. Help the students in providing a good learning environment and also work with real time problems faced in day to day life.

Hardware Requirement: Arduino, IR Sensor etc.

Software Requirement: Arduino IDE

Theory

Infrared or IR proximity sensors emit infrared light and once this light hits an object, it is reflected back to the sensor. Depending on the strength of the reflected light, the sensor will know how far or close an object is. The stronger the reflected signal, the closer the object. The weaker the signal, the farther the object is.

The function of a proximity detector is to be able to tell a moving machine when it is near an object without it having to actually bump against the obstacle. In proximity detection, what we are more interested in are devices that allow machines to detect obstacles without collision. There are several types of proximity sensors available. One of which is the infrared or IR proximity sensor. An example of an infrared proximity sensor is the parking detector. The parking detector is a simple device that can be seen in a modern parking lot. A green light may turn on when there is no car occupying the parking space, while a red light turns on when the space is occupied. The IR sensor in effect detects the presence of a car in a parking space.

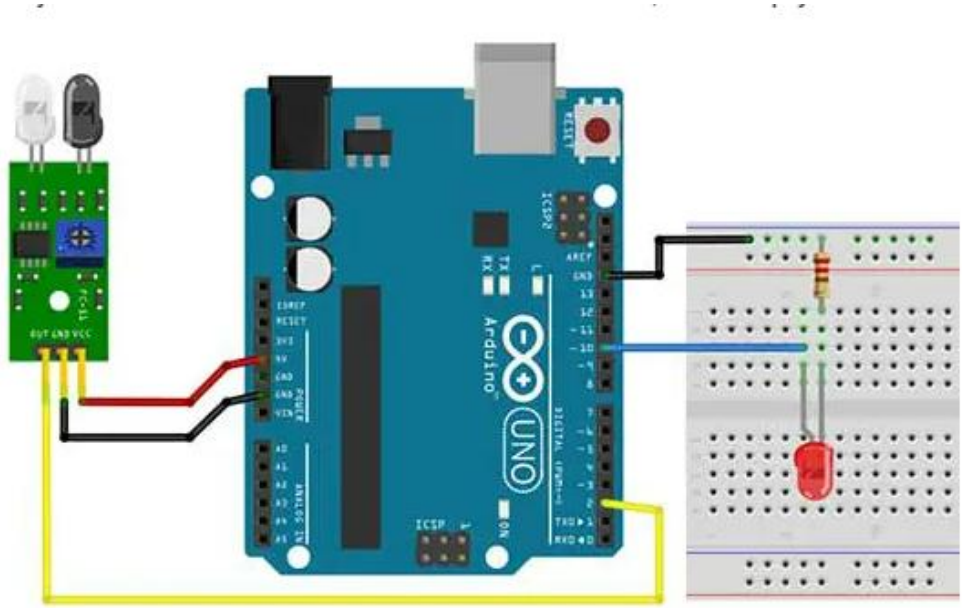
To demonstrate the capability of an IR proximity sensor, we'll be using the HW-201 IR obstacle sensor module.

The detection range of this module is from 2-30cm (depending on the surface). It only has a digital output, which means, it only gives a HIGH or LOW signal, or 1 or 0. HIGH if it detects an object, LOW if no object is detected. This module requires 5 volts DC.An IR sensor is made up of two basic parts: an infrared LED and an infrared photodiode. The IR LED emits the infrared light while the IR photodiode converts the reflected LED light to an electrical current that allows the sensor to interpret the strength of the reflected IR.

There are three header pins for the HW-201 obstacle sensor. The VCC goes to the 5V on the Arduinoboard. GND, of course, should be connected to the GND on the Arduino, and the OUT pin

goes to any digital pin, except 0 and 1.There is also an adjuster in the module to adjust the distance of the object before the sensor is triggered. You can turn the adjuster clockwise or counterclockwise using a screwdriver to adjust the sensitivity level.

To test our IR obstacle sensor, we will try to turn on an LED when an object is detected and turn it off when no object is detected. For the breadboard circuit, we simply add an LED circuit.



Conclusion:

Successfully studied the the connectivity of Raspberry-Pi /Beagle board circuit / Arduino with IR sensor.