LAB MANUAL

Sub: Internet Of Things Lab **Course code: 210248**
Class: Second Year (AIDS) **Pattern: 2019**

Internet of Things Lab (217531) Examination Scheme :**Term Work 50 Marks**
Practical of 25 marks

Table of Contents

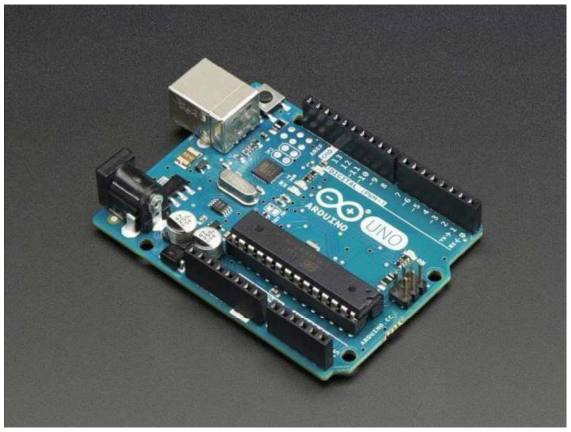
Assignment A1

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

Theory:

1. **Arduino**

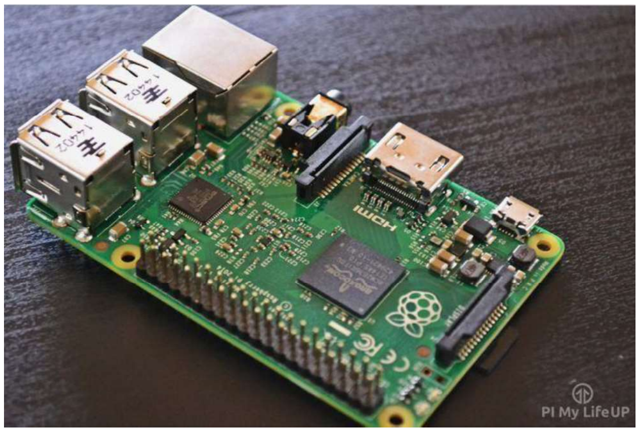
   Arduino is an open source computer hardware and software company, project, and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices and interactive objects that can sense and control objects in the physical and digital world. 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.



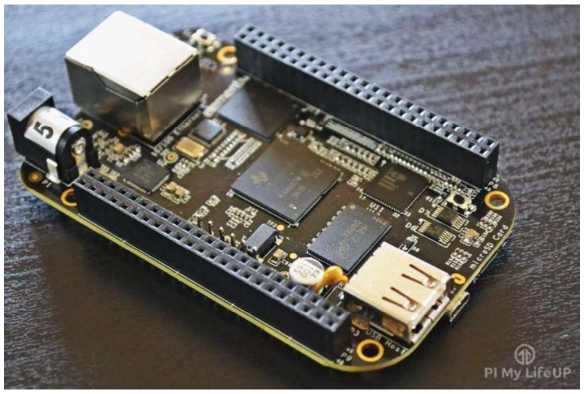
Arduino

2. **Raspberry Pi**

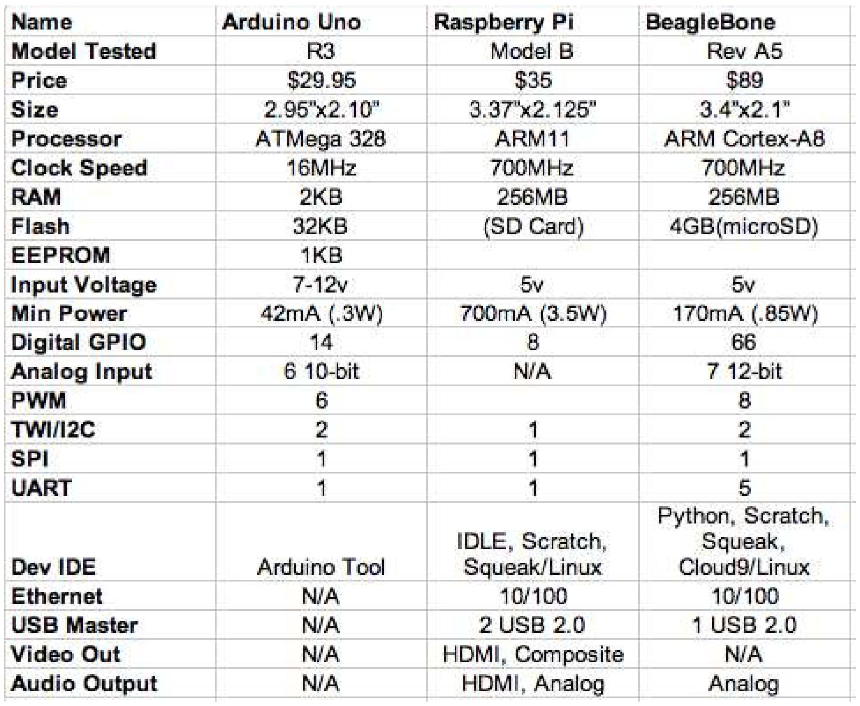
   The **Raspberry Pi** is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and in developing countries. The original model became far more popular than anticipated, selling outside its target market for uses such as robotics. It does not include peripherals (such as keyboards, mice and cases). However, some accessories have been included in several official and unofficial bundles.



3. **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 element 14.The Beagle Board was also designed with open source software development in mind, and as a way of demonstrating the Texas Instrument's OMAP3530system-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.



Beagle Board



Name	Arduino Uno	Raspberry Pi	BeagleBone
Model Tested	R3	Model B	Rev A5
Price	$29.95	$35	$89
Size	2.95"x2.10"	3.37"x2.125"	3.4"x2.1"
Processor	ATMega 328	ARM11	ARM Cortex-A8
Clock Speed	16MHz	700MHz	700MHz
RAM	2KB	256MB	256MB
Flash	32KB	(SD Card)	4GB(microSD)
EEPROM	1KB		
Input Voltage	7-12v	5v	5v
Min Power	42mA (.3W)	700mA (3.5W)	170mA (.85W)
Digital GPIO	14	8	66
Analog Input	6 10-bit	N/A	7 12-bit
PWM	6		8
TWI/I2C	2	1	2
SPI	1	1	1
UART	1	1	5
Dev IDE	Arduino Tool	IDLE, Scratch, Squeak/Linux	Python, Scratch, Squeak, Cloud9/Linux
Ethernet	N/A	10/100	10/100
USB Master	N/A	2 USB 2.0	1 USB 2.0
Video Out	N/A	HDMI, Composite	N/A
Audio Output	N/A	HDMI, Analog	Analog

Comparison of Embedded board

Conclusion: Thus we have studied Raspberry-Pi, Beagleboard, Arduino and other microcontroller.

Assignment A2

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

Theory: About the Different OS used in Aurdino and Rasberry Pi(NOOBS) and beagle board(Debian)

Procedure for installation of OS

A. OS installation Steps on Raspberry-Pi
1. Go to rasberry.org/download and download NOOBS offline installer which consist of Raspberry-Pi desktop pc files.
2. Extract zip file.
3. Format 4GB SD card using gparted.
4. Copy the contents of NOOBS folder into SD card.
5. Put SD card on Raspberry-Pi

B. OS installation Steps on Beagle board
1. Go to http://beagleboard.org/latest-images/ and download latest Debian image files
2. Extract image file
3. Format 4GB SD card using SD card Formatter
4. Using win32 Disk Manager software write ISO image file to SD Card.
5. Put SD card on Beagle board

Conclusion: Thus we have studied different operating systems for Raspberry-Pi /Beagle board and understanding the process of OS installation on Raspberry-Pi /Beagle board.

Assignment A3

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

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 smart phones, tablets or with in memory devices.

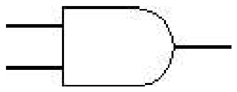
In a circuit, logic gates will make decisions based on a combination of digital signals coming from its inputs. Most logic gates have two inputs and one output. Logic gates are based on Boolean algebra. At any given moment, every terminal is in one of the two binary conditions, *false* or *true*. False 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,andinanother,itison--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

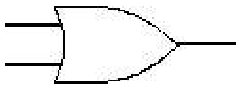


inputsoneANDtwoare1.

Input1	**Input2**	**Output**
	1	
1		
1	1	1

OR gate

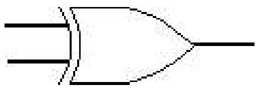
The *OR gate* gets its name from the fact that it behaves after the fashion of the logical inclusive "or." The output is "true" if either or both of the inputs are "true." If both inputs are "false," then the output is "false." In other words, for the output to be 1, at least input one ORtwomustbe1.



Input1	Input2	Outp ut
	1	1
1		1
1	1	1

XOR gate

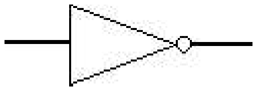
The *XOR* (*exclusive-OR*) *gate* acts in the same way as the logical "either/or." The output is "true" if either, but not both, of the inputs are "true." The output is "false" if both inputs are "false" or if both inputs are "true." Another way of looking at this circuit is to observe thattheoutputis1iftheinputsaredifferent,but0iftheinputsarethesame.



Input1	Input2	Outp ut
	1	1
1		1
1	1	

Inverter or NOT gate

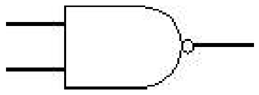
A logical *inverter*, sometimes called a *NOT gate* to differentiate it from other types of electronic inverter devices, has only one input. It reverses the logic state. If the input is 1,thentheoutputis0.Iftheinputis0,thentheoutputis1.



Inpu t	Outp ut
1	
	1

NAND gate

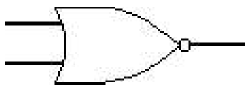
The *NAND gate* operates as an AND gate followed by a NOT gate. It acts in the manner of the logical operation "and" followed by negation. The output is "false" if both inputs are "true." Otherwise, the output is "true."



Input1	**Input2**	**Output**
		1
	1	1
1		1
1	1	

NOR gate

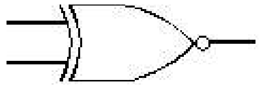
The *NOR gate* is a combination OR gate followed by an inverter. Its output is "true" if both inputs are "false. "Otherwise, the output is "false."



Input1	**Input2**	**Output**
		1
	1	
1		
1	1	

XNOR gate

The *XNOR (exclusive-NOR) gate* is a combination XOR gate followed by an inverter. Its outputis"true"iftheinputsarethesame,and"false"iftheinputsaredifferent.



Input1	Input2	Output
		1
	1	
1		
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

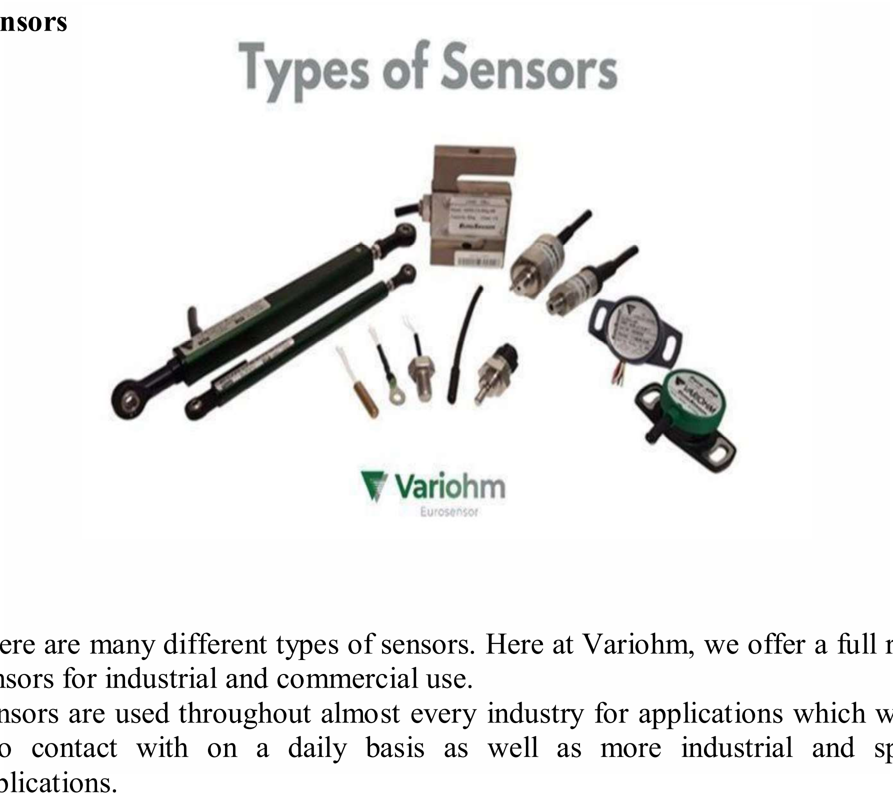
High or low binary conditions are represented by different voltage levels. The logic state of a terminal can, and generally does, often change as the circuit processes data. In most logic gates, the low state is approximately zero volts (0 V), while the high state is approximately five volts positive(+5V).

Logic gates can be made of resistors and transistors or diodes. A resistor can commonly beusedasapull-uporpull-downresistor.Pull-upandpull-downresistorsareusedwhentheareanyunusedlogicgateinputstoconnecttoalogiclevel1or0.Thispreventsanyfalse

switching of the gate. Pull-up resistors are connected to Vcc (+5V), and pull-down resistors are connected to ground(0V).

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 bemarkedasa4000seriesofchips.

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, themeasurementisprocessedandtheoutputcausestheboilertoswitchoff.

- 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 in to subcategories for example, with in 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

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-XPPuck–2partpuckandmagnetdesign
- 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 are a 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 Baror PSI.
Many of our pressure sensors come from our trusted suppliers and we also have our own range of;

Types of Sensors–Load Cells and Force Sensors

Load Cells are available in a wide variety of shapes and sizes. They are used to measure various types of force, the main one being weight. Load cells are used in all types of scales; from bathroom scales to counting scales, industrial scales, truck scales, hopper scales and everything in between.
Most load cells use internal strain gauges to monitor force based on the level of distortion on the strain gauge.
Our load cells come from our trusted suppliers. And can be seen on our website. Further reading on Load Cells

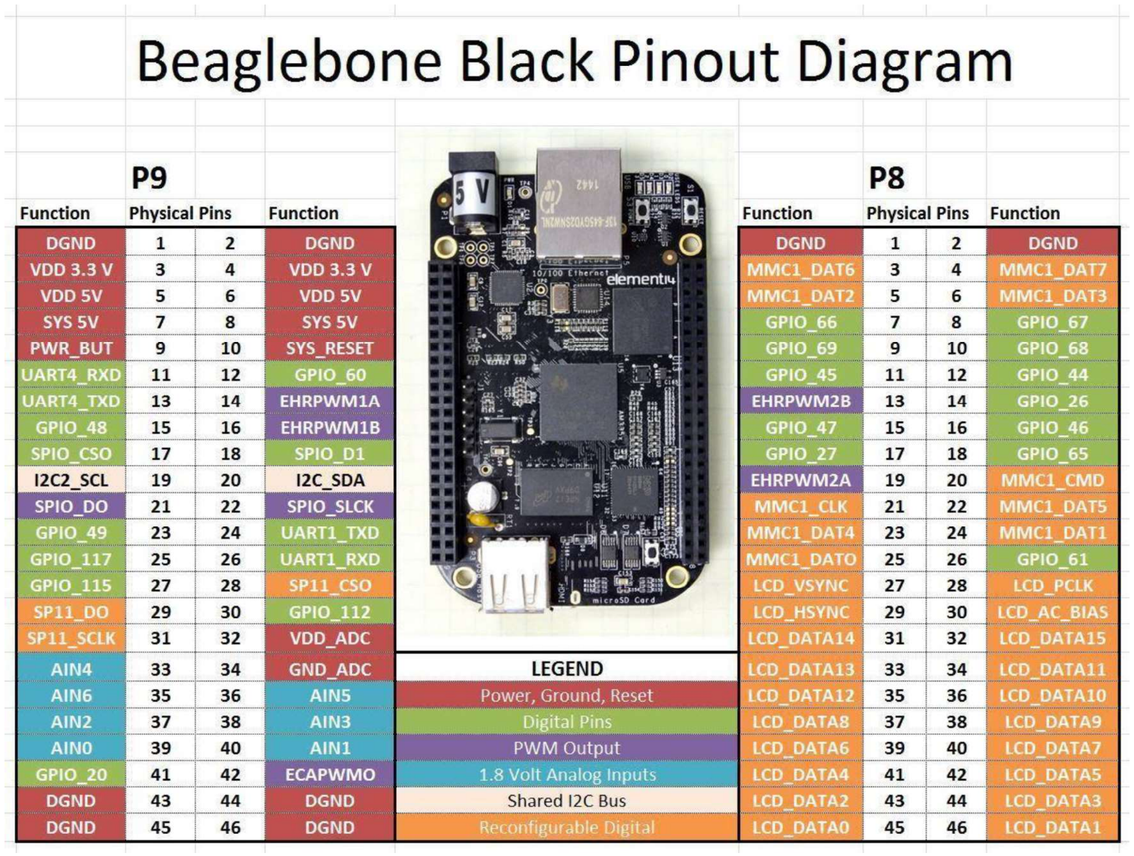
Conclusion:

Assignment A4

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.

Theory:



Beaglebone Black Pinout Diagram

P9

Function	Physical Pins		Function
DGND	1	2	DGND
VDD 3.3 V	3	4	VDD 3.3 V
VDD 5V	5	6	VDD 5V
SYS 5V	7	8	SYS 5V
PWR_BUT	9	10	SYS_RESET
UART4_RXD	11	12	GPIO_60
UART4_TXD	13	14	EHRPWM1A
GPIO_48	15	16	EHRPWM1B
SPIO_CSO	17	18	SPIO_D1
I2C2_SCL	19	20	I2C_SDA
SPIO_DO	21	22	SPIO_SLCK
GPIO_49	23	24	UART1_TXD
GPIO_117	25	26	UART1_RXD
GPIO_115	27	28	SP11_CSO
SP11_DO	29	30	GPIO_112
SP11_SCLK	31	32	VDD_ADC
AIN4	33	34	GND_ADC
AIN6	35	36	AIN5
AIN2	37	38	AIN3
AIN0	39	40	AIN1
GPIO_20	41	42	ECAPWMO
DGND	43	44	DGND
DGND	45	46	DGND

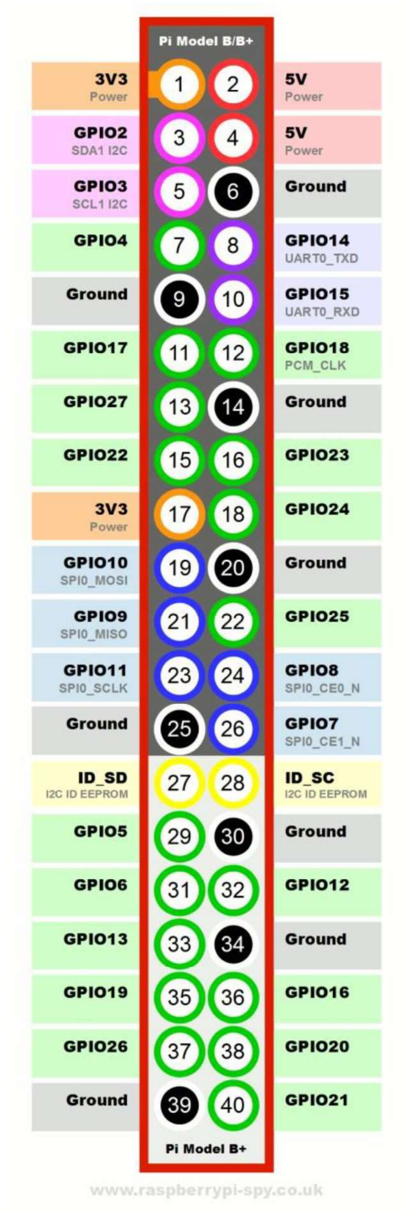
P8

Function	Physical Pins		Function
DGND	1	2	DGND
MMC1_DAT6	3	4	MMC1_DAT7
MMC1_DAT2	5	6	MMC1_DAT3
GPIO_66	7	8	GPIO_67
GPIO_69	9	10	GPIO_68
GPIO_45	11	12	GPIO_44
EHRPWM2B	13	14	GPIO_26
GPIO_47	15	16	GPIO_46
GPIO_27	17	18	GPIO_65
EHRPWM2A	19	20	MMC1_CMD
MMC1_CLK	21	22	MMC1_DAT5
MMC1_DAT4	23	24	MMC1_DAT1
MMC1_DATO	25	26	GPIO_61
LCD_VSYNC	27	28	LCD_PCLK
LCD_HSYNC	29	30	LCD_AC_BIAS
LCD_DATA14	31	32	LCD_DATA15
LCD_DATA13	33	34	LCD_DATA11
LCD_DATA12	35	36	LCD_DATA10
LCD_DATA8	37	38	LCD_DATA9
LCD_DATA6	39	40	LCD_DATA7
LCD_DATA4	41	42	LCD_DATA5
LCD_DATA2	43	44	LCD_DATA3
LCD_DATA0	45	46	LCD_DATA1

LEGEND
Power, Ground, Reset
Digital Pins
PWM Output
1.8 Volt Analog Inputs
Shared I2C Bus
Reconfigurable Digital

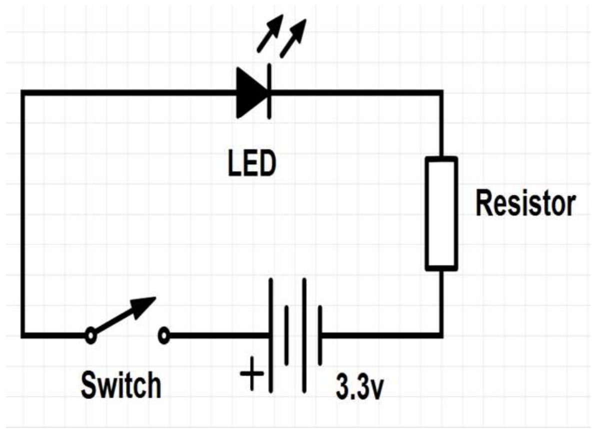
Raspberry Pi 3 Model B Pin Diagram

You can see that the Beagle bone has a large number of pins. There are two headers. Make sure you orient your Beagle bone 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".Thelegend 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.

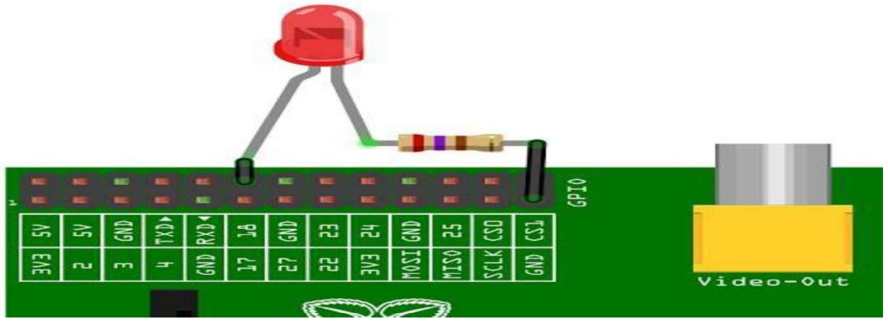

Pi Model B/B+
3V3 Power 1 2 5V Power
GPIO2 SDA1 I2C 3 4 5V Power
GPIO3 SCL1 I2C 5 6 Ground
GPIO4 7 8 GPIO14 UART0_TXD
Ground 9 10 GPIO15 UART0_RXD
GPIO17 11 12 GPIO18 PCM_CLK
GPIO27 13 14 Ground
GPIO22 15 16 GPIO23
3V3 Power 17 18 GPIO24
GPIO10 SPI0_MOSI 19 20 Ground
GPIO9 SPI0_MISO 21 22 GPIO25
GPIO11 SPI0_SCLK 23 24 GPIO8 SPI0_CE0_N
Ground 25 26 GPIO7 SPI0_CE1_N
ID_SD I2C ID EEPROM 27 28 ID_SC I2C ID EEPROM
GPIO5 29 30 Ground
GPIO6 31 32 GPIO12
GPIO13 33 34 Ground
GPIO19 35 36 GPIO16
GPIO26 37 38 GPIO20
Ground 39 40 GPIO21
Pi Model B+
www.raspberrypi-spy.co.uk

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 with in the solid semiconductor material, LEDs are described as solid- state devices. The term solid- state lighting, which also encompasses organic LEDs(OLEDs), distinguishes this lighting technology from other sources that use heated filaments (incandescent and tungsten halogen lamps) or gas discharge (fluorescent lamps)



LED connecting circuit



Interfacing of LED with RPi

Program for blinking LED

```
#led.pyforRaspberrypiwhereledisconnectedtopin3

import  RPi.GPIO  as  GPIO
import                   time
GPIO.setmode(GPIO.BOA
RD)
GPIO.setup(3,GPIO.OUT)
try:
        whileTrue:
                GPIO.output(3,True)
                time.sleep(2)
                GPIO.output(3,False)
                time.sleep(2)
exceptKeyboardInterrupt:
        GPIO.cleanup()
```

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

Assignment B5

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

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

Software Requirement: Arduino IDE

Theory:

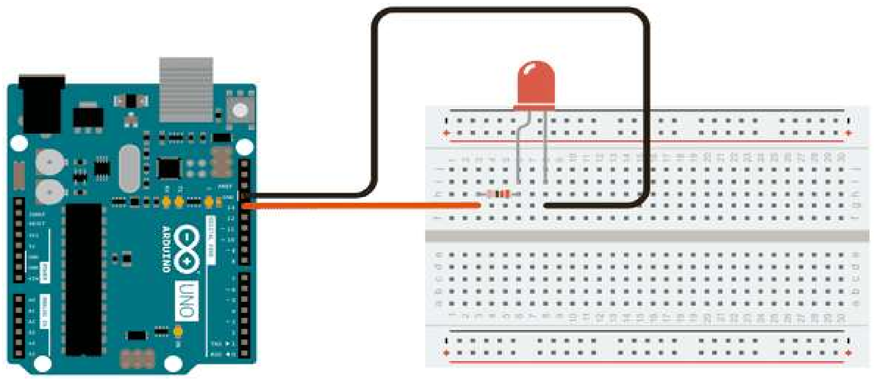
This example shows the simplest thing you can do with an Arduino to see physical output: I t blinks the on-board LED.

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 haveaconstantthatisspecifiedineveryboarddescriptorfile.Thisconstantis*LED_BUILTIN*and allows you to control the built-in LED easily. Here is the correspondence between the constant and the digital pin.

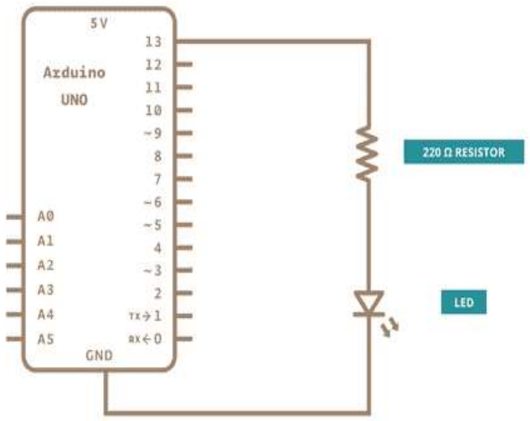
- D13-101
- D13-Due
- D1-Gemma
- D13-IntelEdison
- D13-IntelGalileo Gen2
- D13-LeonardoandMicro
- D13- LilyPad
- D13-LilyPadUSB
- D13- MEGA2560
- D13-Mini
- D6-MKR1000
- D13-Nano
- D13-Pro
- D13-ProMini
- D13-UNO
- D13-Yún
- D13-Zero

If you want to glow 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 valuethan220ohm; the LED will lit 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.ThefirstthingyoudoistoinitializeLED_BUILTINpinasan output pin with the line

Pin Mode (LED_BUILTIN, OUTPUT);

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

Digital Write (LED_BUILTIN, HIGH);

This supplies 5volts 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:

Digital Write (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 Blink Without Delay example to learn how to create a delay while doing other things. Once you've understood this example, check out the Digital Read Serial example to learn how read a switch connected to the board.

Program:

Should be written by student

Conclusion:-

Assignment No. B6

Aim: Create a program that illuminates the green LED if the counter is less than100, illuminates the yellow LED if the counter is between 101 and 200 and illuminates the red LED if the counter is greater than200

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

Hardware Requirement: Arduino, LED, 330 ohm resistor etc.

Software Requirement: Arduino IDE

Theory:

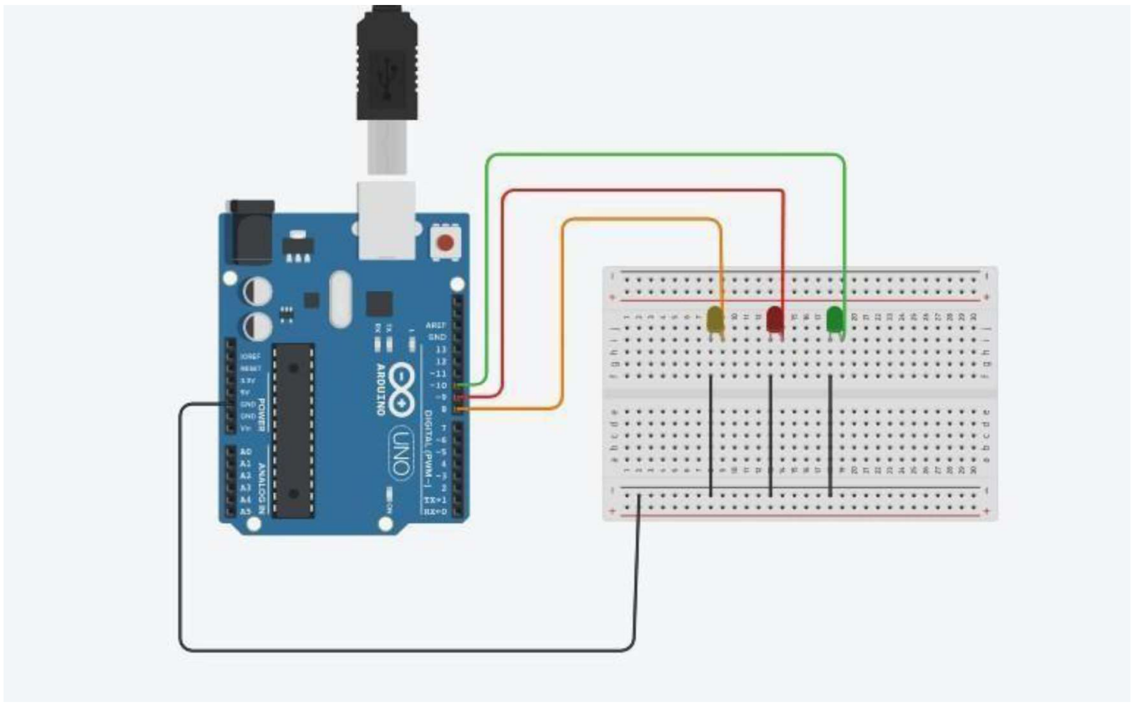
The problem statement is like Arduino traffic light, a fun little project that you can build in under an hour. Here show to build your own using an Arduino, and how to change
The circuit for an advanced variation.

What You Need to Build an Arduino Traffic Light Controller
Apart from the basic Arduino, you'll need:

- 1x10k-ohmresistor
- 3x470-ohmresistors
- A bread board
- Connecting wires
- Red, yellow and green LEDs

Arduino Traffic Light: The Basics
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(viaa470-ohmresistor).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 a number. Start a new Arduino project, and begin with these lines:

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 byname instead:

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:

Upload this code to your Arduino, and run (make sure to select the correct board andportfromthe**Tools**>Boardand**Tools**>**Port**menus).Youshouldhaveaworkingtrafficlightthat changes every 15seconds, like this (sped up):

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

The **change Lights** function consists off our distinct steps:

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

Thesefourstepsreplicatetheprocessusedinrealtrafficlights.Foreachstep,thecodeisvery 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, 1or0, HIGH or LOW). You might assume that a pushbutton switch that isn't actually pressed would be in a LOW state, but in fact, it's said to be 'floating', because no current gets drawn at all.

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

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

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

PROGRAM:

CONCLUSION:

Assignment NO. B7

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 control of LED using Arduino circuit under different conditions.

- **Hardware Requirement:** Arduino, LED,470ohmresistoretc.
- **Software Requirement**: Arduino IDE

- **Theory:**

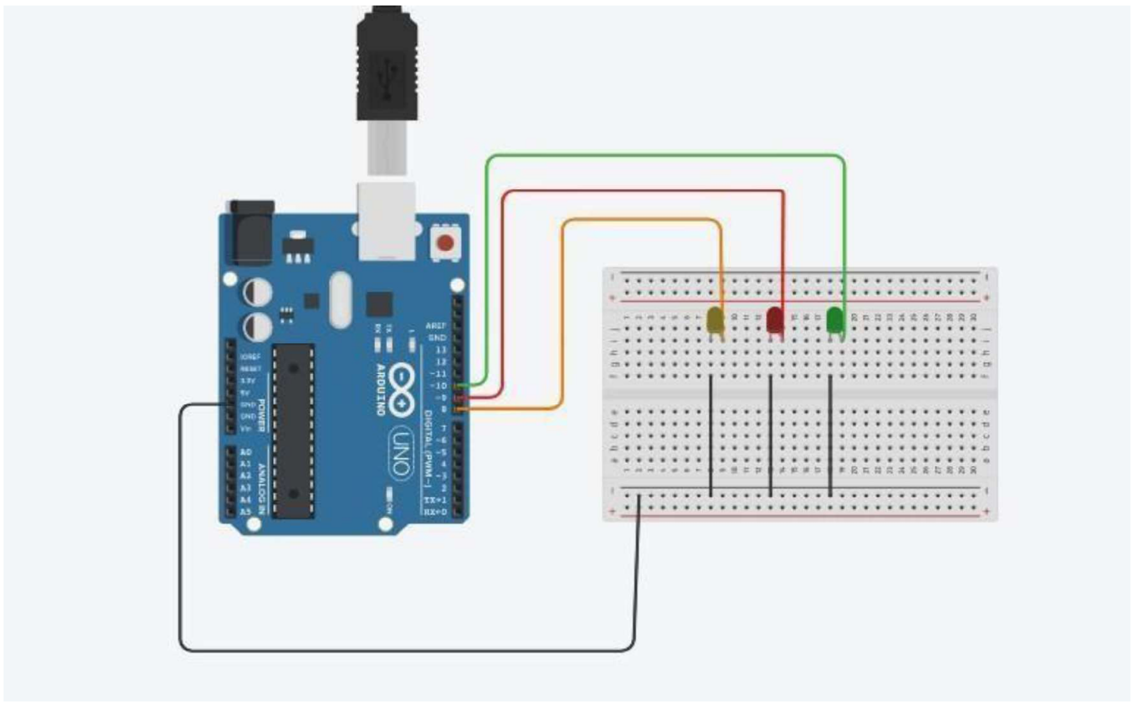
The problem statement is like Arduino traffic light, a fun little project that you can build in under an hour. Here's how to build your own using an Arduino, and how to change
The circuit for an advanced variation.

A part from the basic Arduino, you'll need:

- 3x220-ohmresistors
- A bread board
- Connecting wires
- Red, yellow and green

Basics
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(viaa220-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 a number. 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:

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 traffic light that changes every 15seconds, like this (sped up):

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

The **changeLights** function consists off our distinct steps:

- Green blinking , red off, yellow off - after entering b
- Green on, yellow off, red Off -after entering g
- Yellow off, redon, Green off - after entering r
- Yellow on, red off, Green off after entering y

For each step, the code is very similar. The appropriate LED gets turned on or off using **digitalWrite**. This is an Arduino function used to set output pins to HIGH(for on),or LOW(for off).

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

For entering the data from keyboard we need to enable serial port of Arduino. the functions used for the same are as follows:

1) Serial.begin() : Required to initialize the serial port the desired baud rate is passes as function value, we will required the baud rate of 9600.
2) Serial.avialable(): This function is used to check the status on serial port about the availability of data on serial port.
3) Serial.read(): this function is used to read the values in serial port.
4) Serial.print(): This function is used for printing a data on serial port.

Program:

Conclusion:-

Assignment No. B8

- **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 show show to load an example sketch to an Arduino. It is necessary to know how to load a sketch 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.

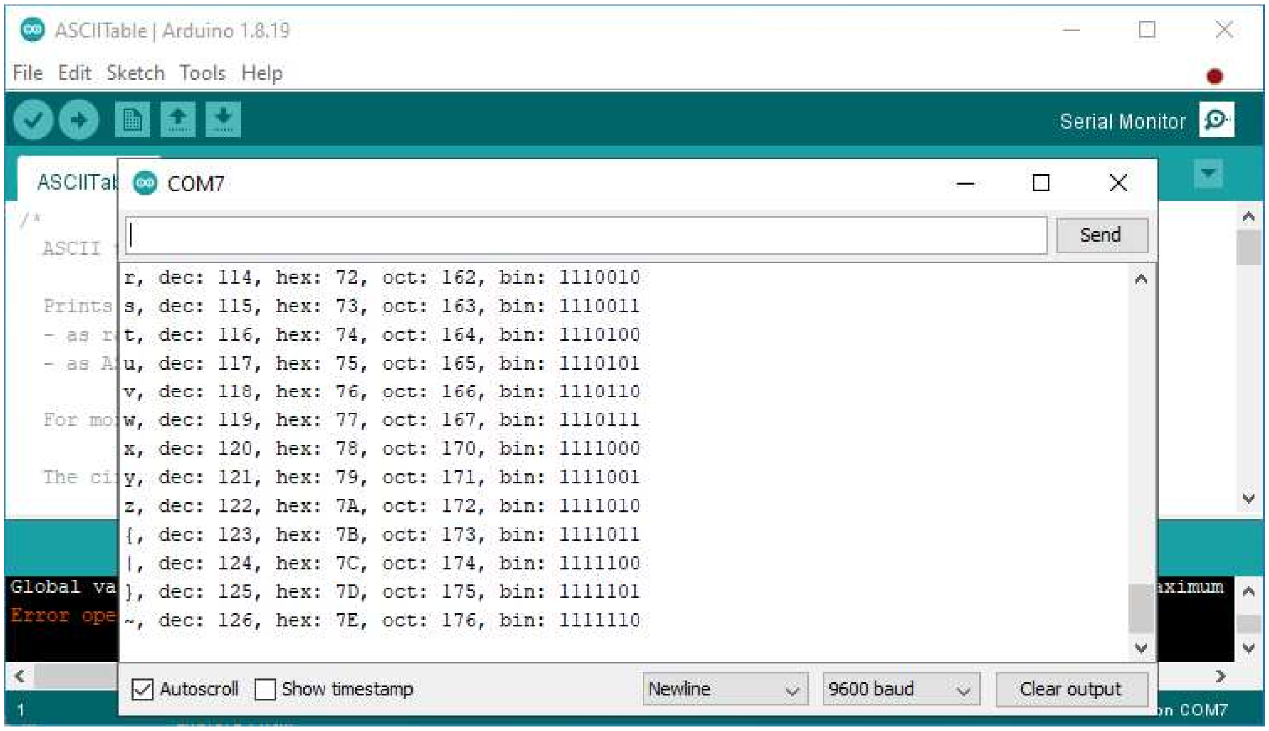
- Load an Example Sketch that uses the Serial Monitor to an Arduino Board

Start the Arduino IDE application. Select File→Examples→04. Communication →ASCII 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 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 tool bar. A red dot near the top right of the image shows the serial monitor tool bar icon location.



```
r, dec: 114, hex: 72, oct: 162, bin: 1110010
s, dec: 115, hex: 73, oct: 163, bin: 1110011
t, dec: 116, hex: 74, oct: 164, bin: 1110100
u, dec: 117, hex: 75, oct: 165, bin: 1110101
v, dec: 118, hex: 76, oct: 166, bin: 1110110
w, dec: 119, hex: 77, oct: 167, bin: 1110111
x, dec: 120, hex: 78, oct: 170, bin: 1111000
y, dec: 121, hex: 79, oct: 171, bin: 1111001
z, dec: 122, hex: 7A, oct: 172, bin: 1111010
{, dec: 123, hex: 7B, oct: 173, bin: 1111011
|, dec: 124, hex: 7C, oct: 174, bin: 1111100
}, dec: 125, hex: 7D, oct: 175, bin: 1111101
~, dec: 126, hex: 7E, oct: 176, bin: 1111110
```

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 ASCIIT able 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 are a 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 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.

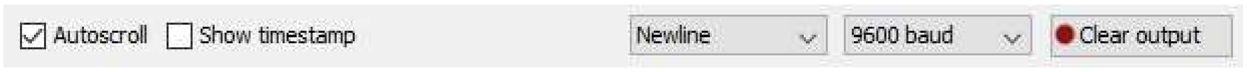
- Reset the Arduino Board with the RESET Button

Press and release the RESET button on the Arduino board and the 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 MEGA2560.

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.



Serial Monitor Window Clear Output Button

- 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 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 and bin.

Program:

Conclusion:

Assignment No. B 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.

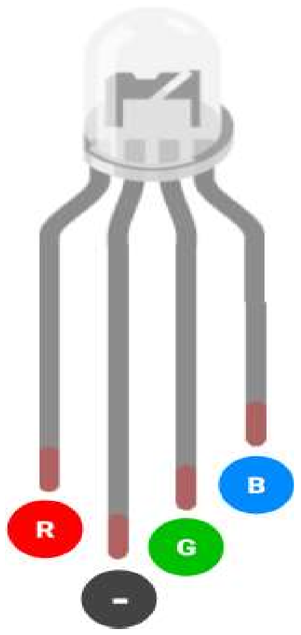
- **Outcome:** Connectivity, configuration and serial communication with Arduino.

- **Hardware Requirement:**
    - Arduino board –
    - Breadboard.
    - Some male-female wires.
    - RGB LED.
    - 3 * 220 Ohm resistors.
    - 3 Potentiometer.
    - Arduino, USB cable etc.

- **Software Requirement**: Arduino IDE

- Theory:

RGB LED
For this project, we will use a **common cathode RGB LED**. Inside this component are three smaller LEDs - one red, one green, and one blue. They are all connected by a shared or *common* cathode.



This RGB LED has four pins: One is the common cathode, and the other three are the red, green, and blue LED anodes.

In the world of digital electronics, colour is often represented in the 8-bit RGB colour model. Every visible color is identified by an encoding of the three base colours: red, green, and blue. In our Arduino code, we will set the intensity of each base colour to a value in the range 0-255.

Try the RGB Colour Picker. Adjust the slider at the bottom of the application. Notice how the three values in the RGB field change as you select different colours.

Recall that LEDs are current driven: we control the intensity of each RGB LED color by increasing/decreasing the current passing through its pin. With the Arduino board, we set the voltage, which changes current flow (recall Ohm's Law).

Here's the rub: digitalWrite () can only set a pin's voltage to 0 or 5V (the supply voltage). If we want to dim an LED, we need a new function that can output voltages between 0 and 5.

We will use a **pulse width modulation** or **PWM** signal - a digital waveform mimicking analog voltage signals. With PWM waves, we can generate analog voltages like 2.3V or 4.1V. In Arduino, we accomplish this using analogWrite().

Only certain Nano pins are capable of PWM, as indicated in the pinout above.

To build the circuit: Make a common ground by connecting a GND pin of the Arduino, to the "minus" line of the breadboard.
For the RGB LED:

- Find the longer leg. Depending on your RGB LED, this can be a cathode or an anode. How to find out? Simply check the manual or description of what you've bought. If no instructions, you can first try the "cathode mode" and see if it works. So, if it's a cathode, plug this to the ground (GND). If it's an anode, plug it to 5V on the Arduino.
- The 3 other legs correspond to red, blue, and green colors. Connect each of these legs to a PWM compatible digital pin on the Arduino (with a "~" next to the number, like on the picture). Add a 220 Ohm resistor in between for each leg.

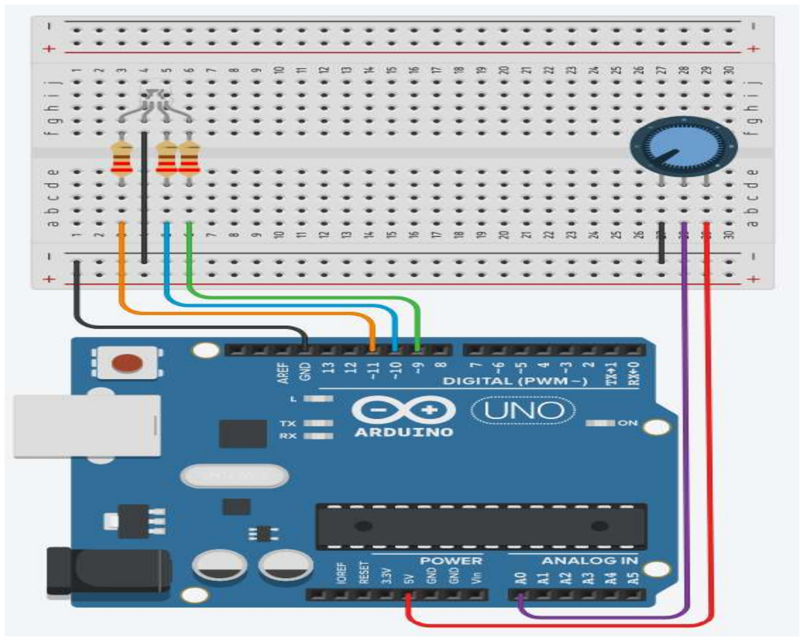
- For the potentiometer:
  - Connect one of the extreme leg (for example left) to the ground. The other extreme leg should be connected to 5V on the Arduino.
  - Connect the middle leg to an analog pin.

Control the RGB LED with the potentiometer – digitalWrite() – 7 colors

In this first application, we are going to modify the color of the RGB LED when we turn the potentiometer knob.
We can see the RGB LED as a combination of 3 different LEDs that you control separately.
We are going to use digitalWrite() – LED fully powered on/off, which means that we have a combination of 7 colors:

- red
- blue
- green
- red + blue
- red + green
- blue + green
- red + blue + green

- **Interfacing Diagram:**



Program:

Output:

Conclusion:

Assignment No. B 10

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

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

- **Hardware Requirement:**
  - Arduino board –
  - Breadboard.
  - Some male-female wires.
  - DHT 11 temperature sensor
  - Arduino, USB cable etc.

- **Software Requirement: ArduinoIDE**

- Theory:

DHT11 Humidity Temperature Sensor

The **DHT11** is a commonly used and cost-effective sensor that provides readings of both **temperature** and **humidity**. Compact and cost-effective, it has become a popular choice in the electronics community for its **relative accuracy** and ease of integration. Operating on a simple **one-wire communication protocol**, the DHT11 can be seamlessly interfaced with a variety of **microcontrollers**, making it suitable for a wide range of applications.

Developed for digital systems, the DHT11 finds its applications in environments where there's a need to monitor or control atmospheric conditions, such as in smart homes, **weather stations**, and agricultural monitoring systems.

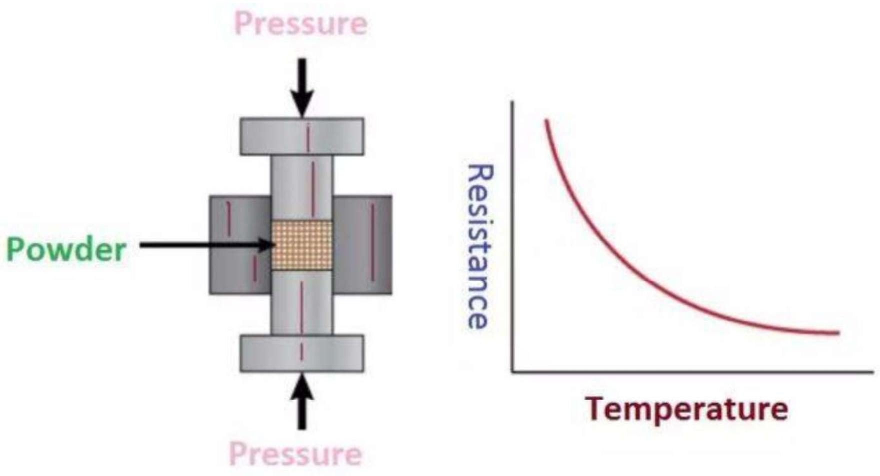
Features/Specifications of DHT11 Sensor

- **Power Supply:** Typically operates from 3.3 to 5V DC, making it suitable for interfacing with most microcontrollers.
- **Temperature Range:** 0°C to 50°C with ±2°C accuracy.
- **Humidity Range:** 20% to 90% RH (Relative Humidity) with ±5% accuracy.
- **Output:** Calibrated digital signal. It employs a single-wire communication protocol which simplifies integration with microcontroller systems.
- **Sampling Period:** Suggested minimum time of 1 second between readings to ensure sensor accuracy.
- **Dimensions:** Compact size, often available in a 4-pin single-row package.
- **Longevity:** Offers excellent long-term stability, thanks to its calibrated digital output.

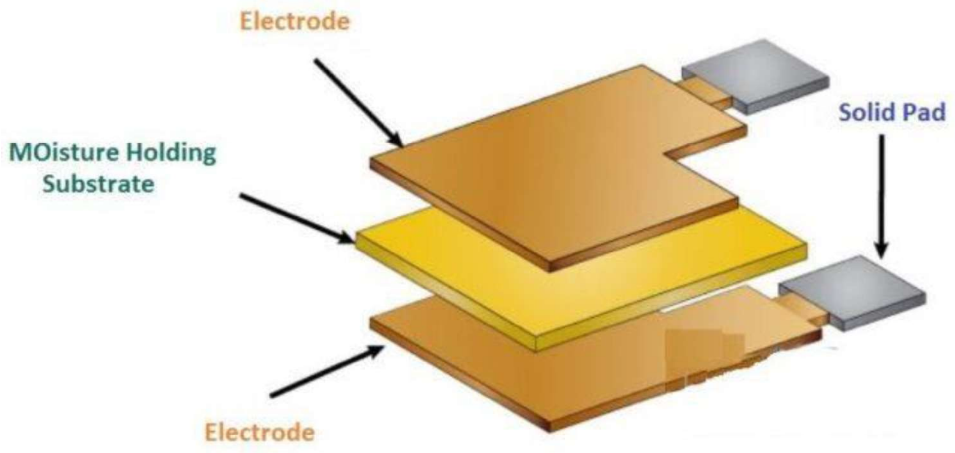
Construction of DHT11 Sensor

- **Sensing Elements:** The DHT11 integrates two primary components for its operations:
    1. A thermistor for temperature measurements.
    2. A capacitive humidity sensor for gauging atmospheric moisture.
- **IC Integration:** An 8-bit microcontroller reads the outputs from the two sensors and translates them into a format suitable for digital systems.
- **Packaging:** The sensor elements are often encased in a plastic casing, which ensures protection against environmental factors while still allowing for accurate readings.

Working of DHT11 Sensor

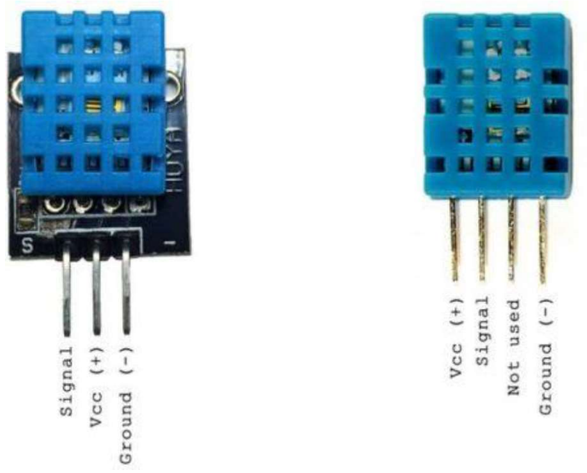


-
    - **Temperature Measurement:** The core of the temperature sensing capability is the thermistor, a type of resistor whose resistance changes with temperature. As the temperature fluctuates, so does the resistance of the thermistor. This variance is then read and converted into temperature values by the onboard microcontroller.



- **Humidity Measurement:** The capacitive humidity sensor operates by having a dielectric material between two plates. As humidity changes, the dielectric constant of the material changes, leading to a variance in capacitance. This change in capacitance is then read and interpreted as a humidity percentage by the onboard microcontroller.
- **Signal Output:** After the microcontroller processes the readings, the data is sent as a digital signal through a single-wire communication protocol. This digital output can then be easily read by microcontrollers such as Arduino, making integration and data interpretation straightforward.

Pinout of DHT11 Sensor
The DHT11 typically comes with a 4-pin package, although only three of them are functionally used:



1. **VCC (Pin 1):** This is the power supply pin, which can accept voltages from 3.3V to 5V, making it compatible with most microcontroller systems.
2. **Data (Pin 2):** This is the pin through which the DHT11 communicates. It uses a proprietary single-wire protocol to transmit temperature and humidity data to the connected microcontroller.
3. **NC (Pin 3):** Not connected or used.

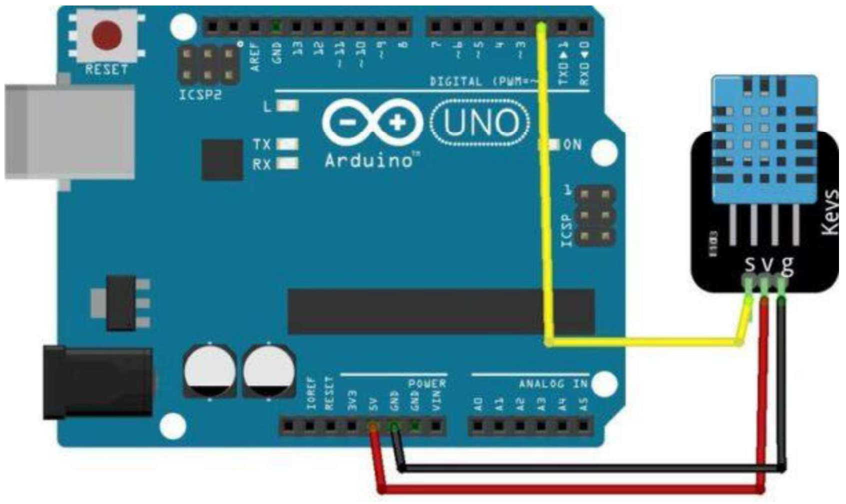
4. **GND (Pin 4):** Ground pin, used to complete the circuit.

- Interfacing DHT11 Sensor with Arduino

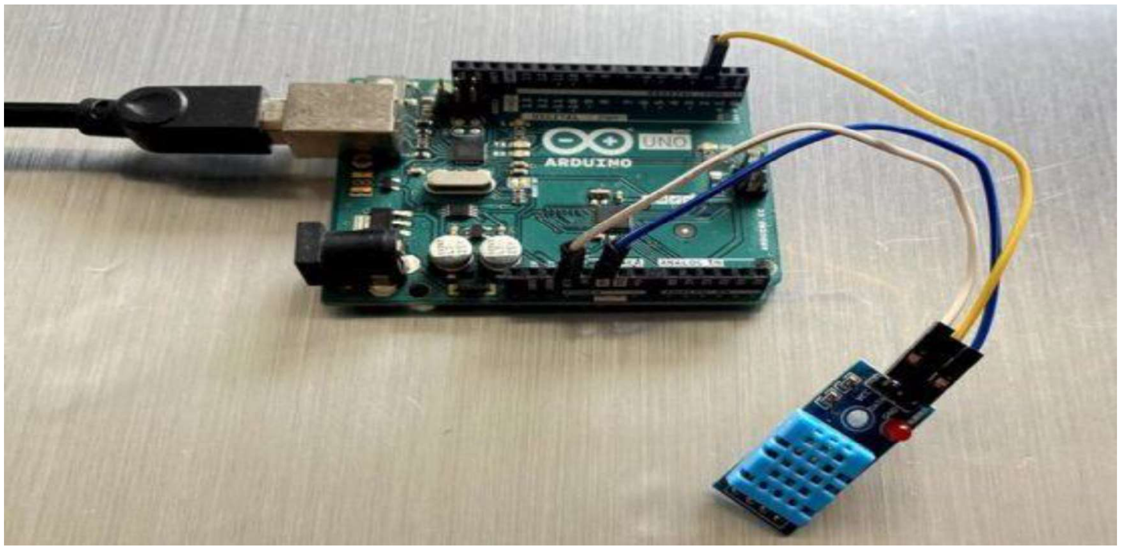
Let us interface the DHT11 Humidity Temperature Sensor with Arduino UNO.

Hardware Connection

The connection diagram is pretty simple as shown in the image below.



Connect the VCC & GND Pin of DHT11 Sensor Module to 3.3V & GND pin of Arduino respectively. Similarly connect the DHT11 Data pin to Arduino Digital Pin 2.



You can use a jumper wire & connect the sensor with Arduino Board.

As shown in fig. DHT11 sensor has three pins: GND, DATA and VCC

Steps:

1. Connect GND and VCC to ground and VCC of Aurdino

2. Connect Data pin of temp sensor to 2 pin.
3. Run Python code temp.py

- Details about the code

You need to follow these instructions to make it work:

1. You need to add the DHT library to the Arduino IDE.

2. Upload the code.

3. When the code is uploaded, open the Serial Monitor and set the baud rate to 9600.

4. You will see the humidity and temperature.

Program:

Output: Temperature and humidity can be seen on serial port

Conclusion

Assignment No. B 11

- **Aim**: Write a program so it displays the temperature in Fahrenheit as well as the maximum andminimum temperatures it has seen

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

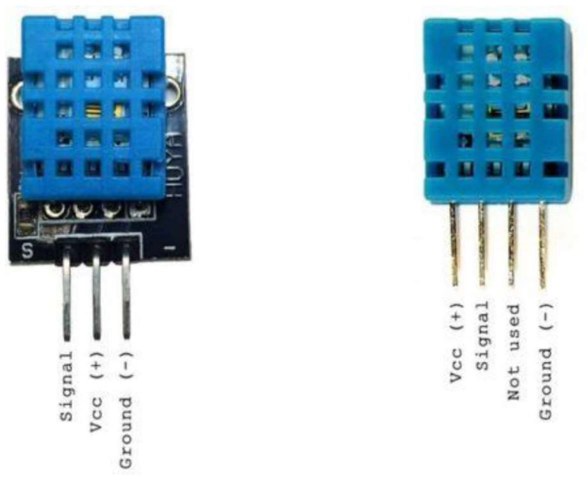
- **Hardware Requirement:**
  - Arduino board –
  - Breadboard.
  - Some male-female wires.
  - DHT 11 temperature sensor
  - Arduino, USB cable etc.

- **Software Requirement: Arduino IDE**

- Theory:

Pinout of DHT11 Sensor

The DHT11 typically comes with a 4-pin package, although only three of them are functionally used:



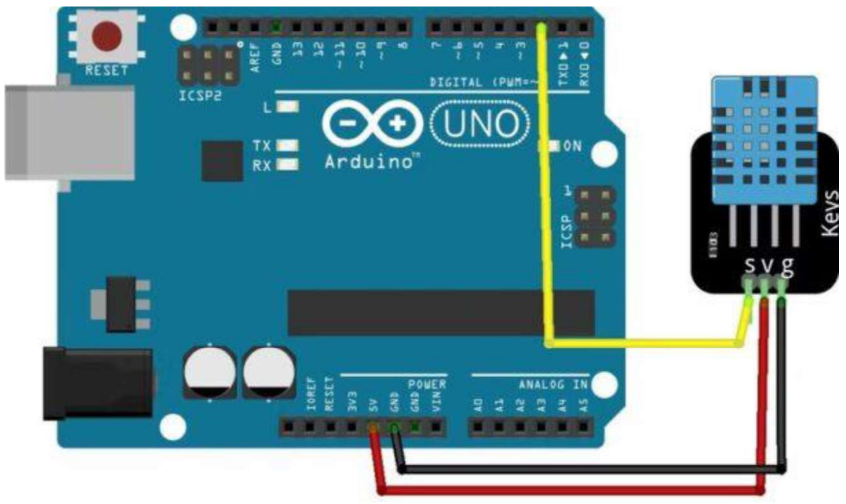
5. **VCC (Pin 1):** This is the power supply pin, which can accept voltages from 3.3V to 5V, making it compatible with most microcontroller systems.
6. **Data (Pin 2):** This is the pin through which the DHT11 communicates. It uses a proprietary single-wire protocol to transmit temperature and humidity data to the connected microcontroller.
7. **NC (Pin 3):** Not connected or used.
8. **GND (Pin 4):** Ground pin, used to complete the circuit.

- **Interfacing DHT11 Sensor with Arduino**

  Let us interface the DHT11 Humidity Temperature Sensor with Arduino UNO.

Hardware Connection

The connection diagram is pretty simple as shown in the image below.



Connect the VCC & GND Pin of DHT11 Sensor Module to 3.3V & GND pin of Arduino respectively. Similarly connect the DHT11 Data pin to Arduino Digital Pin 2.

For finding out the minimum and maximum temperature we need to compare the temperature measured in every turn with the previously measured temperature . for comparison If loop is used

e. g

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

Where tfmax is max temperature in Fahrenheit and tfmin is minimum temperature in minimum

Procedure :

You need to follow these instructions to make it work:

1. You need to add the DHT library to the Arduino IDE.

2. Upload the code.

3. When the code is uploaded, open the Serial Monitor and set the baud rate to 9600.

4. You will see the humidity and temperature.

Program:

Output: Minimum and maximum temperature can be seen on serial port

Conclusion

Assignment B 12

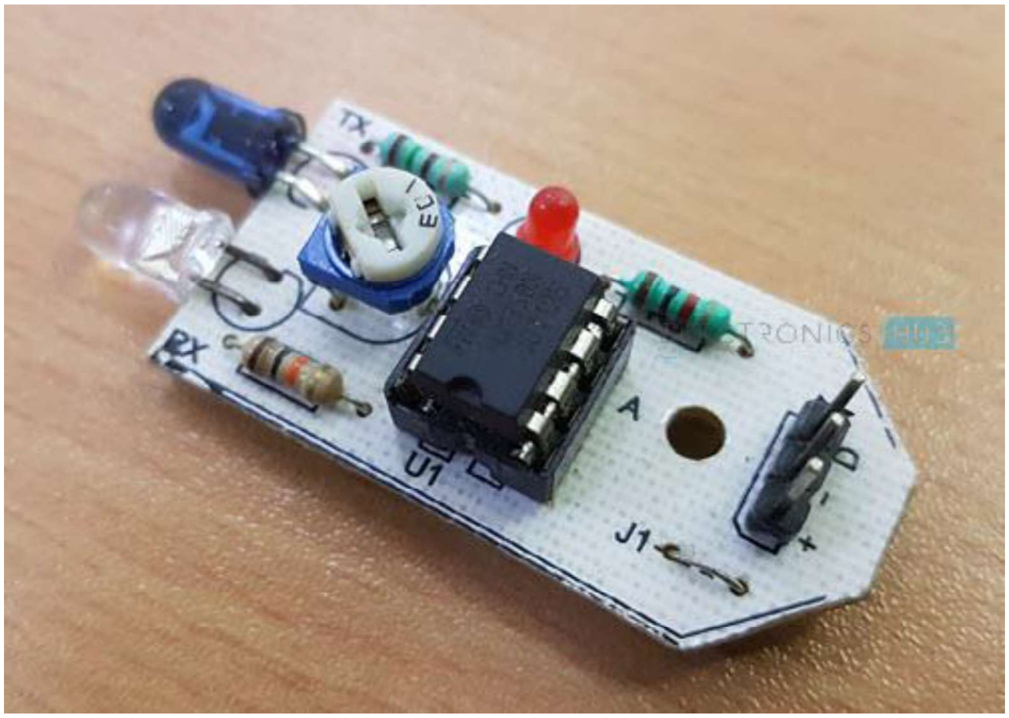
Aim: Understanding the connectivity of Raspberry-Pi /Beagle board circuit / Arduino with IR sensor.Write an application to detect obstacle and notify user using LEDs.

Hardware Requirement: Raspberry-Pi/Beagle, IR sensor, LED, Jumper cables, Bread board.

Theory:

Brief Note on IR Sensor (IR Proximity Sensor)

IR Sensors emit and receive Infrared radiation. They are often used as Proximity Sensors i.e. detect and alarm if an object is close to the sensor. Let us help you understand better about IR Sensors by giving two real life applications of IR Sensors. The first one is Mobile Phones. Almost all mobile phones nowadays have IR Sensors in them. Usually, they will be placed near the earpiece on the phone. When the user make or receives a phone call, the IR Sensor detects how far the phone is from the user's ear. If it is close to the ear, the phone's display will be turned off so that you do not touch anything on the screen accidently. Another important application is in automobiles. All modern cars are equipped with reverse parking sensor that sense how far you can reverse your car without hitting anything. These reverse sensors are implemented using IR Sensors.



An IR Sensor Module basically consists of three parts: an IR Transmitter, an IR Detector and a control circuit. Usually, an IR LED is used as an IR Transmitter and a Photo Diode or a Photo

Transistor (less often) is used as an IR Detector. The control circuit consists of a Comparator IC with necessary components.

Based on the application and requirement, IR Sensors can be implemented in two ways. In the first method, both the IR Transmitter and the IR Detector are placed side-by-side. In the second setup, the IR Transmitter and the IR Detector are placed facing each other.

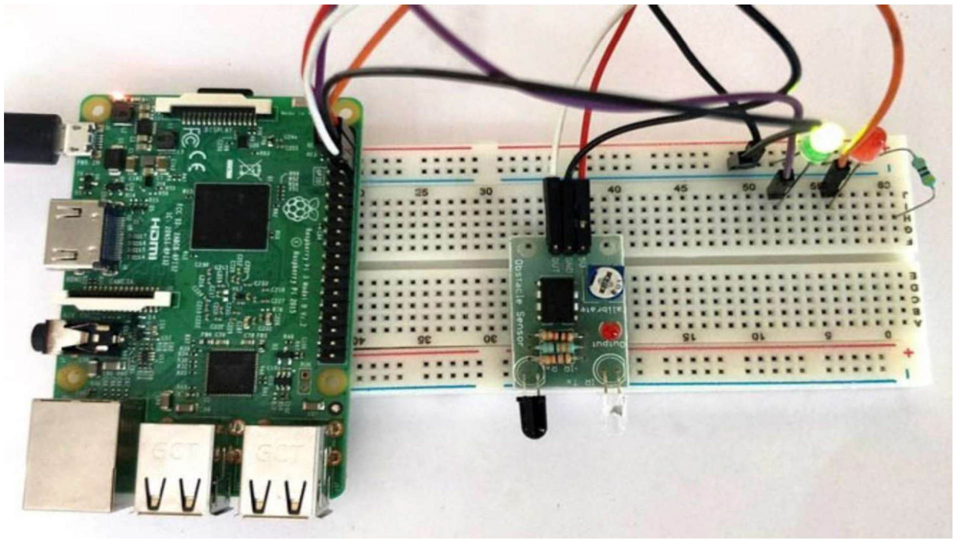
The first way of implementation is known as Reflective Type IR Sensor. In this setup, the IR Transmitter continuously emits infrared light and if there is any obstacle/object in front of the sensor, the infrared light hits the object and bounces back. The reflected signal is captured by the IR Detector and the control circuit will reflect a Logic HIGH on its output.

The second way of implementation, where the IR Transmitter and Detector are positioned face-t-face, is known as Transmissive Type IR Sensor. Here, the infrared light from the IR Transmitter always falls on the Detector.

If there is an object in between the Transmitter and Detector, then there will be an obstruction to the infrared light and the control circuit will detect this and produces appropriate output.

The IR Sensor used in this project is a Reflective Type IR Sensor. You can easily build this type of IR Sensor as a DIY Project as the circuit is very simple.

Interfacing Diagram:



IR interfacing with RPi

Procedure:

1.Connect IRand LED to Raspberry pi according to Raspberry pi pin diagram using breadboard and jumper cables

2.Write code to detect object nearer to IR sensor and notify using LED

3.RunPythoncodeIR.py

Program:

Output: When the object is detected the LED will be on else LED is off

Conclusion: