7.1 Introduction to Arduino

Arduino is an open-source advancement prototyping platform which depends on simple to-utilize equipment and programming. Arduino can read inputs –such as detecting the power of light, events triggered by a Button or a twitter message and can respond into a yield, for example, run the engine, on the LED, send data through online. Instructions to the microcontroller are given by the use of Arduino programming dialect which depends on wiring and handled through the utilization of Arduino Software (IDE-Integrated improvement environment).

The Arduino is a small computer that you can program to read information from the world around you and to send commands to the outside world. All of this is possible because you can connect several devices and components to the Arduino to do what you want. You can do amazing projects with it, there is no limit for what you can do, and using your imagination everything is possible!

Arduino is a tiny computer that you can connect to electrical circuits. This makes it easy to read inputs – read data from the outside – and control outputs - send a command to the outside. The brain of this board (Arduino Uno) is an ATmega328p chip where you can store your programs that will tell your Arduino what to do.

7.1.1 Why Arduino?

- ❖ Arduino is an open source product, software/hardware which is accessible and flexible to customers.
- ❖ Arduino is flexible because of offering variety of digital and analog pins, SPI and PWM outputs.
- ❖ Arduino is easy to use, connected to a computer via a USB and communicates using serial protocol.
- ❖ Inexpensive, around 500 rupees per board with free authoring software.
- ❖ Arduino has growing online community where lots of source code is available for use, share and post examples for others to use too, too!.
- ❖ Arduino is Cross-platform, which can work on Windows, Mac or Linux platforms.
- ❖ Arduino follows Simple, clear programming environment as C language.

7.1.2 Which Arduino?

In the ten years since Arduino was released, hundreds of "Arduino boards" are available in the market serving every kind of purpose. Among all in this book we focus on popular Arduino UNO which is used in almost 99% of projects use.

Some of the Boards from Arduino family are given below.

Arduino Mega is a big sister to the UNO with more memory and pins with a different chip the ATmega2560, useful when your project doesn't fits in an UNO

Arduino Micro is bit smaller with a chip Atmega32u4 that can act like a keyboard or mouse which does its task with native USB. Its slim with downward pins which can be plugged into a breadboard.

The Arduino MKR1000 is a little like an Arduino Micro but has a more powerful32-bit ATSAM ARM chip and built-in WiFi! A great upgrade for when you want to do Internet of Things projects.

Flora is an Arduino compatible from Adafruit which is a round wearable which can be sewed into clothing.

In this textbook all the prototypes are built by using Arduino Uno board from the Arduino family which is discussed in the following section.

7.2 Exploring Arduino Uno Learning Board

In the Figure 7-1 below you can see an Arduino board labeled. Let's see what each part does.

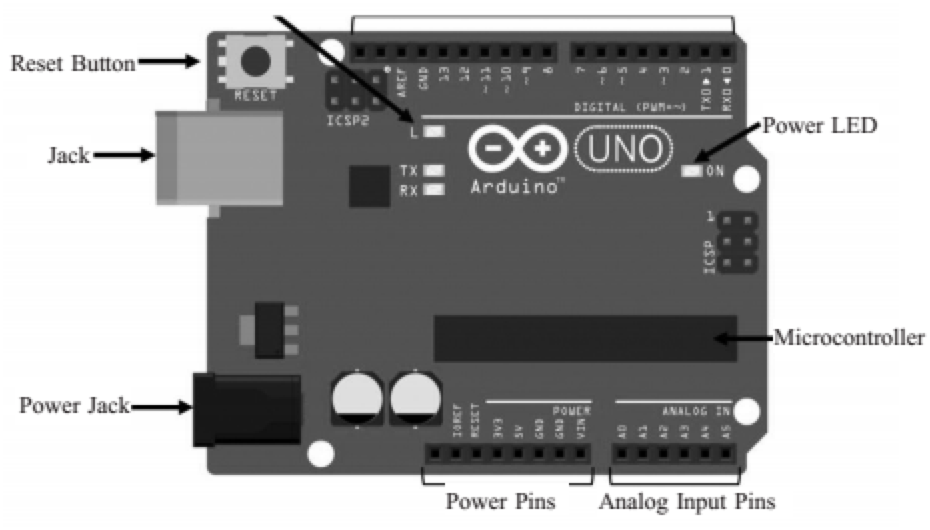


Figure 7-1: Arduino Uno Learning Board

- ❖ Microcontroller: the ATmega328p is the Arduino brain. Everything on the Arduino board is meant to support this microcontroller.
- ❖ Digital pins: Arduino has 14 digital pins, labeled from 0 to 13 that can act as inputs or outputs.
- ❖ When set as inputs, these pins can read voltage. They can only read two different states HIGH or LOW. When set as outputs, these pins can apply voltage. They can only apply 5V (HIGH) or 0V (LOW).

- PWM pins: These are digital pins marked with a ~ (pins 11, 10, 9, 6, 5 and 3). PWM stands for "pulse width modulation" and allows to make digital pins output "fake" varying amounts of voltage. You'll learn more about PWM later.
- TX and RX pins: digital pins 0 and 1. The T stands for "transmit" and the R for "receive". Arduino uses these pins to communicate with the computer. Avoid using these pins, unless you're running out of pins.
- LED attached to digital pin 13: This is useful for an easy debugging of the Arduino sketches.
- TX and RX pins: these pins blink when there are information being sent between the computer and the Arduino.
- Analog pins: the analog pins are labeled from A0 to A5 and are most often used to read analog sensors. They can read different amounts of voltage between 0 and 5V. Additionally, they can also be used as digital output/input pins like the digital pins.
- Power pins: The Arduino has 3.3V or 5V supply, which is really useful since most components require 3.3V or 5V. The pins labelled as "GND" are the ground pins.
- Reset button: when you press that button, the program that is currently being run in your Arduino will start from the beginning. You also have a Reset pin next to the power pins that acts as reset button. When you apply a small voltage to that pin, it will reset the Arduino.
- Power ON LED: will be on since power is applied to the Arduino.
- USB jack: Connecting a male USB A to male USB B cable is how you upload programs from your computer to your Arduino board. This also powers your Arduino.
- Power jack: The power jack is where you connect a component to power up your Arduino (recommended voltage is 5V). There are several ways to power up your Arduino: rechargeable batteries, disposable batteries, wall-warts and solar panel.

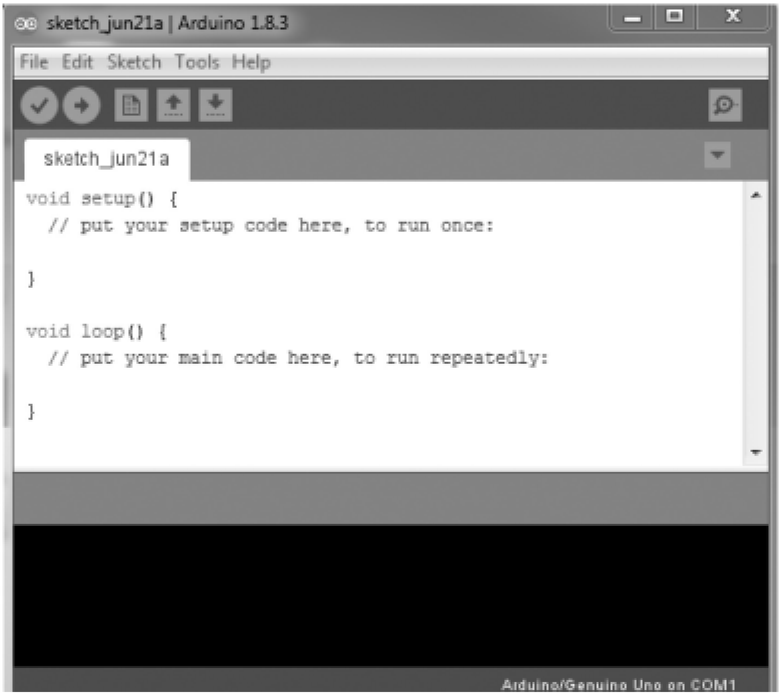
7.2.1 Things that Arduino can do

- The simplest thing you can control with your Arduino is an LED.
- You can also display a message in a display, like the LCD display.
- You can also control DC or servo motors.
- You can also Read data from the outside world
- Motion sensor: The motion sensor allows you detect movement.
- Light sensor: this allows you to "measure" the quantity of light in the outside world.
- Humidity and temperature sensor: this is used to measure the humidity and temperature.
- Ultrasonic sensor: this sensor allows to determine the distance to an object through sonar.
- Shields – an extension of the Arduino
- Shields are boards that will expand the functionalities of your Arduino. You just need to plug them over the top of the Arduino. There are countless types of shields to do countless tasks.

7.3 Installing the Software (Arduino IDE)

The Arduino IDE (Integrated Development Environment) is where you develop your programs that will tell your Arduino what to do. You can load new programs onto the main chip, the ATmega328p, via USB using the Arduino IDE. To download your Arduino IDE, browse on the following link: https://www.arduino.cc/en/Main/Software. Select which Operating System you're using and download it. We won't go into much detail on how to install this software, since the official Arduino web site does a great job explaining how to do it in all three Operating

Systems – Windows, Mac and Linux. When you first open the Arduino IDE, you should see something similar to the Figure 7-2 below:


sketch_jun21a | Arduino 1.8.3
File Edit Sketch Tools Help
sketch_jun21a
Arduino/Genuino Uno on COM1

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Figure 7.2: Arduino IDE

7.3.1 Connecting Arduino Uno Learning Board

After connecting your Arduino with a USB cable, you need to make sure that Arduino IDE has selected the right board you are using. In our case, we're using Arduino Uno, so you should go to Tools > Board: "Arduino/Genuino Uno" > Arduino/Genuino Uno as shown in Figure 7-3. Then, you should select the right serial port where your Arduino is connected to. Go to Tools > Port and select the right port as shown in Figure 7-4 and Figure 7-5 shows the layout of Arduino IDE.

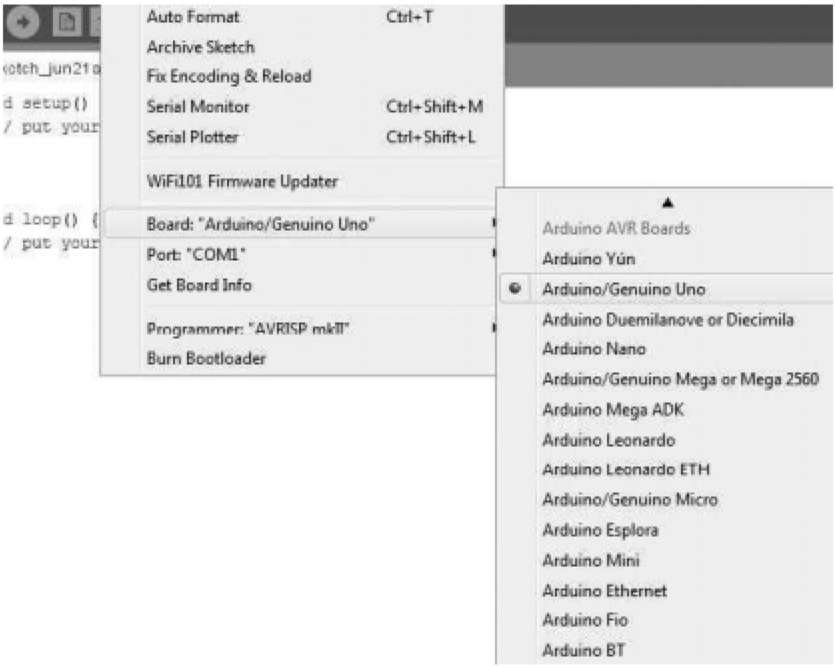
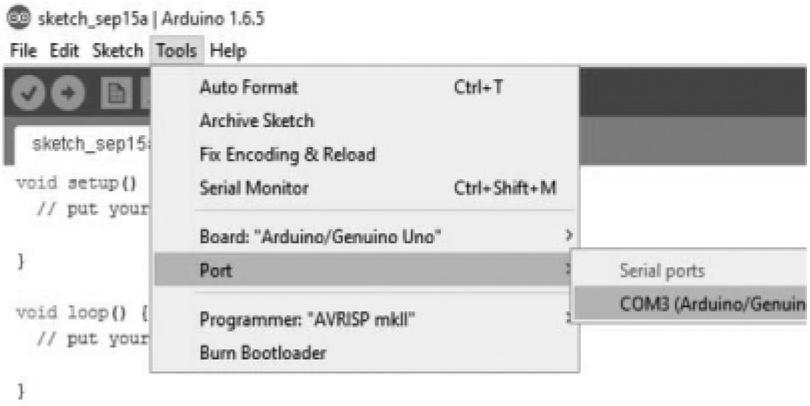


Figure 7-3: Selecting the right Board


sketch_sep15a | Arduino 1.6.5
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board: "Arduino/Genuino Uno"
Port
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
COM3 (Arduino/Genuin

10 selecting the right port

Figure 7-4: showing the layout of Arduino IDE

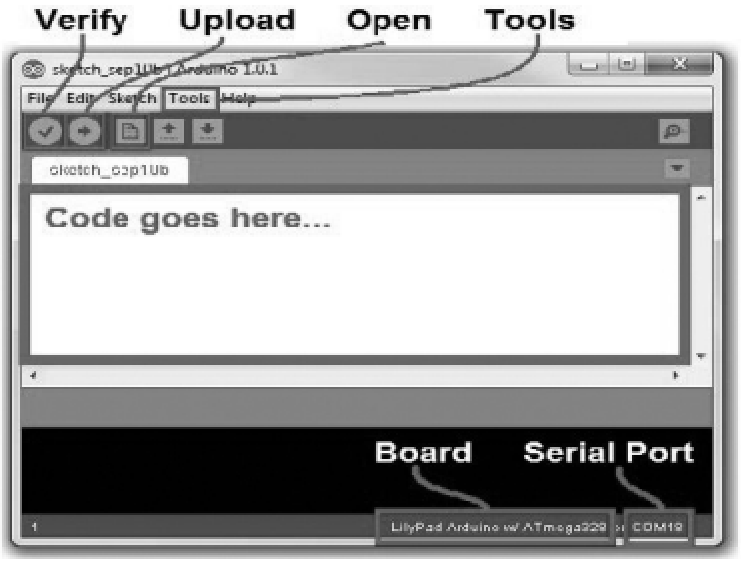

Verify
Upload
Open
Tools
sketch_sep10b | Arduino 1.0.1
File Edit Sketch Tools Help
sketch_sep10b
Code goes here...
Board
Serial Port
LilyPad Arduino w/ ATmega328
COM19

Figure 7-5: Layout of Arduino Uno IDE

The Toolbar buttons and functions of each button are as shown in Table 7-1 as follows:

Table 7-1: Toolbar options in Arduino IDE

Verify/Compile	**Checks the code for errors**
Stop	Stop the serial monitor,or un-highlight other buttons
New	Creates a new blank sketch. enter a name and a location for your sketch
Open	Shows a list of Sketches in your sketchbook
Upload	Uploads the current Sketch to the Arduino. You need to make sure that you have the current board and port selected (in the Tools menu) before Uploading.
Serial Monitor	Display Serial data being sent from the Arduino
Verify/Compile	Button is used to check that your code is correct, before you upload it to your Arduino.
Stop button	Will stop the Serial Monitor from operating. If you need to obtain a snapshot of the serial data so far examined.

Breadboard for prototyping Arduino Uno Circuits

In order to keep your circuit organized you need to use a breadboard, pictured below in **Figure 7-6**. The breadboard allows you to connect components together by plugging them into the little holes. The key is to understand how the holes are connected. As you can see in the diagram, the holes in a column (when oriented as shown in the picture) are connected together. So to connect components together you need to plug the leads you want connected into the same column. Note that the columns are not connected across the "trench" in the center of the board. Also notice that as the long rows at the top and bottom are connected together. These are typically used to create "rails". These are typically used for grounds and supply voltages you might need to connect many components to. Notice some rows are marked (+) and some(-). These are just markings. The row will be set at whatever voltage YOU connect to it.

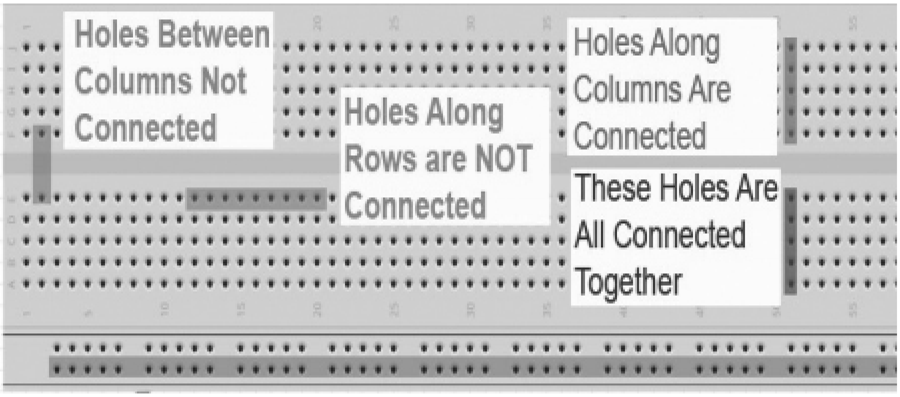

Holes Between Columns Not Connected
Holes Along Rows are NOT Connected
Holes Along Columns Are Connected
These Holes Are All Connected Together

Figure 7-6: Breadboard for prototyping Arduino Uno circuits

Technical Specifications of Arduino UNO which is used in this book is given below in

Table 7-2: Technical Specifications of Arduino UNO

Technical Specifications: Microcontroller Arduino UNO	A Tmega 328P
Operating Voltage	5V
Input Voltage (recommended)	7–12V
Input Voltage (limit)	6–20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pin	6
Analog Input Pins	6
DC Current per I/O Pin	20mA
DC Current for 3.3V Pin	50mA
Flash Memory	32 KB (A Tmega 328P) of which 0.5 KB used by bootloader
SRAM	2KB (A Tmega328P)
EEPROM	1 KB (A Tmega 328P)
Clock Speed	16MHz

7.4 Fundamentals of Arduino Programming

This section in Table 7-3 explains the basic structure of Arduino programming with respect to usage of variables, constants, control flow statement and finally the predefined functions used to read analog and digital inputs.

Table 7-3: Fundamentals of Arduino Programming

Structure	The structure of Arduino programming contains of two parts as shown below
	`void setup()          //Preparation function used to declare variables`
	`{                     //First function that runs only one in the program`
	`      Statement(s);   //used to set pins for serial communication`
	`}`
	`void loop()           //Execution block where instructions are executed repeatedly`
	`{                     //this is the core of the Arduino programming`
	`      Statements();   //Functionalities involve reading inputs, triggering outputs etc.`
	`}`

void setup()	`void setup()`
	`{`
	`      pinMode(pin, INPUT);   //'pin' configure as inpu`
	`}`

void loop()	`void loop()                    //After calling setup(),loop() function does its task`
	`{`
	`      digitalWrite(pin, HIGH);   //sets 'pin' ON`
	`      delay(10000);              //pauses for ten thousand mili second`
	`      digitalWrite(pin, LOW);    //sets 'pin' OFF`
	`      delay(10000);              //pauses for ten thousand mili second`
	`}`

Functions	A function is a piece of code that has a name and set of statements executed when function is called. Functions are declared by its type followed with name of a function. Syntax: type functionName(parameters)

	{
	 Statement(s);
	 }
	 Example:
	 int delayvar()
	 {
	 int var; //create temporary variable var
	 var=analogRead(potent); //read from potentiometer
	 var=var/4; //convert the value of variable var
	 return var; //return var
	 }

{ } curly braces	They define beginning and end of function blocks, unbalanced braces may lead to compile errors.
semicolon	It is used to end a statement and separate elements of a program. Syntax: int x=14;
/*…..*/ block comments	Multiline comments begin with /* with a description of the block and ends with */. Syntax: /*This is an enclosed block of comments Use the closing comment to avoid errors*/
//line comments	Single line comment begins with // and ends with next instruction followed. Syntax: //This is a Single line comment

Variables	A variable is a way of storing value for later use in the program. A variable is defining by its value type as an int, long, float etc by setting a specified name and optionally assigning an initial value. A global variable can be seen in every part of the program which is declared at the beginning of program before setup() function. A local variable is defined inside a function in which it was declared.
	 Example:
	 int var; //variable 'var' visible to all functions
	 void setup()
	 {
	 //nothing is required
	 }
	 void loop()

	{
	for(int local=0;local<5;)
	{
	local++;//variable 'local' is only visible within for loop
	}
	float local_f; //variable 'local_f' is visible only inside the loop
	}

Data Types

Data type	Syntax	Range
Byte	byte x = 100;	0–255
Int	int y = 200;	32767 to–32768
Long	long var = 8000;	2147483647 to –2147483648
Float	float x = 3.14;	3.4028235E+38to – 3.4028235E+38
arrays	int myarray []={10,20,30,40}	Size depends on the data type associated with declaration.

Operators

Operstor	Syntax and its usage
Arithmetic operators	x = x+5;

(+,–,/,*)	y–y–6;
	z=z*2;
	p=p/q;

Assignment operators
(=,++,—,+ =,=,*=./=)
	x++; //same as x=x+1
	x+=y;//same asx=x+y
	x–=y; //same asx=x–y
	x*=y; //same as x=x*y
	x/=y; //same as x=x/y

Comparison operators
(==,!=,<,>,<=,<=)
	x==y //xis equal to y
	x!–y //xis not equal to y
	x<y //xis less that y
	x!–y // x is not equal to y

Logical operators (&&,||,!)	x>2&&x<5 //Evaluates to true only if both expression are true x>2||y>2 //Evaluates to true if any one expression is true !x>2 //true if only expression is false

Constants	Constants	Usage

	TRUE/FALSE	Boolean constants true=2 and false=0 defined in logic levels.
		if(b==TRUE)
		{
		//do something
		}

	INPUT/OUTPUT	Used with pinMode () function to define levels. pinMode (13,OUTPUT):
	HIGH/LOW	Used to define pin levels HIGH–1,ON,5 volts LOW –0,OFF, 0 volts Digital Write (13,HIGH);
Flow control Statements		

if	if(some_variable == value)
	{
	Statement(s); //Evaluated only if comparison results in a true value
	}	

if…else	if(input==HIGH)
	{
	Statement(s); //Evaluated only if comparison results in a true value
	}
	else
	{
	Statement(s); //Evaluated only if comparison results in a false value
	}	

for	for(initialization;condition;expression)
	{
	Dosomething; //Evaluated till condition becomes false
	}
	for(int p=0;p<5;p++) //declares p, tests if less than 5, increments by 1	

	{
	digitalWrite(13,HIGH); //sets pin 13 ON
	delay(250); // pauses for ¼ second
	digitalWrite(13,LOW); //sets pin 13 OFF
	delay(250); //pause for ¼ second
	}	

while	While loop executes until the expression inside parenthesis becomes false.
	while(some_variable ?? value)
	{
	Statement(s); //Evaluated till comparison results in a false value
	}	

do...while	Bottom evaluated loop, works same way as while loop but condition is tested at the end of loop.
	do
	{
	Dosomething;
	}while(somevalue);	

Digital and Analog input output pins and their usage		
Digital i/o	Methods	Usage

	pinMode (pin, mode)	Used in setup () method to configure pin to behave as INPUT/OUTPUT
		pinMode(pin, INPUT) //pin set to INPUT
		pinMode(pin, OUTPUT) // pin set to OUTPUT

	Digital Read (pin)	Read value from a specified pin with result being HIGH/LOW Val=digital Read(pin); // Val will be equal to input pin
	Digital Write(pin, value)	Outputs to HIGH/LOW on a specified pin. Digital Write(pin, HIGH); //pin is set to HIGH

	Example	int x=13; //connect 'x' to pin 13
		int p=7; //connect push button to pin 7
		int val=0; //variable to store the read value
		void setup()

		{
		pin MODE(x, OUTPUT); // sets 'x' as OUTPUT
		}
		void loop()
		{
		val=digital Read (p); //sets 'value' to 0
		digital Write (x,val); //sets 'x' to button value
		}

Analog i/o	**Methods**	**Usage**
	analog Read(pin)	Reads value from a specified analog pin works on pins 0–5. val=anolog Read (pin); // 'val' equal to pin
	analog Write (pin,value)	Writes an analog value using pulse width modulation (PWM) to a pin marked PWM works on pins 3,5,6,9,10.

	Example	int x=10; //connect 'x' to pin 13
		int p=0; //connect potentiometer to analog pin 7
		int val; //variable for reading
		void setup () { } //No setup is needed
		void loop()
		{
		val=analog Read (p); //sets 'value' to 0
		val/=4;
		analog Write (x,val); //outputs PWM signal to 'x'
		}

time	**Methods**	**Usage**
	deaay (ms)	Pauses for amount of time specified in milliseconds. deay(1000); //waits for one second
	millis()	Returns the number of milliseconds since Arduino is running. val=millis();//'val' will be equal to millis()

math	**Methods**	**Usage**
	min(x,y)	Calculates minimum of two numbers val=min (val,10); //sets 'val' to smaller than 10 or equal to 10 but never gets above 10.
	max(x,y)	val=max (val, 10); //sets 'val' to larger than 100 or 100.
random	**Methods**	**Usage**
	Random Seed (value)	Sets a value / seed as starting point for ranction.
	Random (min, max)	Allows to return numbers within the range specified by min and max values. val=random(100,200); //sets 'val' to random number between 100-200

	Example	intr number ; //variable to store random value
		int x=10;
		void setup()
		{
		randomseed (millis()); //set millis() as seed
		number -random(200); //random number from 0–200
		analog Write(x, number); //outputs PWM signal
		delay (500);
		}

Serial	**Methods**	**Usage**

	Serial. begin(rate)	Opens serial port and sets the baud rate for serial data transmission.
		void setup()
		{
		Serial begin(9600); //sets default rate to 9600 bps
		}

	Serial. println (data)	Prints data to the serial port Serial println (value); //sends the 'value'; //sends the ' value ' to serial monitor.

7.4.1 Difference between Analog, Digital and PWM Pins

In analog pins, you have unlimited possible states between 0 and 1023. This allows you to read sensor values. For example, with a light sensor, if it is very dark, you'll read 1023, if it is very bright you'll read 0 If there is a brightness between dark and very bright you'll read a value between 0 and 1023.

In digital pins, you have just two possible states, which are on or off. These can also be referred as High or Low, 1 or 0 and 5V or 0V. For example, if an LED is on, then, its state is High or 1 or 5V. If it is off, you'll have Low, or 0 or 0V.

PWM pins are digital pins, so they output either 0 or 5V. However these pins can output "fake" intermediate voltage values between 0 and 5V, because they can perform "Pulse Width Modulation" (PWM). PWM allows to "simulate" varying levels of power by oscillating the output voltage of the Arduino.

Figure 7-7 shows the representations of Analog, Digital and PWM pins of Arduino.

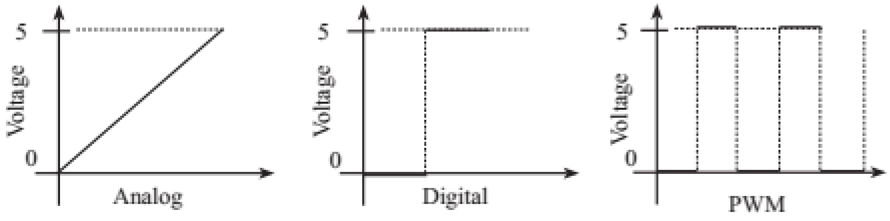


Figure 7-7: Difference between Analog, Digital and PWM Pins

7.5 Introduction to Communications

Arduino has Serial, SPI and I2C communication for data transfer which are explained below.

Serial (UART) communications:

Serial communication on Arduino pins Tx/Rx uses TTL logic levels which operates at either 5V/3.3V depending the type of the board used. Tx/Rx pins should not be connected to any source which operates more than 5V which can damage the Arduino board. Serial communication is basically used for communication between Arduino board and a computer or some other compatible devices. Every Arduino board will have at least one serial port known as UART.

Serial communicates on digital pins Rx(pin 0) and Tx(pin 1) with the computer via USB, pin 0 and pin 1 cannot be used for digital input or output. The built in serial monitor can be used to communicate with an Arduino board by selecting same baud rate that is used in the call to begin () which will come across in the later part of the chapter.

7.5.1 SPI communications

Serial communication Interface (SPI) is a synchronous data protocol used by large microcontrollers for communicating with one or more peripheral devices for a shorter distance and also used for communication between two devices. With SPI there will be always one master device which is a microcontroller like Arduino which controls the functionalities of other peripheral devices. Devices have three lines in common which are as follows

❖ MISO (Master in Slave Out)- Slave line for sending data to the master.

❖ MOSI (Master Out Slave In)- Master sending data to peripherals

❖ SCK Serial clock) - clock pulses which synchronize data transmission generated by the master

And one of the specific line for every device is

❖ SS (slave select) - pin on each device that the master can use to enable and disable specific devices.

When device SS pin is low, communication happens with the master, if SS pin is high device ignores the maser. This allows multiple SPI devices sharing the the same MISO, MOSI and CLK lines.

To program a new SPI device some key points to be noted which are

❖ Maximum SPI speed of the device used?

❖ How data is shifted like MSB/LSB?

❖ Data clock is idle when high/low.

7.5.1.1 I^{2}C communications

Inter-Integrated circuit or I2C (I squared C) is one of the best protocol used when a workload of one Arduino (Master Writer) is shared with another Arduino (Slave receiver). The I2C protocol uses two lines to send and receive data which are a serial clock pin (SCL) which writes data at regular intervals and a serial data pin (SDA) over which data sent between devices. When the clock signal changes from LOW to HIGH the information, the address corresponds to a specific deviceand a command is transferred from board to the I2C device over the SDA line. This information is sent bit by bit which is executed by the called device, executes and transmits the data back. If the device require execution from another a slave device, the data is transferred to the board on the same line using pulse generated from Mater on SCL as timing. Each slave should have unique identity and both Master and slave turns out communicating on the same data line. In this way many of the Arduino boards are communicated using just two pins of microcontroller with each unique address of a device.

7.6 Example Modules on Arduino

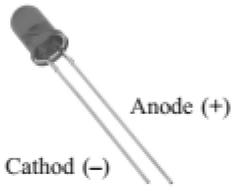
In this section we discuss few prototyping examples by interfacing Arduino using LED, Serial Monitor Potentiometer, Servo motor, ultrasonic sensor, LCD, Bluetooth Module and SD card.

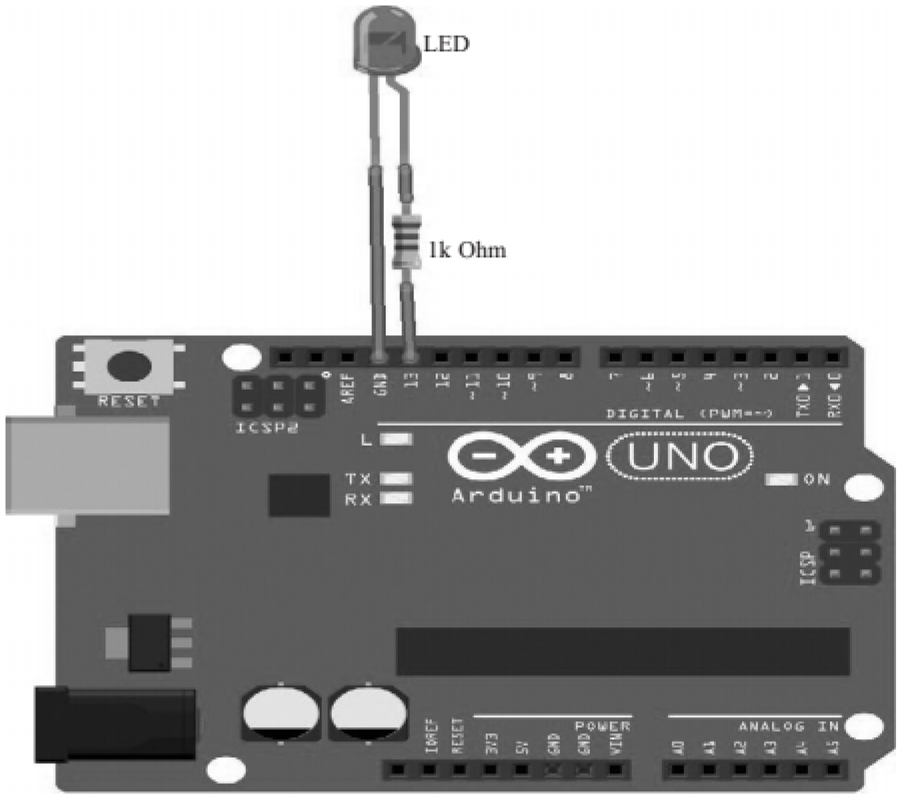
Interfacing programs on Arduino using LED

7.6.1 Programs to interface LED switch and Potentiometer

In this section we discuss three programs on Arduino by interfacing it with an LED which are as follows:

1. Blinking an LED
2. Toggle the state of LED using Switch
3. Traffic light simulation for pedestrians
4. Create Dimmable LED using Potentiometer
5. Toggle the state of LED using switch

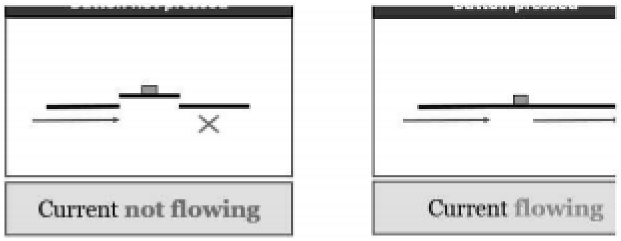
Program #1	**Blinking an LED**
Components required	1-LED, 1-KΩ resistor, Jumper wires, Breadboard
LED	 The longest lead is the anode and the shortest is the cathode
	LED Working: In simple terms here's how an LED works, An LED is a basic semiconductor device which is a p-type semiconductor material containing 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.

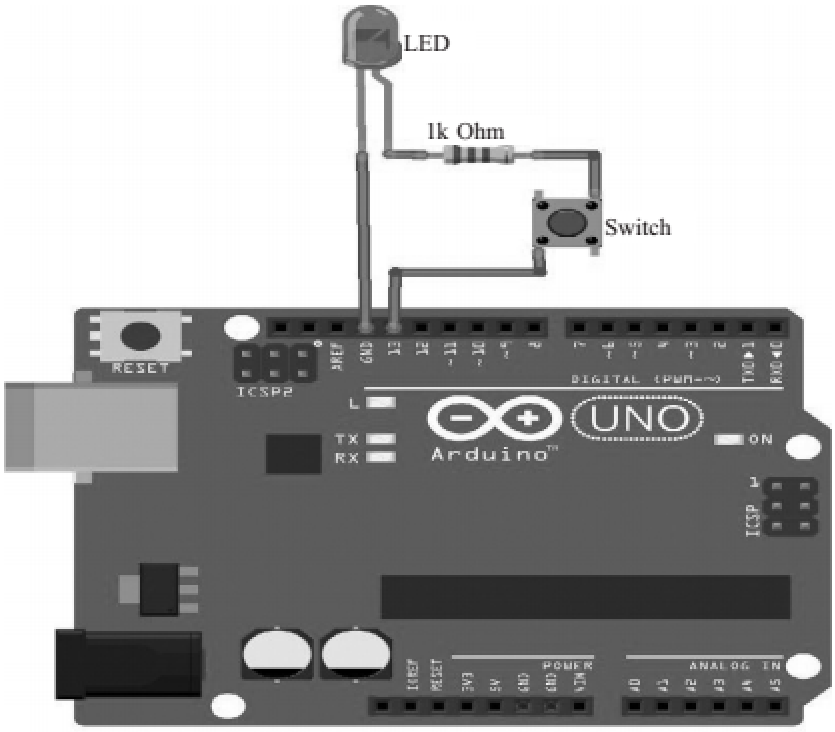
	The type of material and process of creating the n-type and p-type materials dictate the color of the photons as well as the efficiency and other performance characteristics of the LED. After processing the material into an LED chip, the chip is installed in a package that allows electrical connection and directs as much light as possible in the desired direction
Circuit Diagram  LED 1k Ohm RESET ICSP2 AREF GND 13 12 ~11 ~10 ~9 8 7 ~6 ~5 4 ~3 2 TX▶1 RX◀0 DIGITAL (PWM=~) L TX RX UNO Arduino ON ICSP IOREF RESET 3V3 5V GND GND VIN POWER ANALOG IN A0 A1 A2 A3 A4 A5	
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.

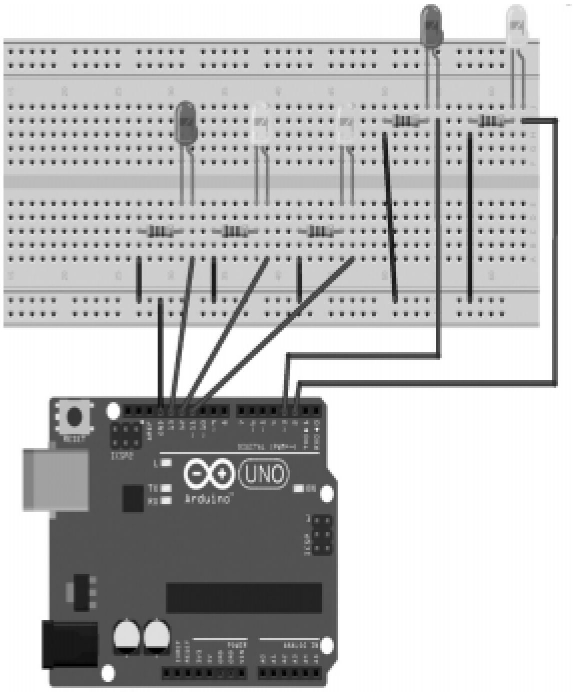
Program # 2	**Toggle the state of LED using Swith**
Components required	1–LED, 1–K Ω resistor, 1–push button, Jumper wires, Breadboard
How a push button works?	 Here an **open push button** mechanism is used. In Normal state(not pushed) of the button current doesn't flow, only when button is pushed flow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

Circuit diagram	
 LED 1k Ohm Switch RESET ICSP2 AREF GND 13 12 ~11 ~10 ~9 8 7 ~6 ~5 4 ~3 2 TX▶1 RX◀0 DIGITAL (PWM=~) L TX RX UNO Arduino™ ON ICSP IOREF RESET 3V3 5V GND GND VIN POWER ANALOG IN A0 A1 A2 A3 A4 A5	
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.

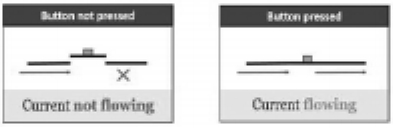
Program # 3	**Traffic light Simulation for Pedestrians**
Components required	2-Red LED, 2-Green LED, 1-Yellow LED, 5-220Ω resistor, Jumper wires, Breadboard
Circuit diagram 	

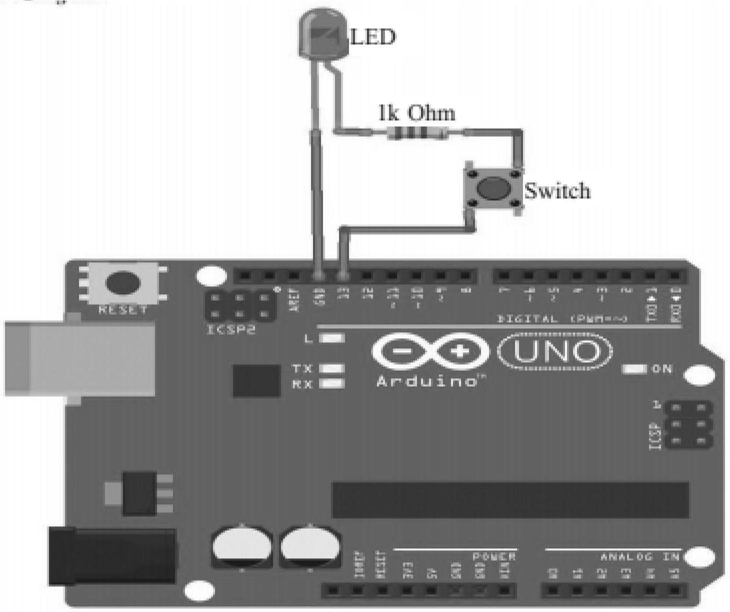
Description	Now, let's take the previous Led blink program to simulate an traffic light application for pedestrians. Design the circuit as shown in the above figure. So, we will need to add another red light and another green light. One thing to notice here is when pedestrian traffic light is green the car traffic light should be red and when car traffic light is green/yellow, the pedestrian traffic light should be red.

Code	`// Declare the variables for different colors of LEDs.`
	`int red_vehicle = 13;`
	`int yellow_vehicle = 12;`
	`int green_vehicle = 11;`
	`int green_Pedestrian =2;`
	`int red_Pedestrian= 3;`
	`void setup( )`
	`{`
	`// Initialize the pins for output`
	`pinMode(red_vehicle, OUTPUT);`
	`pinMode(yellow_vehicle, OUTPUT);`
	`pinMode(green_vehicle, OUTPUT);`
	`pinMode(red_Pedestrian, OUTPUT);`
	`pinMode(green_Pedestrian, OUTPUT);`
	`}`
	`void loop( )`
	`{`
	`digitalWrite(green_Vehicle, HIGH); // green LED turns ON`
	`digitalWrite(red_Pedestrian, HIGH);`
	`delay(5000);`
	`digitalWrite(green_Vehicle, LOW); // green LED turns OFF`
	`digitalWrite(yellow_Vehicle, HIGH); // Yellow LED turns ON for 2second.`
	`delay(2000);`
	`digitalWrite(yellow_Vehicle, LOW); // yellow LED will turn OFF`
	`digitalWrite(red_Pedestrain, LOW);`
	`digitalWrite(red_Vehicle, HIGH); // Red LED turns ON for 5 seconds`

	digitalWrite (green_Pedestrian, HIGH);
	delay(5000);
	digitalWrite(red_Vehicle, LOW); // Red LED turns OFF
	digitalWrite(green_Pedestrian, LOW);
	}

Output	Run the above code and observe the traffic light simulation.

Program #4	**Toggle the state of LED using Switch**
Components required	1-LED, 1-KΩ resistor, 1-push button, Jumper wires, Breadboard
How a push button works?	 Button not pressed Current not flowing Button pressed 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	

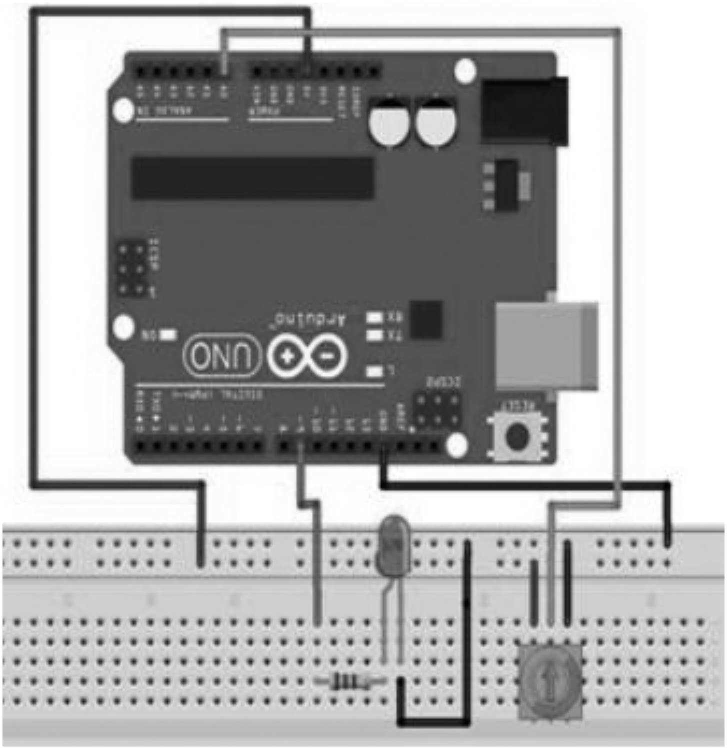


Description	In this program LED is connected to digital pin 13 via switch and shown in circuit diagram above. An infinite loop runs over the loop() function to toggle the state of LED to blink, when an input is given through a switch button press/release and incurring the state switch mechanism by constantly 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 #5	**Creating a Dimmable LED using Potentiometer**
Components Required	1-LED, 220Ω resistor, 1-Potentiometer, Jumper wires, Breadboard
Description	In this program we dim the LED based on the value read from the potentiometer. A "0" value from potentiometer is a "0V" and a value "1023" from potentiometer is a "5V", which means we need to write a value of 255. Hence we need to scale our read values from the potentiometer which falls between 0 to 1023 to suitable write values to be between 0 to 255 using the below given formulae. write value=(255/1023)* read_value

Circuit	
	
	How Potentiometers work: Potentiometers are often referred to as speed paths and our devices for control of speed voltage or frequency. resistive element and a contact that is often referred to as the wiper. resistive element and a contact that is often referred to as the wiper. . As voltage is applied to the resistive ring, the wiper then control the amount of resistance used, since the dial directly controls the wiper we only let enough voltage pass through according to the amount of resistance that is applied. The method used basically is Ohm's law where voltage is equal to the current multiplied by resistance, if the amperage is constant adjusting the value of resistance will directly affect the voltage used using a potentiometer and a drive controlling a motor. The more voltage applied from the potentiometer will signal the drive to put forth more voltage to the LED.

Code	`//Declaring the pins corresponds to an LED-to pin 9 and a Potentiometer-to //pin A0`
	`int pot_Pin= A0;`
	`int LED_Pin= 9;`
	`int read_Value;  // To store the value read by potentiometer`
	`int write_Value; // To write the value to LED`
	`void setup( )`
	`{`
	` pinMode(pot_Pin, INPUT);`
	`  pinMode(LED_Pin, OUTPUT);`
	`   Serial.begin(9600);`
	`}`
	`void loop( )`
	`{`
	`read_Value = analogRead(pot_Pin);  //Potentiometer reading`
	` write_Value = (255./1023.) * readValue; //Write value for LED is calculated`
	`  analogWrite(LEDPin, writeValue);      //Write to the LED`
	`  Serial.print("The writing vlues to the LED is ");  //Debugging purpose`
	` Serial.println(write_Value);`
	`}`

Output	Run the above code, you should be able to set the brightness from the potentiometer. You read the voltage from the potentiometer and then scale the value you write to the LED based on the reading from the potentiometer

7.6.2 Programs to interact with Serial Monitor of our Computer Screen

In this section we discuss the way of interacting with a Serial monitor with a Arduino **"To print the status of our computer Screen".**

Program #1	**Printing the Status of the computer screen using Serial Monitor**
Description	Now, let's introduce the interaction with the Serial monitor. In this program we perform Arithmetic operations on the variables defined in the program, variables are initialized inside the program. Serial monitor communication will be processed when we call the method Serial. begin( ) with appropriate Baud rate. Serial monitor displays the desired message of a program using the method Serial.print( ) method.

Baud Rate	In electronics and telecommunication, baud rate is the unit for symbol rate or tweak rate in symbols every second. Sets the information rate in bits every second (baud) for serial information transmission. For communicating with the PC, utilize one of these rates: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, or 115200. You can, in any case, indicate different rates - for instance, to convey over pins 0 and 1 with a segment that requires a specific baud rate. **Syntax:** Serial.begin(speed) /* to communicate between your computer and Serial monitor */
	Serial.begin(speed, config) Serial.print() #To print desired message on the Serial monitor

Code	//In this program we compute basic aritmetic operations to print the result on //to the Serial monitor.
	int a = 5;
	int b = 10;
	int c = 20;
	void setup( ) // run once, when the sketch starts
	{
	 Serial.begin(9600); // set up Serial library at 9600 bps
	  Serial.println(“Here is some math: “);
	  Serial.print(“a = “);
	  Serial.println(a);
	  Serial.print(“b = “);
	  Serial.println(b);
	   Serial.print(“c = “);
	  Serial.println(c);
	  Serial.print(“a + b = “); // add
	  Serial.println(a + b);
	  Serial.print(“a * c = “); // multiply
	  Serial.println(a * c);

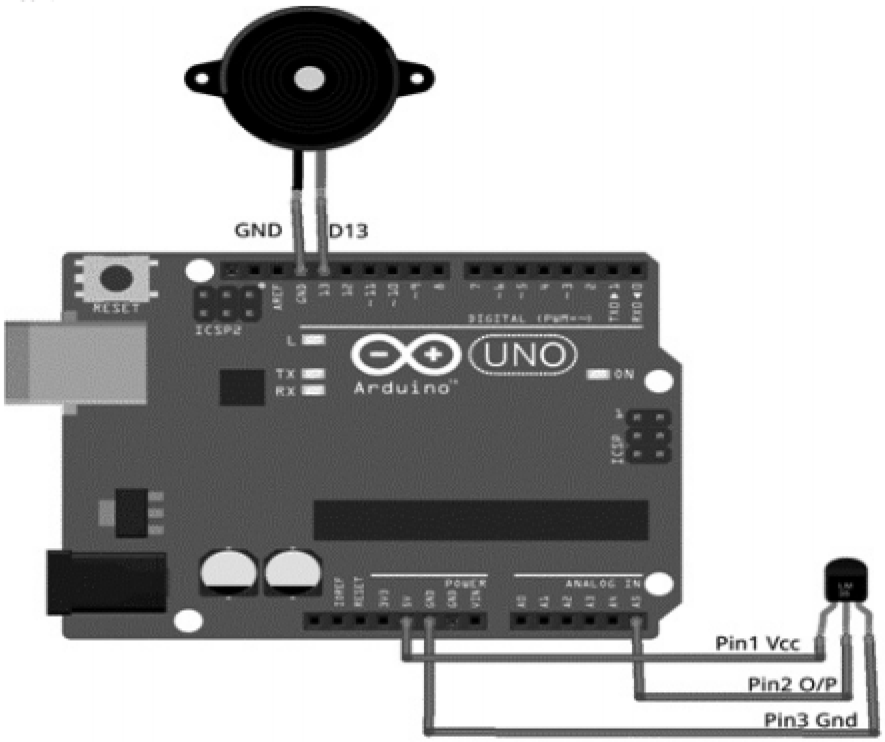
	Serial.print("c / b = "); // divide
	Serial.println(c / b);
	Serial.print("b - c = "); // subtract
	Serial.println(b - c);
	}
	void loop( ) // we need this to be here even though its empty
	{ }

Output	Run the above code and open up a serial monitor to observe the result of arithmetic operations.

7.6.3 Interfacing Sensors to the Arduino

In this section we illustrate example prototypes using different kinds of Sensors such as **Temperature Sensor, Light Sensor, Ultrasonic distance sensor** and a **Line sensor** (infrared).

Program #1	**Interfacing Temperature Sensor**		
Components Required	Buzzer, LM35 Temperature Sensor, Jumper wires, Breadboard		
LM35 Temperature Sensor:	The LM35 series are the gadgets with precision integrated circuit temperature whose yield voltage falls directly corresponding to the Centigrade temperature. ❖ Calibrated Directly in Celsius (Centigrade) ❖ Operates from 4 V to 30 V ❖ Ranges are evaluated from Full −55°C to 150°C. ❖ Suitable for Remote Applications ❖ Used in Battery Management		
Pin Description:	Pin No	Function	Name
	1	Supply voltage; 5V(+35Vto–2V)	Vcc
	2	Output voltage(+6Vto–1V)	Output
	3	Ground(0V)	Ground
Description	In this program an LM35 temperature sensor and a piezo buzzer that beeps a sound when LM35 temperature sensor reads out a temperature which exceeds 32°C. In this example the LM35 is connected to analog pin number A5 since it reads an analog input, and piezo buzzer is connected to GPIO pin 13 which is initialized as output pin in the setup function.		

Circuit  GND D13 RESET ICSP2 DIGITAL (PWM=~) UNO Arduino ON ICSP POWER ANALOG IN Pin1 Vcc Pin2 O/P Pin3 Gnd	

Code	//initialize a variable temPin to Analog pin A%
	int temPin = A5;
	//Set buzzer to pin 13 as OUTPUT
	int buzzer = 13;
	//Variable to store the temperature read
	int value;
	void setup()
	{
	//Initialize Serial baud rate to 9600

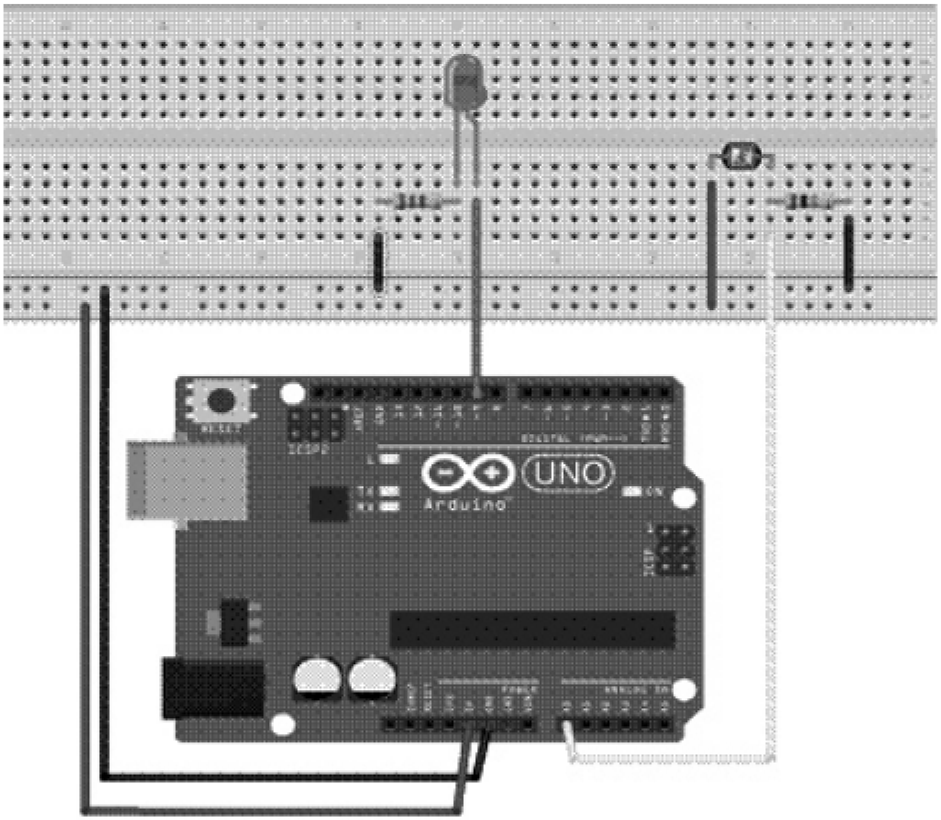
	pinMode(buzzer, OUTPUT);
	}
	void loop()
	{
	//Read temperature value on pin A5 by analogRead() method
	value = analogRead(temPin);
	//Conversion of temperature value read
	float mvalue = (value/1024.0)*5000;
	//Conversion of Temperature to celsius
	float celsius = mvalue/10;
	//conversion of temperature to Fahrenheit
	float fahrenheit = (celsius*9)/5 + 32;
	//print the celsius value onto the serial monitor
	Serial.print(cel);
	//check if the read temperature is greater than 32 degree celsius
	if(cel>32)
	{
	//trigger HIGH value on buzzer
	digitalWrite(buzzer, HIGH);
	//delay for 1 second
	delay(1000);
	// trigger LOW value on buzzer
	digitalWrite(buzzer, LOW);
	//delay for 2 second
	delay(2000);
	//trigger HIGH value on buzzer
	digitalWrite(buzzer, HIGH);
	//delay for 1 second
	delay(1000);
	// trigger LOW value on buzzer

	digitalWrite(buzzer, LOW);
	//delay for 2 second
	delay(2000);
	}
	rin//P t the temperature onto a serial monitor
	Serial.print("TEMPRATURE = ");
	//Print the current read temperature onto serial monitor
	Serial.print(cel);
	//Formatting output to print the temperature
	Serial.print("*C");
	//moving cursor position onto a newline to print next read temperature //onto a next line
	Serial.println();
	}

Output	Whenever the room temperature exceeds 32°C, the Buzzer keeps on beeping until the temperature goes below 32°C. NOTE: Select 9600 baud rate in Serial monitor to see the output.

Program #2	**Automatic lights with light sensor**
Components Required	1x LED , 1x 220Ω resistor , 1x photoresistor , 1x 10kΩ resistor, Jumper wires, Breadboard
What is a photoresistor?	A photoresistor is a light-dependent resistor. The resistance of a photoresistor decreases with increasing of light intensity. So: ❖ When there is light, the resistance decreases, we will have more current flowing. ❖ When there is no light, the resistor increases, we will have less current flowing.
Description	In this program we input data from photoresistor, wherein when there isn't light, LED will turn on, if there is light LED will turn off. Since we read analog value which falls in the range 0 to 1023, however it should converted back to the range between 0 to 255 to set a desired voltage for LEDs we use map() function in our program. Here we give its syntax. Map(value, fromLow, fromHigh, toLow, toHigh).

Circuit



Code

```
int led_Pin = 9;
int led_Brightness = 0;
int sensor_Pin = A0;
int sensor_Value = 0;
void setup(void) {
pinMode(led_Pin, OUTPUT);
// Send some information to Serail  monitor
Serial.begin(9600);
}
void loop(void)
```

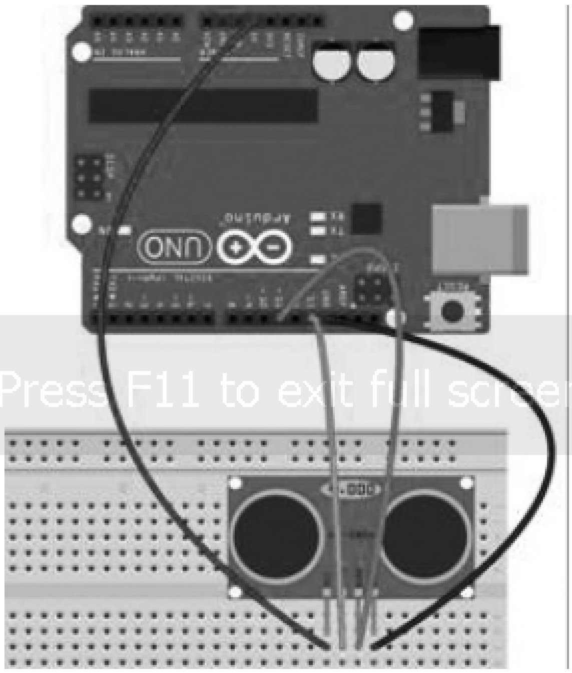
	{
	sensor_Value = analogRead(sensor_Pin);
	Serial.print("Sensor reading: ");
	Serial.println(sensor_Value);
	// LED gets brighter the darker it is at the sensor
	// that means we have to -invert- the reading from 0-1023 back to 1023-0
	sensorValue = 1023 - sensorValue;
	//now we have to map 0-1023 to 0-255 since thats the range analogWrite //uses
	ledBrightness = map(sensorValue, 0, 1023, 0, 255);
	analogWrite(ledPin, ledBrightness);
	delay(50);
	}

Output	With the code above, notice that LED glows on the ambience of light in the environment.

Program #3	**To Measure Speed of Sound using Ultrasonic Sensor**
Components Required	1- HC-SR04 -ultrasonic sensor, Jumper wires, Breadboard
Description	Here we measure the speed of sound using Arduino interfaced with ultrasonic sensor. This sensor measure the time it takes an ultrasonic to ping to an object and come back. The time to leave and come back from the target object depends on the speed of sound and distance to the target.
How Ultrasonic sensor works?	Working of Ultrasonic sensor? ❖ Trigger LOW-HIGH-LOW sequence on the pin which creates a high pitched ultrasonic tone which sent out from the sensor, which will go out and bounce off the first thing in front of it and back to the sensor. ❖ The sensor will output HIGH on the pin and length of pulse in microseconds indicates time it took the ping to travel to target and return. ❖ Measure the length of the pulse using pulseIn command. ❖ Calculate the speed of sound by

	distance= rate * time **rate = time/distance** ❖ convert this to miles per hour as follows: (rate in inches/mircrosecond)*(1000000 microsecond/second)* (3600 seconds/hour)*(1 mile/63360 inches)

Circuit



Ultrasonic sensor connection to Arduino

❖ Connect VCC pin of sensor to the 5V pin of Arduino.

❖ Connect GND pin of sensor to GND pin of arduino.

❖ Connect Trig pin of sensor to pin 13 on the arduino

❖ Connect Echo pin of sensor to pin 11 of arduino.

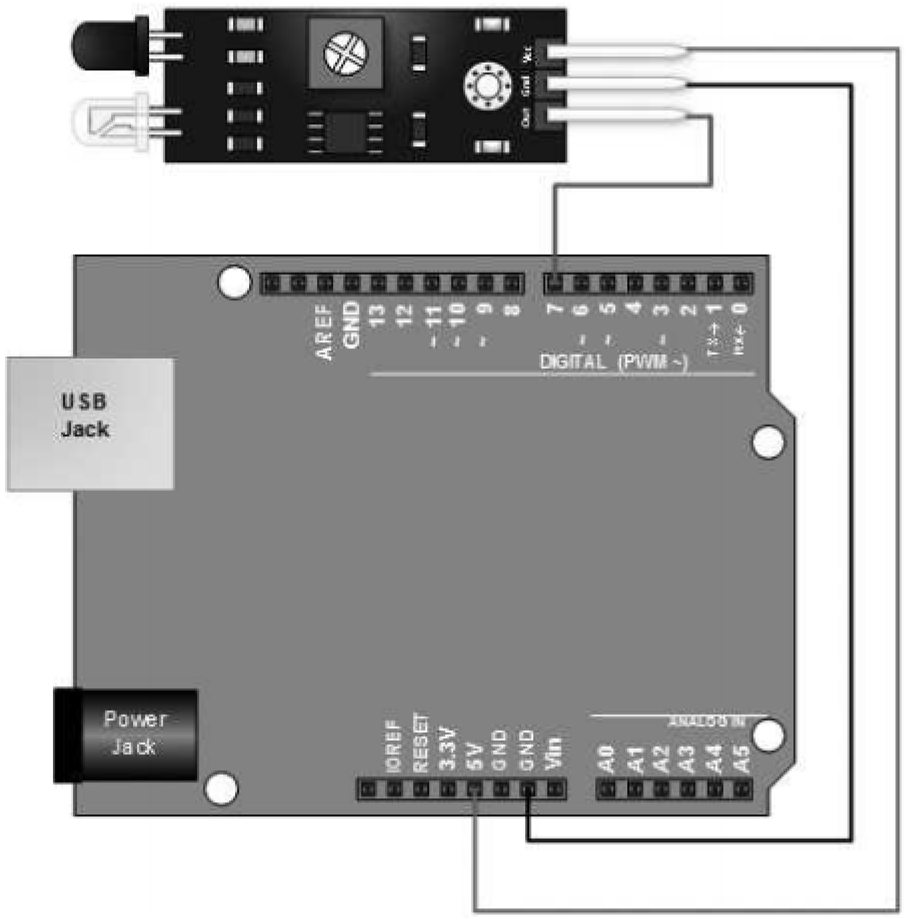
Code	int trig_Pin=13; //Connect Trip pin of sensor to 13 pin of Arduino
	int echo_Pin=11; //Connect sensor echo pin to 11 pin of Arduino
	float pinging_Time;
	float speed_Of_Sound;

```
int target_Distance=6; //Target distance in inches
void setup() {
  Serial.begin(9600);
pinMode(trig_Pin, OUTPUT);
  pinMode(echo_Pin, INPUT);
}
void loop() {
  digitalWrite(trig_Pin, LOW); //trigpin set to LOW
  delayMicroseconds(2000);
  digitalWrite(trig_Pin, HIGH); //trigPin to high
  delayMicroseconds(10);
digitalWrite(trig_Pin, LOW); //Send ping
   pingTime = pulseIn(echo_Pin, HIGH);  /*pingTime is presented in
microceconds */
                  speedOfSound          =          (targetDistance*2)/pinging_
Time*(1000000)*3600/63360;
//converts to miles per hour
  Serial.print("The Speed of Sound is: ");
  Serial.print(speed_Of_Sound);
 Serial.println(" miles per hour");
  delay(1000);
}
```

Output	Check this out and see how close it is to publish values for the speed of sound.

Program #4	**Obstacle collision module using IR(Infrared) sensor on Arduino**
Components Required	1-IR sensor, Jumper wire, Breadboard
IR Sensor	An Infrared sensor is an electronic instrument which is used to sense certain characteristics of its surroundings by either emitting and/ or detecting infrared radiation. Infrared sensors are also capable of measuring the heat being emitted by an object and detecting motion.
Description	Here we detect the obstacle by emitting infrared radiation from a IR sensor. The circuit diagram as shown below.

Circuit



Code

```
// IR Obstacle Collision Detection Module
int LED = 13;
int is_Obstacle_Pin = 7;  // input pin for ostacle
int is_Obstacle = HIGH;  // value HIGH tells there's no obstacle
void setup() {
  pinMode(LED, OUTPUT);
  pinMode(is_Obstacle_Pin, INPUT);
```

	```
	  Serial.begin(9600);
	}
	void loop() {
	  is_Obstacle = digitalRead(is_Obstacle_Pin);
	  if (is_Obstacle == LOW)
	  {
	Serial.println("OBSTACLE!!, OBSTACLE!!");
	    digitalWrite(LED, HIGH);
	  }
	  else
	  {
	Serial.println("clear");
	    digitalWrite(LED, LOW);
	  }
	  delay(200);
	}
	```
Output	Move our hand towards the IR LEDs. As we are near them, the Output LED on the module and the LED for pin 13 on our Arduino will illuminate. Open the serial monitor and vary the distance of your hand while viewing the serial monitor.

7.6.4 Interfacing Display, GSM, GPS to Arduino

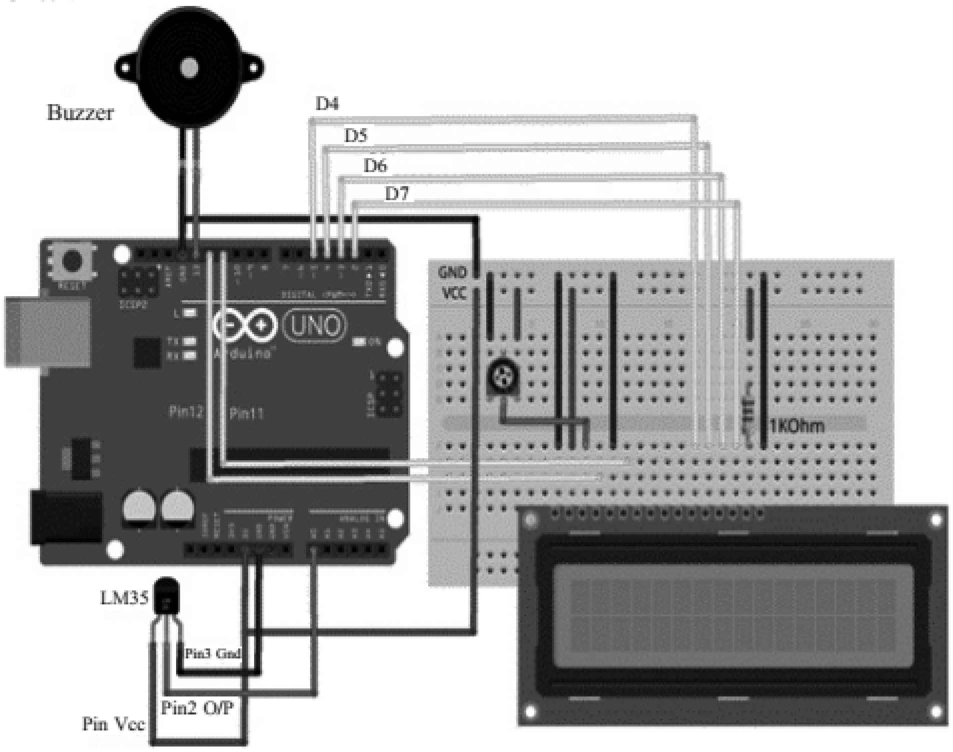
In this section we illustrate example prototypes using different kinds of displays such as LCD and Seven Segment displays.

Program #1	**Temperature and LCD Display**
Components Required	❖ Buzzer, ❖ LM35 Temperature Sensor ❖ LCD 16x2 Display ❖ 10k Potentiometer ❖ 1kOhm Resistor.
LCD Display	LCD (Liquid Crystal Display) screen is an electronic module with an extensive variety of uses. A 16x2 LCD implies it can show 16 characters for each line and there are 2 such lines. In this LCD every character is shown in 5x7 pixel lattice. This LCD has two registers, to be specific, Command and Data. The command register stores the command instructions given to the LCD. A command is a guideline given to LCD to do a predefined assignment like initializing it,

	clearing its screen, setting the cursor position, controlling presentation and so forth. The data register stores the information to be shown on the LCD. The data is the ASCII estimation of the character to be shown on the LCD. The pin Description of the LCD is appeared in underneath Table.

Pin No	Function	Name
1	Ground (0V)	Ground
2	Supply voltage; 5V (4.7V–5.3V)	Vcc
3	Contrast adjustment; through a variable resistor	VEE
4	Selects command register when low; and data register when high	Register Select
5	Low to write to the register; High to read from the register	Read/write
6	Sends data to data pins when a high to low pulse is given	Enable
7	8–bit data pins	DB0
8		DB1
9		DB2
10		DB3
11		DB4
12		DB5
13		DB6
14		DB7
15	Backlight Vcc(5V)	Led+
16	Backlight Ground (0V)	Led–

Description	In this program we are interfacing LM35 temperature sensor, peizo buzzer and an 16X2 LCD display. When temperature read from LM35 sensor exceeds 32°C , a beep output occurs on buzzer and also we display the read temperature on LCD.
NOTE:	Import the Two Libraries which are to be included in this program 1. #include<SoftwareSerial.h> to display temperature readings onto a Serial Monitor. 2. #include<LiquidCrystal.h>to display temperature readings onto a 16x2 LCD display.

Circuit	 Buzzer D4 D5 D6 D7 GND VCC UNO Arduino Pin12 Pin11 1KOhm LM35 Pin3 Gnd Pin2 O/P Pin Vcc

Code	//import the software serial directory
	#include <SoftwareSerial.h>
	//include liquid crystal library to configure the pins on LCD
	#include <LiquidCrystal.h>
	//initialize the pins of LCD with the corresponding Arduino pins
	/*
	Digital pin numbered 12 should be connected to RS pin of LCD
	Digital pin numbered 11 should be connected to enable pin of LCD
	Digital pin numbered 5 should be connected to D$ pin of LCD
	Digital pin numbered 4 should be connected to D% pin of LCD
	Digital pin numbered 3 should be connected to D6 pin of LCD
	Digital pin numbered 2 should be connected to D7 pin of LCD

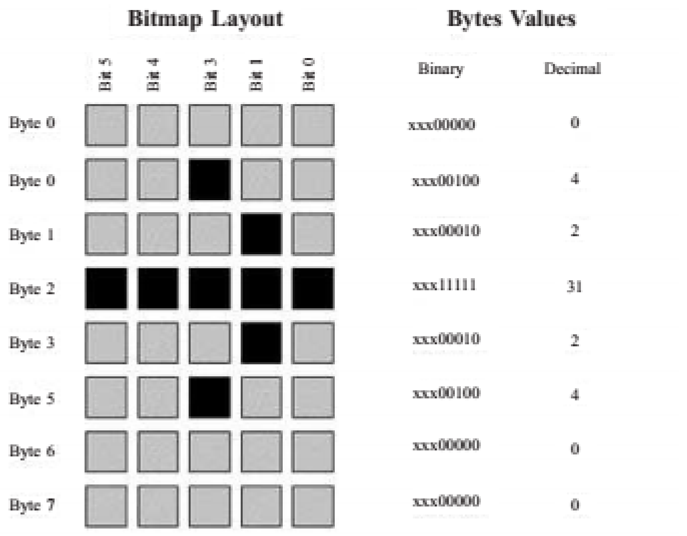
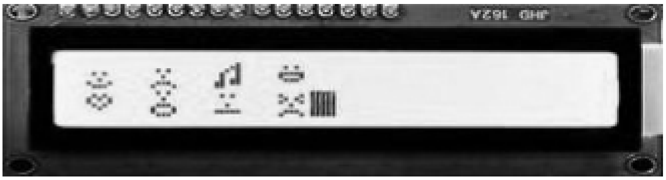
```
Ground pin should be connected to R/W pin of LCD
Ground pin should be connected to VSS pin of LCD
5V pin should be connected to VCC pin of LCD
*/
//Initialize the pins of LCD
LiquidCrystallcd(12, 11, 5, 4, 3, 2);
SoftwareSerialmySerial(9, 10);
Intt empPin = A0;
int buzzer = 13;
int value;
void setup()
{
//Initialize the baud rate to communicate on serial monitor
mySerial.begin(9600);
//set the baud rate of Arduino to serial monitor
Serial.begin(9600);
lcd.begin(16, 2);
pinMode(buzzer, OUTPUT);
delay(100);
}
void loop()
{
//Read the temperature on analog pin A5
value = analogRead(tempPin);
float mvalue = ( value/1024.0)*5000;
float celsius = mvalue/10;
float farhenheit = (celsius*9)/5 + 32;
if(celsius>30)
{
//trigger HIGH value on buzzer
digitalWrite(buzzer, HIGH);
//clear the display screen of LCD
lcd.clear();
```

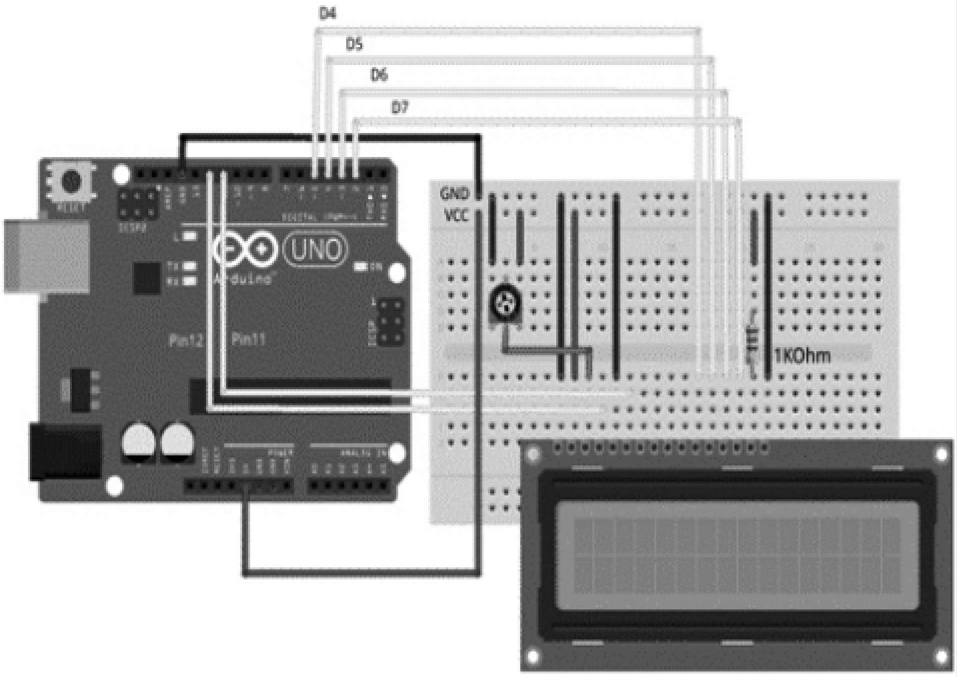
```
//set the starting position of cursor on LCD
lcd.setCursor(0, 0);
delay(10);
//print the temperature onto LCD
lcd.print("Temperature=");
delay(10);
lcd.setCursor(0, 1);
lcd.print(celsius);
delay(10);
lcd.print(" C");
//print the temperature readings onto serial monitor
Serial.print("Temperature=");
Serial.print(celsius);
Serial.println("*C");
Serial.println("BUZZER ON !!!");
delay(1000);
digitalWrite(buzzer, LOW);
delay(100);
digitalWrite(buzzer, HIGH);
delay(1000);
digitalWrite(buzzer, LOW);
delay(100);
digitalWrite(buzzer, HIGH);
delay(1000);
digitalWrite(buzzer, LOW);
delay(100);
}
else
{
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Temperature=");
delay(10);
lcd.setCursor(0, 1);
lcd.print(cel);
```

	delay(10);
	lcd.print(" C");
	Serial.print("Temperature= ");
	Serial.print(cel);
	Serial.println("*C");
	Serial.println("Buzzer OFF");
	delay(1000);
	   }
	}

Output	The room temperature is displayed on the LCD display. Whenever the room temperature exceeds 32°C, the Buzzer keeps on beeping until the temperature goes below 32°C. Also observe the output in the serial monitor in PC/Laptop.

Program #2	**Custom Characters in LCD**
Components Required	❖ LCD 16x2 Display ❖ USB Cable type A/B ❖ 10kohm potentiometer.
CUSTOM CHARACTER GENERATION	Any character can be made to appear on a 5x8 pixel matrix element without knowledge of its ASCII value. The basic technology of LCD is based on 3 type of memories:
	1. CG ROM 2. DD RAM 3. CG RAM
	CG ROM: CG ROM is the memory which holds the perpetual text styles you call to be shown. This holds the example for each and every character of predefined LCD text style and you call the substance of this memory by the setting relating ascii esteem on the LCD port. For instance, for recovery of "A" you need to send the ascii estimation of "A" which is 0x41 to the LCD. CGROM can likewise be viewed as PC hard drive from where you stack your required program into smash to begin working. In any case, it is not altering capable in light of the fact that it's ROM.
	DD RAM: DDRAM is the memory which holds just those characters which are as of now on the screen. implies if there is a message is right now being .

	shown on the screen then it must be on the DDRAM. For instance, in the event that you need to show "hi" on the screen then you need to stack the example of h from the CG ROM to DD RAM then do likewise for 'e', 'l', "l" and 'o'
	CG RAM: This memory works same as CG ROM yet in this smash, we can change its substance at whatever time. So this is where we need to first store our custom character design. at that point that example can be sent to show. The HD44780 LCD Display has a sum of 8 CG RAM memory areas. So we can create just up to 8 custom characters. in any case, you can simply change the substance of CG RAM on the travel to produce new characters.
	 Bitmap Layout Bytes Values Bit 5 Bit 4 Bit 3 Bit 1 Bit 0 Binary Decimal Byte 0 xxx00000 0 Byte 0 xxx00100 4 Byte 1 xxx00010 2 Byte 2 xxx11111 31 Byte 3 xxx00010 2 Byte 5 xxx00100 4 Byte 6 xxx00000 0 Byte 7 xxx00000 0
	This figure shows how Right Arrow is displayed on a single pixel in LCD. The corresponding binary and decimal values are also shown in the table. These binary values have to be given in the C code which is then fed to the CG RAM of LCD Display.
	

	This figure shows some of the custom characters displayed on 16x2 LCD Display.
Description	In this example we generate special charcters by their coding schemes and display on to the LCD.
Circuit  D4 D5 D6 D7 GND VCC Pin12 Pin11 1KOhm UNO Arduino	

Code	#include <LiquidCrystal.h>
	LiquidCrystallcd(12, 11, 5, 4, 3, 2);
	// to display degree centigrade
	byte newChar1[8] = {
	B01000,
	  B10100,
	  B01000,
	  B00011,
	  B00100,
	 B00100,

```
  B00011,
  B00000
};
// to display degree Fahrenheit
byte newChar2[8] =
{
B01000,
  B10100,
  B01000,
  B00011,
  B00100,
B00111,
  B00100,
B00000
};
// to display arrow right
byte newChar3[8] = {
B00000,
  B00100,
  B00010,
  B11111,
  B00010,
B00100,
  B00000,
  B00000
};
// to display arrow left
byte newChar4[8] = {
B00000,
  B00100,
  B01000,
  B11111,
  B01000,
B00100,
  B00000,
  B00000
};
```

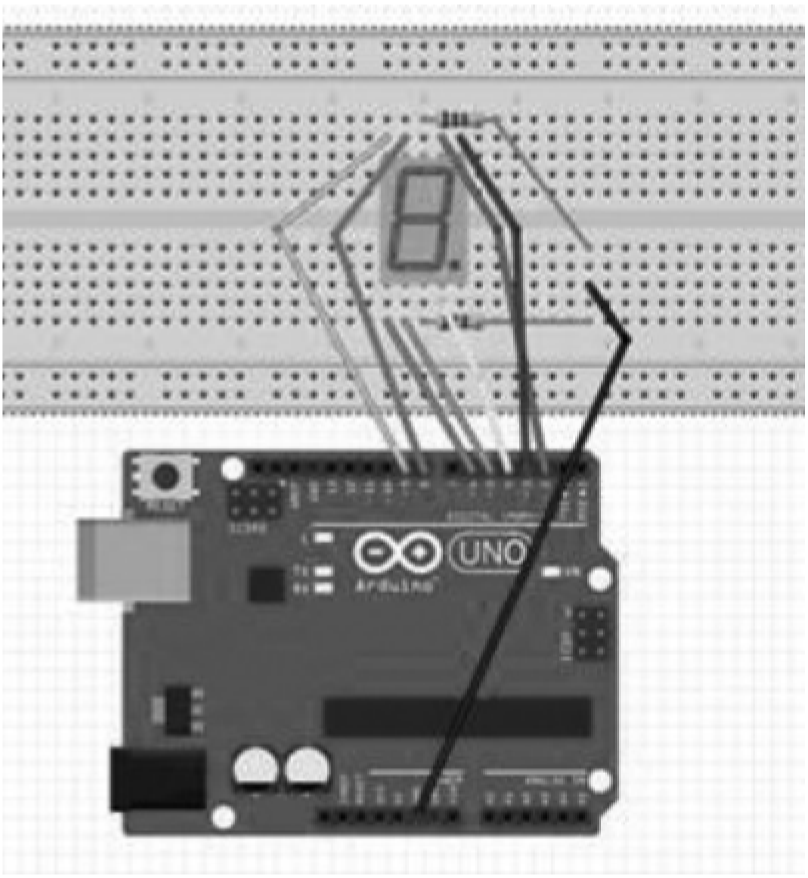
```
// to displaydiameter sign (ø)
byte newChar5[8] = {
B00000,
  B01101,
  B10010,
  B10101,
  B01001,
  B10110,
  B00000,
  B00000
};
// boldface "h"
byte newChar6[8] = {
  B11000,
  B11000,
  B11110
B11111,
  B11011,
  B11011,
  B11011,
  B00000
};
// to display ohm sign
byte newChar7[8] = {
  B00000,
  B01110,
  B10001,
B10001,
  B10001,
  B01010,
  B11011,
  B00000
};
// to display micro sign
byte newChar8[8] = {
  B00000,
  B00000,
B00000,
B10010,
  B10010,
  B10010,
  B11100,
  B10000
};
int i = 0;
void setup()
{
lcd.createChar(0, newChar1);
lcd.createChar(1, newChar2);
lcd.createChar(2, newChar3);
lcd.createChar(3, newChar4);
lcd.createChar(4, newChar5);
lcd.createChar(5, newChar6);
lcd.createChar(6, newChar7);
lcd.createChar(7, newChar8);
lcd.begin(16, 2);
for(int n = 0; n < 8; n++)
{ lcd.setCursor(n*2,0);
lcd.write(n);
  }
lcd.setCursor(0, 1);
lcd.print("Custom Chracters");
}
void loop()
{
}
```

Output	Observe the output on LCD display where in the first row of LCD custom characters (oc, of, →, ←, ø,h, Ω, µ) are displayed, and in the second row "Custom Chracters" message string is displayed.

Program #3	**7 Segment Display on Arduino**
Components Required	Arduino Uno 3, 7 Seven Segment Display, 2 x 220 Ohm Resistors, Jumper Wires
Description	Seven segment displays are of two types: common anode and common cathode. The Internal structure of both types is nearly the same. The difference is the polarity of the LEDs and common terminal. In a common cathode seven-segment display (the one we used in the experiments), all seven LEDs plus a dot LED have the cathodes connected to pins 3 and pin 8. To use this display, we need to connect GROUND to pin 3 and pin 8 and, and connect +5V to the other pins to make the individual segments light up. To display a particular number, turn on the individual segments as shown in the table below:

Digit	gfedcba	abcdefg	a	b	c	d	e	f	g
0	0×3F	0×7E	on	on	on	on	on	on	off
1	0×06	0×30	off	on	on	off	off	off	off
2	0×5B	0×6D	on	on	off	on	on	off	on
3	0×4F	0×79	on	on	off	on	on	off	on
4	0×66	0×33	off	on	on	off	off	on	on
5	0×6D	0×5B	on	off	on	on	off	on	on
6	0×7D	0×5F	on	off	on	on	on	on	on
7	0×07	0×70	on	on	on	off	off	off	off
8	0×7F	0×7F	on	on	on	on	on	on	on
9	0×6F	0×7B	on	on	on	on	off	on	on
A	0×77	0×77	on	on	on	off	on	on	on
B	0×7C	0×1F	off	off	on	on	on	on	on
C	0×39	0×4E	on	off	off	on	on	on	off
D	0×5E	0×3D	off	on	on	on	on	off	on
E	0×79	0×4F	on	off	off	on	on	on	on
F	0×71	0×47	on	off	off	off	on	on	on

Circuit



Connect the pins described below:

1. Arduino Pin 2 to Pin 9.
2. Arduino Pin 3 to Pin 10.
3. Arduino Pin 4 to Pin 4.
4. Arduino Pin 5 to Pin 2..
5. Arduino Pin 6 to Pin 1.
6. Arduino Pin 8 to Pin 7.
7. Arduino Pin 9 to Pin 6.
8. GND to Pin 3 and Pin 8 each connected with 220 ohm resistors.

Code

```
int a = 2;  //For displaying segment "a"
int b = 3;  //For displaying segment "b"
int c = 4;  //For displaying segment "c"
int d = 5;  //For displaying segment "d"
int e = 6;  //For displaying segment "e"
int f = 8;  //For displaying segment "f"
int g = 9;  //For displaying segment "g"
void setup() {
pinMode(a, OUTPUT);  //A
  pinMode(b, OUTPUT);  //B
  pinMode(c, OUTPUT);  //C
  pinMode(d, OUTPUT);  //D
  pinMode(e, OUTPUT);  //E
  pinMode(f, OUTPUT);  //F
  pinMode(g, OUTPUT);  //G
}
void displayDigit(int digit)
{
 //Conditions for displaying segment a
 if(digit!=1 && digit != 4)
 digitalWrite(a,HIGH);
//Conditions for displaying segment b
 if(digit != 5 && digit != 6)
 digitalWrite(b,HIGH);
 //Conditions for displaying segment c
 if(digit !=2)
 digitalWrite(c,HIGH);
//Conditions for displaying segment d
 if(digit != 1 && digit !=4 && digit !=7)
 digitalWrite(d,HIGH);
//Conditions for displaying segment e
 if(digit == 2 || digit ==6 || digit == 8 || digit==0)
digitalWrite(e,HIGH);
//Conditions for displaying segment f
if(digit != 1 && digit !=2 && digit!=3 && digit !=7)
 digitalWrite(f,HIGH);
 if (digit!=0 && digit!=1 && digit !=7)
 digitalWrite(g,HIGH);
}
```

```
void turnOff()
{
  digitalWrite(a,LOW);
  digitalWrite(b,LOW);
  digitalWrite(c,LOW);
digitalWrite(d,LOW);
  digitalWrite(e,LOW);
  digitalWrite(f,LOW);
  digitalWrite(g,LOW);
}
void loop() {
for(int i=0;i<10;i++)
 {
   displayDigit(i);
   delay(1000);
   turnOff();
 }
}
```

Output	Run the code and observe the desired output on 7 segment display

Program #4	**GSM Interface**
Components Required	❖ 1k Ohm Resistor ❖ USB Cable type A/B ❖ 10k Potentiometer ❖ 16x2 LCD Display ❖ SIM900A GSM Module with Antenna.
SIM900A Description	GSM/GPRS TTL UART Modem is build with Dual Band GSM/GPRS motor SIM900, chips away at frequencies 900/1800 MHz. The baud rate is configurable from 9600-115200 through AT command. The GSM/GPRS Modem is having inner TCP/IP stack to empower you to interface with web by means of GPRS. It is appropriate for SMS, Voice and in addition DATA move application in M2M interface. **GSM/GPRS Modem Features** ❖ High Quality Product ❖ Dual-Band GSM/GPRS 900/ 1800 MHz ❖ Quad-Band 850/ 900/ 1800/ 1900 MHz

	❖ Configurable baud rate ❖ SIM Card holder. ❖ Built in Network Status LED ❖ Inbuilt Powerful TCP/IP protocol stack for internet data transfer over GPRS.
	❖ Audio interface Connector ❖ Normal operation temperature: -20 °C to +55 °C ❖ Input Voltage: 3.6- 4.5 VDC
Installation	**Power on GSM/GPRS module** Power the GPRS module by pulling down the PWR (Power status of GPRS module) button or the P pin of control interface for not less than 1 second and discharge. This pin is as of now pulled up to 3V in the module inside, so outside draw up is a bit much. At the point when power on methodology is finished, GSM/GPRS module will send taking after URC to demonstrate that the module is prepared to work at settled baud rate.
Indicator LED and Buttons	**NETSTATUS:** The status of the NETSTATUS LED is listed in following table:

Status	**Description**
Off	SIM900 is not running 64ms On/800ms
64ms On/3000ms Off	SIM900 registered to the network
64ms On/300ms Off	GPRS communication is established

Connection Diagram:

Some of the AT commands of SIM900A has been given in the table below.

Common AT Commands:

Command	**Description**	**Return**	**Result**
AT+CPIN?	Check sim card status	+CPIN:READY	Sim card is found
AT+CSQ	check signal quality	+CSQ:30,0	30 is the quality of signal, max at 31
AT+COPS?	check card service provider	+COPS:0,0,"CHINA MOBILE" or empty	CHINA MOBILE is the service provider
AT+CGMI	Check the module maker	SIMCOM_Ltd	Made by simcom
AT+CGMM	Check the module model type	SIMCOM_ SIM900A	SIM900A

	AT+CGSN	Check the module IMEI, worldwide unique	86998801208905	86998801201890 5
	AT+CNUM	Check the number of current sim card, not all kinds of card support this function	+CNUM:"","159020 20353",129,7,4	phone number 15902020353
	At+ATE1	on/ off AT commands return info	Send either ATE0 or ATE1	

Calling and DTMF:

Command	Description	Return	Result
ATD +number;	dont forget the ";"after AT commands, for example "ATD+18576608994;"	OK	-
ATA	answer the incoming phone call	OK	-
ATH	hang off current phone call	OK	-
AT-COLP	show up the calling number	+COLP;"10086',129,"",""	-
AT+CLIP	sown up the number of incoming call	+CLIP;"15124532672", 161,"",,"ailin",0,	-
AT+VTS"*	Send DTMF audio. for example, when we call the service provider number and need the press number 1 for operation you can do AT+VTS=1	-	-

GPRS:

Command	**Description**	**Return**	**Result**
AT+CGCLASS	support type B and CC,	-	-
AT+CGDCONT	set PDP, e.g.AT+CGDCONT=1,"IP","CMNET", sign1,internet protocol (IP) and connecting port CMNET	-	-
AT+CGATT	attach or dis-attach GPRS service, AT+CGATT=1	-	-
AT+CIPCSGP	set CSD or GPRS link mode, AT+CIPCSGP=1,"CMNET",set GPRS link and connecting port CMNET	-	-
AT+CLPORT	set TCP local port, AT+CLPORT=”TCP”,”8888”,port at 8888	-	-
AT+CIPSTART	initiate a connect or setup a UDP port AT+CIPSTART="TCP","180.120.52.222", "8086",connect to 180.120.52.222 atport 8086	CONNECT OK	-
AT+CIPSEND	send data, module will return ">",send max 1352 bytes and end up with 1A same as sms	SENDOK	-
AT+CIPSTATUS	check module current connection status	-	-
AT+CIPCLOSE	close current TCP/UDP connection, AT+CIPCLOSE=1	OK	-
AT+CIPSHUT	close mobile scene	-	-

SMS:

AT+CMGS (e.g.Chinese SMS)	Set AT+CMGR=1; AT+CSMP=17,167,2,25; message can be by Unicode, AT+CMGS="003100350031003100320034003500330032003600370032-(number 15124532672)	Will return ">"and then type message 00530049004D000390030003000414ED4FE153D190016D4B8D4BSBD5,end with 1A	-
AT+CMGD	Delete message to delete message at position 1: AT+CMGD=1	OK	-
AT+CPMS	Inquiry or set message storage setting	AT+CPMS? to check how many message can be stored maximally and how many message stored	:+CPMS:"SM", 1,50, means support 50 sms max and 1sms are stored now

General procedure to Read and Send SMS using AT commands:

Command	Description	Return	Result
AT+CMGF	Set the new message remind, for example AT+CNMI=2,1	+CMTI:"SM",2	When set is on and message box is NOT full, message is stored at position 2
AT+CMGF	Set the module message mode, set either at PDU (0) or text mode (1)	OK	-
AT+CSCS	Set TE character set,set AT+CSCS="UCS2" for other language	OK	-
AT+CMGR	Read message, for example, AT+CMGR=1 to read message at position 1	-	-
AT+CMGS	Send message, send 180 bytes at GSM mode, or 70 Chinese character at UCS2 mode, AT+CMGS="18576608994"	Will return">" and then type message, then end up with hex value 1A (0X1A,"CTRL+ Z"),send 1B to cancel "ESC"	And finally return : +CMGS:156, in which 156 has meaning.

General read SMS Setps

AT+CMGF=1

AT+CSCS="GSM"

AT+CNMI=2,1//set new message remind

AT+CMGR=2//read message at position 2

AT+CMGD=2 // delete SMS at position 2

General send SMS steps

AT+CSCS="GSM"

AT+CMGF=1

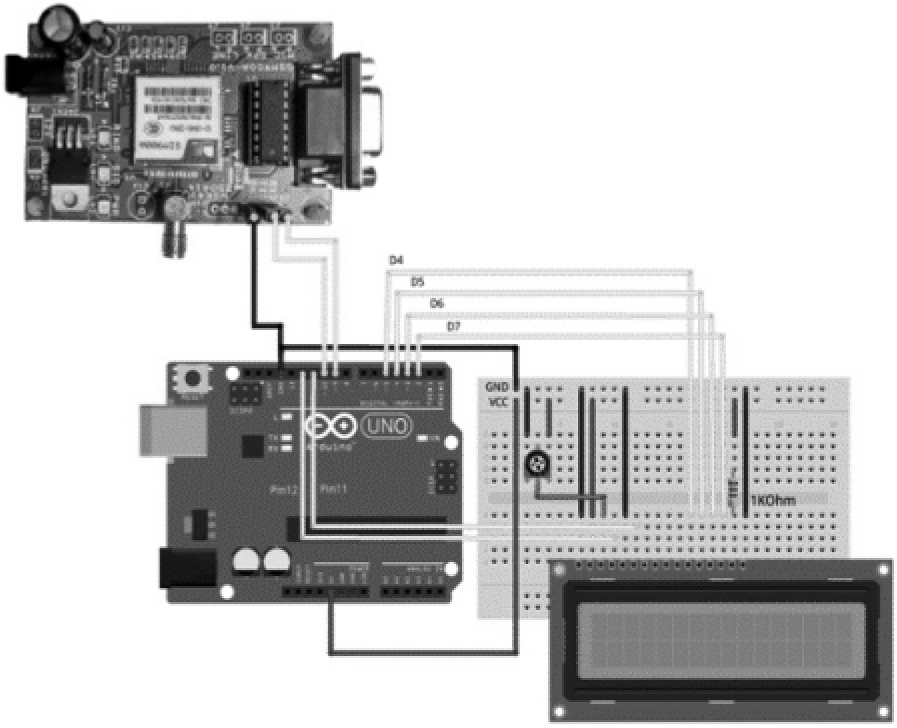
AT+CMGS="+91XXXXXXXXXX"

Dascription	Now let us look at an example for interfacing GSM module to create messages, send messages, receive messages and display messages on LCD display.
Key points	1. Connect the circuit as shown in the pin connection diagram. 2. Write the desired C source code. 3. Replace 'x' with your 10-digit Mobile number.

4. Compile the code and upload it to Arduino UNO board.
5. Connect the sim card and power it up.
6. Insert the sim card and power it up.
7. Ensure that the GSM module is working fine by calling to the number that is in the module.
8. Now go-to Serial monitor and type 's' to send a predefined message in the code to the predefined number.
9. To get the message, first send the message from your mobile and then type 'r' in the serial monitor to receive the message.
10. Observe the messages in LCD Display.

NOTE: The AT command given in the above code to receive the message is to receive a live SMS. So if you send the message 3-5 minute earlier/later, after typing the character 'r' in Serial monitor, there are very less chances of receiving the message. Do not power the GSM form Arduino. Use a separate adapter of minimum 10V-12V rating.

Circuit

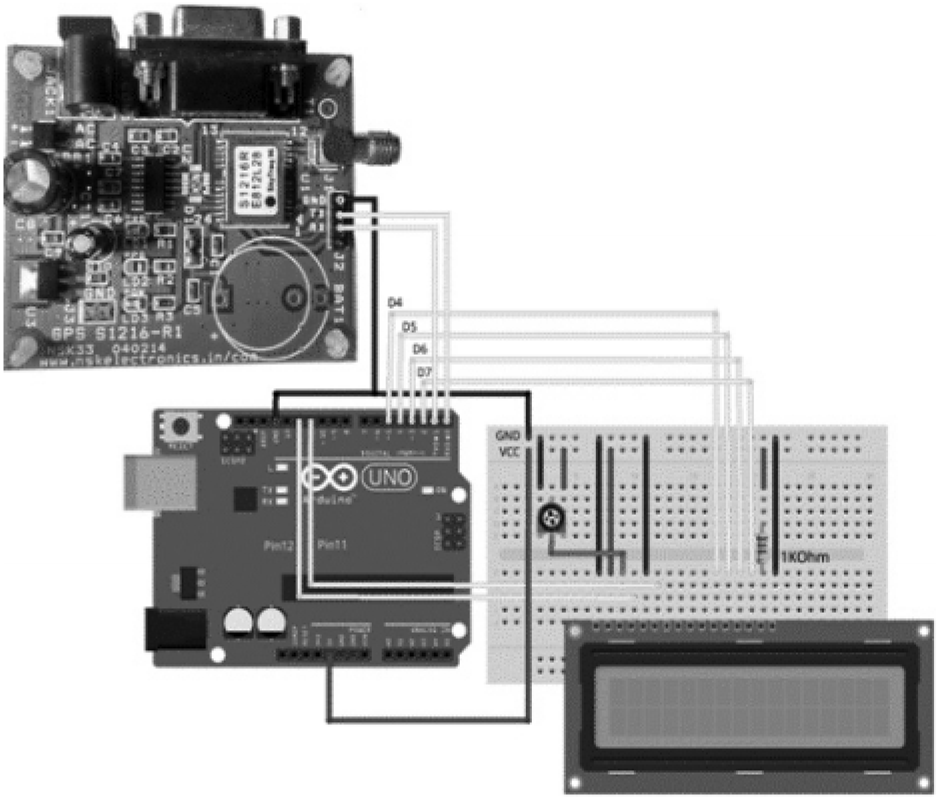


Code	#include <SoftwareSerial.h>
	#include <LiquidCrystal.h>
	//initialize interface pins of LC
	LiquidCrystallcd(12, 11, 5, 4, 3, 2);
	//connect TX pin of GSM module to 9 and RX pin to 10 of UNO board
	SoftwareSerialmySerial(9, 10);
	void setup()
	{
	//configure the baud rate on GSM module
	mySerial.begin(9600);
	//configure baud rate on serial monitor
	Serial.begin(9600);
	lcd.begin(16, 2);
	delay(1000);
	}
	void loop()
	{
	if (Serial.available()>0)
	switch(Serial.read())
	  {
	case 's': SendMsg();
	        break;
	   case 'r': RecieveMsg();
	        break;
	  }
	if (mySerial.available()>0)
	Serial.write(mySerial.read());
	}
	void SendMsg()
	{mySerial.println("AT+CMGF=1"); //Sets the GSM Module in Text Mode
	delay(1000); // Delay of 1 second

	mySerial.println("AT+CMGS=\"+919379146847\"\r"); // Replace x with mobile number
	delay(1000);
	mySerial.println((char)26);// ASCII code of CTRL+Z
	delay(1000);
	Serial.println("Message Sent !");
	lcd.clear();
	lcd.setCursor(0, 0);
	delay(10);
	lcd.print("Message Sent !");
	}
	void RecieveMsg()
	{ Serial.println("Message Received !");
	mySerial.println("AT+CNMI=2,2,0,0,0"); // AT Command to receive a live SMS
	lcd.clear();
	lcd.setCursor(0, 0);
	delay(10);
	lcd.print("Msg Received !");
	delay(1000);
	}
Output	The messages sent from your mobile are displayed in the Serial monitor. The received and sent messages alert is displayed on the LCD panel.

Program #5	**GPS Interface**
Components Required	· 1k Ohm Resistor
	· 10k Potentiometer
	· USB Cable type A/B
	· S1216R GPS Module with external GPS Antenna
	· 16x2 LCD Display.
Description	Now let us look at an example to interface GPS to find the global coordinates of a position. This example shows how to get input from GPS module and to display on LCD. Compile the code given below and upload it to Arduino UNO Board to observe at the desired output.

Circuit



Code

```
#include <SoftwareSerial.h>
#include <LiquidCrystal.h>
SoftwareSerialmySerial(0, 1);
LiquidCrystallcd(12, 11, 7, 6, 5, 4);
int led = 13;
intinByte = 0;// incoming serial byte
void setup()
{lcd.clear();
lcd.begin(0, 0);
lcd.print("Finding GPS");
lcd.setCursor(0, 1);
lcd.print("Location..."); // initialize the led pin as an output.
pinMode(led, OUTPUT);     // start serial port at 9600 bps
Serial.begin(9600);
mySerial.begin(9600);
delay(1000);// wait for a while till the serial port is ready
// send the initial data once //
Serial.print('\n');
Serial.println(" GEOGRAPHICAL CO-ORDINATES :");
digitalWrite(led, HIGH);
delay(1000);
}
void loop ()
{//==================== searching for "GG"
====================//
   do
   {
do
      {
         while ( !mySerial.available() );
      } while ( 'G' != mySerial.read() );// reading a character from the
GPS
while(!mySerial.available());
   } while ( 'G' != mySerial.read() );
//==================== searching for "GG"
====================//
//=============== seeking for north cordinate ===============//
   do
   {
 while ( !mySerial.available() ); // reading a character from the GPS
   } while ( ',' != mySerial.read() );
   do
   {
while ( !mySerial.available() );  // reading a character from the GPS
    } while ( ‘,’ != mySerial.read() );
//============== seeking for north cordinate ==============//
//============== printing the north cordinate ===============//
Serial.print(“ N: “);
    do
    {
while ( !mySerial.available() );
inByte = mySerial.read();   // reading a character from the GPS
Serial.write( inByte );                    // printing the Latitude
    } while ( ‘,’ != inByte );
//============== printing the north cordinate ===============//
//============== seeking for east cordinate ==============//
    do
    {
while ( !mySerial.available() );            // reading a character from the
GPS
    } while ( ‘,’ != mySerial.read() );
//============== seeking for east cordinate ==============//
//============== printing the east cordinate ===============//
Serial.print(“ E: “);
    do
    {
while ( !mySerial.available() );
inByte = mySerial.read();    // reading a character from the GPS
Serial.write( inByte );                  // printing the Longitude
    } while ( ‘,’ != inByte );
//============== printing the east cordinate ===============//
Serial.println();
delay (1000);
}
```

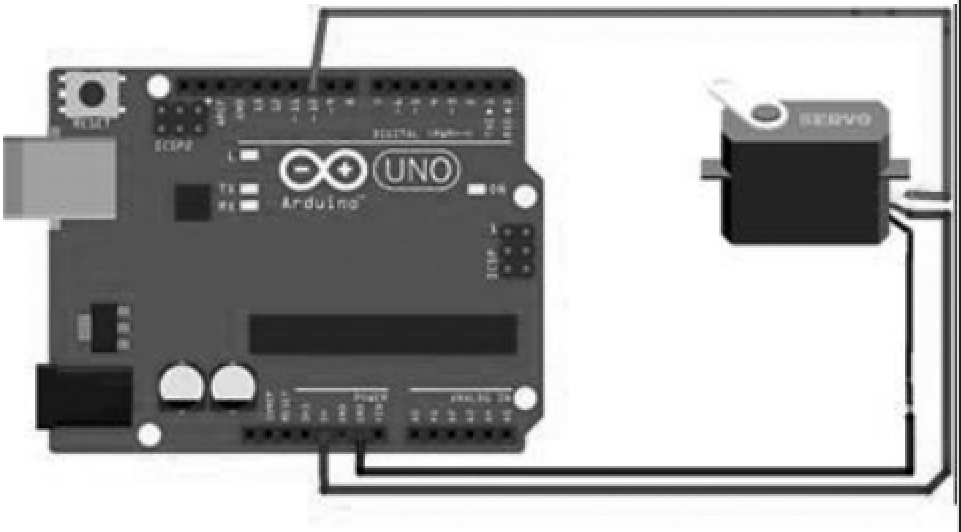
Output	The North and East Geographical co-ordinates of your current position are displayed on the Serial monitor.

7.6.5 Interfacing Motors:

In this section we illustrate example prototypes using different kinds of motors.

Program #1	Servo motor
Components Required	1-Servo motor, Jumper wires, Breadboard
What is a Servo Motor?	Servo motor is position controlled motor. It can easily control physical movement of objects due to its position controlled feature. Servo motor cannot move continuously. Servo motors rotate in angel ranging from 0 to 180 degree. Servo motor have many applications in robotics and industry for position based motion control system. Unlike other motors, servo motor is very easy to interface with Arduino or any other microcontroller due to its built-in controllers. Servo motor respond to change in duration of pulses. A pulse of duration 1 mili second cause the servo motor to move one end and duration of 2 mili second cause the motor to move other end. So we can calculate servo motor position by varying the duration of pulses to servo with the help of Arduino Uno R3. Arduino IDE have built-in library for servo motor control. So you don't need to write lengthy code for servo motor. You just have to call function used in servo.h library.

Circuit



Code	#include<Servo.h>
	Servo myloop; // create servo object to control a servo
	int angle = 0; // variable to store the servo position
	void setup()
	{
	myloop.attach(10); // attached the servo on pin 9 to the servo object
	}
	void loop()
	{
	for(angle = 0; angle < 180; angle += 1) // goes from 0 degrees to 180 degrees
	{ // in steps of 1 degree
	myloop.write(angle); // tell servo to go to position in variable 'angle'
	delay(20); // waits 20ms between servo commands
	}
	for(angle = 180; angle >= 1; angle -= 1) // goes from 180 degrees to 0 degrees
	{
	myloop.write(angle); // move servo in opposite direction
	delay(20); // waits 20ms between servo commands
	}
	}

Output	Observe and control the position of servo motor with specific or calculated angle.

7.7 Summary

In this chapter you learned about Arduino UNO which is a low cost prototype building board. Arduino uses the Arduino programming environment which supports the c coding syntax. Arduino has ATmega328P microcontroller with operating voltage 5V, having 14 digital output pins, 6 analog input pins with a clock speed of 16MHz. You learned how to develop C programs that run on Arduino UNO board. You learned how to interface LED, switch, LCD, RFID, GPS and GSM etc.

7.8 Self test questions

❖ What is a Arduino?

- Can I connect a mouse and keyboard to Arduino Uno?
- What SOC Arduino using?
- What is a SOC?
- Does Arduino Uno overclock?
- Does Arduino uno need a heat sink?
- Does Arduino Uno has any hardware interfaces?
- Does Arduino Uno need an External power source?
- Which IDE environment does Arduino Uno use?
- Does the Arduino supports networking?
- Define a Microcontroller?
- State the use of Serial Monitor in Arduino IDE?
- Define the term Baud Rate?

7.9 Objective questions

1. **Arduino is an open source prototyoing platform.**
   A. True B. False
2. **Arduino Uno is based on which of the following chip.**
   A. ATmega2560 B. ATmega328P C. ATmega32u4 D. ATSAM
3. **How many number of digital I/O pins Arduino Uno is comprised of?**
   A. 12 B. 14 C. 16 D. 18
4. **How many pins are sources of Analog input?**
   A. 4 B. 6 C. 8 D. 10
5. **Crystal clock of Arduino Uno is of __________**
   A. 16MHz B. 20MHz C. 24MHz D. 28MHz
6. **What is the operating voltage of Arduino Uno?**
   A. 3 B. 5 C. 7 D. 10
7. **What is the recommended input voltage for Arduino Uno?**
   A. 5-10v B. 7-12v C. 5-12v D. 6-10v
8. **What is the size of Flash memory Arduino uno comprised of?**
   A. 30KB B. 32KB C. 34KB D. 36KB
9. **Uploading code to Arduino no is dependent on external Hardware.**
   A. True B. False
10. **Arduino Uno supports Networking.**
    A. True B. False

7.10 Review Questions

1. How is Arduino Uno is different from the other available Microcontrollers?
2. What is the use of GPIO pins?
3. What is the use of I2C interfaces on Raspberry Pi?
4. How many pins does the Atmega328P MCU used on the standard Arduino have? Over what range of voltages will it operate?
5. Assume that you have an LED connected to each of the 14 digital-only I/O pins on the Arduino.
6. If all of the LEDs could possibly be on at the same time, what must the current be limited to through each of the LEDs?
7. Assume that a project requires that a high-brightness LED be on any time that the Arduino is powered-on, and that this LED requires 350mA. What is the best way to supply power/ current to this LED?
8. Can you think of a way to use the oscilloscope to measure the time it takes to print out the message in the sketch you are currently running? Possible hint: Digital pin 1 (TX, a.k.a. transmit) is the pin over which serial data is sent to the PC.
9. What's The Difference/Relationship Between AVR And Arduino?
10. How To Unpair Or Delete Paired Bluetooth Device Programmatically On Android?

11 How To Convert Int To String On Arduino?

12 Converting An Int Or String To A Char Array On Arduino

13. How Can I Unit Test Arduino Code?
14. In What Language Is The Arduino IDE Written
15. How Is Programming An Arduino Different Than Standard C?
16. Elicit some points on Arduino Boot loader?

7.11 Glossary

IC: Integrated circuit

GPIO: General purpose Input Output pins

IDE: Integrated development Environment

PWM: Pulse width modulation

PCB: Printed Circuit board

USB: Universal Serial Bus

LED: Light Emitting diode

LCD: Liquid crystal display

CHAPTER
8

IoT Physical Devices and Endpoints: RaspberryPi

This Chapter Covers

- ❖ Introduction to Raspberrypi
- ❖ Exploring the Raspberrypi Learning Board
  - ✓ Description of System on Chip (SoC)
  - ✓ Raspberry Pi interfaces
- ❖ Raspberrypi Operating Systems
  - ✓ Operating Systems (not Linux based)
  - ✓ Operating Sustems (Linux based)
  - ✓ Media center operating systems
  - ✓ Audio operating systems
  - ✓ Recalbox
- ❖ Operating System Setup on RaspberryPi
  - ✓ Formatting SD card
  - ✓ OS installation
  - ✓ First Boot
  - ✓ LOGIN Information
- ❖ RapberryPi commands
- ❖ Programming RaspberryPi with Python

8.1 Introduction to RaspberryPi

The RaspberryPi is a series of credit card sized single-board computers developed in the United Kingdom by the Raspberrypi Foundation to promote the teaching of basic computer science in schools and developing countries. The original model got way popular than anticipated, outside of the target market, enthusiasts use it, and the later models , for various uses, such as robotics. Accessories such as keyboard and mice (not needed for some uses) and even a case, are not included, while planned for; they have frequently been included in unofficial bundles, and later in an official bundle.

Several generations of RaspberryPi have been released. The first generation (RaspberryPi 1 Model B) was released in February 2012, followed by a simpler and inexpensive model A. In 2014, the foundation released a board with an improved design in Raspberry 1 Model B+. The model laid the current "mainline" form factor. Improved A+ and B+ models were released a year later. A cut down "compute module" was released in April 2014, and a RaspberryPi Zero with smaller size and limited input/output (I/O) and general purpose input/output (GPIO) abilities was released in November 2015 for US$5. The RaspberryPi 2 which added more RAM was released in February 2015. RaspberryPi 3 model B released in February 2016 is bundled with on-board Wi-Fi and Bluetooth. As of 2016, RaspberryPi 3 Model b is the newest mainline RaspberryPi. These boards are priced between US$5-35. Furthermore **Table 8-1** gives the distinctions in the specialized particulars of the famous models of RaspberryPi.

All models feature a Broadcom system on a chip(SoC), which includes an ARM compatible central processing unit (CPU) and an on chip graphics processing unit (GPU, a Video Core IV). CPU speed ranges from 700 MHz to 1.2GHz for the Pi 3 and on board memory range from 256MB to 1GB RAM. Secure Digital (SD) cards are used to store the operating system and program memory in either the SDHC or MicroSDHC sizes. Most boards have between one and four USB slots, HDMI and composite video output, and a 3.5 mm phone jack for audio. Lower level output is provided by a number of GPIO pins which support common protocols like I2C. The B-models have an 8P*C Ethernet port and the Pi 3 has on board Wi-Fi 802.11n and Bluetooth.

The Foundation provides Raspbian, a Debian-based Linux distribution for download, as well as third party ubuntu, windows 10 IoT core, RISC OS, and specialized media center distributions. Foundation also provides Python and Scratch as the main programming language, with support for many other languages. The default firmware is closed source, while an unofficial open source is available.

"**Why RaspberryPi**?" **–** Inexpensive, Cross-platform , Simple, clear programming environment, Open source and extensible software **and** Open source and extensible hardware.

Table 8-1: Technical Specification of Raspberry Pi models

RaspberryPi	Model A+	Model B	Model B+	2, Model B	Model 3
Quick Summary	Cheapest,smallest single board computer	The original Raspberry Pi.	More USB and GPIIO than the B.Ideal choice for schools	most advanced Raspberry Pi.	Newest with wireless connectivity
Chip	Broadcom BCM 2835			Broadcom BCM2836	Broadcom BCM 2837
processor	ARMv6 single core			ARMv7 quad core	4×ARM Cortex-A53
Processor speed	700 MHz			900 MHz	1.2GHz
Voltage and power draw	600mA @ 5V			650mA @ 5V	
GPU	Dual core Videocore IV Multimedia Co-Processor				Broadcom Videocone IV
Size	65×56mm	85×56mm			
Memory	256 MB SDRAM @ 400 MHz	512 MB SDRAM @ 400 MHz		1 GB SDRAM @ 400 MHz	1 GB LPDDR2 (900 MHz)
Storage	Micro SD Card	SD Card	Micro SD Card	Micro SD Card	Micro SD Card
GPIO	40	26	40		
USB 2.0	1	2	4		
Ethernet	None	10/100mb Ethernet RJ45J ack			
Wireless	None				2.4GHz 802.11n wireless
Bluetooth	None				Bluetooth 4.1 Classic, Bluetooth Low Energy
Audio	Multi-Channel HD Audio over HDMI, Analog Stereo from 3.5mm Headphone Jack				

Operating Systems	Raspbian RaspBMC,Ar ch Linux,Rise OS,OpenEL EC Pidora
Video Output	HDMI Composite RCA
Supported Resolutions	640×350 to 1920×1200, including 1080p,PAL & NTSC standards
Power Source	Micro USB

8.2 Exploring the Raspberrypi Learning Board

In the **Figure 8-2** below you can see an Raspberrypi board labeled. Let's see what each part does.

❖ **Processor:** The Broadcom BCM2835 SoC used in the first generation Raspberrypi is somewhat equivalent to the chip used in first generation smart phones (its CPU is an older ARMv6 architecture), which includes a 700 MHz ARM 1176JZF-S processor, Video Core IV graphics processing unit (GPU) and RAM. This has a level 1 (L1) cache of 16KB and a level 2 (L2) cache of 128 KB. The level 2 cache is used primarily by the GPU. The SoC is stacked underneath the Ram chip, so only its edge is visible. The Raspberrypi 2 uses a Broadcom BCM2836 SoC with a 900 MHz 32-bit quad-core ARM cortex A7 processor (as do many current smart phones), with 256KB shared L2 cache. The Raspberrypi3 uses a Broadcom BCM2837 SoC with a 1.2GHz 64-bit quad core ARM Cortex A53 processor, with a 512KB shared L2 cache.

❖ **Power Source:** The recommended and easiest way to power the Raspberrypi is via the Micro USB port on the side of the unit. The recommended input voltage is 5V, and the recommended input current is 2A. the Raspberrypi can function on lower current power supplies e.g. 5V @ 1A. However, any excessive use of the USB ports or even heavy CPU/GPU loading can cause the voltage to drop, and instability during use. The latest versions of the Raspberrypi B+/A+/2 have a "low voltage indicator icon" to notify the user if there is a problem with the power.

❖ **SD Card:** The Raspberry Pi does not have any locally available storage accessible. The working framework is stacked on a SD card which is embedded on the SD card space on the Raspberry Pi.

❖ **GPIO (General Purpose Input Output):** General-purpose input/output (GPIO) is a non specific pins on a coordinated circuit to know is an input or output pin which can be controlled by the client at run time. GPIO pins have no exceptional reason characterized, and go unused as a matter of course. GPIO capabilities may include:

✓ GPIO pins can be designed to be input or output

✓ GPIO pins can be empowered/crippled

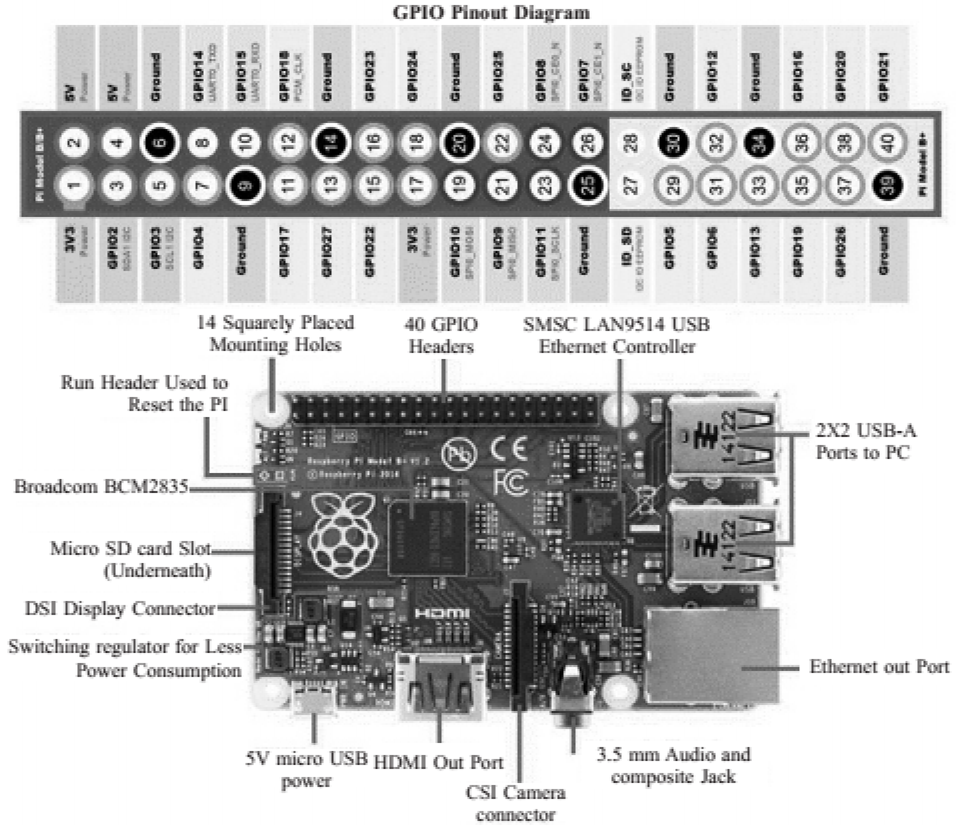


Figure 8-1: Raspberry Pi2 Model B and its GPIO

- ✓ Input values are meaningful (normally high=1, low=0)
- ✓ Yield values are writable/meaningful
- ✓ Input values can frequently be utilized as IRQs (regularly for wakeup occasions)

In programming environment The GPIO.BOARD alternative alludes to the pins by the number of the pin (e.g. P1) and amidst the outlines beneath. The GPIO.BCM choice alludes to the pins by the "Broadcom SOC channel" number; these are the numbers after "GPIO" in the green rectangles around the outside of the underneath graphs:

- ❖ **DSI Display X:** the Raspberrypi Connector S2 is a display serial interface (DSI) for connecting a liquid crystal display (LCD) panel using a 15-pin ribbon cable. The mobile industry processor interface (MIPI) inside the Broadcom BCM2835 IC feeds graphics data directly to the display panel through this connector. This article looks at the connector pin out and some of the display panels compatible with the port.

- **Audio Jack:** A standard 3.5 mm TRS connector is accessible on the RPi for stereo sound yield. Any earphone or 3.5mm sound link can be associated straightforwardly. In spite of the fact that this jack can't be utilized for taking sound information, USB mics or USB sound cards can be utilized.
- **Status LEDs:** There are 5 status LEDs on the RPi that demonstrate the status of different exercises as takes after:
  - — OK - SDCard Access (by means of GPIO16) - named as "OK" on Model B Rev1.0 sheets and "ACT" on Model B Rev2.0 and Model A sheets
  - — POWER - 3.3 V Power - named as "PWR" on all the boards
  - — FDX - Full Duplex (LAN) (Model B) - marked as "FDX" on all the boards
  - — LNK - Link/Activity (LAN) (Model B) - marked as "LNK" on all the boards

— 10M/100 - 10/100Mbit (LAN) (Model B) - named (erroneously) as "10M" on Model B Rev1.0 boards and "100" on Model B Rev2.0 and Model A boardsUSB Ports.

There is 1 port on Model A, 2 on Model B and 4 on Model B+ operates at a current upto 100mA, An external USB powered hub is required to draw current more than 100mA.

- **Ethernet port:** Ethernet port is accessible on Model B and B+. It can be associated with a system or web utilizing a standard LAN link on the Ethernet port. The Ethernet ports are controlled by Microchip LAN9512 LAN controller chip.
- **CSI connector (CSI) –** Camera Serial Interface is a serial interface outlined by MIPI (Mobile Industry Processor Interface) organization together went for interfacing computerized cameras with a portable processor. The RPi establishment gives a camera uncommonly made to the Pi which can be associated with the Pi utilizing the CSI connector.
- **JTAG headers :** JTAG is an acronym for Joint Test Action Group', an association that began back in the mid 1980's to address test point get to issues on PCB with surface mount gadgets. The association formulated a technique for access to gadget pins by means of a serial port that got to be distinctly known as the TAP (Test Access Port). In 1990 the strategy turned into a perceived universal standard (IEEE Std 1149.1). A large number of gadgets now incorporate this institutionalized port as a component to permit test and configuration architects to get to pins.
- **HDMI:** High Definition Multimedia Interface to give both video and sound yield.

8.1.1 Description of System on Chip (SoC)

A System on a chip (SoC) is an integrated circuit (IC) that coordinates all parts of a PC or other electronic framework into a solitary chip. It might contain advanced, simple, blended flag, and regularly radio-recurrence works—all on a solitary chip substrate. SoCs are exceptionally regular in the portable gadgets advertise in view of their low power utilization. A run of the mill application is in the range of implanted frameworks.

An SoC comprises of:

- ✓ A microcontroller, chip or DSP core(s). Some SoCs—called multiprocessor framework on chip (MPSoC)— incorporate more than one processor center.
- ✓ memory pieces including a choice of ROM, RAM, EEPROM and streak memory
- ✓ timing sources including oscillators and stage bolted circles
- ✓ peripherals including counter-clocks, ongoing clocks and power-on reset generators
- ✓ outer interfaces, including industry norms, for example, USB, FireWire, Ethernet, USART, SPI
- ✓ simple interfaces including ADCs and DACs
- ✓ voltage controllers and power administration circuits

Accessories

Numerous embellishments and peripherals for the Raspberry Pi go from USB center points, engine controllers to temperature sensors. There are some official embellishments for the RPi as takes after:

- ❖ **Camera –**On 14 May 2013, the establishment and the merchants RS Components and Premier Farnell/Element 14 propelled the Raspberry Pi camera board with a firmware redesign to bolster it.The Raspberry Pi camera board contains a 5 MPixel sensor, and interfaces through a strip link to the CSI connector on the Raspberry Pi. In Raspbian support can be empowered by the introducing or moving up to the most recent variant of the OS and after that running Raspi-config.

A bus - either exclusive or industry-standard, for example, the AMBA bus from ARM Holdings - interfaces these squares. DMA controllers course information straightforwardly between outside interfaces and memory, bypassing the processor center and accordingly expanding the information throughput of the SoC. also, selecting the camera choice. The cost of the camera module is 20 EUR in Europe (9 September 2013). what's more, backings 1080p, 720p, 640x480p video. The impression measurements are 25 mm x 20 mm x 9 mm.

- ❖ **Gertboard –**A Raspberry Pi Foundation authorized gadget intended for instructive purposes, and grows the Raspberry Pi's GPIO pins to permit interface with and control of LEDs, switches, simple signs, sensors and different gadgets. It likewise incorporates a discretionary Arduino perfect controller to interface with the Pi.
- ❖ **USB Hub –**In spite of the fact that not an official embellishment, it is an exceptionally suggested extra for the Pi. A fueled USB Hub with 7 additional ports is accessible at all online stores. It is mandatory to utilize a USB Hub to associate outer hard plates or different adornments that draw control from the USB ports, as the Pi can't offer energy to them.

8.1.2 Raspberry Pi interfaces

Raspberry Pi has serial, SPI and I2C interfaces as shown in **Figure 8-2.**

- **Serial:** The serial interface on Raspberry Pi has receive(rx) and transmit(Tx) pins for communication with serial peripherals.
- **SPI:** Serial Peripheral Interface(SPI) is a synchronous serial data protocol used for communicating with one or more peripheral devices. In an SPI connection, there is one master device and one or more peripheral devices. There are five pins on Raspberry Pi for SPI interface:
  - ✓ MISO(Master In Slave Out): Master line for sending data to the peripherals.
  - ✓ MOSI(Master out Slave In): Slave line for sending data to the master.
  - ✓ SCK(Serial Clock): Clock generated by master to synchronize data transmissions.
  - ✓ CE0(Chip Enable 0): To enable or disable devices.
  - ✓ CE0(Chip Enable 1): To enable or disable devices
- **I2C:** The I2C interface pins on Raspberry Pi allow you to connect hardware modules. I2C interface allows synchronous data transfer with just two pins-SDA(data line) and SCL(clock line).

8.3 Raspberrypi Operating Systems

Various operating systems can be installed on Raspberrypi through SD cards. Most use a MicroSD slot located on the bottom of the board. The Raspberrypi primarily uses Raspbian, a Debian-based Linux operating system. Other third party operating systems available via the official website include Ubuntu MATE, Snappy Ubuntu Core, Windows 10 IoT Core, RISC OS and specialized distributions for the Kodi media center and classroom management. Many other operating systems can also run on the Raspberrypi.

8.3.1 Operating Systems (not Linux based):

- ✓ RISC OS Pi (a special cut down version RISC OS Pico, for 16MB cards and larger for all models of Pi 1 and 2, has also been made available.
- ✓ FreeBSD
- ✓ NetBSD
- ✓ Plan 9 from Bell Labs and Inferno
- ✓ Windows 10 IoT core- a no cost edition of Windows 10 offered by Microsoft that runs natively on the RaspberryPi 2.
- ✓ xv6-is a modern reimplementation of sixth edition Unix OS for teaching purposes, it is ported to RaspberryPi from MIT Xv6, which can boot NOOBs.
- ✓ Haiku-is an open source BeOS clone that has can be compile for the RaspberryPi and several other ARM boards

8.3.2 Operating Sustems (Linux based):

- ✓ Xbian-using Kodi open source digital media center
- ✓ openSUSE
- ✓ Raspberry Pi Fedora remix
- ✓ Pidora, another Fedora Remix optimized for the RaspberryPi
- ✓ Gentoo Linux
- ✓ Diet Pi
- ✓ CentOS\OpenWrt
- ✓ Kali Linux
- ✓ Ark OS
- ✓ Kano OS
- ✓ Nard SDK

8.3.3 Media center operating systems

- ✓ OSMC
- ✓ OpenELEC
- ✓ LibreELEC
- ✓ Xbian
- ✓ Rasplex

8.3.4 Audio operating systems

- ✓ Volumio
- ✓ Pimusicbox
- ✓ Runeaudio
- ✓ moOdeaudio

8.3.5 Recalbox

- ✓ Happi Game Center
- ✓ Lakka
- ✓ ChameleonPi
- ✓ Piplay

8.4 Operating System Setup on Raspberrypi

Preinstalled NOOBS operating system is already available in many authorized as well as independent seller, there are many other operating system for Raspberrypi in the market like NOOBS, Raspbian and third party operating systems are also available like UBUNTU MATE, OSMC, RISC OS etc. To setup an operating system we need a SD card with minimum capacity of 8GB.

8.4.1 Formatting SD card:

Format the SD card before copying NOOBS onto it. To do this

1. Download SD formatter 4.0 from SD Association website for either Windows or Mac.
2. Follow the instructions to install the software.
3. Insert the SD card into the computer or laptops SD card reader and make a note of the drive letter allocated to it
4. In SD formatter, select the drive letter the SD card is and format it.

8.4.2 OS installation:

Follow the steps to install operating system in the SD card.

1. Go to raspberry pi foundation website and click on DOWNLOAD section.
2. Click on NOOBS, then click on the “Download ZIP” button under NOOBS(offline and network install) and select a folder to save this ZIP file
3. Extract all the files from ZIP.
4. Once SD card has been formatted, drag all the files in the extracted NOOBS folder and drop them onto the SD card drive.
5. The necessary files will then be transferred to the SD card.
6. When this process has finished, safely remove the SD card and insert it into the Raspberry Pi.

8.4.3 First Boot:

1. Plug in the keyboard, mouse, and monitor cables.
2. Now plug the USB cable into the Raspberrypi.
3. Now Raspberrypi will boot, and a window will appear with a list of different operating systems that we can install. We recommend using Raspbian- tick the box next to Raspbian and clicking on install.
4. Raspbian will then run through its installation process. Note that this can take a while.
5. When the install process has completed, the Raspberrypi configuration menu (raspi-config) will load. Here we should set the time and date for our region, enable a Raspberrypi camera board, or even create users.

8.4.5 LOGIN Information:

The default login for Raspbian is username “pi” with the password “raspberry”. To load the graphical user interface, type “startx” and press “Enter”.

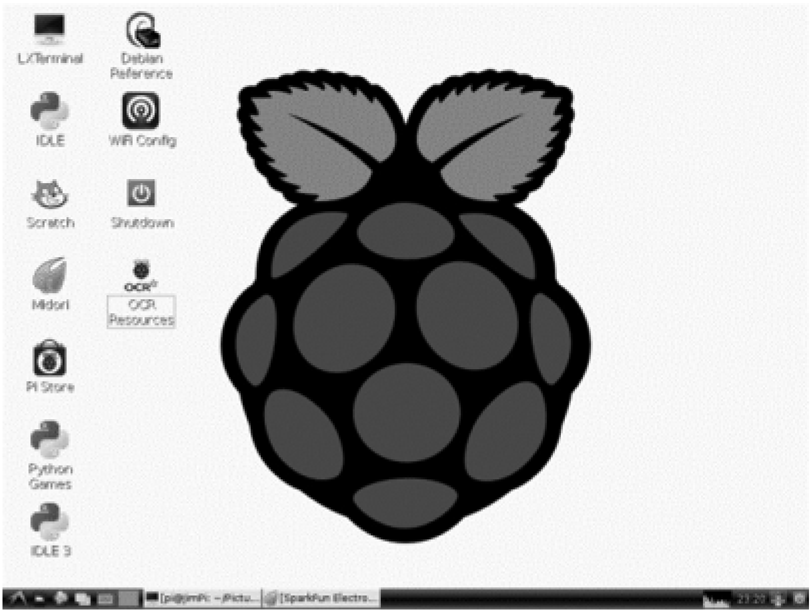


Figure 8-3: Raspbian desktop

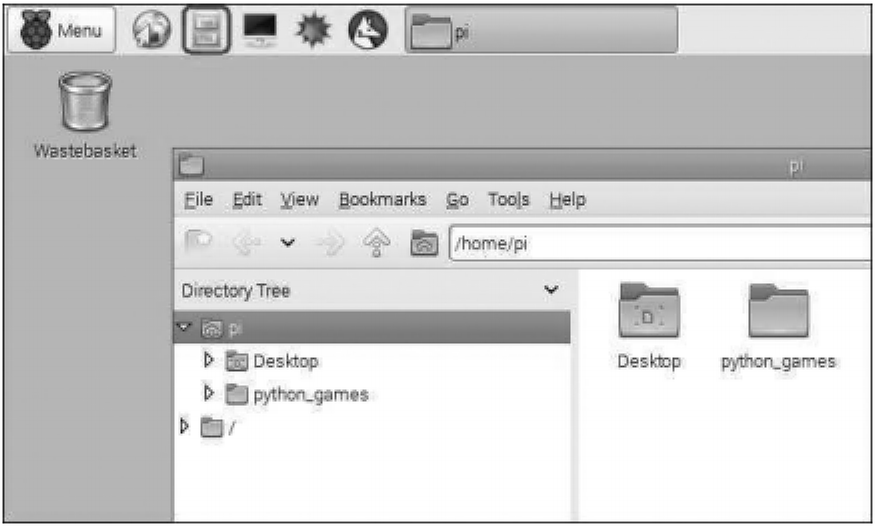
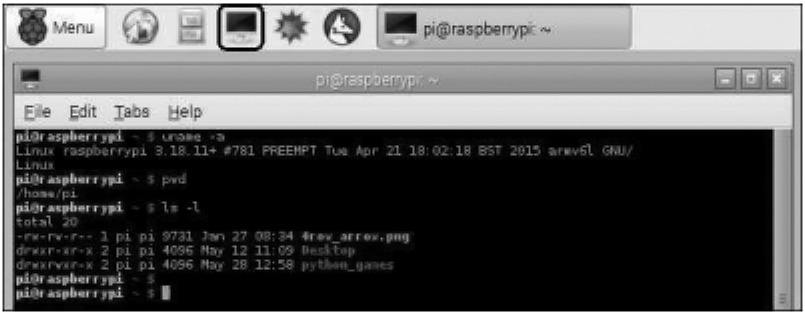


Figure 8-4: File explorer on Raspberry Pi

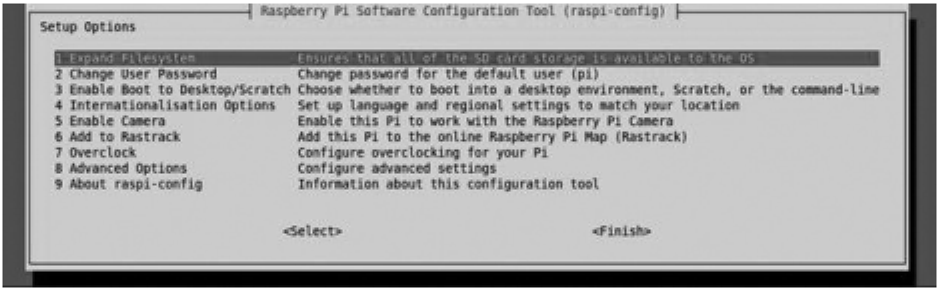


```
pi@raspberrypi ~ $ uname -a
Linux raspberrypi 3.18.11+ #781 PREEMPT Tue Apr 21 18:02:18 BST 2015 armv6l GNU/
Linux
pi@raspberrypi ~ $ pwd
/home/pi
pi@raspberrypi ~ $ ls -l
total 20
-rw-rw-r-- 1 pi pi 9731 Jan 27 08:34 4rev_arrow.png
drwxr-xr-x 2 pi pi 4096 May 12 11:09 Desktop
drwxrwxr-x 2 pi pi 4096 May 28 12:58 python_games
pi@raspberrypi ~ $
pi@raspberrypi ~ $ 
```

Figure 8-5: Console on RaspberryPi.

Menu
Raspberry Pi - Teach, Lear...
https://www.raspberrypi.org/
LATEST BLOG POST
EMOTI – VISUALISING OUR EMOTIONS
A project from some of our Creative Technologists at the Art Hackathon
WHAT IS A RASPBERRY PI?
Loading "www.raspberrypi.org"...

Figure 8-6: Browser on RaspberryPi



```
┤ Raspberry Pi Software Configuration Tool (raspi-config) ├
Setup Options

1 Expand Filesystem               Ensures that all of the SD card storage is available to the OS
2 Change User Password            Change password for the default user (pi)
3 Enable Boot to Desktop/Scratch  Choose whether to boot into a desktop environment, Scratch, or the command-line
4 Internationalisation Options    Set up language and regional settings to match your location
5 Enable Camera                   Enable this Pi to work with the Raspberry Pi Camera
6 Add to Rastrack                 Add this Pi to the online Raspberry Pi Map (Rastrack)
7 Overclock                       Configure overclocking for your Pi
8 Advanced Options                Configure advanced settings
9 About raspi-config              Information about this configuration tool

                <Select>                                <Finish>
```

Figure 8-7: RaspberryPi configuration tool

Figure 8-3 shows Raspbian pi desktop and **Figure 8-4** shows File explorer on Raspberry Pi. **Figure 8-5** shows console on Raspberry Pi, **Figure 8-6** shows browser on Raspberry Pi, raspi-config tool shown in **Figure 8-7** is used to configure Raspberry Pi to expand root partition to fill SD card, change password, setting time zone, enable SSH(Secure shell) server and change root behavior.

8.5 RapberryPi commands

The **Table 8-2** below gives out some of the useful commands used with Raspberrypi.

Table 8-2: Raspberrypi commands

General commands for RaspberryPi:

a. **raspi-config:** Configuration settings menu

b. **clear:** clears data from terminal.

c. **date:** current date.

d. **reboot:** Reboot immediately

e. **shutdown –**h now: Shutdown system immediately

f. **nano example.txt:** Opens the example.txt in the text editor nano.

g. **poweroff:** To shutdown immediately.

h. **shutdown –**h 01:22: To shutdown at 1:22 AM.

i. **apt-get update:** Synchronizes the list of packages on our system to the list in the repositories. Use this before installing new packages to make sure we are installing the latest version

j. **apt-get upgrade:** Upgrades all of the software packages we have installed.

k. **startx:** Opens the GUI

Directory and File commands:

a. **mkdir new_directory**: Creates a new directory named new_directory.

b. **mv new_folder:** Moves the file or directory named "new_folder" to a specified location

c. **rm new_file.txt:** Deleted the file new_file.txt.

d. **rmdir new_directory:** Deletes the directory "new_directory" only if it is empty.

e. **touch new_file.txt:** creates a new, empty file named new_file.txt in the current directory.

f. **cat new_file.txt:** Displays the contents of the file new_file.txt

g. **cd /xyz/abc:** Changes the current directory to the /xyz/abc directory.

h. **ls –l:** lists files in the directory.

Networking and Internet Commands:

a. **iwconfig:** Tp check which wireless adapter is currently active.

b. **ifconfig**: wireless connection ststus.

c. **ping:** tests connectivity between two devices connected on a network.

d. **wget http://www.website.com/new_file.txt**: Downloads the file new_file.txt from the web and saves it to the current directory.

e. **nmap:** Scans the network and lists connected devices, protocol, port number and other information.

f. **iwlist wlan0 scan:** list of currently available wireless networks.

System information commands:

a. **cat /proc/meminfo:** shows details about memory

b. **cat /proc/version:** shows which version of rsapberrypi we are using.

c. **df –h:** shows information about available disk space.

d. **df /:** shows how much free disk space is available.

e. **free:** shows how much free memory is available.

f. **hostname –I:** shows th ip address of the raspberrypi.

g. l**susb;** lists the usb hardware connected to raspberrypi.

h. **vcgencmd measure_temp:** shows the temperature of the cpu.

i. v**cgencmd get_mem arm && vcgencmd get_mem gpu:** shows the memory split between the cpu and gpu.

8.6 Programming RaspberryPi with Python:

In this section you will learn how to get started with developing python programs on Raspberry Pi. Raspberry Pi runs Linux and supports python out of the box. Henceforth you can run any python program that runs on a normal computer. However it is the general purpose input/output capability provided by the GPIO pins on Raspberry Pi that makes it useful device for Internet of Things. Raspberry pi can be interfaced with variety of sensors, actuators using GPIO pins and also SPI, I2C and serial interfaces. Input from the Raspberry Pi can be processed and actions can be taken, for instance, sending data to server, sending an email, triggering a relay switch. The **Table 8-3** below gives out the simple python programs that can be executed on Raspberrypi

Table 8-3: Simple python programs on RaspberryPi

Program	**Code**
Print hello world	print(“hello world”)

program to add two numbers
code
	a=1.2
	b=5.3
	sum=float(a)+float(b)
	print(“the sum of {0} and {1} is {2}”.format(a,b,sum))

program to roll a dice	import random
	min=1
	max=6
	roll_again=”yes”

	while roll_again=="yes" or roll_again=="y"
	        print("rolling the dices")
	        print("the values are")
	         print(random.randomint(min,max)
	         print(random.randint(min,max)

program to find the ip address of raspberrypi	import urllib
	import re
	print("we will try to open this url, in order to get ip address")
	url=http://checkip.dyndns.org
	print(url)

program to generate password	import string
	from random import*
	characters=string.ascii_letters+string.punctuation+string.digits
	password="".join(choice(charcters) for x in range(randint(8,16)))
	print(password)

program to print fibonacci series	a,b=0,1
	while b<200:
	      print(b)
	       a,b=b,a+b

program to check for armstrong number	num=int(input("enter a number:"))
	initial_sum=0
	temp=num
	while temp>0:
	digit=temp%10
	   initial_sum+=digit**3
	temp//=10
	if num==initial_sum:
	print(num,"is an armstrong number")
	else:
	   print(num,"is not an armstrong number")

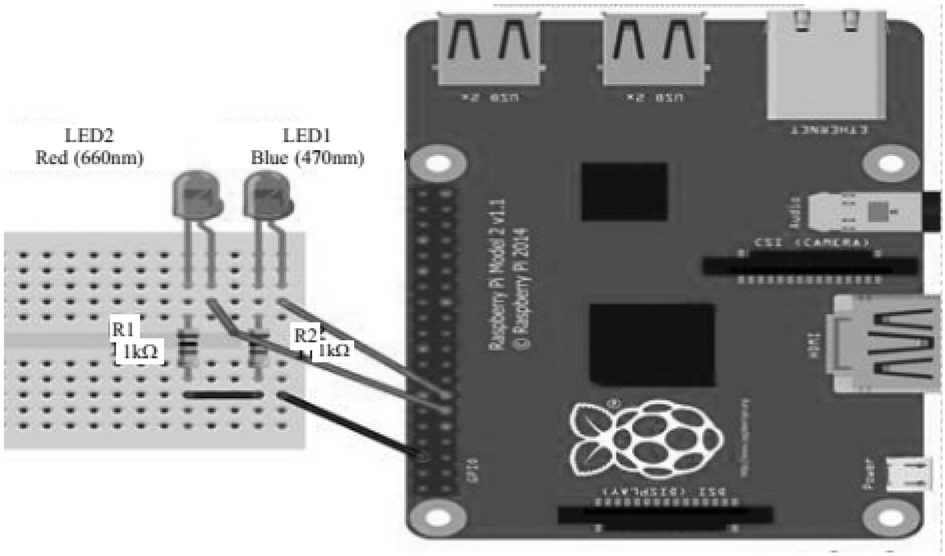
program to display calendar of given month of the year	import calendar
	yy=2017
	mm=11
	print(calendar.month(yy,mm))

Interfacing programs on Raspberrypi

Program #1	**Printing to a terminal**
Components required	Raspberry Pi + SD card, Monitor + HDMI Cable, Keyboard & Mouse and 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 #2	**Blinking an LED**
Components required	Raspberry Pi + SD card, Monitor + HDMI Cable, Keyboard & Mouse and Power supply, 1 Red LED and Blue LED, 2 1K resistors.
Description	Now let us look at an example of controlling the state of LEDs from ON to OFF state or vice versa from Raspberry Pi. Figure shows the schematic diagram of connecting an LEDs to Raspberry Pi. Box shows how to turn the LED on/off from by running the code given. In this example the LEDs are connected to GPIO pin 17 and 27 respectively which is initialized as output pin. The state of the LED is toggled by the executing the two programs given in the Box. Box
	having the code changes the state of the LED twice whereas Box having the code runs an infinite loop to flicker LEDs from ON/OFF state every second. Run the code given below and observe at the desired output.
Circuit Diagram  LED2 Red (660nm) LED1 Blue (470nm) R1 1kΩ R2 1kΩ USB 2x USB 2x ETHERNET Raspberry Pi Model 2 v1.1 © Raspberry Pi 2014 Audio CSI (CAMERA) HDMI GPIO DSI (DISPLAY) Power	
Key points	1. Create file "blink.py" 2. Create file "blink_ever.py" 3. Enter the above code 4. Run the python file "sudo python blink.py" << Watch the LEDs blink 2 times

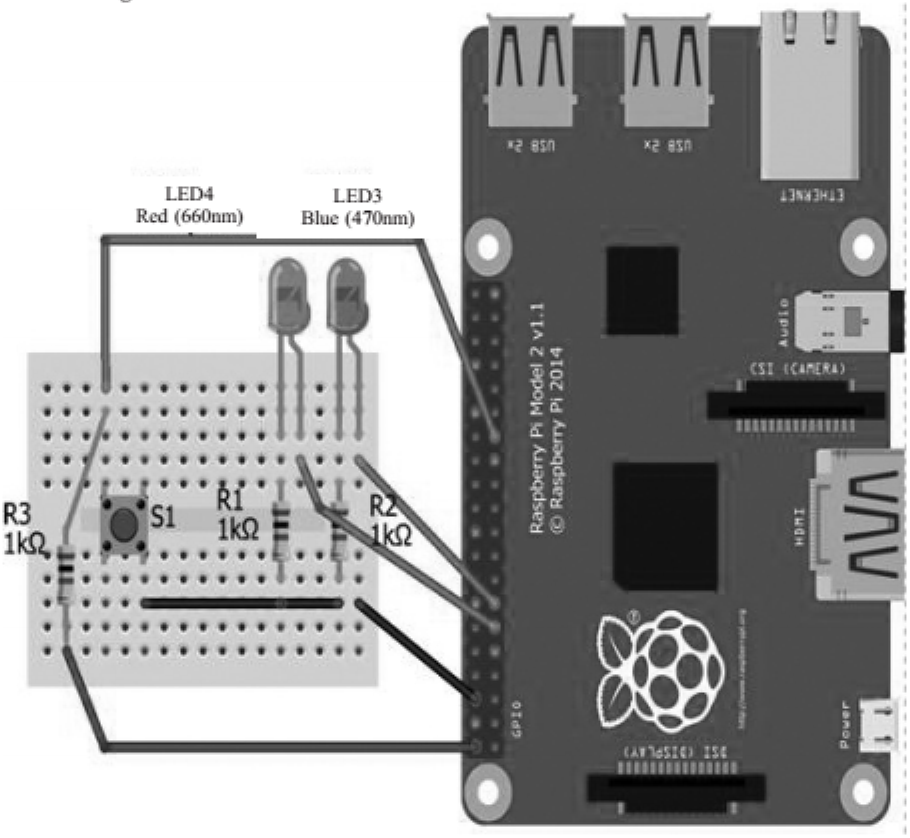
	5. Run the python file "sudo python blink_ever.py" << Watch the LEDs blink forever

Code	**File: blink.py**
	#Access the python working environment
	#!/usr/bin/python
	#import the time module so as to switch LEDs on/off with the time elapsed
	#import the RPI.GPIO library
	import RPi.GPIO as GPIO
	#use one of the two numbering system either BOARD numbers/BCM
	# Refer to the channel numbers on the Broadcom SOC.
	GPIO.setmode(GPIO.BCM)
	#Configure Pin 17 as an OUTPUT
	GPIO.setup(17,GPIO.OUT)
	#Configure Pin 27 as an OUTPUT
	GPIO.setup(27,GPIO.OUT)
	#Turn up LEDs on pin 17
	GPIO.output(17,GPIO.HIGH)
	#Turn up LEDs on pin 27
	GPIO.output(27,GPIO.HIGH)`
	#wait for 1 second
	time.sleep(1)
	#Turn up LEDs off on pin 17
	GPIO.output(17,GPIO.LOW)
	#Turn up LEDs off on pin 27
	GPIO.output(27,GPIO.LOW)
	#wait for 1 second
	time.sleep(1)
	File:blink_ever.py
	#Access the python working environment
	#!/usr/bin/python
	#import the time module so as to switch LEDs on/off with the time elapsed
	import time
	#import the RPI.GPIO library

```
import RPi.GPIO as GPIO
#use one of the two numbering system either BOARD numbers/BCM
# Refer to the channel numbers on the Broadcom SOC.
GPIO.setmode(GPIO.BCM)
#Configure pin 17 and 27 to be an OUTPUT pins
GPIO.setup(17,GPIO.OUT)
GPIO.setup(27,GPIO.OUT)
#Use while construct which runs infinite number of times there by blinking
        LEDs forever
while 1:
        #Turn up LEDs on
       GPIO.output(17,GPIO.HIGH)
        GPIO.output(27,GPIO.HIGH)
       time.sleep(1)
        #Turn up LEDs off
       GPIO.output(17,GPIO.LOW)
       GPIO.output(27,GPIO.LOW)
       time.sleep(1)
```

Output	LEDs turns on/off twice when blink.py file is executed and LEDs keeps changing their state forever when blink_forever.py file is executed.

Program #3	**Push button for physical input**
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.
Description	Now let us look at an example involving LEDs and a switch that is used to control LED. Figure shows the schematic diagram of connecting an LEDs and a switch to Raspberry Pi. Box shows a python program for controlling an LED with a switch.In the infinite while loop the value of pin 10 is checked and the state of the LED is toggled if the switch is pressed. This example shows how to get input from GPIO pins and process the state of the LEDS. Run the code given below and observe at the desired output.

Circuit Diagram	
 LED4 Red (660nm) LED3 Blue (470nm) R3 1kΩ S1 R1 1kΩ R2 1kΩ USB 2x USB 2x ETHERNET Raspberry Pi Model 2 v1.1 © Raspberry Pi 2014 Audio CSI (CAMERA) HDMI GPIO DSI (DISPLAY) Power	
Key points	1. Create file "button.py" 2. Enter the above code 3. To run the python code "sudo python button.py"
Code	File: button.py #Access the python working environment #!/usr/bin/python #Import os module to enable interrupts from a push button import os #import the time module so as to know the time the user as given the input from a push button

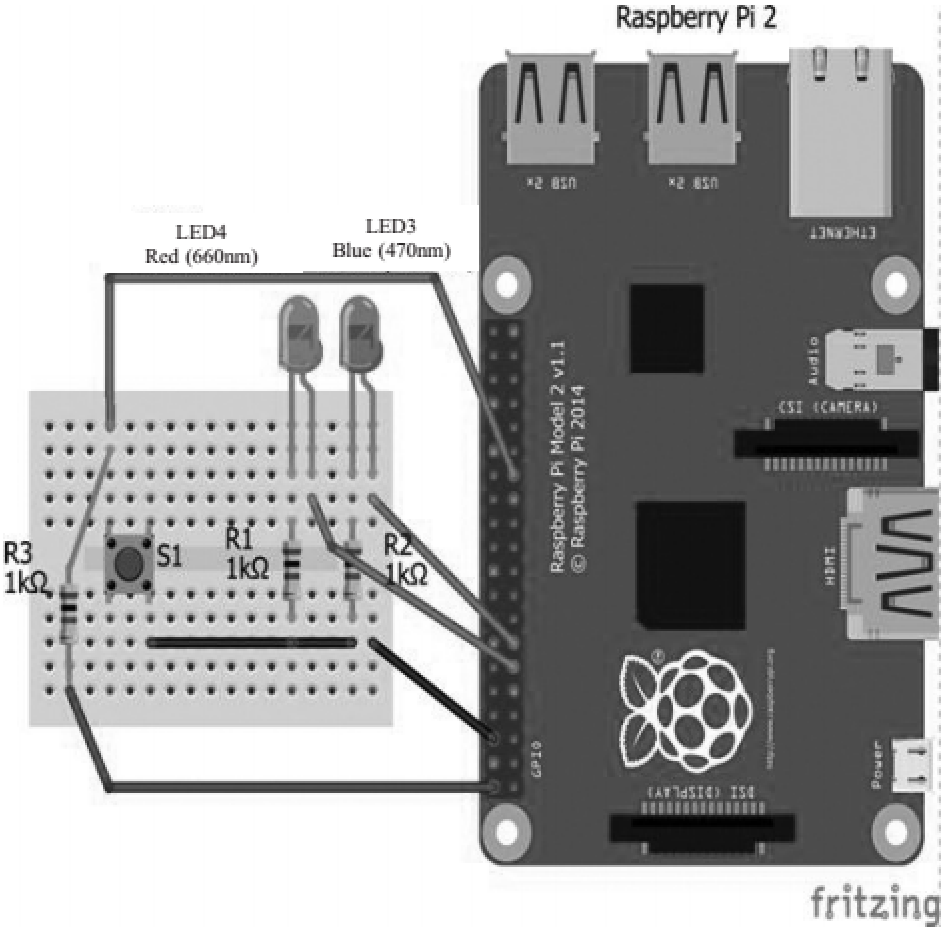
```
import time
#import the RPI.GPIO library
import RPi.GPIO as GPIO
#use one of the two numbering system either BOARD numbers/BCM
# Refer to the channel numbers on the Broadcom SOC.
GPIO.setmode(GPIO.BCM)
#configure pin 10 to be INPUT which reads the status of a switch button
GPIO.setup(10, GPIO.IN)
#print a message on to the terminal
print(" Button + GPIO ")
#Read the status of a button from GPIO pin 10
print GPIO.input(10)
#Run a infinite loop on the status of the button read
while True:
  if ( GPIO.input(10) == True ):
      print("Button Pressed")
     #Print the time when the input was given from the push button
      os.system('date')
     #Read the status of a button from GPIO pin 10
      print GPIO.input(10)
     #wait for 5 seconds
      time.sleep(5)
  else:
      #clear the system variables
     os.system('clear')
      #prompt the user to give an input
     print ("Waiting for you to press a button")
time.sleep(1)
```

Output	Press the Push button switch to Turn ON/OFF the LED

Program #4	**Interact with the user**
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.
Description	Now let us look at an example involving LEDs and a switch that is used to control LED depending on what a user choose. Figure shows the schematic

	diagram of connecting an LEDs and a switch to Raspberry Pi. Box shows a python program for controlling an LED by reading two input values, one for which LED would user like to blink(option 1-for Red, 2- for Blue) and one more parameter to set the maximum number of times the LED should be flickered. This example shows how to get input from a user and process the state of the LEDS. Run the code given below and observe at the desired output.

Circuit Diagram



Key points	1. Create file "user_input.py" 2. Enter the above code

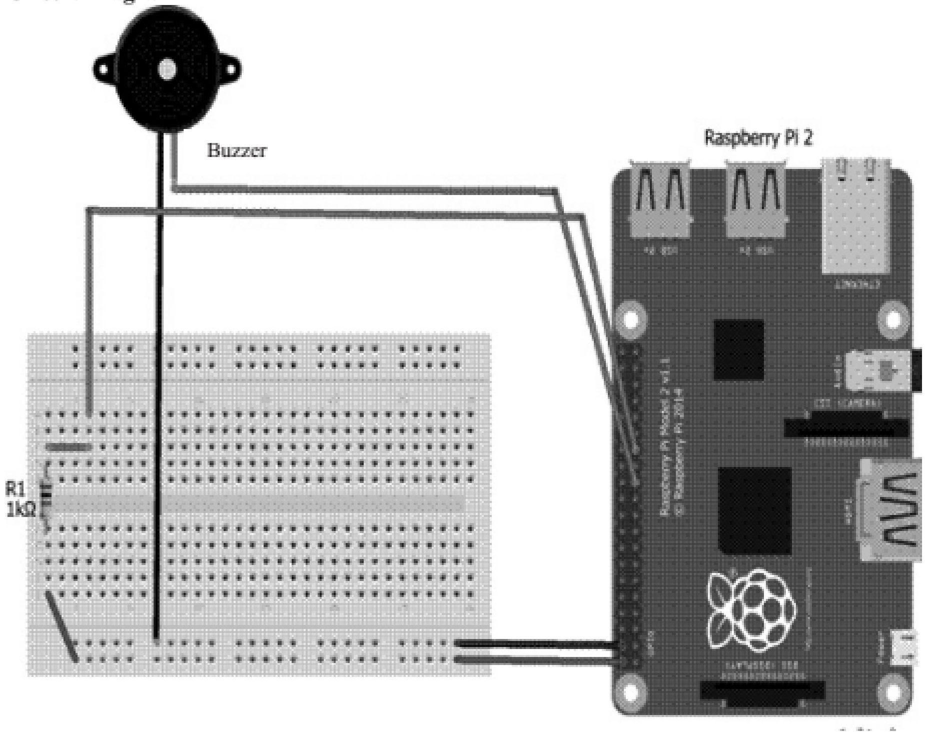
	3. Run the python file by "sudo python user_input.py" << Run through the questions and make an LED blink.

Code	**File: user_input.py**
	`#Access the python working environment`
	`#!/usr/bin/python`
	`#Import os module to enable interrupts from a push button`
	`import os`
	`#import the time module so as to switch LEDs on/off with the time elapsed`
	`import time`
	`#import the RPI.GPIO library`
	`import RPi.GPIO as GPIO`
	`#use one of the two numbering system either BOARD numbers/BCM`
	`# Refer to the channel numbers on the Broadcom SOC`
	`GPIO.setmode(GPIO.BCM)`
	`#configure pin 17 to be OUTPUT pin`
	`GPIO.setup(17,GPIO.OUT)`
	`#configure pin 27 to be OUTPUT pin`
	`GPIO.setup(27,GPIO.OUT)`
	
	`#Initialize variables for user input`
	`led_ch = 0`
	`counter = 0`
	
	`#clear the python interpreter console`
	`os.system('clear')`
	
	`print "Which LED would you like to blink"`
	`#Accept 1 for Red`
	`print "1: Red?"`
	`#Accept 2 for Blue`
	`print "2: Blue?"`
	
	`led_ch = input("Choose your option: ")`
	
	`if led_ch == 1:`

```
    #clear the python interpreter console
    os.system('clear')
    print "You picked the Red LED"
    counter = input("How many times would you like it to blink?: ")
    while counter > 0:
        #on LED on Pin 27
        GPIO.output(27,GPIO.HIGH)
        time.sleep(1)
        #off LED on pin 27
        GPIO.output(27,GPIO.LOW)
time.sleep(1)
        #Record the number of counts on LED
        counter = counter – 1
if led_ch== 2:
    #clear the python interpreter console
    os.system('clear')
    print "You picked the Red LED"
    counter = input("How many times would you like it to blink?: ")
    while counter > 0:
        #on LED Pin 27
        GPIO.output(17,GPIO.HIGH)
        time.sleep(1)
        #off LED on pin 27
        GPIO.output(17,GPIO.LOW)
        time.sleep(1)
        #Record the number of counts on LED
        count = count – 1
```

Output	LED gets flickered on the inputs given by the user.

Program #5	**Buzzer**
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.
Description	Now let us look at an example involving a Piezo buzzer and a switch that is used to beep a number of times the user choose. Figure shows the schematic

	diagram of connecting an piezo buzzer to pin 22 and a switch to Raspberry Pi. Box shows a python program for controlling an piezo buzzer by reading an input value which runs over a loop to beep number of times the user has choose. Run the code given below and observe at the desired output.
Circuit Diagram 	
Key points	1. Create file "buzzer.py" 2. Enter the code above code 3. To run python code "sudo python buzzer.py"<<listen to it beep

Code	**File: buzzer.py**
	#!/usr/bin/python
	import os
	import time
	import RPi.GPIO as GPIO
	GPIO.setmode(GPIO.BCM)

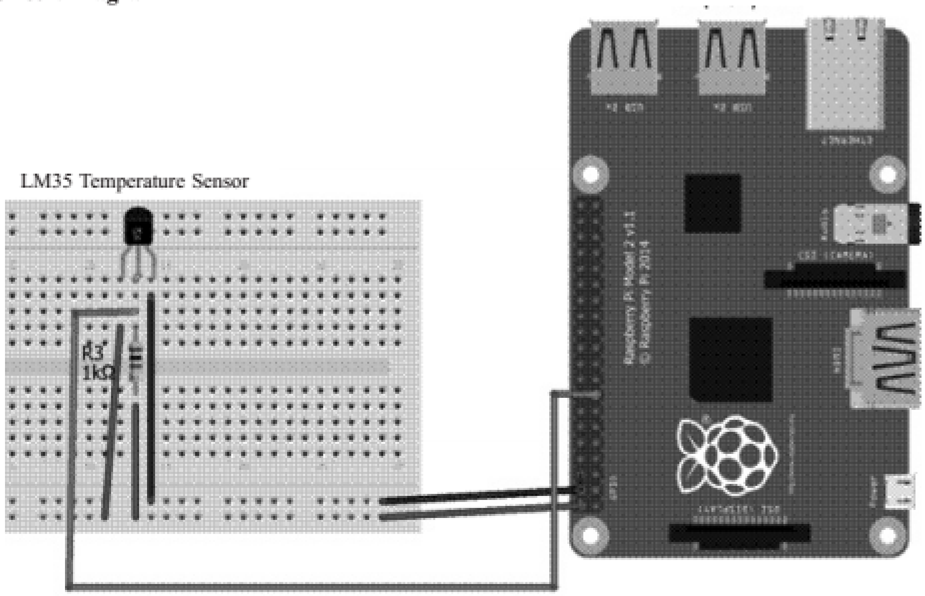
```
GPIO.setwarnings(False)
GPIO.setup(22,GPIO.OUT)
loop_counter = 0
def morsecode ():
        #Dot Dot Dot
        GPIO.output(22,GPIO.HIGH)
        time.sleep(.1)
        GPIO.output(22,GPIO.LOW)
        time.sleep(.1)
        GPIO.output(22,GPIO.HIGH)
        time.sleep(.1)
        GPIO.output(22,GPIO.LOW)
        time.sleep(.1)
        GPIO.output(22,GPIO.HIGH)
        time.sleep(.1)
        #Dash Dash Dah
        GPIO.output(22,GPIO.LOW)
        time.sleep(.2)
        GPIO.output(22,GPIO.HIGH)
        time.sleep(.2)
        GPIO.output(22,GPIO.LOW)
        time.sleep(.2)
        GPIO.output(22,GPIO.HIGH)
        time.sleep(.2)
        GPIO.output(22,GPIO.LOW)
        time.sleep(.2)
        GPIO.output(22,GPIO.HIGH)
        time.sleep(.2)
        GPIO.output(22,GPIO.LOW)
        time.sleep(.2)
```

```
          #Dot Dot Dot
          GPIO.output(22,GPIO.HIGH)
          time.sleep(.1)
          GPIO.output(22,GPIO.LOW)
          time.sleep(.1)
          GPIO.output(22,GPIO.HIGH)
          time.sleep(.1)
          GPIO.output(22,GPIO.LOW)
          time.sleep(.1)
          GPIO.output(22,GPIO.HIGH)
          time.sleep(.1)
          GPIO.output(22,GPIO.LOW)
          time.sleep(.7)
os.system('clear')
print "Morse Code"
loop_count = input("How many times would you like SOS to loop?: ")
while loop_count > 0:
          loop_counter = loop_counter - 1
          morsecode ()
```

Output	listen to beep of piezo buzzer

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.

Circuit Diagram



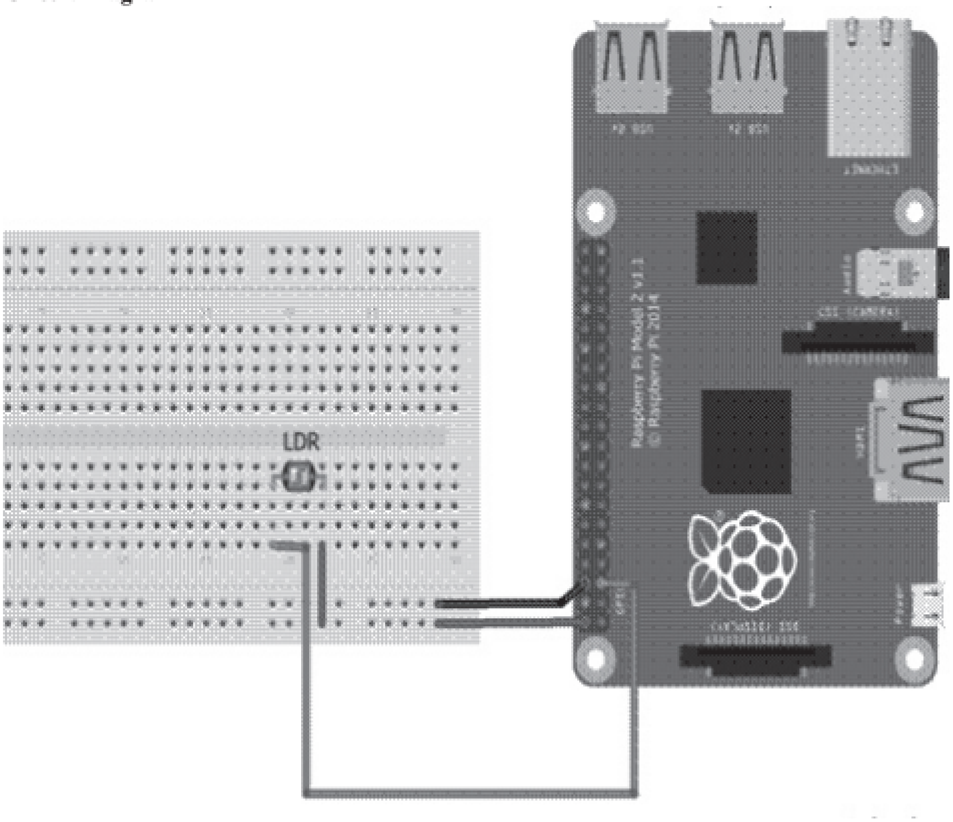
Code

```
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.

Program #7	**Light 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.LM35 temperature sensor, LDR Light Dependent resistor
Description	Now let us look at an example involving an LDR sensor which reads the intensity of light and records it an text file. Figure shows the schematic diagram of connecting an LDR sensor to Raspberry Pi board. Box shows a python program which records the intensity of light to a text file. 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 intensity of light with date and time stamps every second Compile the code given below and upload it to Arduino UNO Board to observe at the desired output.

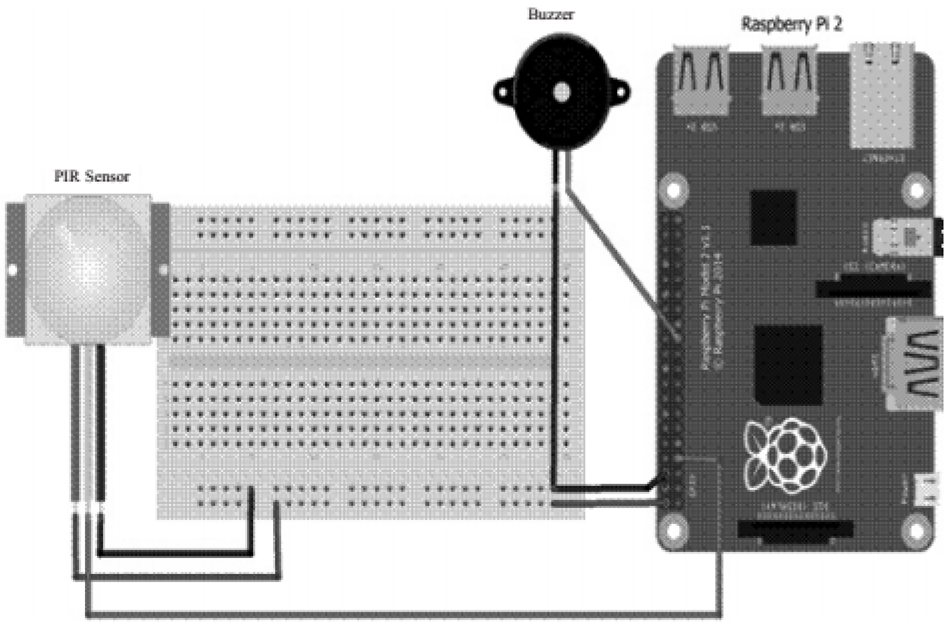
Circuit Diagram	
	
Key points	1. Create file "ldr.py" then "touch foo.txt" 2. To run the python code "sudo python ldr.py" << See what the light levels in the room are. 3. The check the file "more foo.txt" you can see your results.

Code	**File: ldr.py**
	#!/usr/bin/env python
	import os
	import datetime
	import time
	import RPi.GPIO as GPIO

```
DEBUGER = 1
GPIO.setmode(GPIO.BCM)
def RCtimer (RCpins):
        readings = 0
       GPIO.setup(RCpins, GPIO.OUT)
        GPIO.output(RCpins, GPIO.LOW)
       time.sleep(.1)

       GPIO.setup(RCpins, GPIO.IN)
        # iterates 1 miliseconds over one cycle
       while (GPIO.input(RCpins) == GPIO.LOW):
              readings += 1
       return readings
while True:
              GetDateTime = datetime.datetime.now().strftime("%Y-%m-%d
%H:%M:%S")\
              LDRReading = RCtimes(3)
              print RCtimes(3)

              # Open a file
              fo = open("/home/pi/10x10/foo.txt", "wb")
              fo.write (GetDateTime)
              LDRReading = str(LDRReading)
              fo.write ("\n")
              fo.write (LDRReading)

              # Close opend file
              fo.close()
              time.sleep(1)
```

Output	Intensity of the light in the room is recorded on to a terminal as well as to text file.

Program #8	**Passive Inferred 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.LM35 temperature sensor, LDR Light Dependent resistor, PIR(Passive Infrared sensor) sensor.
Description	Now let us look at an example involving an PIR sensor which detects the motion of an object. Figure shows the schematic diagram of connecting an PIR sensor to Raspberry Pi board. Box shows a python program which the motion of an object and triggers a message as "Motion detected".This example shows how to get an analog input from GPIO pins and process the input. An infinite loop runs over the PIR sensor which waits for any of the movements across its boundary. Compile the code given below and upload it to Arduino UNO Board to observe at the desired output.
Circuit Diagram	
Key points	1. Create file "touch python pir.py" 2. To run the python code "sudo python pir.py" << Move in front of the PIR to activate it.

Code	**File: PIR.py**
	#!/usr/bin/python
	import RPi.GPIO as GPIO
	import time

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(27,GPIO.OUT)
GPIO_PIR_sensor = 7

print "PIR Module Test (CTRL-C to exit)"
# configure pin to be input
GPIO.setup(GPIO_PIR_sensor,GPIO.IN)

CurrentState  = 0
PreviousState = 0

try:
 print "Waiting for PIR to settle ..."
  # iterate till PIR  outputs the value 0
 while GPIO.input(GPIO_PIR_sensor)==1:
    CurrentState  = 0
 print "  Ready"
  # Iterate until user types CTRL-C
 while True :
    # status of the PIR to be read
   CurrentState = GPIO.input(GPIO_PIR_sensor)
    if CurrentState==1 and PreviousState==0:
     # Trigger action on PIR
      print "  Motion detected!"
     # Previous status of the PIR to be recorded
      GPIO.output(27,GPIO.HIGH)
     time.sleep(1)
      GPIO.output(27,GPIO.LOW)
     PreviousState=1
    elif CurrentState==0 and PreviousState==1:
     # check if PIR has arrived to the ready state
      print "  Ready"
      PreviousState=0
    # stop for 10 milliseconds
```

	`    time.sleep(0.01)`
	`except KeyboardInterrupt:`
	`  print "  Quit"`
	`   # GPIO settings to be reset`
	`    GPIO.cleanup()`
Output	Move in front of the PIR to activate it and sensor generates a message..

8.7 Summary

In this chapter you learned about Raspberry Pi which is a low cost mini computer. Raspberry Pi supports various flavors of Linux operating system. The official recommended operating system is Raspbian Linux. Raspberry Pi has an ARM processor, 512 MB RAM, two USB ports, HDMI, RCA and audio outputs, Ethernet port, SD card slot and DSI and CSI interfaces which depends on the version of Raspberry Pi used. Raspberry Pi has serial, SPI and I2C interfaces for data transfer. Raspberry Pi supports python. You learned how to develop Python Programs that run on the Raspbery Pi. You learned how to interface LED, switch, LDR etc

8.8 Objective Questions

1. **Raspberry Pi supports the concept of cross platform.**

   A. True   B. False

Ans: A. True

2. **Raspberry Pi is an Open Source and extensible software but not hardware.**

   A. True   B. False

Ans: B. False

3. **Which of the following Raspberry Pi model support wireless connectivity**

   A. Model A+   B. Model B   C. Model 3   D. Model B+

Ans: C. Model 3

4. **How many GPIO pins are there in Raspberry Pi model 3?**

   A. 26   B. 30   C. 40   D. 42

Ans: C.40

5. **Which of the following programming dialect doesnot come with operating system of Raspberry Pi?**

   A. C   B. C++   C. Java   D. Python

Ans: D. Python

6. **What is the processing speed of Model A+ of Raspberry Pi?**

   A. 700MHz   B. 900MHz   C. 1.2GHz   D. 1.4GHz

Ans: A. 700MHz

7. **How many USB ports are ther in Model 3 of Raspberry Pi?**

A. 1 B. 2 C. 3 D. 4

Ans: D. 4

8. What is the voltage on which Raspberry Pi is to be operated?

A. 4 B. 5 C. 6 D. 10

Ans: B. 5

9. By default what is the username of raspberry pi?

A. Pi B. Raspberry C. Raspberrpi D. Piraspberry

Ans: A. Pis

10. Which is the XBMC media center distribution of raspberry Pi?

A. Pidora B. Raspbian C. RISC OS D. OpenELEC

Ans: D. OpenELEC

8.9 Self test Questions

1. What is a raspberry pi?
2. What is the username and password for the raspberry pi?
3. Why does nothing happen when i type in my password, did my raspberry pi freeze?
4. What is the difference between model a and model b?
5. How do i connect a mouse and keyboard?
6. Where is the on/off switch?
7. Who or what is noobs?
8. What soc are you using?
9. What is a soc?
10. Does raspberry pi overclock?
11. Does raspberry pi need a heatsink?
12. What hardware interfaces does raspberry pi have?
13. Why is there no real time clock (rtc)?
14. What displays can i use with raspberry pi ?
15. Does the hdmi port support cec?
16. Why is there no vga support?
17. What are the power requirements for rapberry pi?
18. Can i power the raspberry pi from a usb hub?
19. Can i power the raspberry pi from batteries as well as from a wall socket?
20. What operating system (os) does rapberry pi use?
21. Does rapberry pi have an official programming language?
22. Will rapberry pi run wine (or windows, or other x86 software)?
23. Will rapberry pi run the windows 8 arm edition?
24. Will rapberry pi run android?

25. Does the device support networking?
26. Why is there no gigabit ethernet?

8.10 Review Questions

1. How is Raspberry Pi different from a desktop computer?

2. What is the use of GPIO pins?

3. What is the use of SPI and I2C interfaces on Raspberry PI?

8.11 Glossary

BGA: ball grid array. A type of surface mount packaging for electronics.

SoC: system on chip. A computer on a single chip.

GPIO: General purpose input/output. A pin that can be programmed to do stuff.

GPU: graphics processing unit. The hardware the handles the graphics.

Distro: a specific package ("flavour") of Linux and associated software.

Brick: to accidentally render a device inert by making chsanges to software or firmware.

Pxe: preboot execution environment. A way to get a device to boot by via the network.

PoE: power over Ethernet. Powering a device via an Ethernet cable.