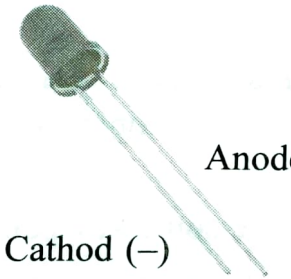
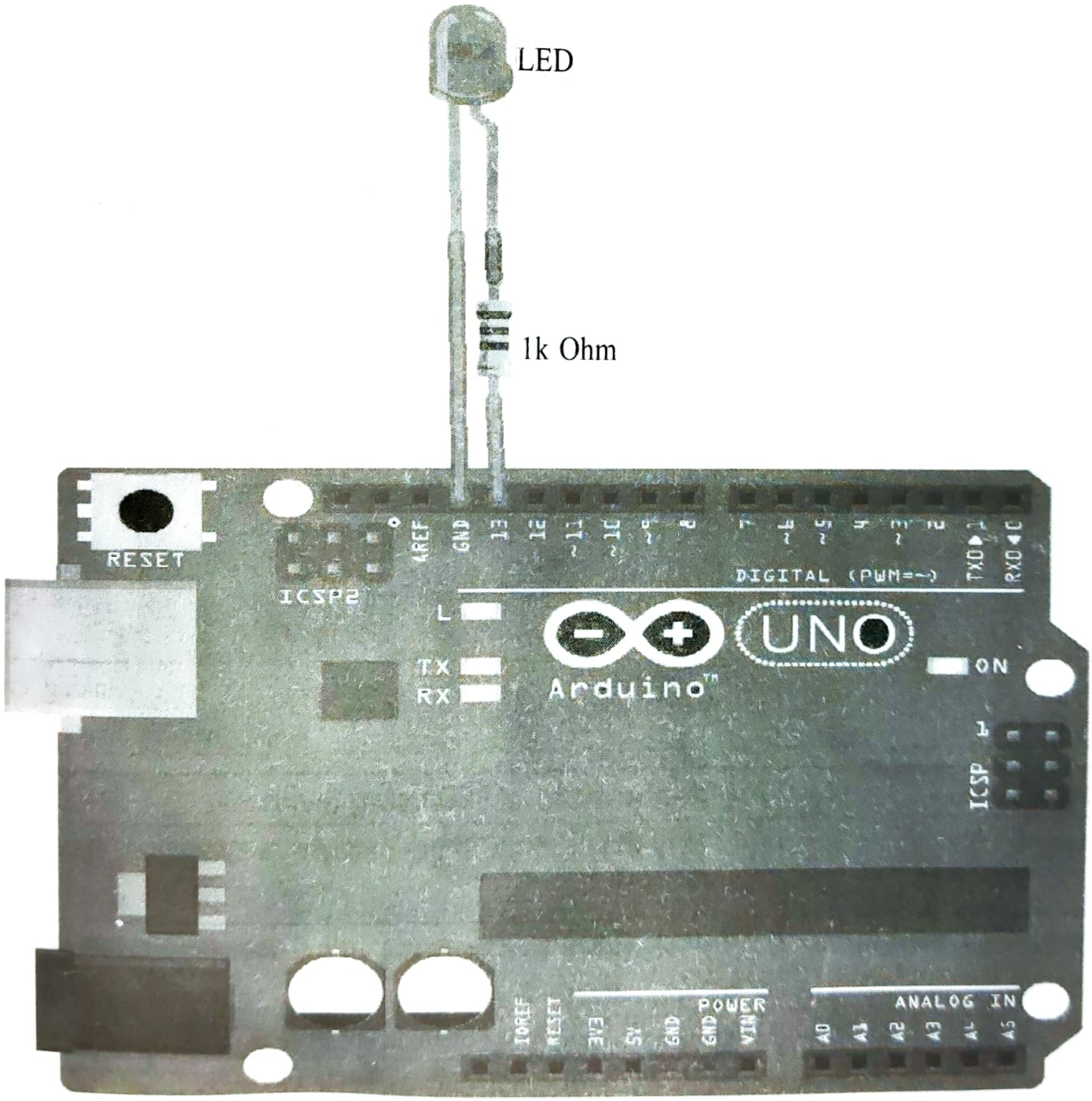
Program #1	**Blinking an LED**
Components required	1-LED, 1-KΩ resistor, Jumper wires, Breadboard
LED	 The longest lead is the anode and the shortest is the cathode
	LED Working: In simple terms here's how an LED works, An LED is a basic semiconductor device which is a p-type semiconductor material containing positively charged carriers called **Holes**. Holes are combined with n-type semiconductor material containing negatively charged carriers called electrons to create a **diode**. When a current supply is connected to the diode the negative charged electrons are forced to move in one direction and the positive holes move in the opposite direction.
	When a free electron comes near a hole it combines with the hole, the holes exist at a lower energy level than the free electron so the electron must lose energy to combine with the hole. This energy is released in the form of a photon or unit of light, the amount of photon energy released determines the frequency or color of the light.

	The type of material and process of creating the n-type and p-type materials dictate the color of the photons as well as the efficiency and other performance characteristics of the LED. After processing the material into an LED chip, the chip is installed in a package that allows electrical connection and directs as much light as possible in the desired direction

Circuit Diagram

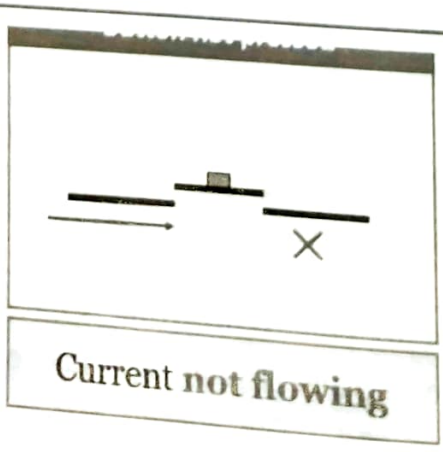


How to choose Resistor for a circuit	In simple context Resistor is chosen using Ohm's Law i.e R=V/I where R-Value of the Resistance required, V-Voltage across the resistor and I-The current that is flown through the resistor. For example consider an LED require a voltage of 3V which is from a 9V power source, for a current of 0.02Amps. Then the equation results in R=(9V-3V)/0.02=300 KΩ.

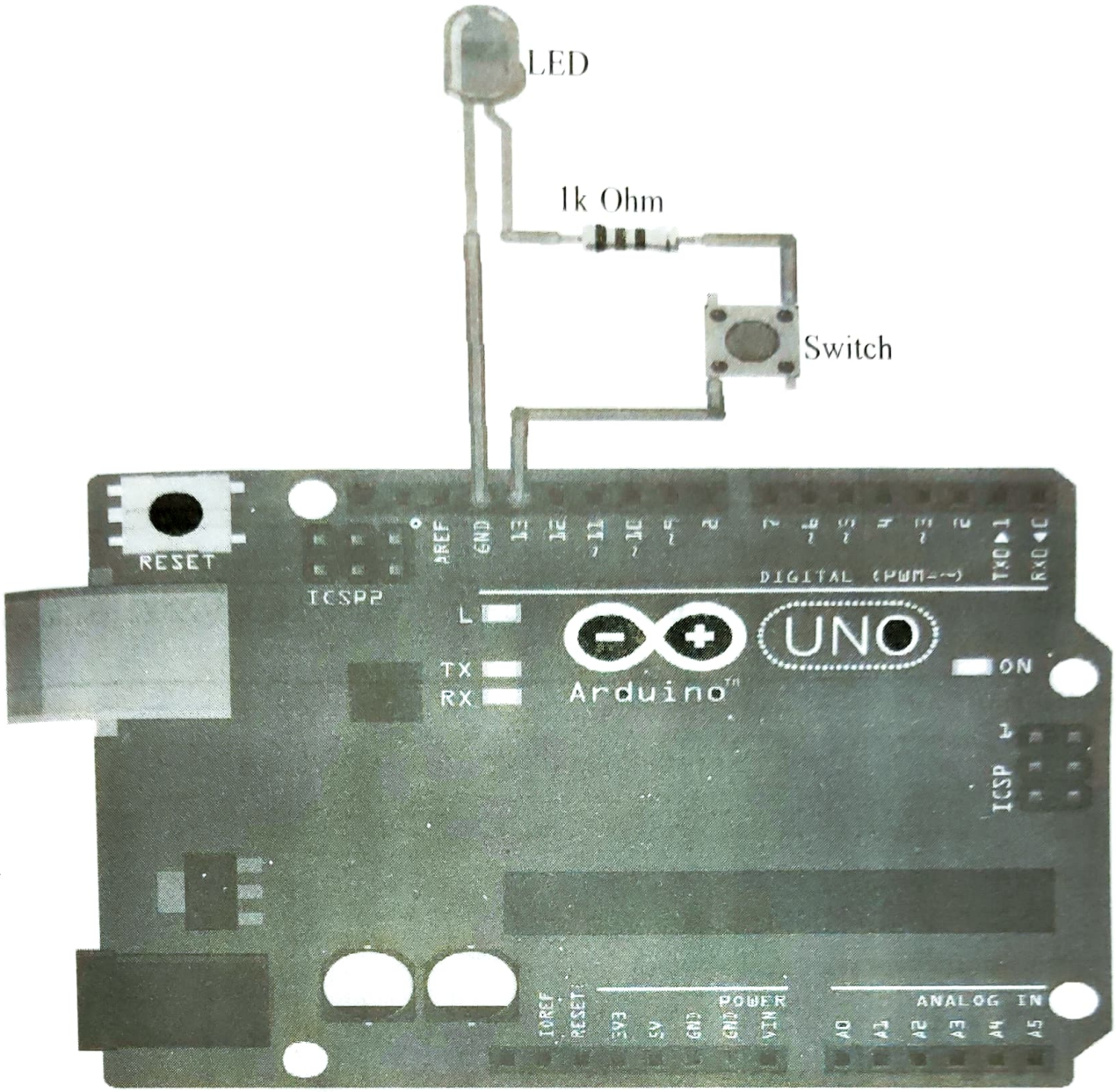
Description	In this program LED is connected to digital pin 13 as shown in circuit diagram above. An infinite loop runs over the loop() function to toggle the state of LED to blink and incurring the state switch mechanism by constantly using the delay() function.

Code	/*The Function setup runs only once when Arduino board is first powered up or a rest button the board is pressed */
	void setup()
	{
	//pin 13 is set as an OUTPUT pin
	pinMode(13, OUTPUT);
	}
	//loop function iterates forever
	void loop()
	{
	//Sets LED to HIGH voltage
	digitalWrite(13, HIGH);
	//delay by a second
	delay(1000);
	//Sets LED to LOW voltage
	digitalWrite(13, LOW);
	//delay by a second
	delay(1000); // wait for a second
	}

Output	Run the above code and notice the LED starts blinking every second.

Program # 2	**Toggle the state of LED using Swith**
Components required	1–LED, 1–K Ω resistor, 1–push button, Jumper wires, Breadboard
How a push button works?	 Current **not flowing**  Current flowing Here an **open push button** mechanism is used. In Normal state(not pushed) of the button current doesn't flow, only when button is pushed flow of current is allowed as shown in above figure.

Circuit diagram



Description	In this program LED is connected to digital pin 13 via switch and shown in circuit diagram above. An infinite loop runs over the loop() function to toggle the state of LED to blink, when an input is given through a switch button press/release and incurring the state switch mechanism by constantly using the delay() function.

Code	/*The Function setup runs only once when Arduino board is first powered up or a rest button the board is pressed */
	void setup()
	{
	//pin 13 is set as an OUTPUT pin
	pinMode(13, OUTPUT);
	}

```
//loop function iterates forever
void loop()
{
//Sets LED to HIGH voltage when a button is pressed else it remains
LOW
digitalWrite(13, HIGH);
//delay by a second
delay(1000);
}
```

Output	Run the above code and Press the Push button switch to Turn ON OFF the LED.
Program # 3	**Traffic light Simulation for Pedestrians**
Components required	2-Red LED, 2-Green LED, 1-Yellow LED, 5-220Ω resistor, Jumper wires, Breadboard

Program #1	Printing to a terminal
Components required	Raspberry Pi + SD card, Monitor + HDMI Cable, Keyboard & Mouse, Power supply
Description	Now let us start with a basic example of printing a message "Hello World" using Python Programming. Box shows how to print a greeting message to the console.
Key points	1. Find your customized Raspberry Pi. 2. Mount the SD card. 3. Plug in the HDMI cable into the Pi and the monitor. 4. Plug in the keyboard into the USB ports 5. Plug in the mouse into the USB ports 6. Plug in the power cable 7. Type in user name "pi" 8. Type in password "raspberry" 9. Double click on "Terminal" 10. This will load the "terminal" 11. Type the follow commands ✓ Change the directory by the command $ cd Desktop ✓ Create a new directory $ mkdir python_code ✓ Change the directory to python_code $ cd python_code ✓ create new file helloworld.py ✓ Now enter the code given in the box below ✓ Run the python code "sudo python helloworld.py ✓ You will see it print "Hello World!" to the screen

Code	**File:Helloworld.py**
	#Access the python working environment
	#!/usr/bin/python
	#Print a message Hello world on to the terminal
	print("Hello World!")

Output	A message "Hello world" will prints on the console

Program #6	Temperature sensor
Components required	Raspberry Pi + SD card, Monitor + HDMI Cable, Keyboard & Mouse and Power supply, 1 Red LED and Blue LED, 2 1K resistors, push button and jumper wires, Breadboard, buzzer.1 LM35 temperature sensor.
Description	Now let us look at an example involving an DS18B22 temperature sensor which reads out a temperature and records on to a terminal. Figure shows the schema -tic diagram of connecting an DS18B22 temperature sensor to Raspberry Pi board. Box shows a python program which records the temperature read by LM35 temperature sensor.This example shows how to get an analog input from GPIO pins and process the input. An infinite loop runs over the sensor which records a temperature every second Compile the code given below and upload it to Arduino UNO Board to observe at the desired output.

Code

```
import os
import glob
import time
#initialize the device
os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')

base_dir = '/sys/bus/w1/devices/'
device_folder = glob.glob(base_dir + '28*')[0]
device_file = device_folder + '/w1_slave'

def read_temp_raw():
    f = open(device_file, 'r')
    lines = f.readlines()
    f.close()
```

```
    return lines


def read_temp():
    lines = read_temp_raw()
    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
        lines = read_temp_raw()
    equals_pos = lines[1].find('t=')
    if equals_pos != -1:
        temp_string = lines[1][equals_pos+2:]
        temp_c = float(temp_string) / 1000.0
        temp_f = temp_c * 9.0 / 5.0 + 32.0
        return temp_c, temp_f


while True:
        print(read_temp())
        time.sleep(1)
```

Output	Current Room Temperature is recorded.