Empowerment Through Quality Technical Education
AJEENKYA DY PATIL SCHOOL OF ENGINEERING
Dr. D. Y. Patil Knowledge City, Charholi (Bk), Lohegaon, Pune – 412 105
Website: https://dypsoe.in/

LAB MANUAL

INTERNET OF THINGS LABORATORY

(217531)

SE (AI&DS) 2020 COURSE

Course Coordinator

Prof. Rohini Shrikhande

DEPARTMENT OF

ARTIFICIAL INTELLIGENCE & DATA SCIENCE

Department of Artificial Intelligence & Data Science

Vision:

Imparting quality education in the field of Artificial Intelligence and Data Science

Mission:

- To include the culture of R and D to meet the future challenges in AI and DS.
- To develop technical skills among students for building intelligent systems to solve problems.
- To develop entrepreneurship skills in various areas among the students.
- To include moral, social and ethical values to make students best citizens of country.

Program Educational Outcomes:

1. To prepare globally competent graduates having strong fundamentals, domain knowledge, updated with modern technology to provide the effective solutions for engineering problems.

2. To prepare the graduates to work as a committed professional with strong professional ethics and values, sense of responsibilities, understanding of legal, safety, health, societal, cultural and environmental issues.

3. To prepare committed and motivated graduates with research attitude, lifelong learning, investigative approach, and multidisciplinary thinking.

4. To prepare the graduates with strong managerial and communication skills to work effectively as individuals as well as in teams.

Program Specific Outcomes:

1. Professional Skills- The ability to understand, analyze and develop computer programs in the areas related to algorithms, system software, multimedia, web design, networking, artificial intelligence and data science for efficient design of computer-based systems of varying complexities.

2. Problem-Solving Skills- The ability to apply standard practices and strategies in software project development using open-ended programming environments to deliver a quality product for business success.

3. Successful Career and Entrepreneurship- The ability to employ modern computer languages, environments and platforms in creating innovative career paths to be an entrepreneur and to have a zest for higher studies.

Table of Contents

Contents

1. Guidelines to manual usage

This manual assumes that the facilitators are aware of collaborative learning methodologies.

This manual will provide a tool to facilitate the session on Digital Communication modules in collaborative learning environment.

The facilitator is expected to refer this manual before the session.

Icon of Graduate Attributes

K Applying Knowledge	**A** Problem Analysis	**D** Design & Development	**I** Investigation of problems
M Modern Tool Usage	**E** Engineer & Society	**E** Environment Sustainability	**T** Ethics
T Individual & Team work	**O** Communication	**M** Project Management & Finance	**I** Life-Long Learning

Disk Approach- Digital Blooms Taxonomy

1: Remembering / Knowledge
2: Comprehension / Understanding
3: Applying
4: Analyzing
5: Evaluating
6: Creating / Design

Program Outcomes:

1. **Engineering knowledge:** An ability to apply knowledge of mathematics, including discrete mathematics, statistics, science, computer science and engineering fundamentals to model the software application.
2. **Problem analysis:** An ability to design and conduct an experiment as well as interpret data, analyze complex algorithms, to produce meaningful conclusions and recommendations.
3. **Design/development of solutions**: An ability to design and development of software system, component, or process to meet desired needs, within realistic constraints such as economic, environmental, social, political, health & safety, manufacturability, and sustainability.
4. **Conduct investigations of complex problems**:An ability to use research based knowledge including analysis, design and development of algorithms for the solution of complex problems interpretation of data and synthesis of information to provide valid conclusion.
5. **Modern tool usage:** An ability to adapt current technologies and use modern IT tools, to design, formulate, implement and evaluate computer based system, process, by considering the computing needs, limits and constraints.
6. **The engineer and society:** An ability of reasoning about the contextual knowledge of the societal, health, safety, legal and cultural issues, consequent responsibilities relevant to IT practices.
7. **Environment and sustainability:** An ability to understand the impact of engineering solutions in a societal context and demonstrate knowledge of and the need for sustainable development.
8. **Ethics:** An ability to understand and commit to professional ethics and responsibilities and norms of IT practice.
9. **Individual and team work:** An ability to apply managerial skills by working effectively as an individual, as a member of a team, or as a leader of a team in multidisciplinary projects.
10. **Communication:** An ability to communicate effectively technical information in speech,presentation, and in written form
11. **Project management and finance:** An ability to apply the knowledge of Information Technology and management principles and techniques to estimate time and resources needed to complete engineering project.
12. **Life-long learning:** An ability to recognize the need for, and have the ability to engage inindependent and life-long learning.

Course Name: Internet of Things Laboratory

Course Code: (217531)

Course Outcomes

CO1: Understand IOT Application Development using Raspberry Pi/ Beagle board/ Arduino board

CO2: Develop and modify the code for various sensor based applications using wireless sensor modules and working with a variety of modules like environmental modules.

CO3: Make use of Cloud platform to upload and analyze any sensor data

CO to PO Mapping:

PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	2	1	3	-	-	-	-	-	-	-	-	-
CO2	-	2	3	-	-	-	-	-	-	-	-	-
CO3	-	-	3	2	-	-	-	-	-	-	-	3

Laboratory 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.

Laboratory Equipment/Software

1. Computer/PC
2. Arduino Software

2. Laboratory Experiment list

List of Assignments
Group A
1. Study of Raspberry-Pi, Beagle board, Arduino and other micro controller
2. Study of different operating systems for Raspberry-Pi/Beagle board/Arduino. Understanding the process of OS installation.
3. Study of different GATES (AND, OR, XOR), Sensors and basic binary operations.
4. 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.
Group B
5. Write a program using Arduino to control LED (One or more ON/OFF). Or Blinking
6. 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.
7. 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.
8. Write a program that asks the user for a number and outputs the number squared that is entered.
9. Write a program to control the color of the LED by turning 3 different potentiometers. One will be read for the value of Red, one for the value of Green, and one for the value of Blue.
10. Write a program read the temperature sensor and send the values to the serial monitor on the computer.
11. Write a program so it displays the temperature in Fahrenheit as well as the maximum and minimum temperatures it has seen.
12. Write a program to show the temperature and shows a graph of the recent measurements.

Experiment No. 1

Aim: Study of Raspberry-Pi, Beagle board, Arduino and other micro controller

Problem Statement: Study of Raspberry-Pi, Beagle board, Arduino and other micro controller (History & Elevation)

Prerequisites: Data Communication

Course Objectives: To understand hardware details of Raspberry-Pi, Beagle board, Arduino and other micro controller

Course Outcomes:

CO1: To understand fundamentals of IoT and Embedded System

CO3: To discover various components of IoT to build IoT System

Theory:

- **Raspberry Pi**

The Raspberry Pi was created in February 2012 by the Raspberry Pi Foundation, Originally setup to promote and teach basic computer science in schools and colleges around the UK. They initially released 2 Devices, the Model A and the Model B, these computers ranged in spec and capabilities. Soon after the release of these products a community was formed and thousands of "Tech-Heads" bought one and started to create new projects with it, for instance setup a Home Media Centre and played the popular game Mine craft etc. The products were so popular due to their lower cost. They were efficient and durable which made them easy to modify and crate projects on. The device ran Linux a popular OS for developers due to it being open-source. On the Raspberry Pi website they created 2 images that could be installed easily onto a SD card which would then act as the OS for the device, one of the images was based off of Debian a popular lightweight Linux OS and was called Raspbian, the other was called Raspbmc and was based off the popular media Centre software Codi (Formally known as XBMC). In February 2014 they had been reported to have sold 4.5 million boards, soon after this success they released the Model A+ and Model b+ which provided more GPIO‟s and used less power to run. In early 2015 the Raspberry PI 2 was announced with increased MHz by 200 to bring it to 900 MHz and doubled the ram to make it 1GB. Experiment level outcome (ELO1): Perform pre-processing of a text document such as stop word removal, stemming using NLTK. The Raspberry Pi 2 which added more RAM was released in February 2015.

The Figure shows structure of Raspberry Pi.

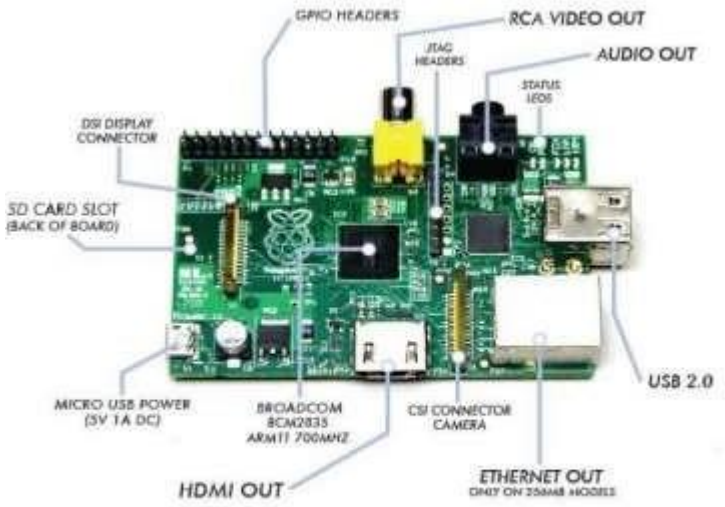


Figure: Raspberry Pi Structure

A Raspberry Pi Zero with smaller size and reduced input/output (I/O) and general- purpose input/output (GPIO) capabilities was released in November 2015 for US$5. Raspberry Pi 3 Model B was released in February 2016 and has on-board WiFi, Bluetooth and USB boot capabilities. By 2017, it became the newest mainline Raspberry Pi. On 28 February 2017, the Raspberry Pi Zero W was launched, which is like Raspberry Pi Zero with Wi-Fi and Bluetooth, for US$10. The Foundation provides Raspbian, a Debian-based Linux distribution for download, as well as third-party Ubuntu, Windows10 IOT Core, RISC OS, and specialized media center distributions. It promotes Python and Scratch as the main programming language, with support for many other languages.

- **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 or CAD for schematics and Cadence Allegro for PCB manufacturing; no simulation software was used. Table shows versions of beagle board.

Table:BeagleBoard version Introduced

Sr.No	Version	Year
1	**BeagleBoard**	July 28, 2008
2	**BeagleBoard rev.C**	May 13, 2009
3	**BeagleBoard-xM**	September 14, 2010
4	**BeagleBone**	October 31, 2011
5	**BeagleBone Black**	April 23, 2013
6	**BeagleBoard-X15**	November 1, 2015

Features of Beagle Board

	BeagleBoard.org BeagleBone Black	BeagleBoard.org BeagleBone (original)	SeeedStudio BeagleBone Green	SanCloud BeagleBone Enhanced
Processor	AM3358 ARM Cortex-A8	AM3358 ARM Cortex-A8	AM3358 ARM Cortex-A8	AM3358 ARM Cortex-A8
Maximum Processor Speed	1GHz	720MHz (1GHz on latest)	1GHz	1GHz
Analog Pins	7	7	7	7
Digital Pins	65 (3.3V)	65 (3.3V)	65 (3.3V)	65 (3.3V)
Memory	512MB DDR3 (800MHz x 16), 2GB (4GB on Rev C) on-board storage using eMMC, microSD card slot	256MB DDR2 (400MHz x 16), microSD card slot	512MB DDR3 (800MHz x 16), 4GB on-board storage using eMMC, microSD card slot	1GB DDR3 (800MHz x 16), 4GB on-board storage using eMMC, microSD card slot
USB	miniUSB 2.0 client port, USB 2.0 host port	miniUSB 2.0 client port, USB 2.0 host port	microUSB 2.0 client port, USB 2.0 host port	miniUSB 2.0 client port, 4 USB 2.0 Ports (2 A-type connectors, 2 on pin headers)
Video	microHDMI, cape add-ons	cape add-ons	cape add-ons	microHDMI, cape add-ons
Audio	microHDMI, cape add-ons	cape add-ons	cape add-ons	microHDMI, cape add-ons

Supported Interfaces	4x UART, 8x PWM, LCD, GPMC, MMC1, 2x SPI, 2x I2C, A/D Converter, 2xCAN Bus, 4 Timers	4x UART, 8x PWM, LCD, GPMC, MMC1, 2x SPI, 2x I2C, A/D Converter, 2xCAN Bus, 4 Timers, FTDI USB to Serial, JTAG via USB	4x UART, 8x PWM, LCD, GPMC, MMC1, 2x SPI, 2x I2C, A/D Converter, 2xCAN Bus, 4 Timers, 2 Grove (I2C, UART)	4x UART, 8x PWM, LCD, GPMC, MMC1, 2x SPI, 2x I2C, A/D Converter, 2xCAN Bus, 4 Timers
Sensors	n/a	n/a	n/a	Barometer, Accelerometer, Gyro, Temperature
MSRP	$49	$89	$39	$69

- **Arduino**

In 2005, in Ivrea, Italy, a project was initiated to make a device for controlling student-built interactive design projects that was less expensive than other prototyping systems available at the time. One of the cofounders, Massimo Banzi, named this piece of hardware Arduino in honor of Bar di Re Arduino (In 1002, King Arduin became the ruler of the Italy. Today, the Bar di Re Arduino, a pub on a cobblestoned street in town, honors his memory), and began producing boards in a small factory located in the same region as the computer company Olivetti. It was in the Interactive Design Institute that a hardware thesis was contributed for a wiring design by a Colombian student named Hernando Barragan. The title of the thesis was "Arduino– La rivoluzione dell"open hardware" ("Arduino – The Revolution of Open Hardware"). Yes, it sounded a little different from the usual thesis but none would have imagined that it would carve a niche in the field of electronics.

Arduino is an open source computer 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 world. The project's products are distributed as open-source hardware and software, which are licensed under the GNU Lesser General Public License (LGPL) or the GNU General Public License (GPL), permitting the manufacture of Arduino boards and software distribution by anyone. Arduino boards are available commercially in preassembled form, or as do-it-yourself (DIY) kits. 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 (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 toolchains, the Arduino project provides an integrated development environment (IDE) based on the Processing language project.

Conclusion: Thus we have studied Raspberry-Pi, Beagle board, Arduino and other micro controller

FAQs:

Q1.What is IOT?

Ans: The Internet of things (IoT) is the network of physical devices, vehicles, home appliances and other items embedded with electronics, software, sensors, actuators, and connectivity which enable these objects to connect and exchange data.

Q2. List limitations of IOT?

Ans: 1.Privacy

2.Safety

3. Compatibility

4. Complexity

5. Unemployment

Q3. List applications of IOT?

Ans :

1.Smart home 4. Smart grids 7. Connected Health (Digital health/Telehealth/Telemedicine)

2.Wearables 5. Industrial internet 8. Smart retail

3. Smart City 6. Connected car 9. Smart supply chain 10. Smart farming

Q4.Draw pin diagram of raspberry pi?

Ans:

Q5.Can we accept analog input through arduino uno?

Ans: Yes, Pin no.A0-A5 used for Analog input

Conclusion: Thus we have studied Raspberry-Pi, Beagle board, Arduino.

Experiment No. 2

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

Aim: To Study different operating systems for Raspberry-Pi/Beagle board/Arduino. And to understand the process of OS installation for each.

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) Beagle Bone 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.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 SD 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: 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 4 .

• Step 6: 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. 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: -

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 rite 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 studied of how to install operating systems for platforms such as Raspberry-Pi/Beagle board/Arduino.

Experiment No. 3

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:

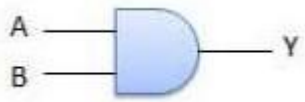
A logic gate is a device that acts as a building block for digital circuits. They perform basic logical functions that are fundamental to digital circuits. Most electronic devices we use today will have some form of logic gates in them. For example, logic gates can be used in technologies such as smartphones, 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, 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 .



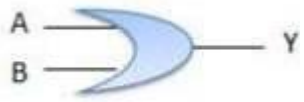
Truth Table:

Inputs		Output
A	B	AB
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.

Logic Diagram



Truth Table

Inputs		Output
A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

XOR Gate

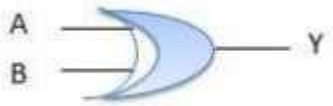
The XOR (exclusive-OR) gate acts in the same way as the logical "either/or." The output is "true" if either, but not both, of the inputs are "true." The output is "false" if both inputs are "false" or if both inputs are "true." Another way of looking at this circuit is to observe that the output is 1 if the inputs are different, but 0 if the inputs are the same**.**

$$Y = A \text{ XOR } B \text{ XOR } C \ldots\ldots N$$

$$Y = A \oplus B \oplus C \ldots\ldots N$$

$$Y = \overline{A}B + A\overline{B}$$

Logic Diagram



Truth Table

Inputs		Output
A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
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 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).

Sensors

There are many different types of sensors. 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

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;

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.

Temperature Sensors

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.

Basic Binary Operations

The basic operations of mathematics- addition, subtraction, division and multiplication are performed on two operands. Even when we try to add three numbers, we add two of them and then add the third number to the result of the two numbers. Thus, the basic mathematical operations are performed on two numbers and are known as binary operations

Types of Binary Operation

There are four main types of binary operations which are:

- Binary Addition
- Binary Subtraction
- Binary Multiplication
- Binary Division

Binary Addition

The result obtained after adding two binary numbers is the binary number itself. Binary addition is the simplest method to add any of the binary numbers. It can be calculated easily if we know the following rules.

Rules

• 0 + 0 = 0

• 0 + 1 = 1

• 1 + 0 = 1

• 1 + 1 =10

Let us take any two binary numbers and add them.

Add: 10001 + 11101 = 101110

Binary Subtraction

The result obtained after subtracting two binary numbers is the binary number itself. Binary subtraction is also the simplest method to subtract any of the binary numbers. It can be calculated easily if we know the following rules.

Rules

• 0 – 0 = 0

• 0 – 1 = 1 (with a borrow of 1)

• 1 – 0 = 1

• 1 – 1 = 0

Let us take any two binary numbers and subtract them.

Binary Multiplication

The binary multiplications are calculated similarly as the other arithmetics numerals are calculated. Let us take any two binary numbers and multiply them.It can be calculated easily if we know the following rules.

Rules

• 0 × 0 = 0

• 0 × 1 = 0

• 1 × 0 = 0

• 1 × 1 = 1

Example: 1101 * 1010 = 10000010

Binary Division

The method of binary division is similar to the 10 decimal system other than the base 2 system.

It can be calculated easily if we know the following rules.

• 1 ÷ 1=1

• 1 ÷ 0 =0

• 0 ÷ 1 = Meaningless

• 0 ÷ 0= Meaningless

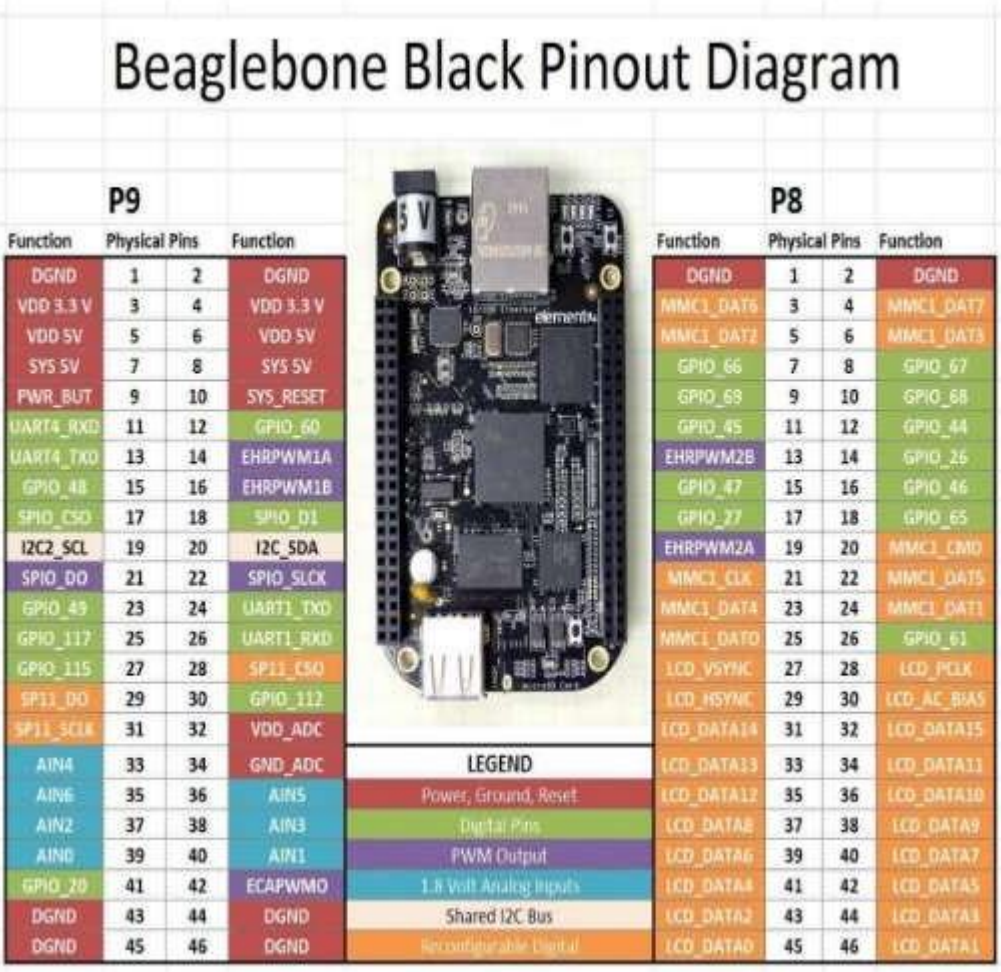
Conclusion:- Thus, we have studied different GATES (AND, OR, XOR), Sensors and basic binary operations.

Experiment No. 4

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

Requirement: Raspberry-Pi /Beagle board circuit, basic peripherals like LEDS, buzzer.

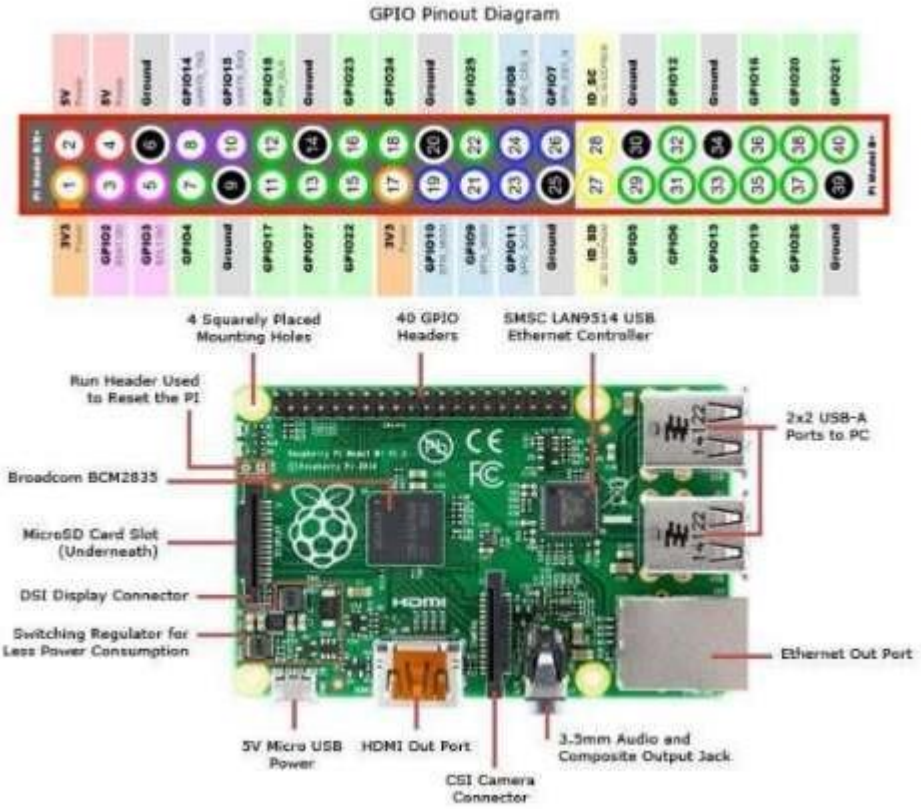
Theory:



You can see that the Beagle bone has a large number of pins. There are two headers. Make sure you orient your Beagle bone n the same direction as mine in the picture, with the five volt plug on the top. In this orientation, the pin header on the left is referred to as “P9” and the pin header on the right is referred to as “P8”. The legend in the diagram above shows the functions, or the possible functions of the various pins. First, we have shaded in red the various 5V, 3.3V, 1.8V and ground pins. Note that VDD_ADC is a 1.8 Volt supply and is used to provide a reference for Analog Read functions. The general purpose GPIO pins have been shaded in green. Note some of these green pins can also be used for UART serial communication. If you want to simulate

analog output, between 0 and 3.3 volts, you can use the PWM pins shaded in purple. The light blue pins can be used as analog in. Please note that the Analog In reads between 0 and 1.8 volts. You should not allow these pins to see higher voltages that 1.8 volts. When using these pins, use pins 32 and 34 as your voltage reference and ground, as pin 32 outputs a handy 1.8 volts. The pins shaded in light orange can be used for I2C. The dark orange pins are primarily used for LCD screen applications.

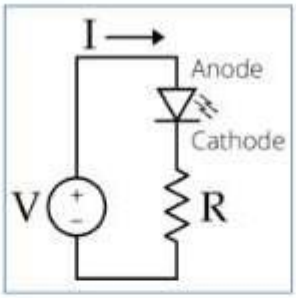
Raspberry Pi 3 Model B Pin Diagram



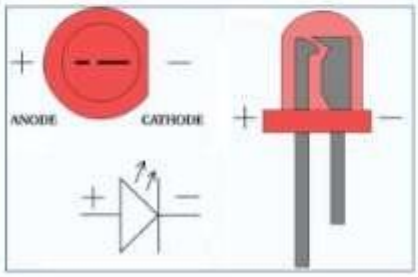
Basic Peripherals

1. LED

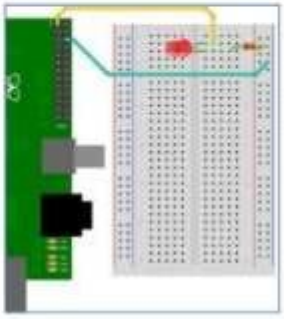
A light-emitting diode (LED) is a semiconductor device that emits light when an electric current is passed through it. Light is produced when the particles that carry the current (known as electrons and holes) combine together within the semiconductor material. Since light is generated within the solid semiconductor material, LEDs are described as solid-state devices. The term solid-state lighting, which also encompasses organic LEDs (OLEDs), distinguishes lighting technology from other sources that use heated filaments (incandescent and tungsten halogen lamps) or gas discharge (fluorescent lamps).



LED Circuit Diagram



LED Symbol



LED Connection with Raspberry pi

Program to Glow LED

- LED has two pins one is positive (long end) and one is negative(small end)
- Connect positive end to GPIO pin p8 10 of beaglebone or pin 17 of raspberry pi using jumper cable
- Connect negative end to any GND pin on beaglebone or raspberry pi as shown in diagram.
- Once connection is done run python code led.py

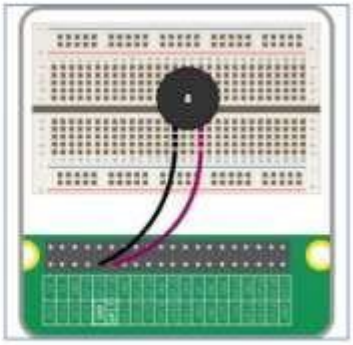
```
# led.py for Beaglebone black where led is connected to pin p8 10

    import Adafruit_BBIO.GPIO as GPIO

    import time

    GPIO.setup("P8_10", GPIO.OUT)

    while True:

        GPIO.output("P8_10", GPIO.HIGH)

        time.sleep(0.5)
```

```
        GPIO.output("P8_10", GPIO.LOW)
        time.sleep(0.5)

#led.py for Raspberry pi where led is connected to pin 17
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(17,GPIO.OUT)
try:
        while True:
        GPIO.output(17,True)
        time.sleep(2)
        GPIO.output(17,False)
        time.sleep(2)
except KeyboardInterrupt:
        GPIO.cleanup()
```

3. Buzzer

A buzzer is an electrical device that is used to make a buzzing sound.

Buzzer



Buzzer Connected to Raspberry pi

Program to ON/OFF Buzzer

- Buzzer has two pins one is positive (long end) and one is negative(small end)
- Connect positive end to GPIO pin 17 of raspberry pi using jumper cable
- Connect negative end to any GND pin of raspberry pi as shown in diagram.
- Once connection is done run python code buzzer.py

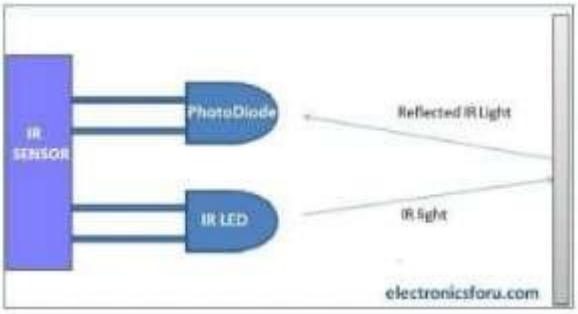
```
#buzzer.py
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)
GPIO.setup(17,GPIO.OUT)
try:
        while True:
                GPIO.output(17,True)
                time.sleep(2)
                GPIO.output(17,False)
                time.sleep(2)
except KeyboardInterrupt:
GPIO.cleanup()
```

2. IR Sensor

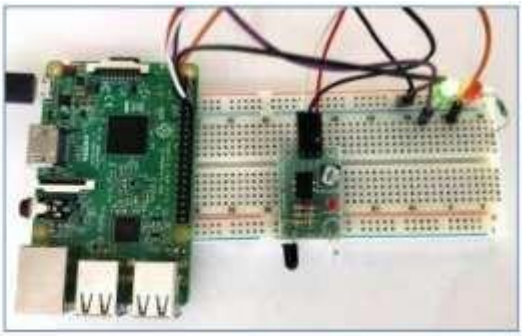
An IR sensor is a device that detects IR radiation falling on it. Proximity sensors (used in touchscreen phones and edge avoiding robots), contrast sensors (used in line following robots) and obstruction counters/sensors (used for counting goods and in burglar alarms) are some applications involving IR sensors.

Principle of Working

An IR sensor consists of two parts, the emitter circuit and the receiver circuit. This is collectively known as a photo-coupler or an optocoupler. The emitter is an IR LED and the detector is an IR photodiode. The IR phototdiode is sensitive to the IR light emitted by an IR LED. The photo-diode‟s resistance and output voltage change in proportion to the IR light received. This is the underlying working principle of the IR sensor. The type of incidence can be direct incidence or indirect incidence. In direct incidence, the IR LED is placed in front of a photodiode with no obstacle in between. In indirect incidence, both the diodes are placed side by side with an opaque object in front of the sensor. The light from the IR LED hits the opaque surface and reflects back to the photodiode.



IR Sensor



IR connected to Raspberry pi

Program for IR Sensor

- Connect IR Sensor on breadboard.

- IR sensor has three pins: VCC, GND and Data.
- Connect VCC and GND pins to any VCC pin and GND pin of raspberry pi
- Connect data pin to 37 GPIO pin of raspberry pi.
- Once connection is done run IR.py code

```
#IR.py
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)
GPIO.setup(37,GPIO.IN)#IR or PIR
try:
        while(True):
                if (GPIO.input(37)==1):
                        print("IR Detected")
                else:
                        print("IR not detected")
GPIO.cleanup()
```

Conclusion: Thus we have studied, connectivity and configuration of Raspberry-Pi with basic peripherals, LEDS, buzzer and IR understanding GPIO and its use in program.

Experiment No. 5

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, 220 ohm resistor etc.

Software Requirement: Arduino IDE

Theory:

This example shows the simplest thing you can do with an Arduino to see physical output: it blinks the on-board LED. This example uses the built-in LED that most Arduino boards have. This LED is connected to a digital pin and its number may vary from board type to board type. To make your life easier, we have a constant that is specified in every board descriptor file. This constant is LED_BUILTIN and allows you to control the built-in LED easily. Here is the correspondence between the constant and the digital pin.

- D13 - 101
- D13 - Due
- D1 - Gemma
- D13 - Intel Edison
- D13 - Intel Galileo Gen2
- D13 - Leonardo and Micro
- D13 - LilyPad
- D13 - LilyPad USB
- D13 - MEGA2560
- D13 - Mini
- D6 - MKR1000
- D13 - Nano
- D13 - Pro
- D13 - Pro Mini

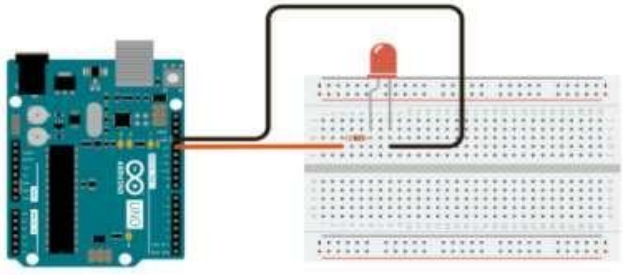
● D13 - UNO

● D13 – Yún

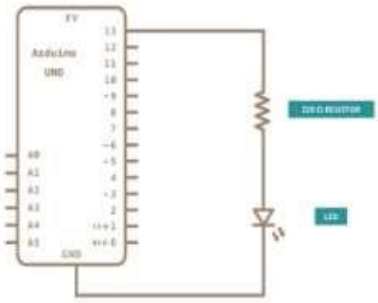
● D13 – Zero

If you want to lit an external LED with this sketch, you need to build this circuit, where you connect one end of the resistor to the digital pin correspondent to the LED_BUILTIN constant. Connect the long leg of the LED (the positive leg, called the anode) to the other end of the resistor. Connect the short leg of the LED (the negative leg, called the cathode) to the GND. In the diagram below we show an UNO board that has D13 as the LED_BUILTIN value. The value of the resistor in series with the LED may be of a different value than 220 ohm; the LED will lit up

up also with values up to 1K ohm.



Schematic



Code

After you build the circuit plug your Arduino board into your computer, start the Arduino Software (IDE) and enter the code below. You may also load it from the menu

File/Examples/01.Basics/Blink . The first thing you do is to initialize LED_BUILTIN pin as an output pin with the line

pinMode(LED_BUILTIN, OUTPUT);

In the main loop, you turn the LED on with the line:

digitalWrite(LED_BUILTIN, HIGH);

This supplies 5 volts to the LED anode. That creates a voltage difference across the pins of the LED, and lights it up. Then you turn it off with the line:

digitalWrite(LED_BUILTIN, LOW);

That takes the LED_BUILTIN pin back to 0 volts, and turns the LED off. In between the on and the off, you want enough time for a person to see the change, so the delay() commands tell the board to do nothing for 1000 milliseconds, or one second. When you use the delay() command, nothing else happens for that amount of time. Once you've understood the basic examples, check out the BlinkWithoutDelay example to learn how to create a delay while doing other things. Once you've understood this example, check out the DigitalReadSerial example to learn how read a switch connected to the board.

Conclusion: - We have successfully studied and implemented Connectivity and configuration of Raspberry-Pi /Beagle board/Arduino circuit with basic peripherals like LEDS

Experiment No. 6

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:

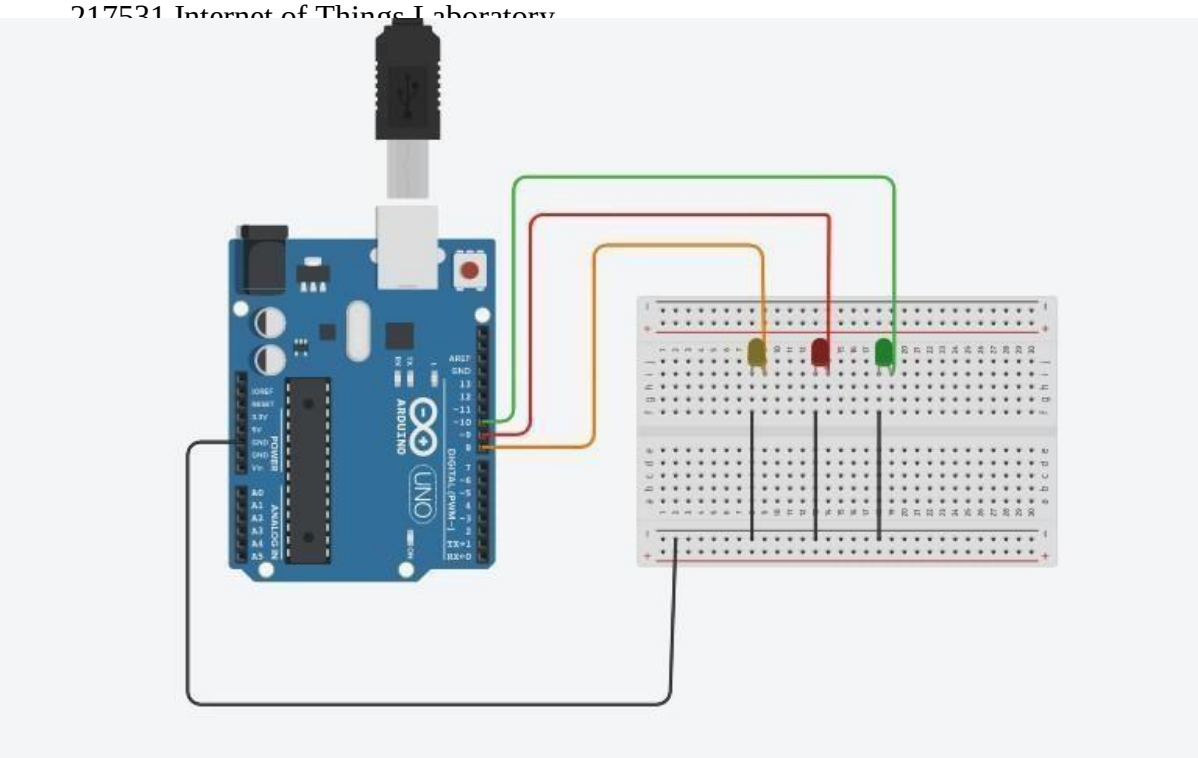
The problem statement is like Arduino traffic light, a fun little project that you can build in under an hour.

Here's how to build your own using an Arduino, and how to change the circuit for an advanced variation.

What You Need to Build an Arduino Traffic Light Controller Apart

from the basic Arduino, you'll need:

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



Code for the Arduino Traffic Light

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:

```
    // 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. Thisrotates 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 isvery 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 HIGHnor 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'regoing 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 toset 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);
}
```

```
int counter = 0;

void setup()

{

Serial.begin(9600);

pinMode(7,OUTPUT);
```

```
pinMode(8,OUTPUT);
pinMode(9,OUTPUT);
}
void loop() {
if(counter == 31)
{
counter=0;
}
if(counter < 31)
{
Serial.println(counter);
}
counter = counter + 1;
delay(100);
if(counter > 0 && counter < 11 )
{
digitalWrite(7,HIGH);
digitalWrite(8,LOW);
digitalWrite(9,LOW);
}
if(counter > 10 && counter < 21 )
{
digitalWrite(7,LOW);
digitalWrite(8,HIGH);
digitalWrite(9,LOW);
```

```
}
if(counter > 20 && counter < 31 )
{
digitalWrite(7,LOW);
digitalWrite(8,LOW);
digitalWrite(9,HIGH);
}
```

Conclusion - We have successfully studied and implemented Connectivity,configuration and control of LED using Arduino circuit under different conditions.

Experiment no. 7

Aim: 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.

Outcome: Connectivity, configuration and serial communication with Arduino.

Hardware Requirement: Arduino, USB cable etc.

Software Requirement: Arduino IDE

Theory:

The problem statement is like Arduino traffic light, a fun little project that you can buildin under an hour. Here's how to build your own using an Arduino, and how to change
the circuit for an advanced variation.

What You Need to Build an Arduino Traffic Light ControllerApart

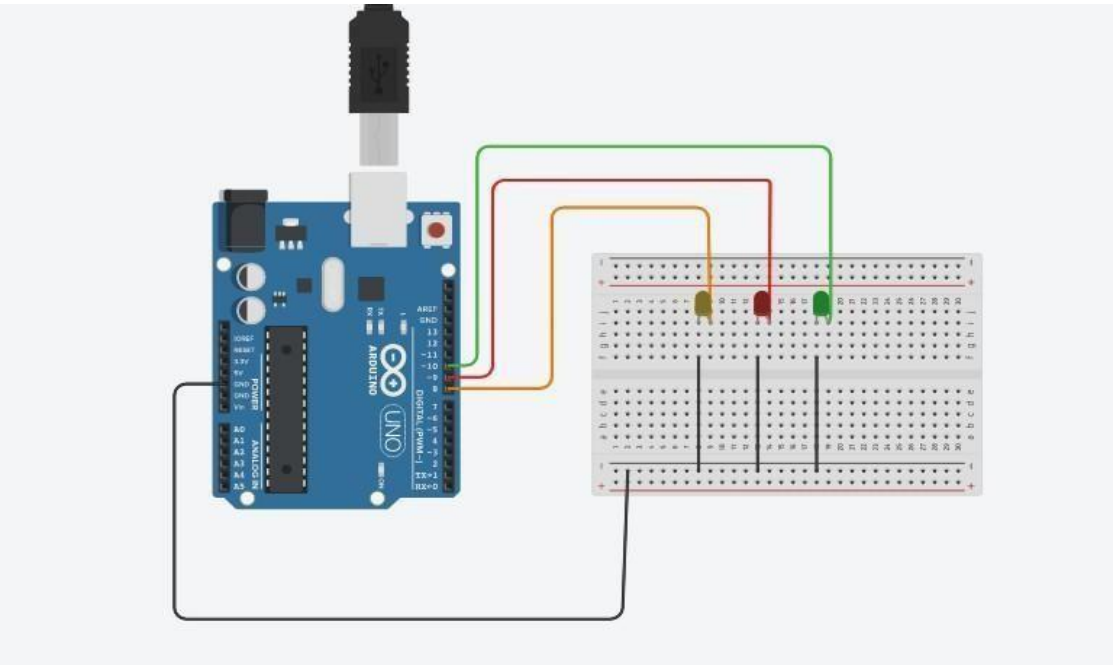
from the basic Arduino, you'll need:

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

Arduino Traffic Light: TheBasics

Let's start small. A basic, single traffic light is a good place to start. Here's the circuit:

Connect the anode (long leg) of each LED to digital pins eight, nine, and ten (via a 220-ohm resistor). Connect the cathodes (short leg) to the Arduino's ground.



Code for the Arduino Traffic Light

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. Thisrotates 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 isvery 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 HIGHnor 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'regoing 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 toset 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);
}
```

Conclusion

Successfully done the Connectivity, configuration and control of LED using Arduino circuit under different conditions.

Experiment no. 8

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

Arduino serial monitor for beginners in electronics. Send and receive data between the serial monitor window on a computer and an Arduino. 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.

Demonstration of the Arduino Serial Monitor for Beginners

Part 2 of this Arduino tutorial for beginners shows how to install the Arduino IDE. In addition, it shows how to load an example sketch to an Arduino. It is necessary to know how to load a ketch to an Arduino board in this part of the tutorial. Therefore, first finish the previous parts of this tutorial before continuing with this part. A sketch loaded to an Arduino board demonstrates how the serial monitor works in the sub-sections that follow.

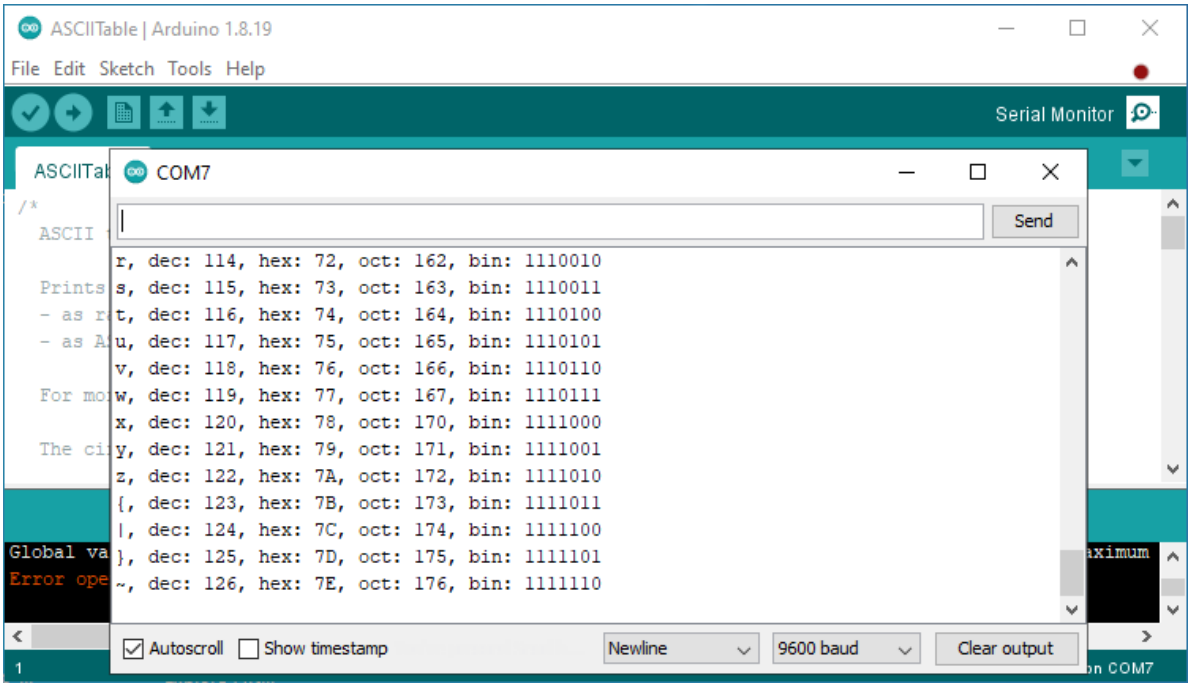
Load an Example Sketch that uses the Serial Monitor to an Arduino Board

Start the Arduino IDE application. Select File → Examples → 04.Communication →SCII Table from the top Arduino IDE menu bar. As a result, the ASCII Table example sketch opens in a new Arduino IDE window. Upload the ASCII Table example sketch to the Arduino Uno or MEGA 2560 board. After the ASCII Table sketch is uploaded, nothing is seen to happen. This is because this example sketch sends text out of the USB port of the Arduino board. Because there is nothing running on the computer to receive this text, nothing is seen.

How to Open the 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 ASCII table 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 ASCII Table sketch to run from the beginning again. The ASCII Table 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 ASCII Table, and most other built-in example sketches, set the Arduino to communicate at 9600 baud. If you're 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.

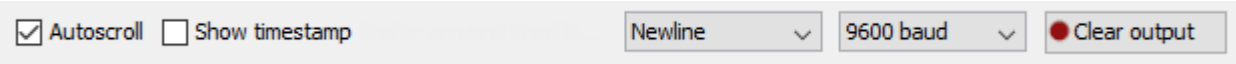
Reset the Arduino Board with the RESET Button

Press and release the RESET button on the Arduino board and the ASCII Table 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 ASCII Table sketch again.



What the ASCII Table 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 ASCII Table 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 andbin.

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);


}
}
```

Conclusion – We have successfully studied and implemented connectivity, configuration and serial communication with Arduino.

Experiment no. 9

Aim: Write a program to control the color of the LED by turning 3 different potentiometers. One will be read for the value of Red, one for the value of Green, and one for the value of Blue.

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:

Control the color of an RGB LED

In this part you will create a color RGB led whose color can be changed through three potentiometers.

Steps to build the circuit:

As a best practice we'll start with the ground (GND). It's very important to make a common ground for all components. To do this we will first plug a black wire (black is convention for GND) between a GND pin of the Arduino and the "minus" line on the breadboard. From this "minus" line, then we will be able to connect all other grounds, which will make things easier to manage.

Add the LED:

Connect the shorter leg of the LED to the ground. You can directly plug this leg into the "minus" line of the breadboard, or add a small black wire. Plug the other (longer) leg of the LED to an independent line on the bread board. From this leg, add a 220 Ohm resistor to yet another line. Add a wire between the other side of the resistor and a PWM-compatible digital pin (so we can control the brightness). Here on Arduino Uno you can choose between pins 3, 5, 6, 9, 10, and 11 – you can recognize PWM compatibility with the "~" next to the pin number.

Add the Potentiometer:

Plug the 3 legs of the potentiometer to 3 different lines on the breadboard.

Connect the extreme left (or right) leg to GND.

Connect the other extreme leg to 5V on the Arduino.

Add a wire between the middle pin and an analog pin.

Arduino code to control LED brightness with the potentiometer

Code:

```
void setup()
{
pinMode(A0,  INPUT);
pinMode(3, OUTPUT);
pinMode(A1,  INPUT);
pinMode(5, OUTPUT);
pinMode(A2,  INPUT);
pinMode(6, OUTPUT);
}
void loop()
{
analogWrite(3, analogRead(A0));
analogWrite(5, analogRead(A1));
analogWrite(6, analogRead(A2));
delay(10); // Delay a little bit to improve simulation performance
}
```

Conclusions: We have successfully studied and implemented connectivity, configuration and control of LED using Arduino circuit under different conditions.

Experiment no. 10

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

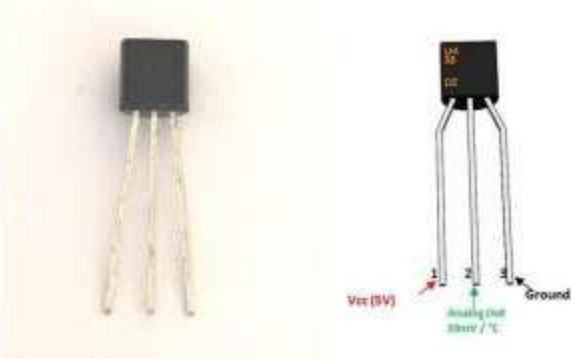
Outcome: Understanding working principle of DHT11, LM35 temperature sensor.

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

Software Requirement: Arduino IDE

Theory:

LM35 Temperature Sensor



LM35 Temperature Sensor PinoutLM35

Sensor Pinout Configuration

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. Canrange from -1V(-55°C) to 6V(150°C)
3	Ground	Connected to ground of circuit

LM35 Sensor Features

- Minimum and Maximum Input Voltage is 35V and -2V respectively. Typically 5V.
- Can measure temperature ranging from -55°C to 150°C

● Output voltage is directly proportional (Linear) to temperature (i.e.) there will be a rise of 10mV (0.01V) for every 1°C rise in temperature.

● ±0.5°C Accuracy

● Drain current is less than 60uA

● Low cost temperature sensor

● Small and hence suitable for remote applications

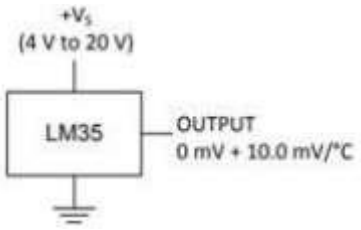
● Available in TO-92, TO-220, TO-CAN and SOIC package

● LM35 Temperature Sensor Equivalent

LM34, DS18B20, DS1620, LM94022

How to use LM35 Temperature Sensor:

LM35 is a precession Integrated circuit Temperature sensor, whose output voltage varies, based on the temperature around it. It is a small and cheap IC which can be used to measure temperature anywhere between - 55°C to 150°C. It can easily be interfaced with any Microcontroller that has ADC function or any development platform like Arduino. Power the IC by applying a regulated voltage like +5V (VS) to the input pin and connected the ground pin to the ground of the circuit. Now, you can measure the temperature in form of voltage as shown below.



If the temperature is 0°C, then the output voltage will also be 0V. There will be rise of 0.01V (10mV) for every degree Celsius rise in temperature. The voltage can converted into temperature using the below formulae.

$$V_{OUT} = 10\ mv/°C \times T$$

where

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

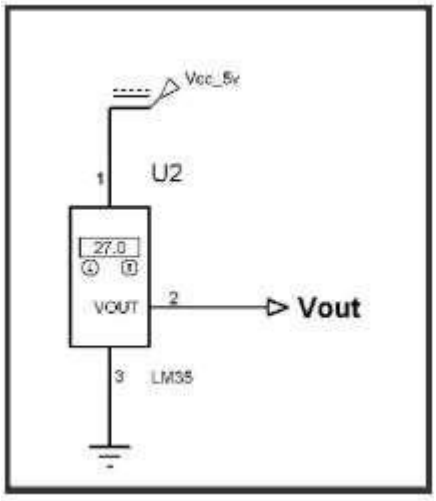
LM35 Temperature Sensor Applications

- Measuring temperature of a particular environment
- Providing thermal shutdown for a circuit/component
- Monitoring Battery Temperature
- Measuring Temperatures for HVAC applications.

How Does LM35 Sensor Work?

Main advantage of LM35 is that it is linear i.e. 10mv/°C which means for every degree rise in temperature the output of LM35 will rise by 10mv. So if the output of LM35 is 220mv/0.22V the temperature will be 22°C. So if room temperature is 32°C then the output of LM35 will be 320mv i.e. 0.32V.

LM35 Interfacing Circuit



DHT11 interfacing with arduino and weather station

DHT11 sensor is used to measure the temperature and humidity. It has a resistive humidity sensing component and a negative temperature coefficient (NTC). An 8 bit MCU is also connected in it which is responsible for its fast response. It is very inexpensive but it gives values of both temperature and humidity at a time.

Specification of DHT11

- It has humidity range from 20 to 90% RH
- It has temperature range from 0 – 50 C
- It has signal transmission range of 20 m

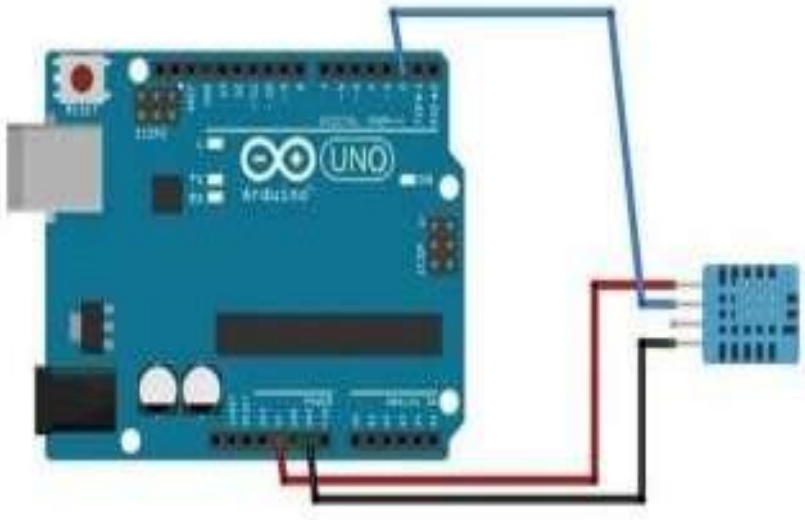
- It is inexpensive
- It has fast response and it is also durable

DHT11 Pin out

- The first pin of the DHT11 is vcc pin.
- The second pin of the DHT is Data pin.
- The third pin is not used.
- The fourth pin of the DHT sensor is ground.

DHT11 interfacing with arduino

First of all connect the ground and the VCC of the DHT11 temperature and humidity sensor to the ground and 5v of the Arduino. Then connect the data pin of the DHT11 sensor to the pin 2 of the Arduino.



Installing the DHT11 Library

To run the following code in Arduino IDE you will first have to install the DHT library in you Arduino directory. Download the zip file from here and place it in your Arduino library folder. The path to Arduino library folder for my computer is Documents/ Arduino/ Libraries Unzip the

downloaded file and place it in this folder. After copying the files, the Arduino library folder should have a new folder named DHT containing the dht.h and dht.cpp. After that copy the following code in the Arduino IDE and upload the code.

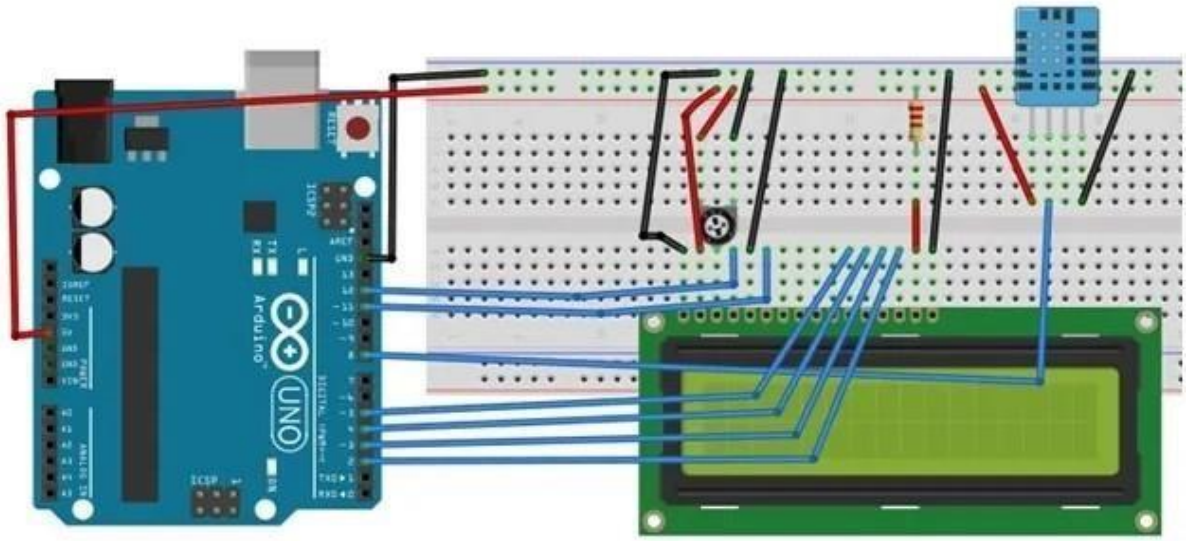
Code of DHT11 interfacing with arduino

```
// Code for DHT11 Temperature and humidity sensor.

#include " DHT.h " // including the library of DHT11 temperature andhumidity sensor

#define DHTPIN 2 // Selecting the pin at which we have connectedDHT11

#define DHTTYPE DHT11 // Selecting the type of DHT sensorsDHT dht ( DHTPIN,

DHTTYPE ) ;

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

dht.begin ( ) ; // The sensor will start working

}

void loop ( ) {

// Reading temperature or humidity may take about 2 seconds because itis a very slow sensor.

float humidity = dht.readHumidity ( ) ; // Declaring h a variable andstoring the humidity in it.

float temp = dht.readTemperature ( ) ; // Declaring t a variable andstoring the temperature in it.

// Checking if the output is correct. If these are NaN, then there issomething in it.

if ( isnan ( t ) || isnan ( h ) ) { Serial.println ( " Sensor not working " ) ;

}

else

{

Serial.print ( " Temp is " ) ;

Serial.print ( temp ) ; // Printing the temperature ondisplay.
```

```
Serial.println ( " *C " ) ; // Printing “ *C ” on display.Serial.print ( "

Humidity in % is : " ) ;

Serial.print ( humidity ) ; // Printing the humidity on displaySerial.print ( " %
\t " ) ; // Printing “%” on display

}

}
```

Weather Station using DHT11 and arduino

In this example we will make a weather station that will sense the humidity and temperature and will show it on the lcd attached to the Arduino. Make the circuit as shown in the diagram. The resistor in the circuit will make the black light darker. We have used the 220 ohm resistor but you can use any resistor having value near to that. The potentiometer we used in the circuit is used to set the screen contrast. We have used the 10 K ohm value but you can choose any value relative to that one.



Components Required

● Arduino Uno (you can use any)

- 16 x 2 LCD
- DHT11 Temperature and humidity sensor
- 10 K ohm potentiometer
- 220 ohm resistor

Code of weather station using arduino and DHT11

// This code is for the weather station using the DHT11 humidity and temperaturesensor.

//Install the library ofthe DHT before uploading the code in the Arduino IDE#include < dht.h > // including the

DHT library

#include <LiquidCrystal.h > // including the LCD library LiquidCrystal lcd (12, 11, 5, 4,

3, 2) ; // initializing the lcd pinsdht DHT ; // declaring dht a variable

#define DHT11_PIN 8 // initializing pin 8 for dhtvoid setup ( ) {

lcd.begin (16, 2) ; //starting the 16 x 2 lcd

}

void loop ( )

{

int chk = DHT.read11(DHT11_PIN) ; //Checking that either the dht isworking or not

lcd.setCursor (0, 0) ; // starting the cursor from top left

lcd.print (" Temperature is: ") ; // printing the “ Temperature is: ” onthe lcd

lcd.print(DHT.temperature); // printing the temperature on the lcdlcd.print ((char)

223) ;

lcd.print (" C ") ; // Printing “ C“ on the displaylcd.setCursor (0 ,

1);

lcd.print (" Humidity is: ") ; // printing “ humidity is: ” on thedisplay

lcd.print(DHT.humidity) ; // printing humidity on the displaylcd.print (" % ") ;

```
// printing " % " on display
delay ( 1000 ) ; // Giving delay of 1 second.
}
```

Conclusion: We have successfully studied and implemented working principle of DHT11, LM35 temperature sensor.

Experiment no - 11

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

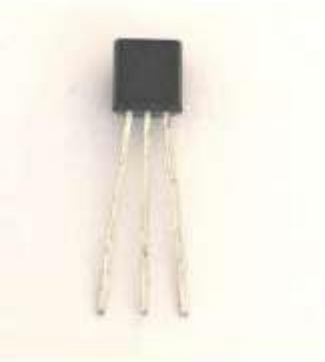
Outcome: Understanding working principle of DHT11, LM35 temperature sensor, Relationship between different temperature scales

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

Software Requirement: Arduino IDE

Theory:

LM35 Temperature Sensor



LM35 Temperature Sensor PinoutLM35

Sensor Pinout Configuration

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. Canrange from -1V(-55°C) to 6V(150°C)
3	Ground	Connected to ground of circuit

LM35 Sensor Features

- Minimum and Maximum Input Voltage is 35V and -2V respectively. Typically 5V.
- Can measure temperature ranging from -55°C to 150°C
- Output voltage is directly proportional (Linear) to temperature (i.e.) there will be a rise of 10mV (0.01V) for every 1°C rise in temperature.
- ±0.5°C Accuracy
- Drain current is less than 60uA
- Low cost temperature sensor
- Small and hence suitable for remote applications
- Available in TO-92, TO-220, TO-CAN and SOIC package

Temperature Scales:

Thermometers measure temperature according to well-defined scales of measurement. The three most common temperature scales are the Fahrenheit, Celsius, and Kelvin scales.

Celsius Scale & Fahrenheit Scale:

The Celsius scale has a freezing point of water as 0ºC and the boiling point of water as 100ºC. On the Fahrenheit scale, the freezing point of water is at 32ºF and the boiling point is at 212ºF. The temperature difference of one degree Celsius is greater than a temperature difference of one degree Fahrenheit. One degree on the Celsius scale is 1.8 times larger than one degree on the Fahrenheit scale 180/100=9/5.

Kelvin scale:

Kelvin scale is the most commonly used temperature scale in science. It is an absolute temperature scale defined to have 0 K at the lowest possible temperature, called absolute zero. The freezing and boiling points of water on this scale are 273.15 K and 373.15 K, respectively. Unlike other temperature scales, the Kelvin scale is an absolute scale. It is extensively used in scientific work. The Kelvin temperature scale possesses a true zero with no negative temperatures. It is the lowest temperature theoretically achievable and is the temperature at which the particles in a perfect crystal would become motionless.

Relationship between Different Temperature Scales:

The relationship between three temperature scales is given in the table below:

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	$T_{F^0} = \frac{9}{5}(T(K) - 273.15) + 32$

Code:

```
int val;
int tempPin = 0; float tfmax; float tfmin
= 100; void setup()
{
Serial.begin(9600);
}
void loop()
{
val = analogRead(tempPin); float mv = (
val/1024.0)*500; float cel = mv;
```

```
float farh = (cel*9)/5 + 32;
// Serial.print("TEMPRATURE = ");
// Serial.print(cel);
// Serial.print("*C");
// Serial.println();
// delay(1000);

if (farh > tfmax)
{
tfmax = farh;
}
if (farh < tfmin)
{
tfmin = farh;
}
```

Conclusion: We have successfully studied and implemented the working principle of DHT11, LM35 temperature sensor, Relationship between different temperature scales.

Experiment no - 12

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

Outcome: Understanding working principle of DHT11 temperature sensor, Blynk IOT Platform

Hardware Requirement: Arduino (Node MCU), LM35, DHT11, etc

Software Requirement: Arduino IDE

Theory:

1. Introduction:

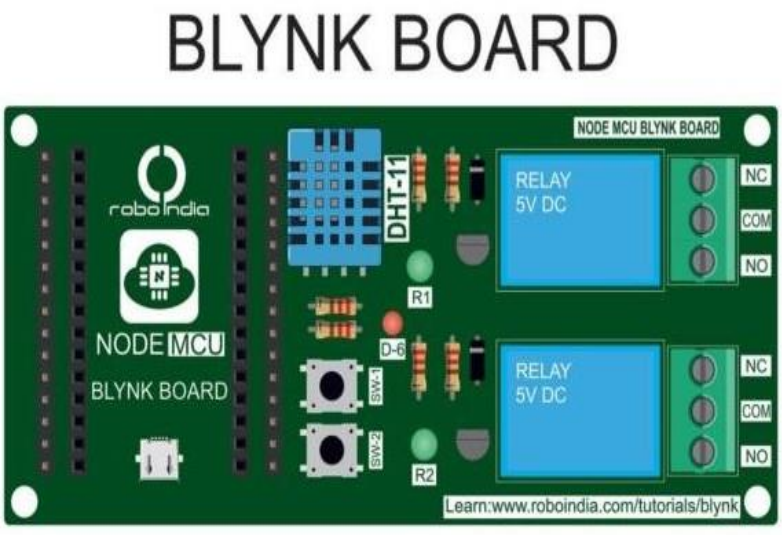
In this project using an esp8266, to show the temperature and humidity DHT11 sensor on your Smartphone or tablet. The NodeMCU collects the temperature and humidity from DHT11 sensor and sends it to Blynk app every second.

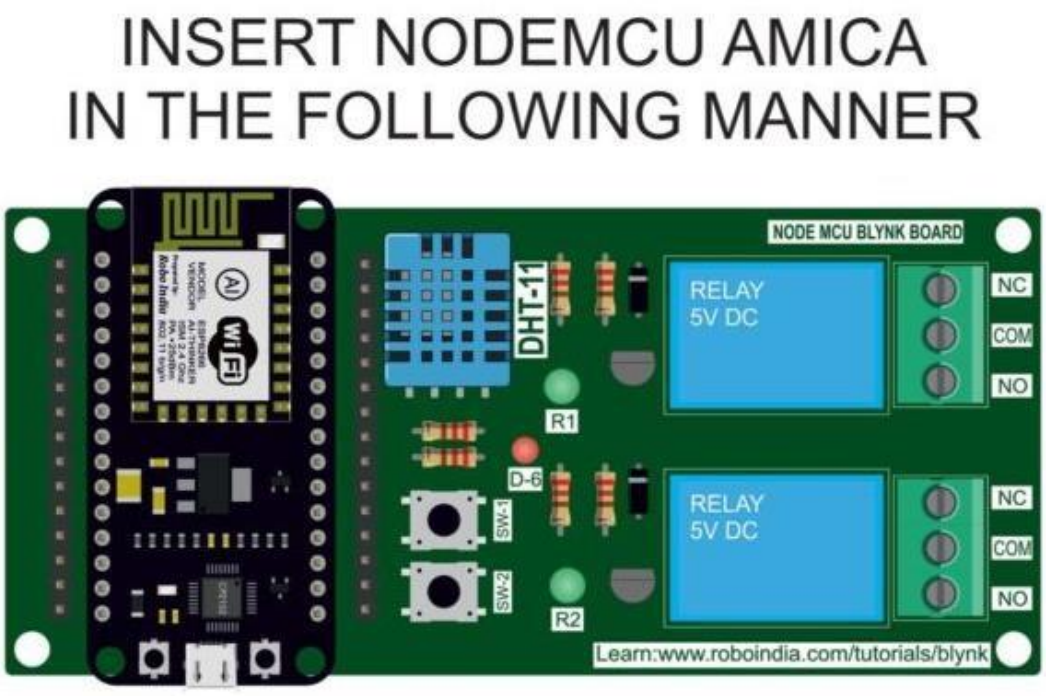
1.2 Hardware required

Blynk Board and NodeMCU is used in this example. Inset NodeMCU to the Blynk board as shown in the image ahead then connect NodeMCU to PC or Laptop through USB cable.

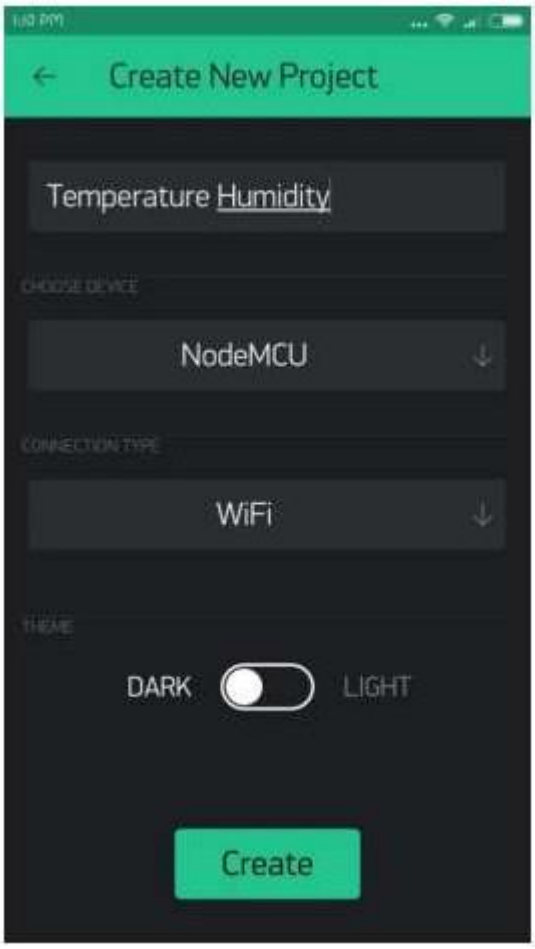
2. On Blynk App

You need to perform following steps on Blynk App.

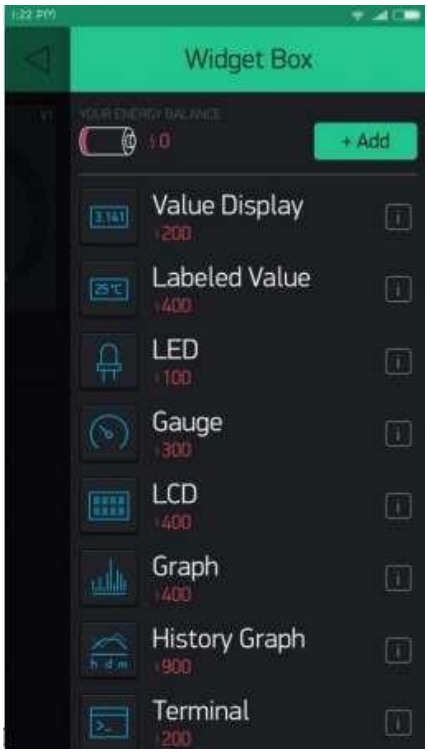



INSERT NODEMCU AMICA
IN THE FOLLOWING MANNER
NODE MCU BLYNK BOARD
DHT-11
RELAY
5V DC
NC
COM
NO
R1
D-6
SW-1
SW-2
R2
RELAY
5V DC
NC
COM
NO
Learn:www.roboindia.com/tutorials/blynk

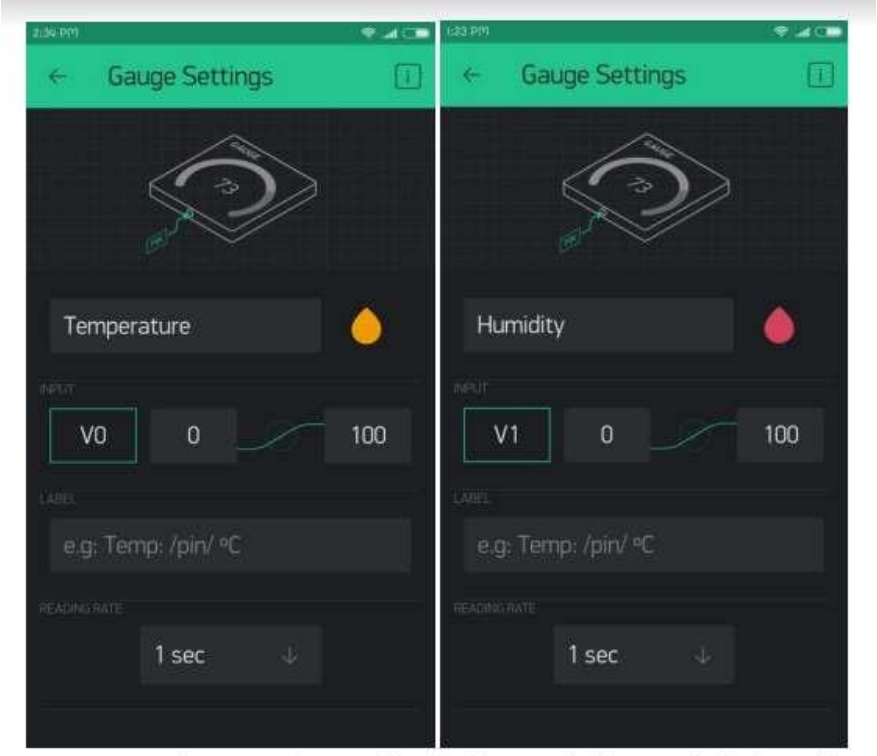
2.1 Create a New Project in BLYNK app. Write Project name Temperature Humidity and Select Node MCU from drop down.



2.2 An AUTH token will be sent to your registered email, note this down. Tap on the screen and add a 2 Gauges.



2.3 Tap on the Widget and select the respective Virtual pins for temperature and humidity data (V0 for temperature and V1 for humidity).



Note: Make sure to setup Reading rate as '1' second for all Widgets. And add gauges for both Humidity and Temperature.

3. Code the NodeMCU with the following code.

Before uploading, make sure to paste your authorization token into the auth [] variable. Also make sure to load your Wifi network settings into the Blynk.begin(auth, "ssid", "pass") function.

Following code may be downloaded from here.

```
// Robo India Tutorial

// Digital Output on LED

// Hardware: NodeMCU Blynk Board#define

BLYNK_PRINT Serial

#include <ESP8266WiFi.h> #include

<BlynkSimpleEsp8266.h>

#include "DHT.h" // including the library of DHT11 temperatureand humidity sensor

#include <SimpleTimer.h> //including the library of SimpleTimer#define

DHTTYPE DHT11 // DHT 11

#define dht_dpin 14

DHT dht(dht_dpin, DHTTYPE);SimpleTimer timer;

char auth[] = "Your Auth. Key"; // You should get Auth Token in

the Blynk App.(nut

icon).


// Go to the Project Settings


char ssid[] = "Your Wifi Network name"; // Your WiFi credentials.

char pass[] = "Password of your network"; // Set password to "" for opennetworks.

float t; // Declare the variables
```

```
float h;


void setup()
{
Serial.begin(9600);// Debug consoleBlynk.begin(auth, ssid,
pass); dht.begin();
timer.setInterval(2000, sendUptime);
}
void sendUptime()
{
float h = dht.readHumidity(); float t =
dht.readTemperature();
Serial.println("Humidity and temperature\n\n");Serial.print("Current
humidity = "); Serial.print(h);
Serial.print("% "); Serial.print("temperature = ");
Serial.print(t); Blynk.virtualWrite(V0, t);
Blynk.virtualWrite(V1, h);
}
void loop()
{
Blynk.run();
timer.run();
}
```

4. Output

After Uploading the Ardunio code IDE. Press the play button on blynk app to show the output.

Conclusion – We have successfully studied and implemented the working principle of DHT11 temperature sensor, BlynkIOT Platform.