GURU GOBIND SINGH FOUNDATION'S

GURU GOBIND SINGH COLLEGE OF ENGINEERING AND RESEARCH CENTRE, NASHIK

Certificate

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & DATA SCIENCE

This is to certify that Mr./Miss _____

of _____ *Year* ***AI &DS*** *Div. :*_____*Roll No.:*_____*University Examination Seat No. :*_____*has completed all the practical work/Term work in Subject* ---------- *satisfactorily in* ***Department of Artificial Intelligence & Data Science*** *as prescribed by the* ***Savitribai Phule Pune University*** *in the academic year 20 - 20*

Practical In-charge **Head of Department** **Principal**

PRACTICAL NO. 1 (Group A)

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

- **Outcome:** At end of this PRACTICAL, student will be able to study of different microcontrollers like Raspberry-Pi/ Beagle board/ Arduino

- **Hardware Requirement:** Arduino

- **Software Requirement:** Arduino IDE

- **Theory:**

Internet of Things: - IoT is short for Internet of Things. The Internet of Things refers to the evergrowing 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 network.

Examples of IoT: -
1) Apple Watch and Home Kit.
2) Smart Refrigerator.
3) Smart cars.
4) Google Glass.
5) Smart thermostats.

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

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

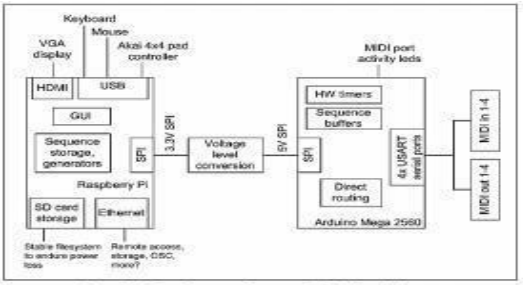


Fig. A.2: -**Raspberry-Pi Architecture**

Here are the **various components on 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.
- **LED's** -- 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 Model B.

B) Beagle board: - Beagle Board is a low-power open-source single-board computer produced by Texas Instruments in association with Digi-Key and Newark element14. 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.

Fig.B.1: -Beagle Board Black

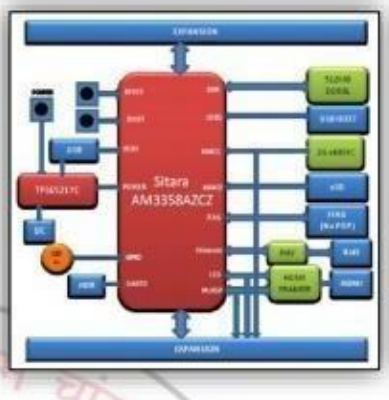


Fig.B.1: - Beagle Board Black architecture

Here are the **various components on Beagle board**:

Processor: AM335x 1GHz ARM® Cortex-A8

- 512MB DDR3 RAM
- 4GB 8-bit eMMC on-board flash storage
- 3D graphics accelerator
- NEON floating-point accelerator
- 2x PRU 32-bit microcontrollers

Connectivity

- USB client for power & communications
- USB host
- Ethernet
- HDMI
- 2x 46 pin headers

Software Compatibility

- Debian
- Android
- Ubuntu
- Cloud9 IDE on Node.js w/ BoneScript library 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.

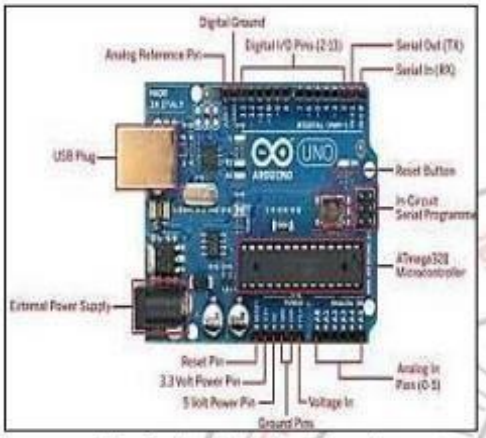


Fig.C.1: - Arduino Board

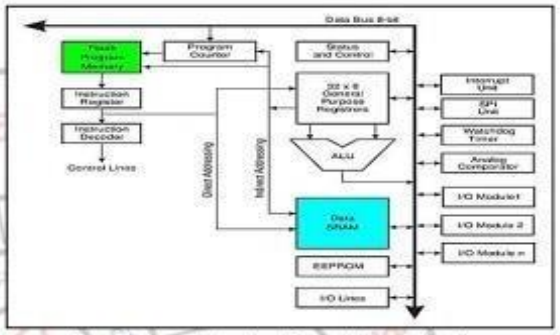


Fig.C.1: - Arduino Board Architecture.

Here are the **various components on Arduino board:**

- **Microcontrollers**

- ATmega328P (used on most recent boards)
- ATmega168 (used on most Arduino Diecimila and early Duemilanove)
  ATmega8 (used on some older board)

- **Digital Pins**

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

- **Analog Pins**

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

- **Power Pins**

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

Other Pins

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

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

Questions:

1) What is Microcontroller?

2) What is the stable version of Arduino software?

3) Why we should use Raspberry-Pi/ Beagle board/ Arduino?

4) What are the advantages of Raspberry-Pi/ Beagle board/ Arduino?

PRACTICAL NO. 2 (Group A)

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

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

- **Hardware Requirement:** Raspberry-Pi
- **Software Requirement:** Raspbian OS
- **Theory:**

1) **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.

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

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

➢ **How to install Debian on BeagleBone Black:**

- **Step 1**: Download Debian img.xz file.
- **Step 2**: Unzip the file.
- **Step 3:** Insert your MicroSD (uSD) card into the proper slot. Most uSD cards come with a full-sized SD card that is really just an adapter. If this is what you have then insert the uSD into the adapter, then into your card reader.
- **Step 4**: Now open Win32 Disk imager, click the blue folder icon, navigate to the debian img location, and double click the file. Now click Write and let the process complete. Depending on your processor and available RAM it should be done in around 5 minutes.
- **Step 5:** Alright, once that's done, you'll get a notification pop-up. Now we're ready to get going. Remove the SD adapter from the card slot, remove the uSD card from the adapter. With the USB cable disconnected insert the uSD into the BBB.
- **Step 6**: Now, this next part is pretty straight forward. Plug the USB cable in and wait some more. If everything is going right you will notice that the four (4) leds just above the USB cable are doing the KIT impression. This could take up to 45 minutes; I just did it again in around 5 minutes. Your mileage will vary. Go back and surf reddit some more.

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

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

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

3) **Arduino**: -

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

Conclusion: - Thus, we have implemented of how to install operating systems for platforms such as Raspberry-Pi/Beagle board/Arduino.

FAQ:

1. Which operating system sre used in Raspberry-Pi/Beagle board/Arduino? Explain in brief.

PRACTICAL NO. 3 (Group A)

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

Outcome: To study different GATES (AND, OR, XOR), Sensors

- **Hardware Requirement:** Logical Gates, Sensors etc.
- **Software Requirement:** Raspbian OS
- **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).

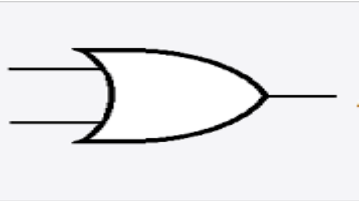
- **Basic logic gates**
  There are seven basic logic gates: **AND, OR, XOR, NOT, NAND, NOR, XNOR.**

- 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.

AND gate

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

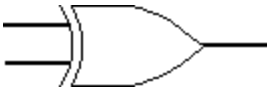
- The ***OR gate*** gets its name from the fact that it behaves after the fashion of the logical inclusive "or." The output is "true" if either or both of the inputs are "true." If both inputs are "false," then the output is "false." In other words, for the output to be 1, at least input one OR two must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



OR gate

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

- 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.



XOR gate

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

- **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.

Inverter / NOT gate

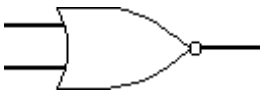
Input	Output
1	0
0	1

- 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."

NAND gate

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

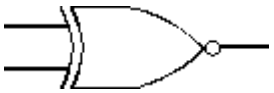
- The ***NOR gate*** is a combination **OR gate followed by an inverter**. Its output is "true" if both inputs are "false." Otherwise, the output is "false."



NOR gate

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

- The ***XNOR (exclusive-NOR) gate*** is a combination **XOR gate followed by an inverter**. Its output is "true" if the inputs are the same, and "false" if the inputs are different.



XNOR gate

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

- 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 gates**

- 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.

- **Types of Sensors -**

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

- Position Sensors
- Pressure Sensors

- Temperature Sensors
- Force Sensors
- Vibration Sensors
- Piezo Sensors
- Fluid Property Sensors
- Humidity Sensors
- Strain gauges
- Photo Optic Sensors
- 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;

- Hall effect
- Capacitive
- Eddy Current
- 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

- 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.

➢ Linear position Sensors

- VLP
- VXP
- ELPM
- VLPSC

➢ Rotary Position Sensors

- Euro-X Hall Effect
- Euro-XP Puck – 2 part puck and magnet design
- Euro – XPD – D shaft
- 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.

➢ 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.

➢ 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
- ETP – Eurosensor Temperature probes – these are fully customisable to your requirements.

Conclusion: - Thus, we have implemented different GATES (AND, OR, XOR), Sensors and basic binary operations.

Questions:

1) What are different types of Sensors?

2) Explain different basic binary operations with example?

3) Explain different types of Logical Gates?

PRACTICAL NO. 4 (Group A)

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

Outcome: Connectivity and configuration of Raspberry-Pi /Beagle board/Arduino circuit with basic peripherals like LEDS

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

- **Theory:**

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

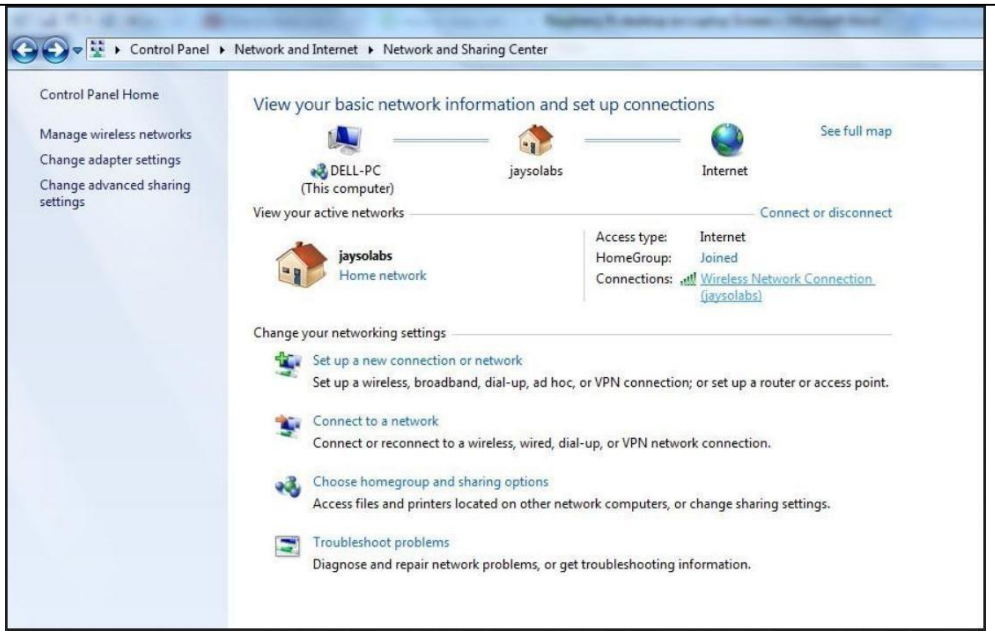
- **How to Connect a Raspberry-Pi to the Laptop or PC Display**

  Required devices:

  - Raspberry Pi
  - Ethernet Cable
  - Laptop/ PC
  - SD Card with Raspbian
  - Micro USB Cable

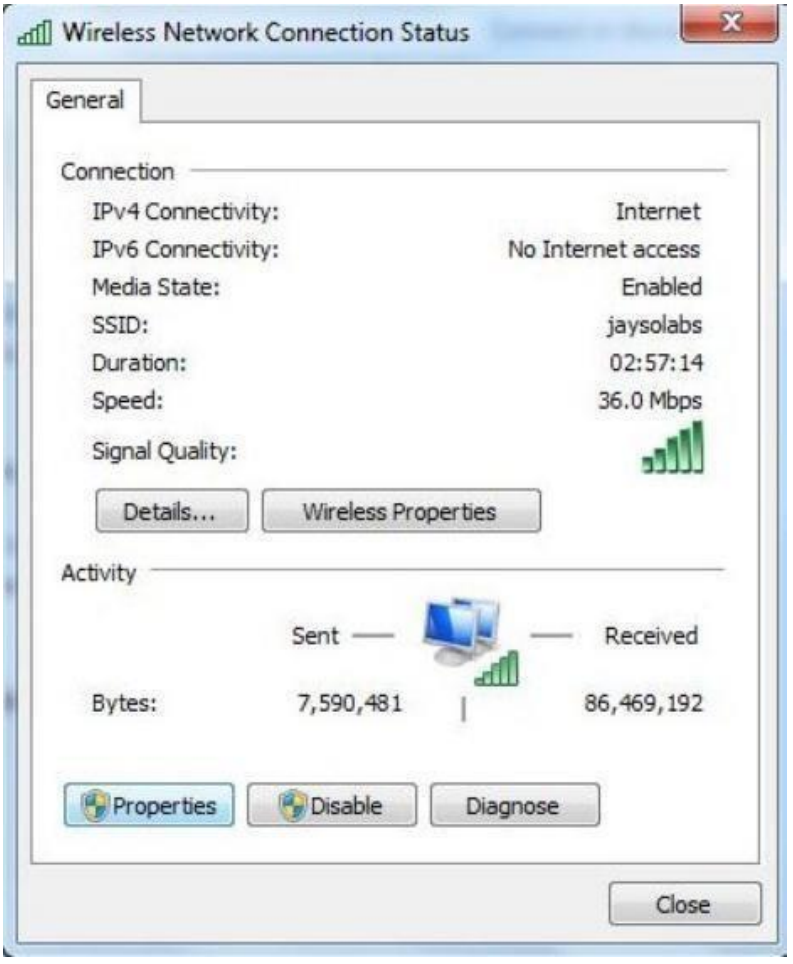
- **How does it Work?**
- To connect a Raspberry Pi to a laptop or PC display, you can simply use an Ethernet cable.
- The Raspberry Pi's desktop GUI can be viewed through the laptop or PC display using a 100mbps Ethernet connection between the two.
- We used VNC server software to connect the Raspberry-Pi 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

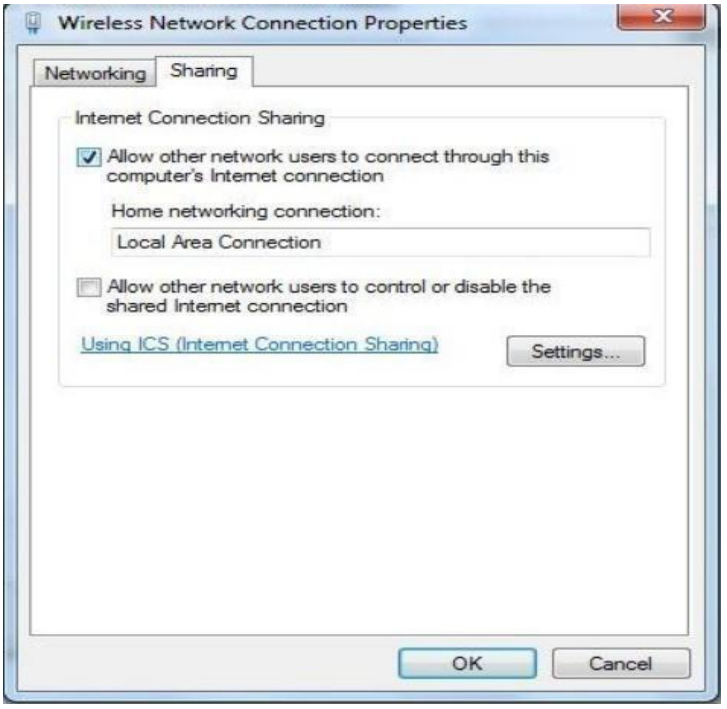


This step explains how you can share your laptop or PC with the Raspberry Pi via Ethernet cable.

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

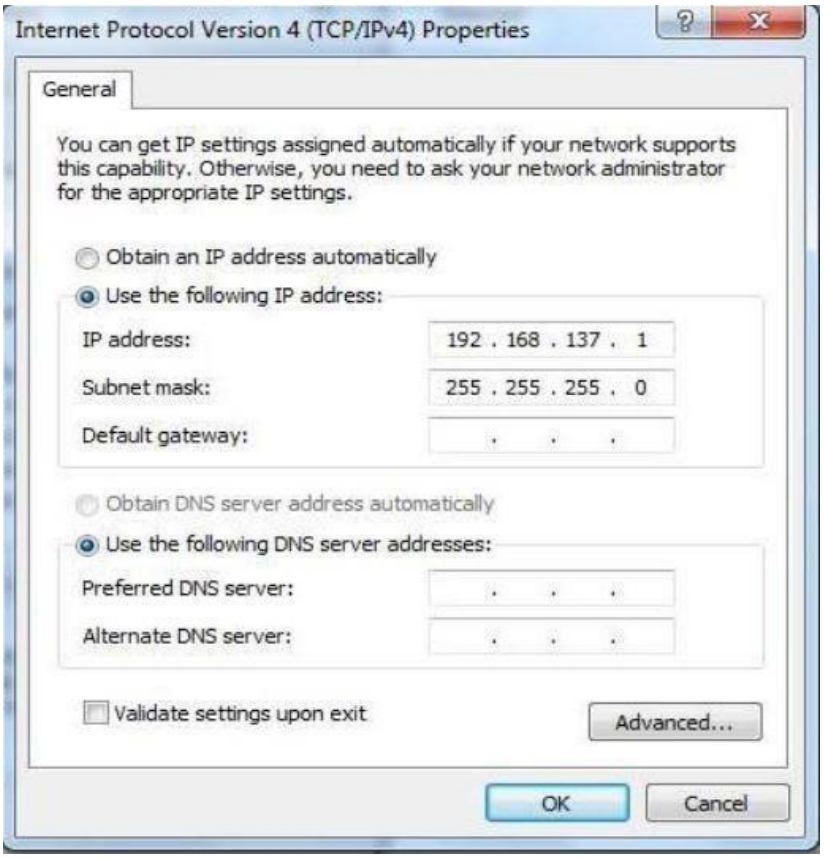


Go to “Sharing” tab and click on “Allow other network users to connect”.



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 “Advance IP Scanner”. It is free software.

Setting up the VNC Server to Connect Your Raspberry Pi to the Laptop or PC Display

- First install VNC server and Putty on your laptop/ PC
- Open Putty Software, and enter login ID: pi and Password: raspberry.

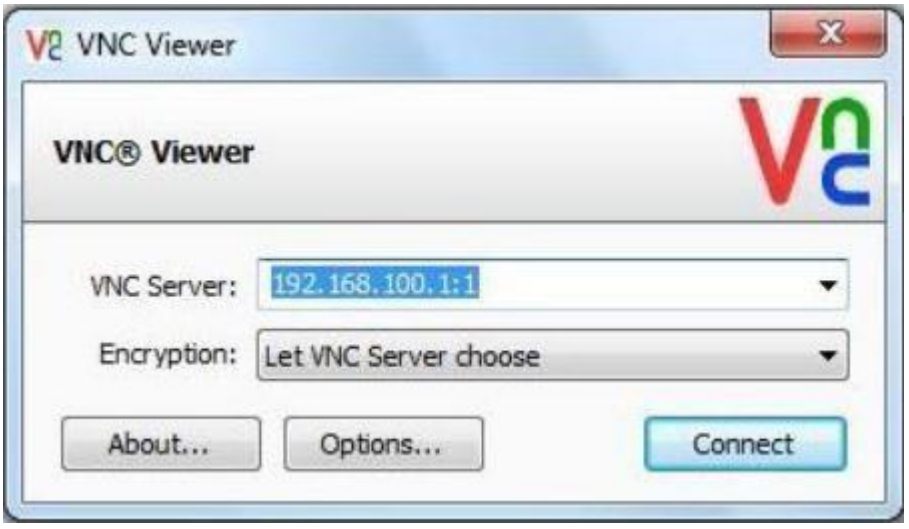
After that, enter commands into Putty i.e,

- $ 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)

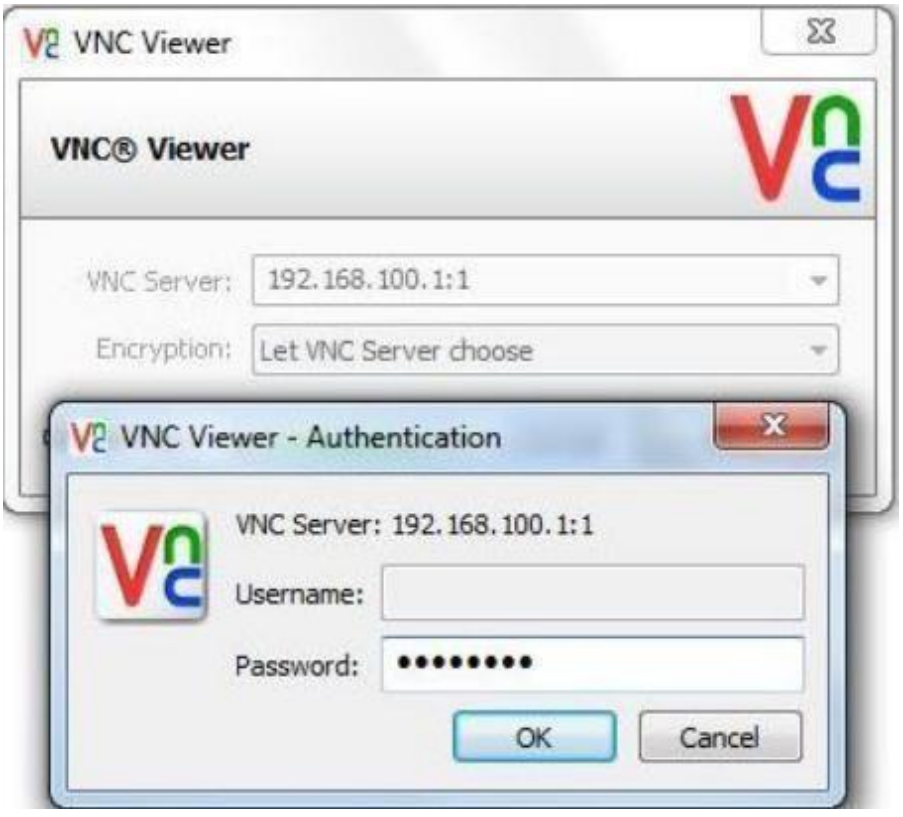
- Download VNC client and install it.
- When you first run VNC viewer, you will see following screen



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

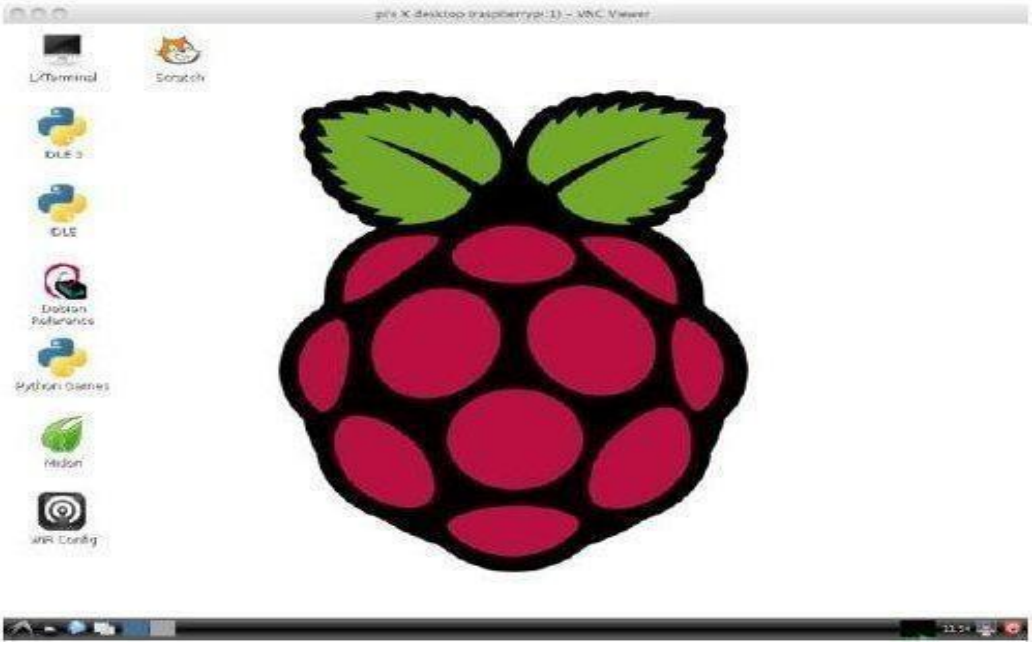
You will get a warning message, press 'Continue':

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, and monitor 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 with the 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 to analogue 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 provide sufficient 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.

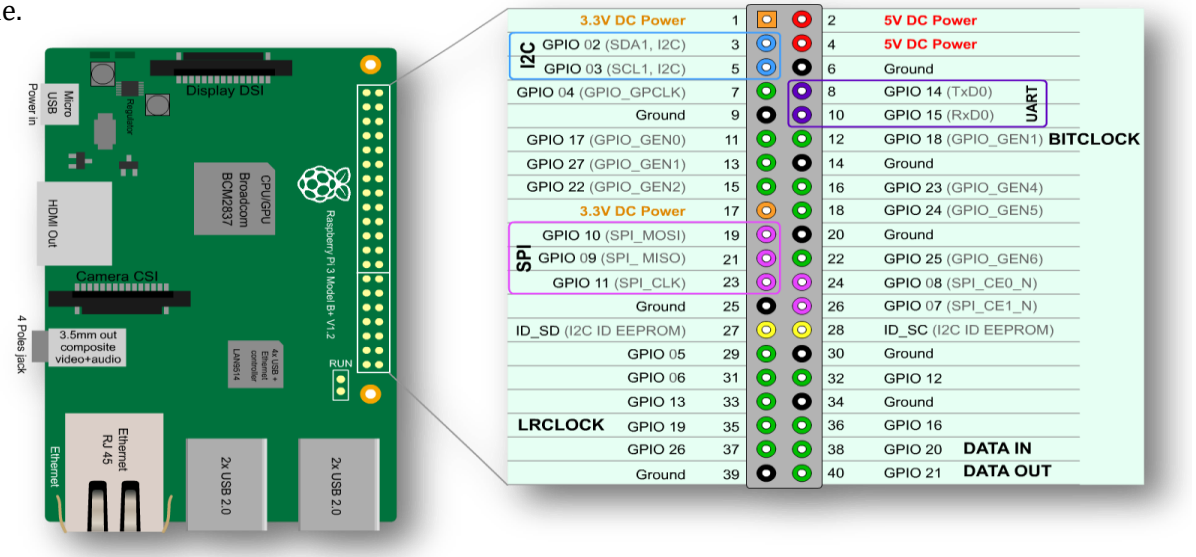
- **Understanding GPIO pins on Raspberry Pi board and its use**
- **Aim/Objectives:**
  1. To understand the GPIO pins of Raspberry-Pi 3
  2. To program the GPIO pins of Raspberry-Pi 3 using Python

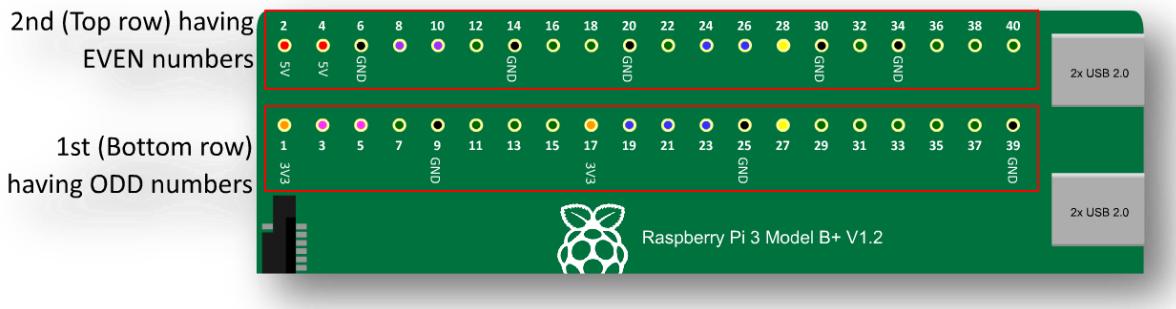
Introduction:

➢ 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 whose behavior—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 above
- Out 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 your program.
- 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 use with 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 goes to 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

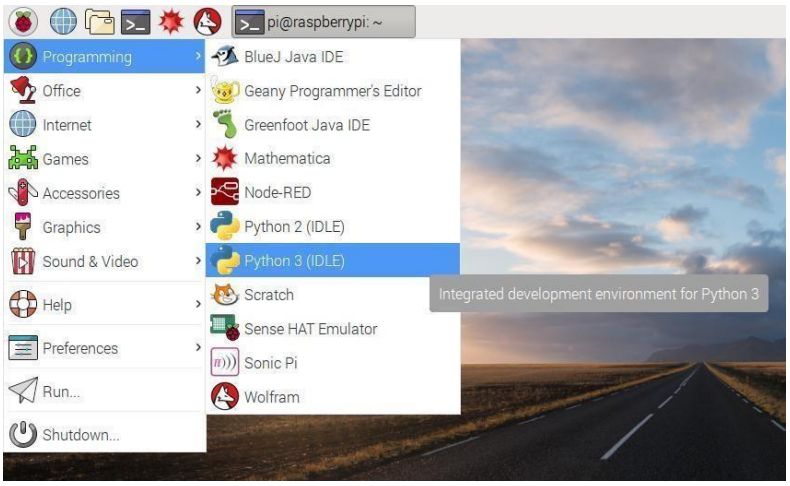
The below table summarizes the **pinout of Raspberry-Pi in both number systems**.

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 use the following

Open Python 3 from main menu:



- Or open terminal window and type the command **sudo idle 3.5** and press enter
- Install 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 interesting electronic 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.
- To include 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 on and 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 installed in Raspbian.

- If you are using any other operating system, the package can be installed by using the following command:
  $ 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 use physical 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 command
  GPIO.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 will persist, 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 the python program, we need to give the super user permission .

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

Aim/Objectives:

- To understand the concept of Led bar
- To understand the common anode & common cathode configuration.
- To interface LED bar with Raspberry Pi.
- Generate various patterns on LED bar.

Software: Raspbian OS , IDLE editor

Hardware Modules: Raspberry Pi Board, LED bar, Monitor

Introduction to "LED"

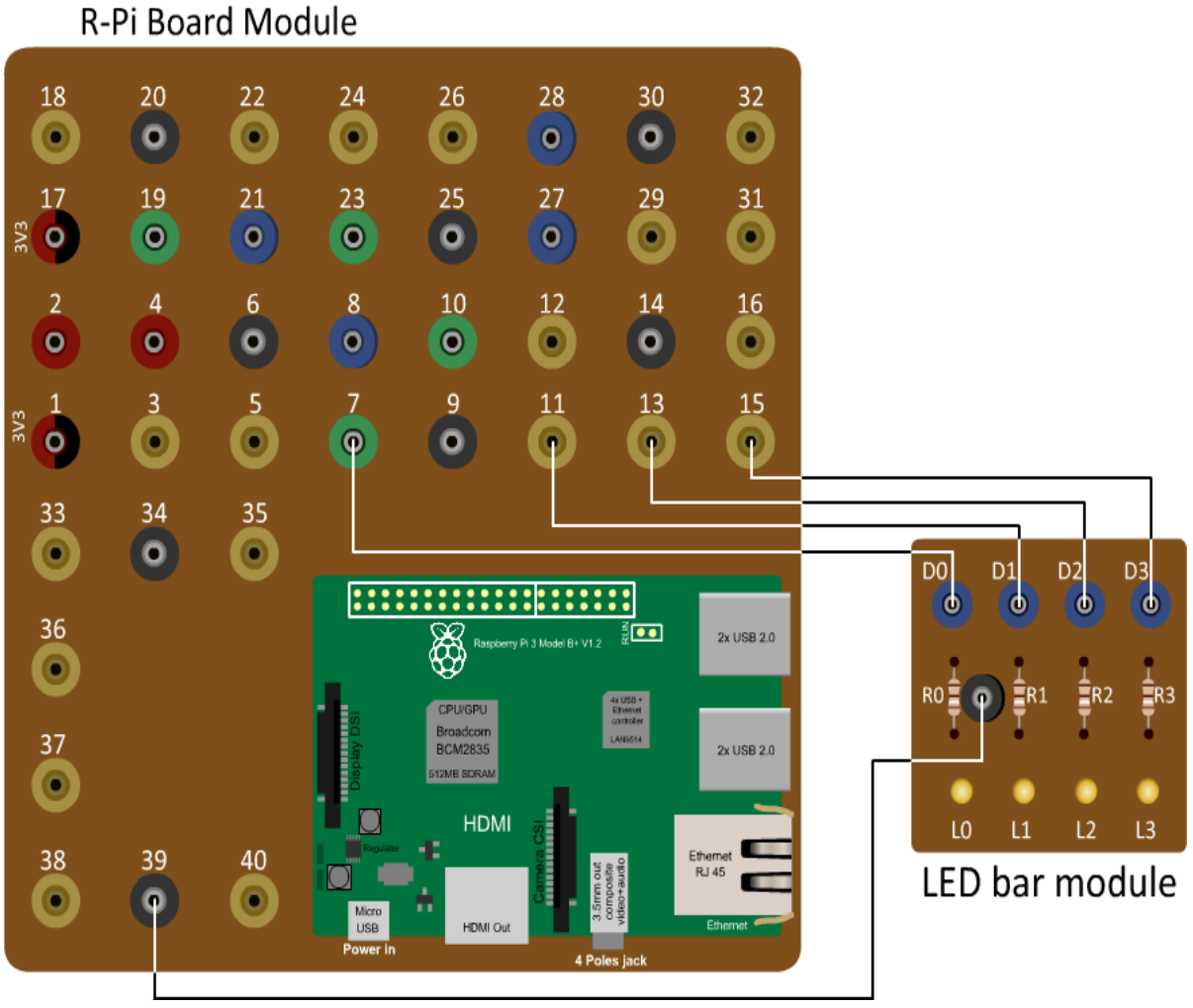
LED is a Light Emitting Diode. Light emitting diode is a two lead semiconductor light source. It is a p-n junction diode, which emits light when it is activated. When suitable voltage is applied to the leads, electrons are able to recombine with electron holes within the device, and the color of light (corresponding to the energy of photon) is determined by the energy band gap of 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 a sensor or an actuator.
- Ensure that signal voltage coming to the Raspberry-Pi from any sensor or actuator does not exceed 3.3V.

- If signal/data coming to Raspberry-Pi is greater than 3.3V then use voltage level shifter module to decrease the incoming voltage.
- The Raspberry-Pi is a costly device, hence you should show the circuit connections to your instructor before starting your PRACTICAL.

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.

Write the program as per algorithm given below.

- Save program.
- Run code using Run module.
- Algorithm:
- Import GPIO and Time library
- Set mode i.e. GPIO.BOARD
- Set GPIO 8 pins as a Output pin
- Print message “ON”
- After 1 second time delay, Make all the led’s ON one by one
- Print message “OFF”
- After 1 second time delay, Make all the led’s OFF one by one

---

Conclusion: - Thus, we have implemented Understanding GPIO and its use in the program and Connectivity and configuration of Raspberry-Pi /Beagle board/Arduino circuit with basic peripherals

---

Questions:

1) Explain GPIO pins connectivity in details.
2) Explain programming of Raspberry-Pi using Python.

PRACTICAL NO. 5 (Group B)

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

Outcome: Connectivity and configuration of Raspberry-Pi /Beagle board/Arduino circuit with basic peripherals like LEDS

- **Hardware Requirement:** Arduino, LED etc.
- **Software Requirement**: Arduino IDE

Procedure:

1. Write the program as per the algorithm given below and save the program.
2. Connect the Arduino board to PC/Laptop using USB cable.
3. Select the board and the port in the Arduino IDE.
4. Then compile the program.
5. Then Upload the program and check the output.

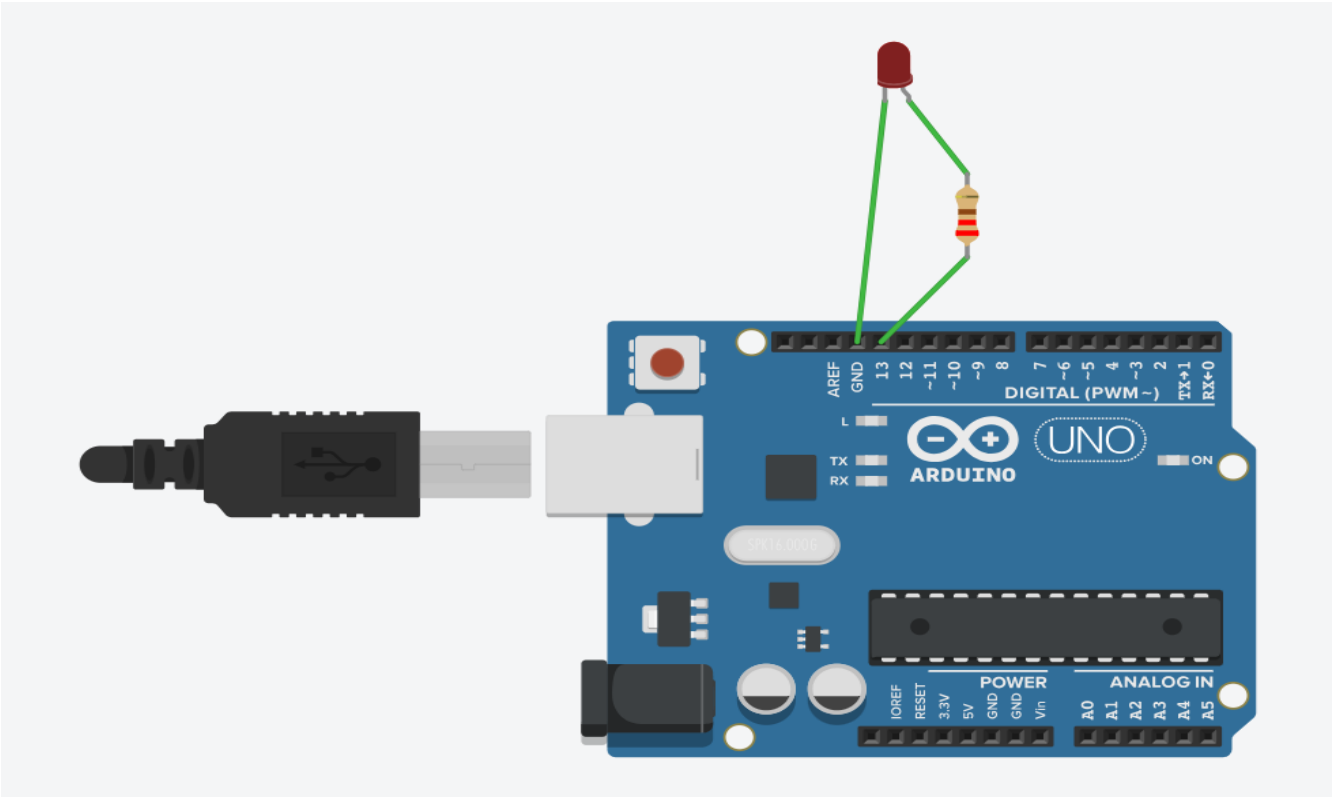
Algorithm:

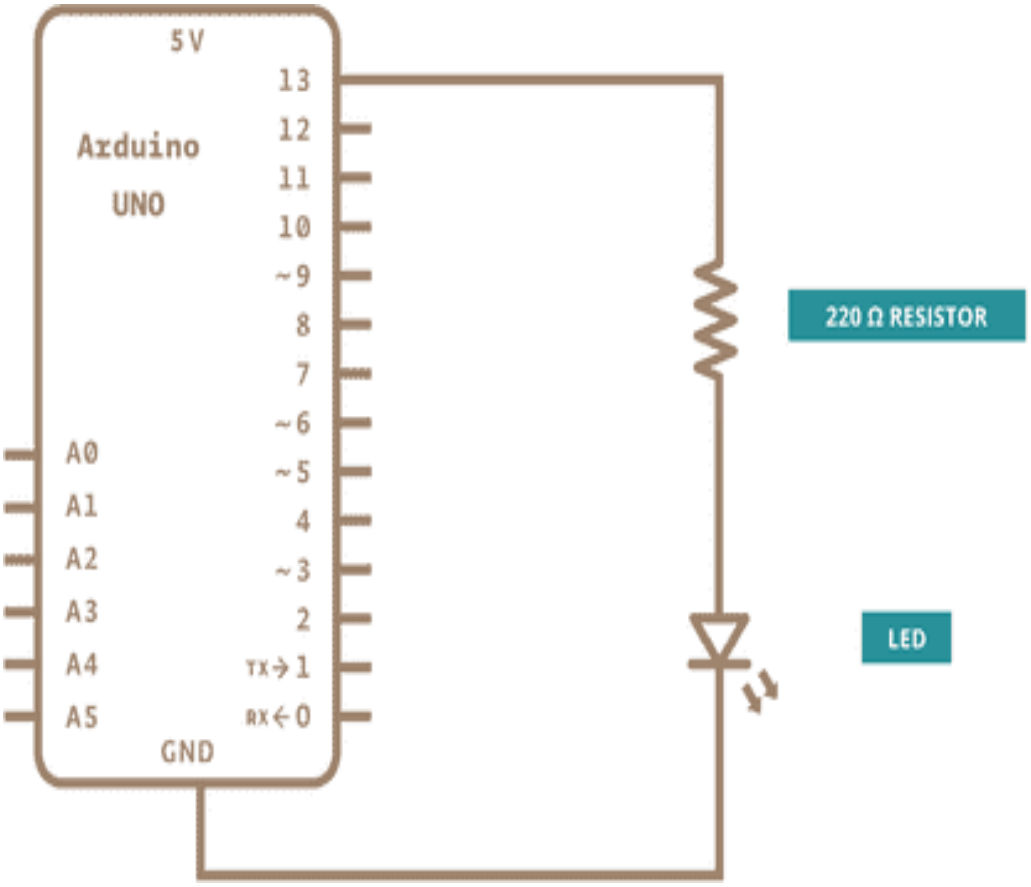
1. Start IDE
2. Configure the pin number '13' as 'OUTPUT' pin
3. Make the 'OUTPUT' as 'HIGH'
4. Give delay of one second
5. Make the 'OUTPUT' as 'LOW'
6. Give delay of one second

Observe the LED connected to pin number '13', it starts blinking as soon as program is uploaded.

Program:

```
void setup()
{
 pinMode(13, OUTPUT);
}
void loop()
{
 digitalWrite(13, HIGH);
delay(1000);
digitalWrite(13, LOW);
delay(1000);
}
```


AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
L
TX
RX
ARDUINO
UNO
ON
POWER
ANALOG IN
IOREF
RESET
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5


5V
Arduino
UNO
13
12
11
10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
A0
A1
A2
A3
A4
A5
GND
220 Ω RESISTOR
LED

Conclusion: - Thus, we have implemented Arduino program to control LED

PRACTICAL NO. 6 (Group B)

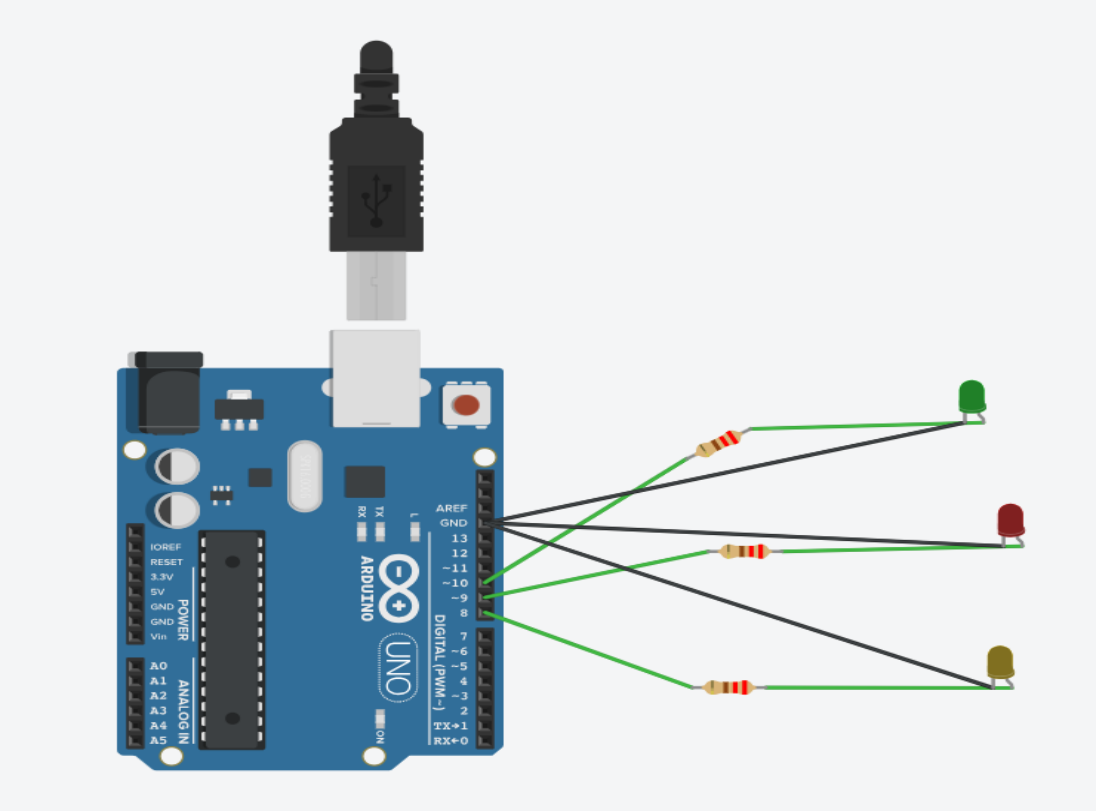
Aim: Create a program that illuminates the green LED if the counter is less than 100, illuminates the yellow LED if the counter is between 101 and 200 and illuminates the red LED if the counter is greater than 200

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

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

  Introduction to "LED"

1. LED is a Light Emitting Diode.
2. Light emitting diode is a two lead semiconductor light source.
3. It is a p-n junction diode, which emits light when it is activated.
4. When 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 semiconductor.
5. It has two terminals named as '**anode (+ve)**' and '**cathode (- ve)**'.
6. Battery is connected to these two terminals.
7. When LED is forward biased, it emits light.



PROGRAM

```
int red = 10;
int yellow = 9;
int green = 8;
int counter=0;
void setup() {
pinMode(red, OUTPUT);
 pinMode(yellow, OUTPUT);
 pinMode(green, OUTPUT);
}
void loop()
{
 changeLights();
 delay(100);

}
void changeLights(){
 if(counter<100)
 {
 digitalWrite(yellow, HIGH);
 digitalWrite(red, LOW);
 digitalWrite(green, LOW);
 delay(50);
 }
```

```
 else if(counter>100 && counter<200)
 {
 digitalWrite(red, HIGH);
 digitalWrite(yellow, LOW);
 digitalWrite(green, LOW);
 delay(50);
 }
 else if (counter>200 && counter<500)
 {
 digitalWrite(green, HIGH);
 digitalWrite(red, LOW);
 digitalWrite(yellow, LOW);
 delay(50);
 counter=0;
 }
 counter=counter+50;
}
```

Conclusion: - Thus, we have implemented Arduino program to control LEDs by counter value.

PRACTICAL NO. 7 (Group B)

Aim: Create a program so that when the user enters "b" the blue light blinks, "g" the green light is illuminated, and "r" the red light is illuminated

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

- **Hardware Requirement:** Arduino, LED, resistors, wires etc
- **Software Requirement**: Arduino IDE
- **Theory:**

1. An **LED**, or Light Emitting Diode, is a semiconductor device that emits light when an electric current flows through it, offering advantages like energy efficiency, long lifespan, and environmental friendliness.
2. **Semiconductor Device:**
   LEDs are made from semiconductor materials, which are neither good conductors nor insulators of electricity.
3. **Light Emission:**
   When electricity passes through an LED, electrons and holes (positively charged "empty" electron spots) recombine, releasing energy in the form of light.
4. **Efficiency**:
   LEDs are highly energy-efficient, meaning they convert a large percentage of electrical energy into light, unlike older technologies like incandescent bulbs.
5. **Long Lifespan:**
   LEDs have a much longer lifespan than traditional light sources, requiring less frequent replacement.
6. **Environmental Friendly**:
   LEDs are more environmentally friendly as they do not contain toxic materials like mercury.

How it works:

1. **Forward Bias:**
   LEDs only light up when electricity flows through them in the correct direction, known as forward bias.
2. **Anode and Cathode:**
   LEDs have two terminals: an anode (positive) and a cathode (negative). The current must flow from the anode to the cathode for the LED to light up.
3. **P-N Junction**:
   The core of an LED is a P-N junction, where a positively charged "P-type" semiconductor material meets a negatively charged "N-type" semiconductor material.
4. **Recombination:**
   When electrons from the N-type material and holes from the P-type material recombine at the junction, they release energy in the form of light.

Advantages of LEDs:

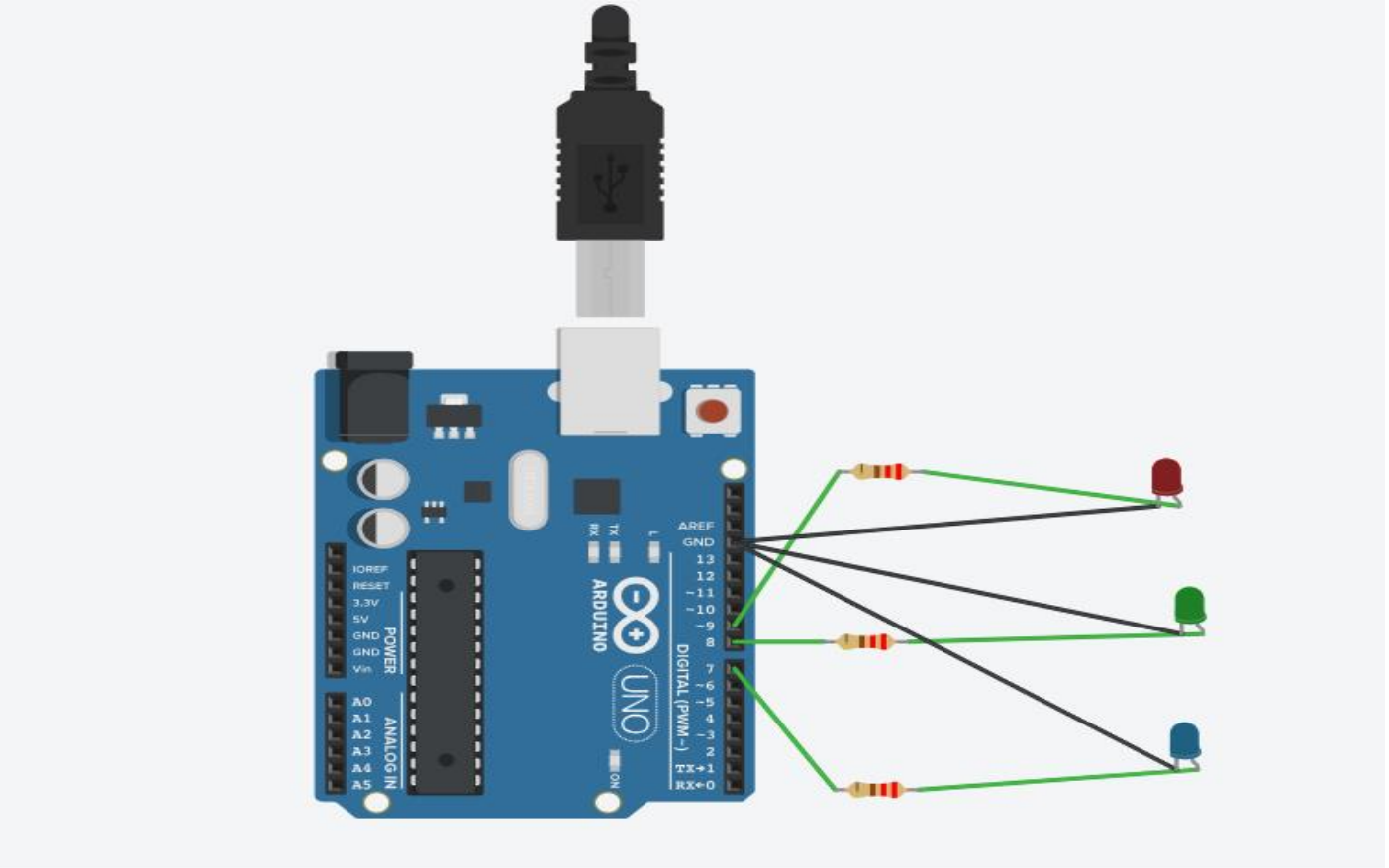
1. **Energy Efficiency:** LEDs consume significantly less energy than traditional light sources.

2. **Long Lifespan**: LEDs can last for many years, reducing maintenance costs.
3. **Durable**: LEDs are resistant to vibration and shock, making them suitable for various applications.
4. **Design Flexibility**: LEDs can be made in various shapes, sizes, and colors, allowing for creative lighting solutions.
5. **Low Heat Emission**: LEDs generate very little heat compared to traditional light sources, making them safer and more energy-efficient.
6. **Environmental Friendliness**: LEDs do not contain harmful materials like mercury, making them a more sustainable choice.

Applications of LEDs:

1. **Lighting:**
   LEDs are used in a wide range of lighting applications, from streetlights and home lighting to vehicle headlights and backlighting for displays.
2. **Displays:**
   LEDs are used in digital displays, such as televisions, smartphones, and computer screens.
3. **Signage:**
   LEDs are used in electronic signs and billboards due to their brightness and energy efficiency.
4. **Other Applications**:
   LEDs are also used in flashlights, traffic lights, and other electronic devices.

ASCII stands for American Standard Code for Information Interchange, A character encoding standard that assigns unique numerical values to letters, digits, punctuation marks, and other symbols.



PROGRAM

```
int mychar = 0; // for incoming serial data
void setup()
{
        Serial.begin(9600); // opens serial port, sets data rate to 9600 bps
        pinMode(7,OUTPUT);
       pinMode(8,OUTPUT);
       pinMode(9,OUTPUT);
}
void loop()
{
 // send data only when you receive data:
 if (Serial.available() > 0)
 {
        // read the incoming byte:
        mychar = Serial.read();
        // say what you got:
       Serial.print("I received: ");
       Serial.println(mychar);
 }
if(mychar == 114 ) //r
{
       digitalWrite(7,HIGH);
       digitalWrite(8,LOW);
       digitalWrite(9,LOW);
}
if(mychar == 103 ) //g
{
       digitalWrite(7,LOW);
       digitalWrite(8,HIGH);
       digitalWrite(9,LOW);
}
if(mychar == 98 ) //b
{
       digitalWrite(7,LOW);
       digitalWrite(8,LOW);
       digitalWrite(9,HIGH);
}
}
```

CONCLUSION – Thus we have implemented arduino code for LEDs using r,g,b characters ASCII value

PRACTICAL NO. 8 (Group B)

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

- **Outcome:** Connectivity, configuration and serial communication with Arduino.

- **Hardware Requirement:** Arduino, USB cable etc.

- **Software Requirement**: Arduino IDE

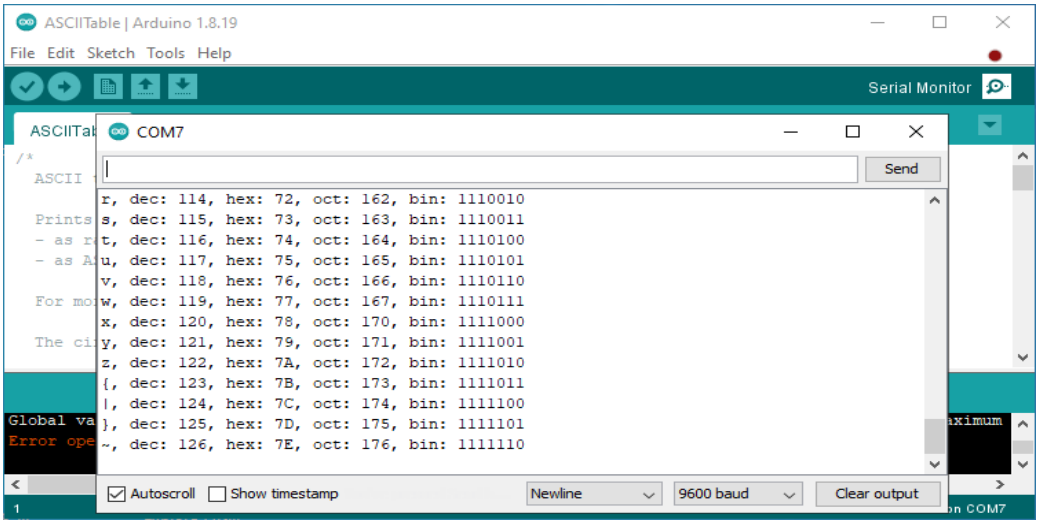
- **Theory:**

- **Arduino Serial Monitor for Beginners**

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

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

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



Click the Serial Monitor icon near the top right of the Arduino IDE to open the serial monitor window. The above image shows the serial monitor window opened, and on top of the Arduino IDE window. Because the ASCIITable example is loaded on the Arduino board, when the serial monitor window opens, the Arduino sends text to the serial monitor window. This is also because opening the serial monitor window resets the Arduino board, causing the ASCIITable sketch to run from the beginning again.

The ASCIITable sketch sends text out of the USB port of the Arduino. Because the serial monitor is connected to the USB port, it receives the text and displays it in the big receive area of the window. As a result, text scrolls on the serial monitor window for a while. The text then stops because the Arduino has finished sending text. Use the right scrollbar in the serial monitor window to scroll up. Scrolling up reveals all of the text that the Arduino sent.

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

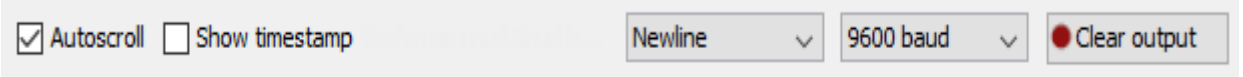
When junk, or garbage characters, or even nothing is displayed in the serial monitor, it is usually because of an incorrect baud rate setting. Look at the bottom of the serial monitor in the above image. Notice the value 9600 baud in a box. This is the baud setting of communications between the Arduino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

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

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

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

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

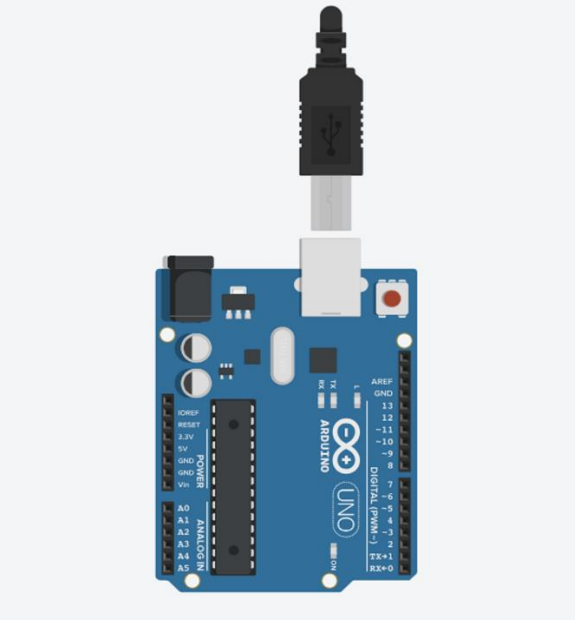


Serial Monitor Window Clear Output Button

- **What the ASCIITable Sketch Does**

ASCII stands for **American Standard Code for Information Interchange**. ASCII is a standard way that uses numbers to represent various characters. For example, the decimal number 65 represents the letter A. Another example is the decimal number 125 represents a closing brace: }. This allows computers to send and receive text by sending and receiving numbers. For example when a computer receives the number 65, it knows to display the letter A.

The ASCIITable sketch sends the numbers 33 through to 126 out of the USB port. This results in the printable text characters from the ASCII table displayed in the serial monitor window. In addition to the ASCII characters, the number that represents each character is displayed. Each number is shown in four different numbering systems. These are the decimal, hexadecimal, octal and binary number systems. In the serial monitor window, these number systems are abbreviated to **dec, hex, oct, bin.**



Program Code:

```
int out;

void setup()
{

  Serial.begin(9600); // opens serial port, sets data rate to 9600 bps

}

void loop()
{
  // send data only when you receive data:
  if (Serial.available() > 0)
  {
    // read the incoming byte:
    int num=Serial.readString().toInt();
    // say what you got
   Serial.print("I received: ");
   Serial.println(num);
   out = num*num;
   Serial.print("Sq of no.: ");
   Serial.println(out);
  }
}
```

- **Important functions used:**

1. **Serial.begin()** is an Arduino function used to initialize serial communication between the Arduino board and a connected device. It sets up the communication parameters, such as baud rate, and prepares the hardware serial port for data transmission and reception.
2. **Serial.println()** is an Arduino function used to print data to the serial monitor. It prints the specified data followed by a newline character ('\n'). Top of Form
3. **Serial.available()** is an Arduino function used to check if there is data available to read from the serial input buffer. It returns the number of bytes available for reading.

Conclusion:

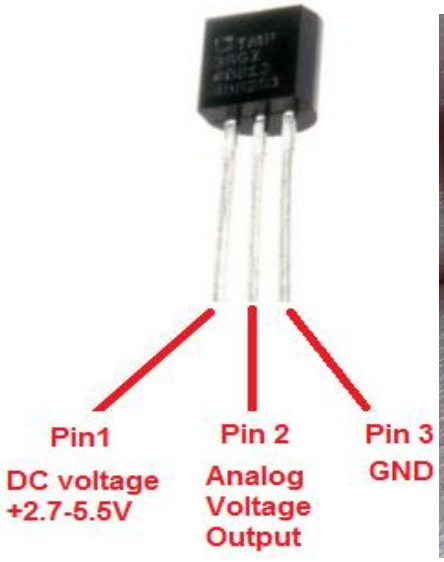
We have implemented arduino code to find square of a number

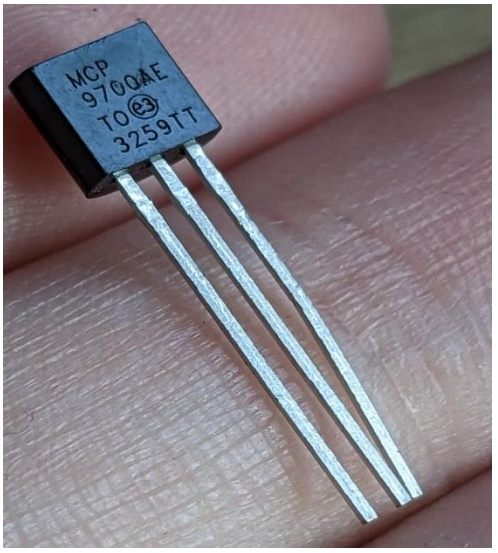
PRACTICAL NO. 9 (Group B)

- **Aim:** Write a program read the temperature sensor and send the values to the serial monitor on the computer.
- **Outcome:** Understanding working principle of temperature sensor.
- **Hardware Requirement:** Arduino, TMP sensor, etc
- **Software Requirement**: Arduino IDE
- **Theory:**

TMP Temperature Sensor

The TMP35/TMP36/TMP37 are low voltage, precision centigrade temperature sensors. They provide a voltage output that is linearly proportional to the Celsius (centigrade) temperature. The TMP35/TMP36/TMP37 do not require any external calibration to provide typical accuracies. The low output impedance of the TMP35/TMP36/TMP37 and its linear output and precise calibration simplify interfacing to temperature control circuitry and ADCs. All three devices are intended for single-supply operation from 2.7 V to 5.5 V maximum. The supply current runs well below 50 µA, providing very low self-heating—less than 0.1°C in still air. In addition, a shutdown function is provided to cut the supply current to less than 0.5 µA.

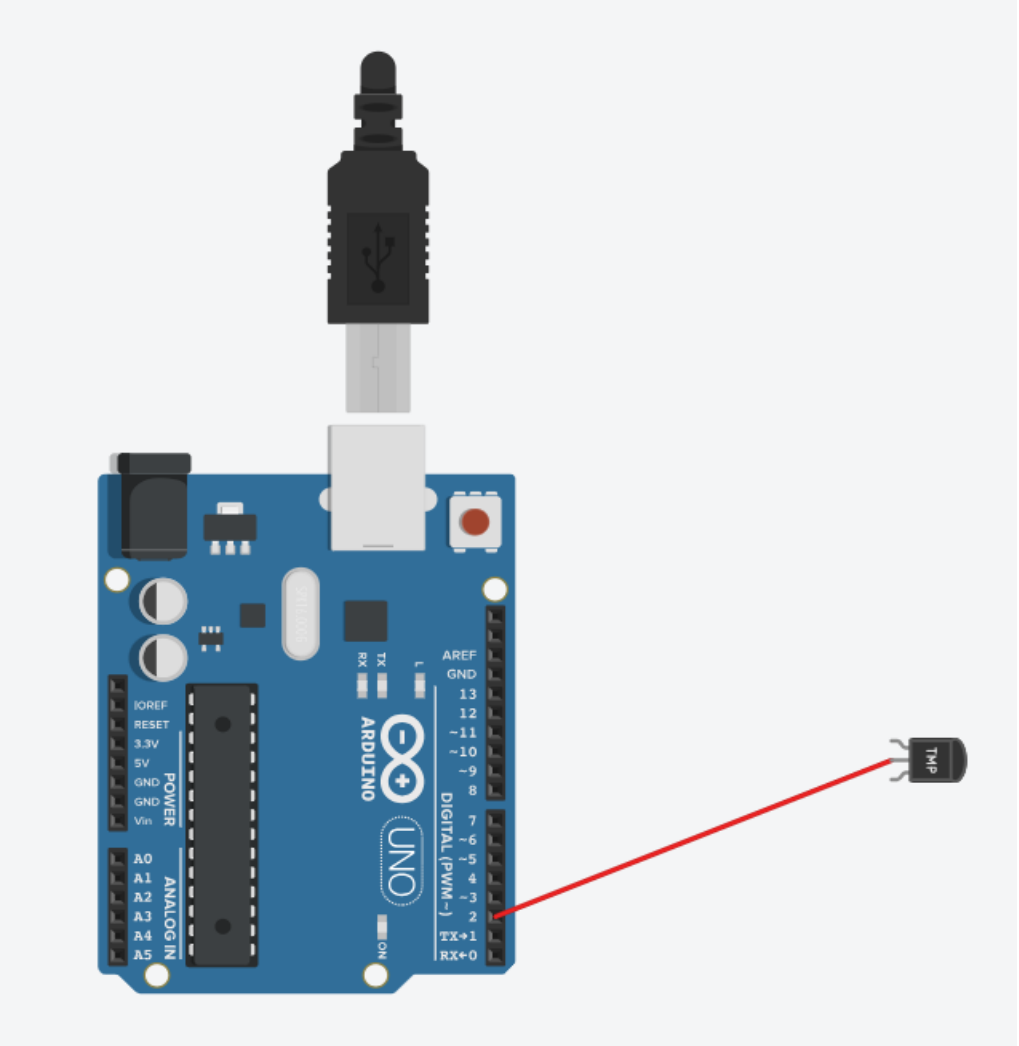




Pin Number	Pin Name	Description
1	Vcc	Input voltage is +5V for typical applications
2	Analog Out	There will be increase in 10mV for raise of every 1°C. Can range from -1V(-55°C) to 6V(150°C)
3	Ground	Connected to ground of circuit

- **TMP Sensor Features**

1. **Low Voltage Operation:** The TMP36 operates on a low voltage, typically between 2.7 and 5.5V DC.
2. **Precision Temperature Sensing:** It's designed for accurate temperature measurement, providing a linear voltage output that corresponds to the Celsius temperature.
3. **No External Calibration:** The sensor doesn't require any external calibration to achieve its typical accuracy.
4. **Accuracy:**
    a. ±1°C at +25°C
    b. ±2°C over the -40°C to +125°C temperature range
5. **Output:** The sensor provides an analog voltage output that is linearly proportional to the Celsius temperature.
6. **Temperature Range:** The sensor can measure temperatures between -40°C and +125°C.
7. **How it Works:**
   The TMP36 uses the property of diodes; as a diode changes temperature the voltage changes with it at a known rate. The sensor measures the small change and outputs an analog voltage between 0 and 1.75VDC based on it.



PROGRAM CODE

```
#define DHTPIN 2
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  int temperature = analogRead(DHTPIN);
  Serial.print("Temperature is: ");
  Serial.print(temperature);
  Serial.println(" degree Fahrenheit ");
 delay(2000);
}
```

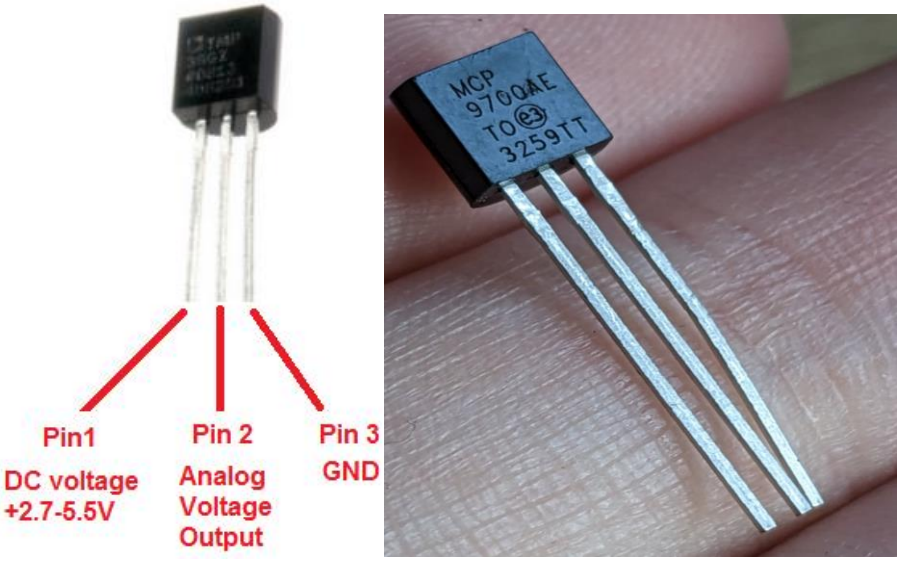
Conclusion: - Thus we have implemented arduino code for temperature reading.

PRACTICAL NO. 10 (Group B)

- **Aim:** Write a program so it displays the temperature in Fahrenheit as well as the maximum and minimum temperatures it has seen.
- **Outcome:** Understanding working principle of TMP temperature sensor.
- **Hardware Requirement:** Arduino, TMP, etc
- **Software Requirement**: Arduino IDE
- **Theory:**

TMP Temperature Sensor

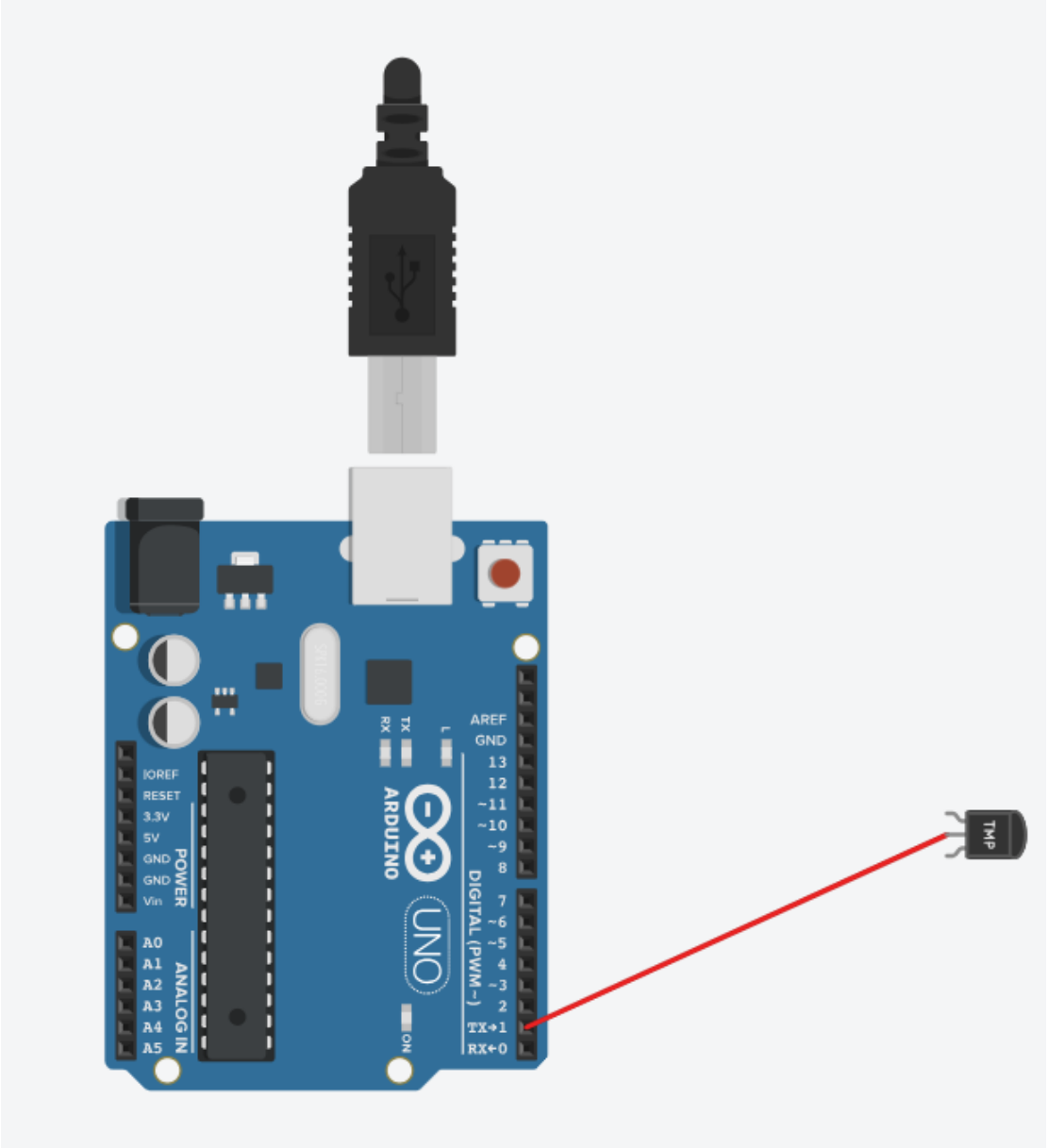
The TMP35/TMP36/TMP37 are low voltage, precision centigrade temperature sensors. They provide a voltage output that is linearly proportional to the Celsius (centigrade) temperature. The TMP35/TMP36/TMP37 do not require any external calibration to provide typical accuracies. The low output impedance of the TMP35/TMP36/TMP37 and its linear output and precise calibration simplify interfacing to temperature control circuitry and ADCs. All three devices are intended for single-supply operation from 2.7 V to 5.5 V maximum. The supply current runs well below 50 µA, providing very low self-heating—less than 0.1°C in still air. In addition, a shutdown function is provided to cut the supply current to less than 0.5 µA.



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

- **TMP Sensor Features**

1. **Low Voltage Operation:** The TMP36 operates on a low voltage, typically between 2.7 and 5.5V DC.
2. **Precision Temperature Sensing:** It's designed for accurate temperature measurement, providing a linear voltage output that corresponds to the Celsius temperature.
3. **No External Calibration:** The sensor doesn't require any external calibration to achieve its typical accuracy.
4. **Accuracy:**
    a. ±1°C at +25°C
    b. ±2°C over the -40°C to +125°C temperature range
5. **Output:** The sensor provides an analog voltage output that is linearly proportional to the Celsius temperature.
6. **Temperature Range:** The sensor can measure temperatures between -40°C and +125°C.
7. **How it Works:**
    The TMP36 uses the property of diodes; as a diode changes temperature the voltage changes with it at a known rate. The sensor measures the small change and outputs an analog voltage between 0 and 1.75VDC based on it.



PROGRAM CODE

```
int val;
int tempPin = 1;

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

void loop()
{
 val = analogRead(tempPin);
 float mv = ( val/1024.0)*5000;
 float cel = mv/10;
 float farh = (cel*9)/5 + 32;
 Serial.print("TEMPRATURE = ");
 Serial.print(cel);
 Serial.print("*C");
 Serial.println();
 delay(1000);
float tfmax = 100;
float tfmin = 0;
if (farh > tfmax)
{
        tfmax = farh;
}

if (farh < tfmin)
{
         tfmin = farh;
}
Serial.print("TEMPRATURE = ");
Serial.print(farh);
Serial.print("*F");
Serial.println();
Serial.print("Max Temp");
Serial.print(tfmax);
Serial.println();
Serial.print("Min Temp");
Serial.print(tfmin);
Serial.println();
Serial.println();
}
```

Conclusion: -

Thus we have implemented arduino code for temperature reading as well as displayed minimum and maximum temperature.

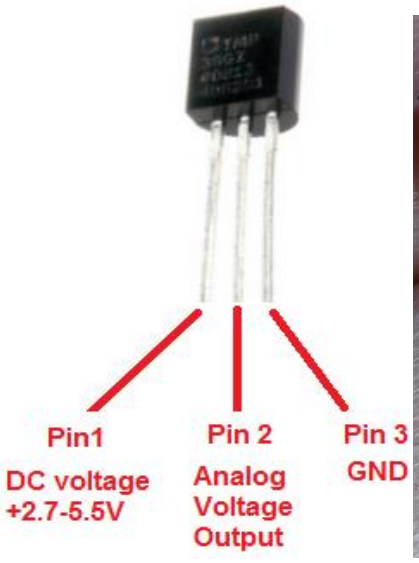
PRACTICAL NO. 11 (Group B)

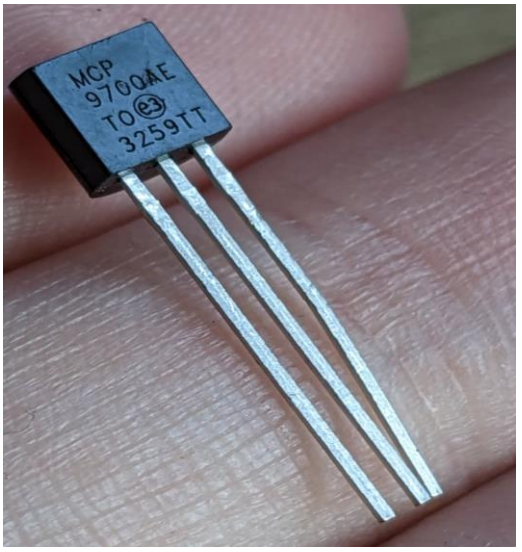
- **Aim:** Write a program to show the temperature and shows a graph of the recent measurements
- **Hardware Requirement:** Arduino, TMP, etc
- **Software Requirement**: Arduino IDE
- **Theory:**

1. Introduction:

TMP Temperature Sensor

The TMP35/TMP36/TMP37 are low voltage, precision centigrade temperature sensors. They provide a voltage output that is linearly proportional to the Celsius (centigrade) temperature. The TMP35/TMP36/TMP37 do not require any external calibration to provide typical accuracies. The low output impedance of the TMP35/TMP36/TMP37 and its linear output and precise calibration simplify interfacing to temperature control circuitry and ADCs. All three devices are intended for single-supply operation from 2.7 V to 5.5 V maximum. The supply current runs well below 50 µA, providing very low self-heating—less than 0.1°C in still air. In addition, a shutdown function is provided to cut the supply current to less than 0.5 µA.

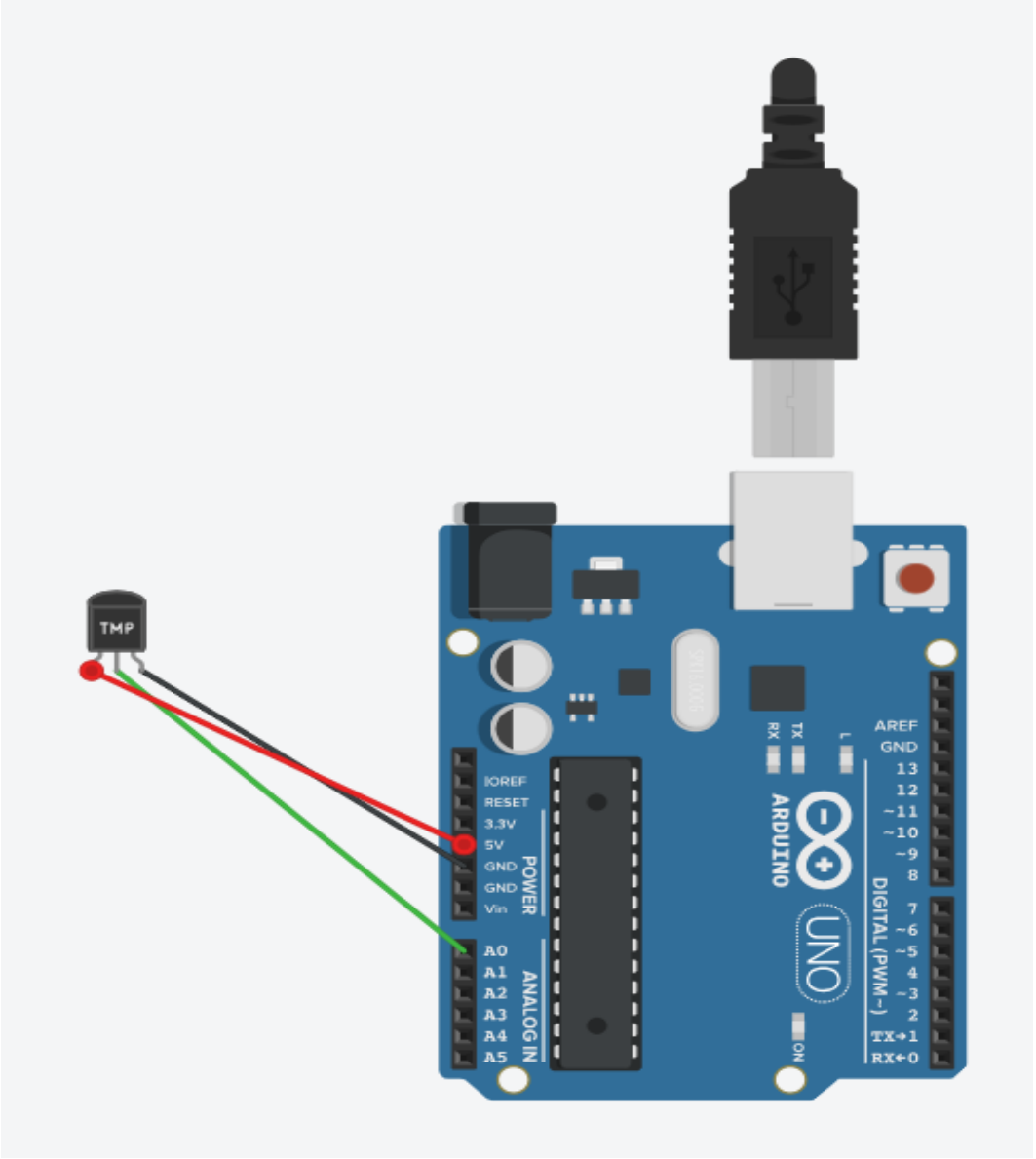




Pin Number	Pin Name	Description
1	Vcc	Input voltage is +5V for typical applications
2	Analog Out	There will be increase in 10mV for raise of every 1°C. Can range from -1V(-55°C) to 6V(150°C)
3	Ground	Connected to ground of circuit

- **TMP Sensor Features**

1. **Low Voltage Operation:** The TMP36 operates on a low voltage, typically between 2.7 and 5.5V DC.
2. **Precision Temperature Sensing:** It's designed for accurate temperature measurement, providing a linear voltage output that corresponds to the Celsius temperature.
3. **No External Calibration:** The sensor doesn't require any external calibration to achieve its typical accuracy.
4. **Accuracy:**
    a. ±1°C at +25°C
    b. ±2°C over the -40°C to +125°C temperature range
5. **Output:** The sensor provides an analog voltage output that is linearly proportional to the Celsius temperature.
6. **Temperature Range:** The sensor can measure temperatures between -40°C and +125°C.
7. **How it Works:**
    The TMP36 uses the property of diodes; as a diode changes temperature the voltage changes with it at a known rate. The sensor measures the small change and outputs an analog voltage between 0 and 1.75VDC based on it.



Program code

```
int inPin = A0;
float tempC = 0;
int n = 0;

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

void loop()
{
 int value = analogRead(inPin); // read the value from the sensor
 tempC = ((value * 0.00488) - 0.5) * 100;
 //Serial.print("Temperature in Celsius = ");
 Serial.println(tempC);
 if(n==30)
{
   n=0;
 delay(5000);
 }
}
```

Conclusion: -

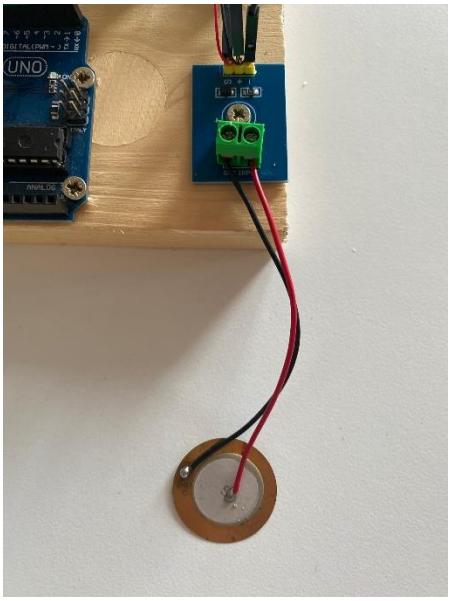
Thus we have implemented arduino code with graph display

PRACTICAL NO 12. (GROUP B)

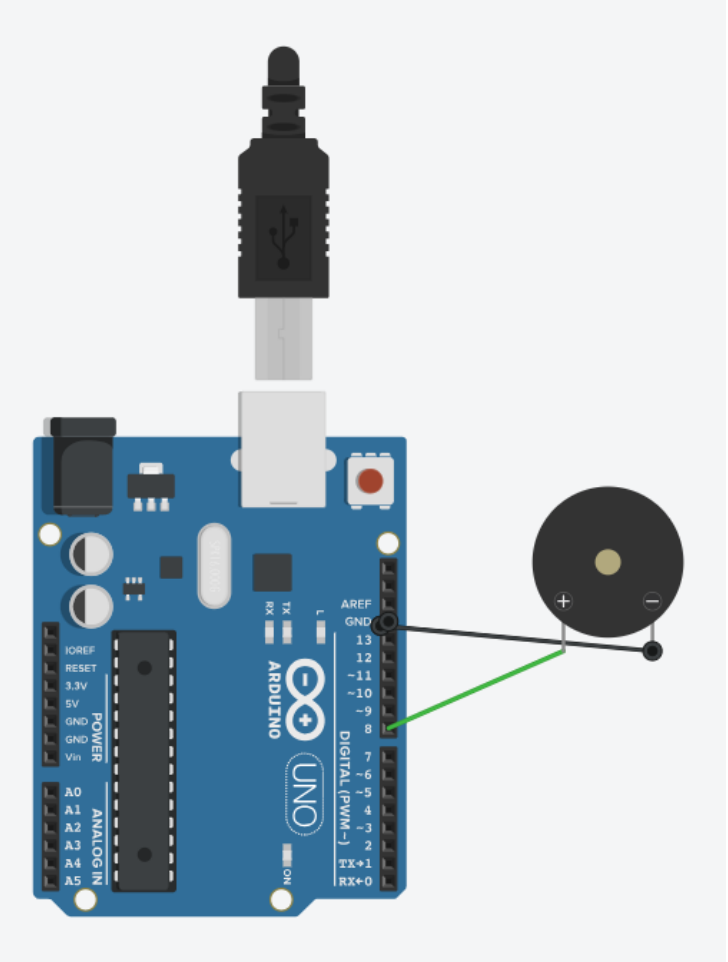
- **Aim:** Write a program using piezo element and use it to play a tune after someone knocks
- **Outcome**: Understanding Working Principle of Piezo element.

- **Hardware Requirement:** Arduino kit, piezo element, jumper wires etc
- **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. Piezos can be used both to play tones and to detect tones.
- 
- The sketch reads the piezo output using the
- **analogRead() 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.
- Open the serial monitor to see this text.

- Piezos are **polarized**, meaning that voltage passes through them (or out of them) in a specific direction.
- 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.



Procedure:

1) Connecting the Piezo Element:

• Connect positive terminal of the piezo element to pin 8 on the Arduino board.

• Connect the negative terminal of the piezo element to the ground (GND) pin on the Arduino board.

2) Uploading the Code:

• Copy the provided Arduino code into the Arduino IDE.

• Connect the Arduino board to your computer using a USB cable.

• Upload the code to the Arduino board.

3) **Testing**:

- Knock on the piezo element or trigger the input in any desired way.
- Observe the melody being played by the piezo element.

PROGRAM

```
int pin=8;
void setup()
{
}
void loop()
{
 tone(8,200);
 delay(2000);// Wait for 2000 millisecond(s)
 noTone(8);
 delay(2000);// Wait for 2000 millisecond(s)
}
```

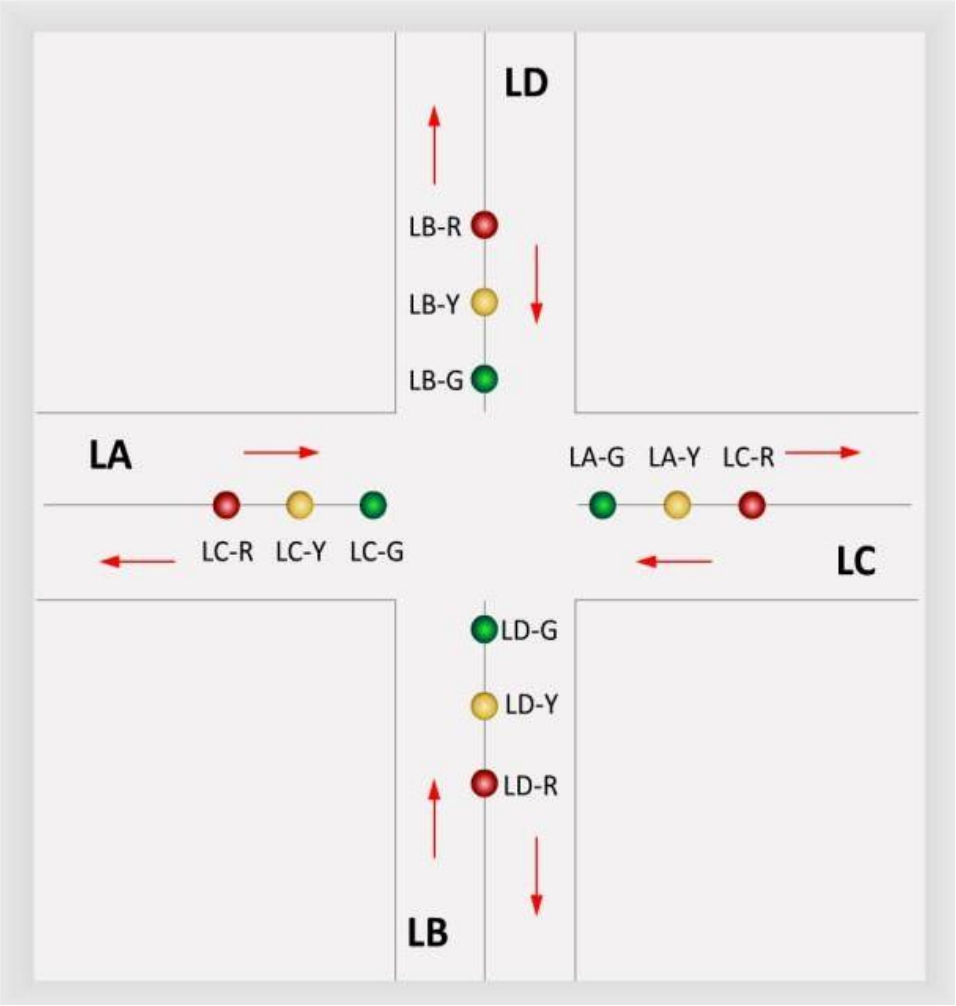
CONCLUSION

Thus we have implemented arduino code for piezo element.

PRACTICAL NO. 13 (Group C)

- **Aim:** Write an application to control the operation of hardware simulated traffic signals.
- **Outcome**: Understanding Working Principle of Traffic Light Controller.

- **Hardware Requirement:** Traffic Light Controller Kit
- **Software Requirement:** Arduino IDE
- **Hardware Modules**: Arduino Board, PC / Laptop, 12 LED's , 12 resistors of 220 ohm .
- **Theory:**



- A simple traffic light system for a **4-way intersection** is implemented using arduino where the traffic is controlled in a pre-defined timing system.
- There are **4 lanes LA, LB, LC and LD** going towards the signal.
- At the cross road there are 4 sets of Traffic lights opposite to each lane.

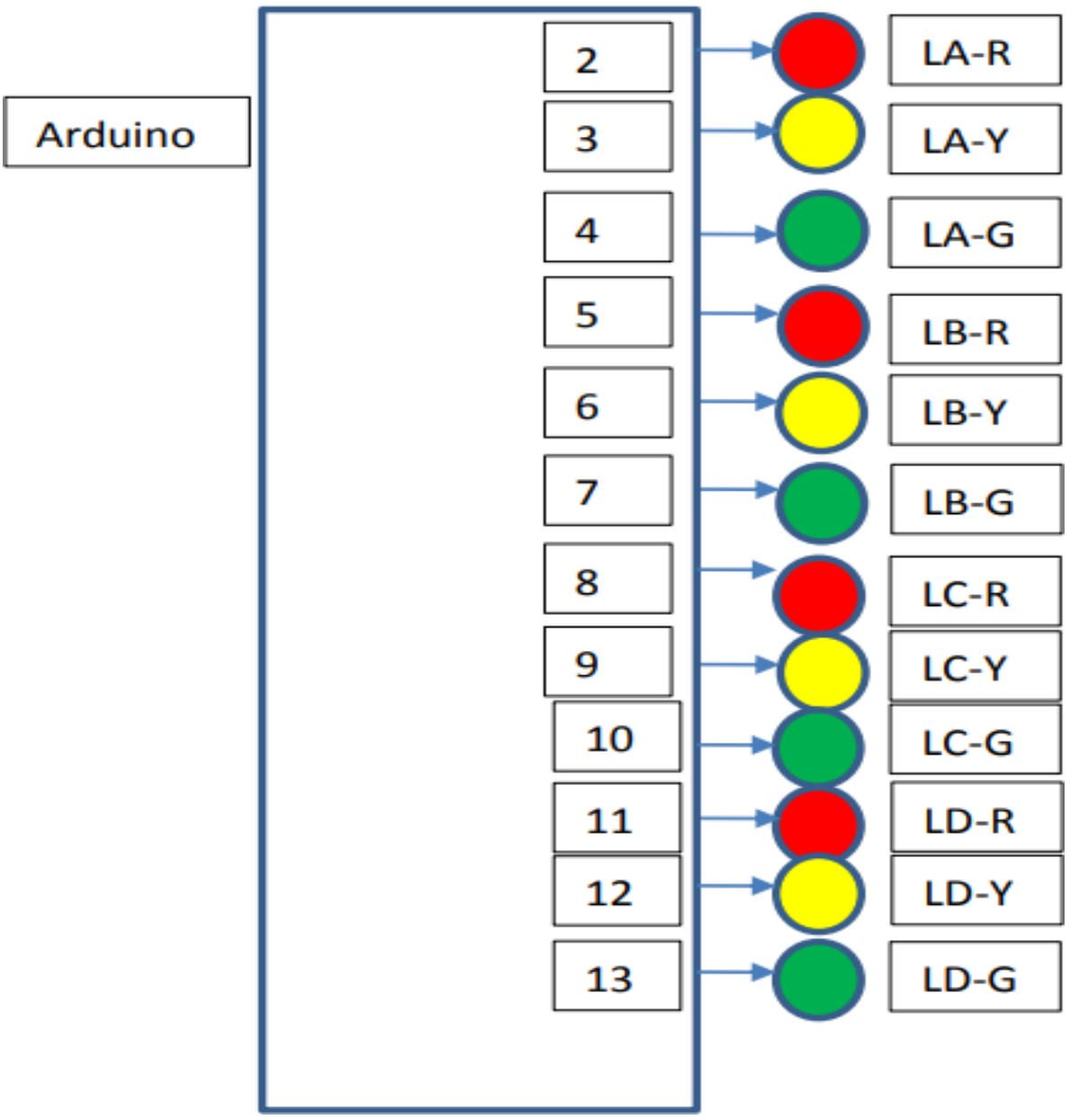
- These sets **are**
    - **LA(LA-G, LA-Y, LA-R)**
    - **LB(LB-G, LB-Y, LB-R)**
    - **LC(LC-G, LC-Y, LC-R)**
    - **LD(LD-G, LD-Y, LD-R)**

- Traffic from any lane moves when its corresponding Green light is ON.
- "ON time" of any Red light is dependent on the "ON time" of Yellow light and Green light of other 3 signal lights.
- "ON time" of Yellow light is same for all lanes.
- User can specify and change the "ON time" of the Green light and Red light of each signal separately.
- The Traffic light pattern keeps on repeating till the next change made by the user

Safety precautions:

- First, make all the connections as per steps given below
- Then connect Arduino board to PC/Laptop

Schematic diagram:


Arduino
2
3
4
5
6
7
8
9
10
11
12
13
LA-R
LA-Y
LA-G
LB-R
LB-Y
LB-G
LC-R
LC-Y
LC-G
LD-R
LD-Y
LD-G

1) Connect the R, Y, G pins of “Lane A” to 2, 3, 4 pins of Arduino Board respectively.
2) Connect the R, Y, G pins of “Lane B” to 5, 6, 7 pins of Arduino Board respectively.
3) Connect the R, Y, G pins of “Lane C” to 8, 9, 10 pins of Arduino Board respectively.
4) Connect the R, Y, G pins of “Lane D” to 11, 12, 13 pins of Arduino Board respectively.
5) Connect the “COM” pin of the Traffic Signal module to the GND pin of Arduino Board.

Procedure

- Write the program as per the algorithm given below.
  Save and compile the program.
- Connect the Arduino board to PC/Laptop using USB cable.
- Upload the program and check the output.

Algorithm:

- Declare the 12 led’s data pins.
- Declare **pinmode** of 12 led’s as output.
- Firstly, for the Lane-A, the signal becomes Green.
- Hence, for all other Lanes, the corresponding Red signal is on.
- As a warning indicator, the Yellow light in Lane-A is tuned on indicating that the red light is about to light up.
- Similarly, the yellow light in the Lane-C is also turned as an indication that the green light about to be turned on.
- After a time delay for the Lane-C, the signal becomes Green.
- So at the same time the signal for Lane-A becomes Red.
- As a warning indicator, the Yellow light in Lane-C is tuned on indicating that the red light is about to light up.
- Similarly, the yellow light in the Lane-D is also turned as an indication that the green light about to be turned on.
- After a time delay for the Lane-D, the signal becomes Green.
- So at the same time the signal for Lane-C becomes Red.
- As a warning indicator, the Yellow light in Lane-D is tuned on indicating that the red light is about to light up.
- Similarly, the yellow light in the Lane-B is also turned as an indication that the green light about to be turned on.
- After a time delay for the Lane-B, the signal becomes Green.
- So at the same time the signal for Lane-D becomes Red.
- As a warning indicator, the Yellow light in Lane-B is tuned on indicating that the red light is about to light up.
- Similarly, the yellow light in the Lane-A is also turned as an indication that the green light about to be turned on.
- The system then loops back to Lane 1 where the process mentioned above will be repeated all over again.

PROGRAM

```
byte R1 = 2;
byte Y1 = 3;
byte G1 = 4;

byte R2 = 5;
byte Y2 = 6;
byte G2 = 7;

byte R3 = 8;
byte Y3 = 9;
byte G3 = 10;

byte R4 = 11;
byte Y4 = 12;
byte G4 = 13;

int Lane_A[] = {R1, Y1, G1};          // Lane 1 Red, Yellow and Green
int Lane_B[] = {R2, Y2, G2};          // Lane 2 Red, Yellow and Green
int Lane_C[] = {R3, Y3, G3};          // Lane 3 Red, Yellow and Green
int Lane_D[] = {R4, Y4, G4};          // Lane 4 Red, Yellow and Green

void setup()
{
  for (int i = 0; i < 3; i++)
  {
    pinMode(Lane_A[i], OUTPUT);
    pinMode(Lane_B[i], OUTPUT);
    pinMode(Lane_C[i], OUTPUT);
    pinMode(Lane_D[i], OUTPUT);
  }
  for (int i = 0; i < 3; i++)
  {
    digitalWrite(Lane_A[i], LOW);
    digitalWrite(Lane_B[i], LOW);
    digitalWrite(Lane_C[i], LOW);
    digitalWrite(Lane_D[i], LOW);
  }
}

void loop()
{
  digitalWrite(Lane_A[2], HIGH);     //LaneA Green ON
  digitalWrite(Lane_A[0], LOW);      //LaneA Red OFF
  digitalWrite(Lane_C[0], HIGH);     //LaneA Red OFF
  digitalWrite(Lane_D[0], HIGH);     //LaneA Red OFF
  digitalWrite(Lane_B[0], HIGH);     //LaneA Red OFF
  delay(7000);

  digitalWrite(Lane_A[2], LOW);      //LaneA Green OFF
```

```
 digitalWrite(Lane_A[1], HIGH);     //LaneA Yellow ON
 delay(3000);

 digitalWrite(Lane_A[0], HIGH);     //LaneA Red ON
 digitalWrite(Lane_A[1], LOW);      //LaneA Yellow OFF
 digitalWrite(Lane_B[0], LOW);      //LaneB Red OFF
 digitalWrite(Lane_B[2], HIGH);     //LaneB Green ON
 delay(7000);

 digitalWrite(Lane_B[2], LOW);      //LaneB Green OFF
 digitalWrite(Lane_B[1], HIGH);     //LaneB Yellow ON
 delay(3000);

 digitalWrite(Lane_B[0], HIGH);     //LaneB Red ON
 digitalWrite(Lane_B[1], LOW);      //LaneB Yellow OFF
 digitalWrite(Lane_C[0], LOW);      //LaneC Red OFF
 digitalWrite(Lane_C[2], HIGH);     //LaneC Green ON
 delay(7000);

 digitalWrite(Lane_C[2], LOW);      //LaneC Green OFF
 digitalWrite(Lane_C[1], HIGH);     //LaneC Yellow ON
 delay(3000);

 digitalWrite(Lane_C[0], HIGH);     //LaneC Red ON
 digitalWrite(Lane_C[1], LOW);       //LaneC Yellow OFF
 digitalWrite(Lane_D[0], LOW);      //LaneD Red OFF
 digitalWrite(Lane_D[2], HIGH);     //LaneD Green ON
 delay(7000);

 digitalWrite(Lane_D[2], LOW);      //LaneD Green OFF
 digitalWrite(Lane_D[1], HIGH);     //LaneD Yellow ON
 delay(3000);
 digitalWrite(Lane_D[1], LOW);      //LaneD Yellow OFF
}
```

- **Observation**:

  Observe the output on LEDs.

Conclusion: -

Thus we have implemented arduino code to control traffic lights

_____

_____

_____

_____