FUTURE VISION BIE

By K B Hemanth Raj

Visit : https://hemanthrajhemu.github.io

A Small Contribution Would Support Us.

Dear Viewer,

Future Vision BIE is a free service and so that any Student/Research Personal **Can Access Free of Cost**.

If you would like to say **thanks**, you can make a **small contribution** to the author of this site.

Contribute whatever you feel this is worth to you. This gives **us support** & to bring **Latest Study Material** to you. After the Contribution Fill out this Form (https://forms.gle/tw3T3bUVpLXL8omX7). To Receive a **Paid E-Course for Free**, from our End within 7 Working Days.

Regards

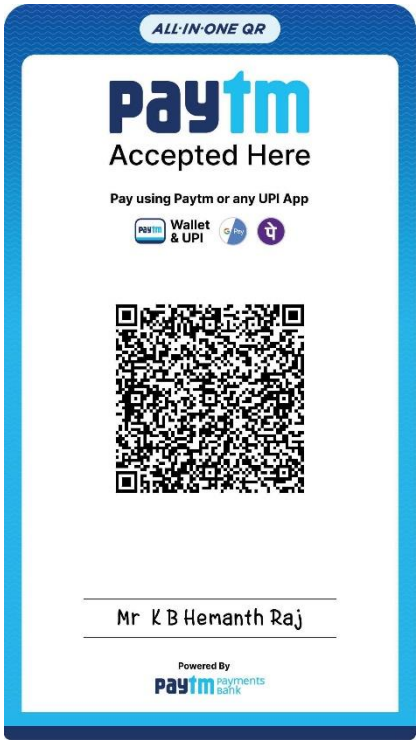
- K B Hemanth Raj (Admin)

Contribution Methods

UPI ID

1. futurevisionbie@oksbi
2. futurevisionbie@paytm

Scan & Pay



Account Transfer

Account Holder's Name: K B Hemanth Raj

Account Number: 39979402438

IFSC Code: SBIN0003982

MICR Code: 560002017

More Info: https://hemanthrajhemu.github.io/Contribution/

Gain Access to All Study Materials according to VTU,
CSE – Computer Science Engineering,
ISE – Information Science Engineering,
ECE - Electronics and Communication Engineering & MORE...

Stay Connected... get Updated... ask your queries...

Join Telegram to get Instant Updates: https://bit.ly/VTU_TELEGRAM

Contact: MAIL: futurevisionbie@gmail.com

INSTAGRAM: www.instagram.com/futurevisionbie/

WHATSAPP SHARE: https://bit.ly/FVBIESHARE

```
A0 = dict(zip(('a','b','c','d','e'),(1,2,3,4,5)))
A1 = range(10)
A2 = sorted([i for i in A1 if i in A0])
A3 = sorted([A0[s] for s in A0])
A4 = [i for i in A1 if i in A3]
A5 = {i:i*i for i in A1}
A6 = [[i,i*i] for i in A1]
```

CHAPTER 7

IoT Physical Devices and Endpoints: Arduino UNO

This Chapter Covers

- Introduction to Arduino
  - ✓ Why Arduino?
  - ✓ Which Arduino?
- Exploring Arduino Uno Learning Board
  - ✓ Things that Arduino can do
- Installing the Software (Arduino IDE)
  - ✓ Connecting Arduino Uno Learning Board
- Fundamentals of Arduino Programming
  - ✓ Difference between Analog, Digital and PWM Pins
- Introduction to Communications
  - ✓ SPI communications
- Example Modules on Arduino
  - ✓ Programs to interface LED switch and Potentiometer
  - ✓ Programs to interact with Serial Monitor of our Computer Screen
  - ✓ Interfacing Sensors to the Arduino
  - ✓ Interfacing Display, GSM, GPS to Arduino
  - ✓ Interfacing Motors

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.

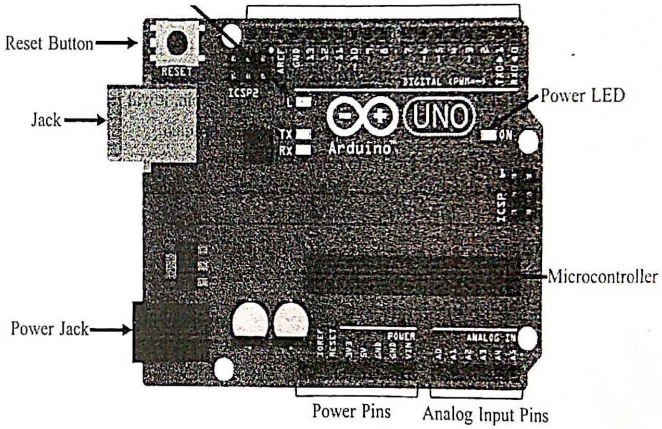


Figure 7-1: Arduino Uno Learning Board

- 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:

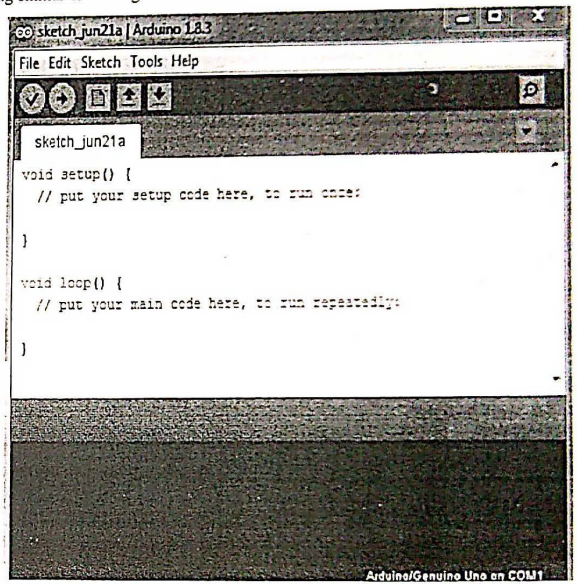


Figure 7.2: Arduino IDE

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.

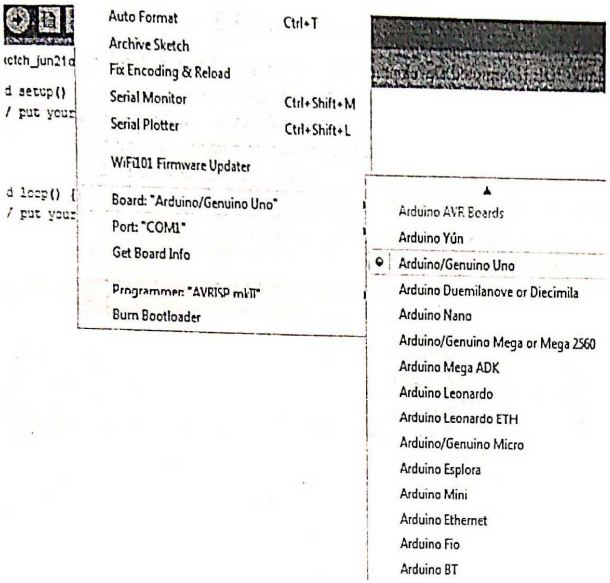
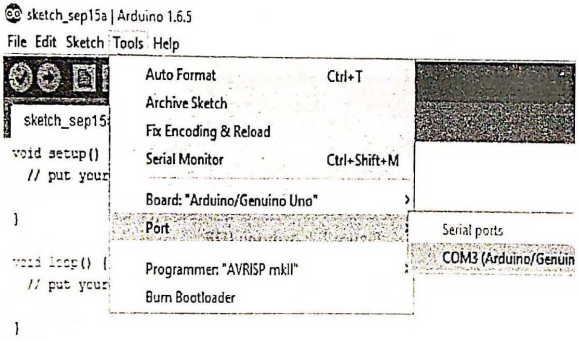


Figure 7-3: Selecting the right Board



10 selecting the right port

Figure 7-4: showing the layout of Arduino IDE

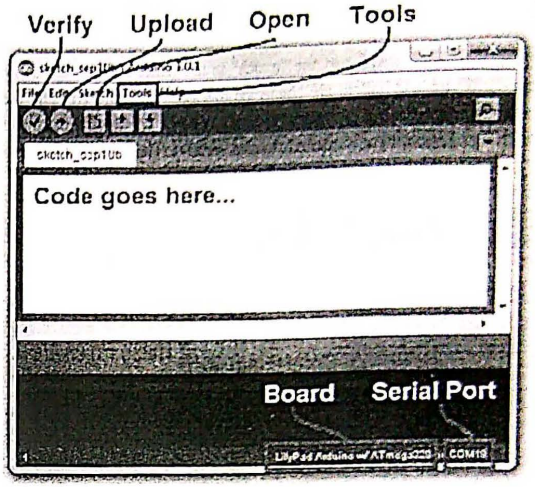


Figure 7-5: Layout of Arduino Uno IDE

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.

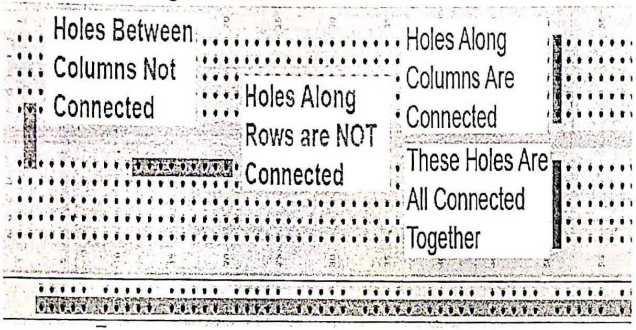


Figure 7-6: Breadboard for prototyping Arduino Uno circuits

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	16MIlz

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 of function Name(parameters)`

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.

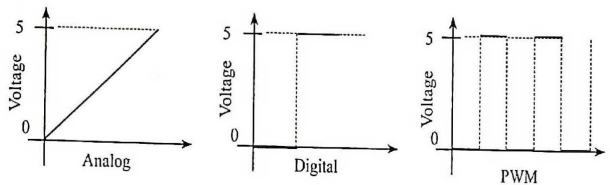


Figure 7-7: Difference between Analog, Digital and PWM Pins

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 I²C 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.

7.10 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

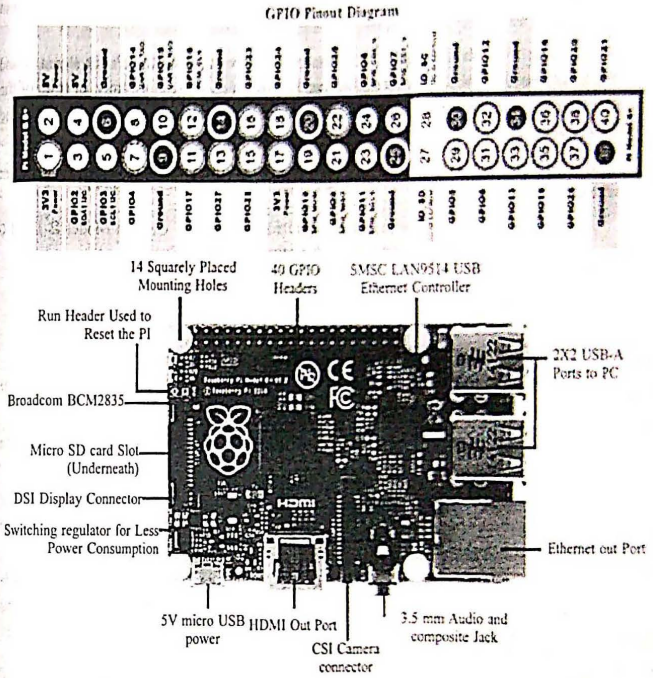


Figure 8-1: Raspberry Pi2 Model B and its GPIO

- ✓ 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 Systems (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".

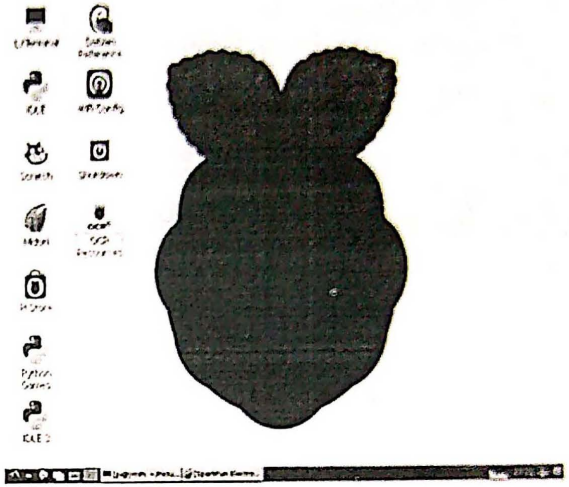


Figure 8-3: Raspbian desktop

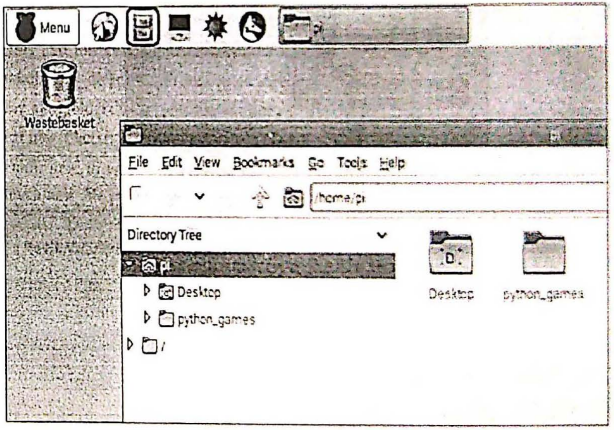


Figure 8-4: File explorer on Raspberry Pi

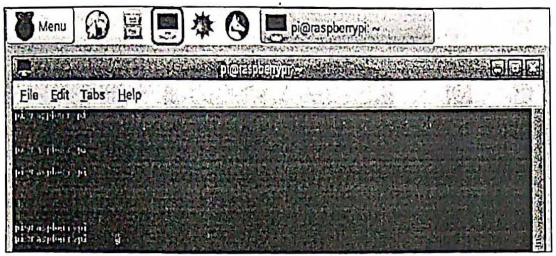


Figure 8-5: Console on RaspberryPi.

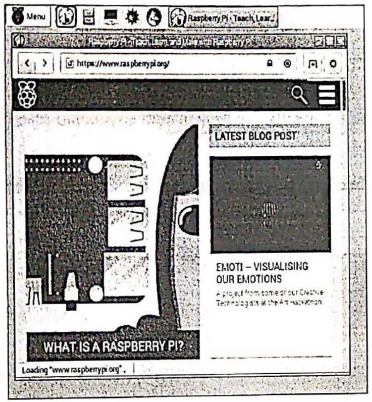


Figure 8-6: Browser on RaspberryPi

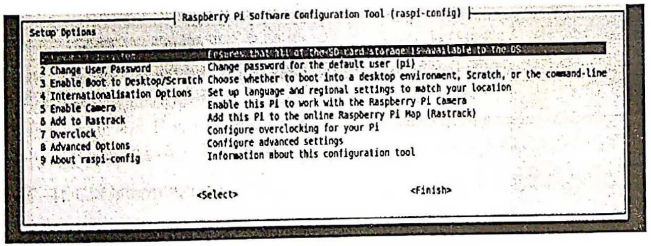


Figure 8-7: Raspberry Pi configuration tool

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. **lsusb;** lists the usb hardware connected to raspberrypi.
h. **vcgencmd measure_temp:** shows the temperature of the cpu.
i. **vcgencmd 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")

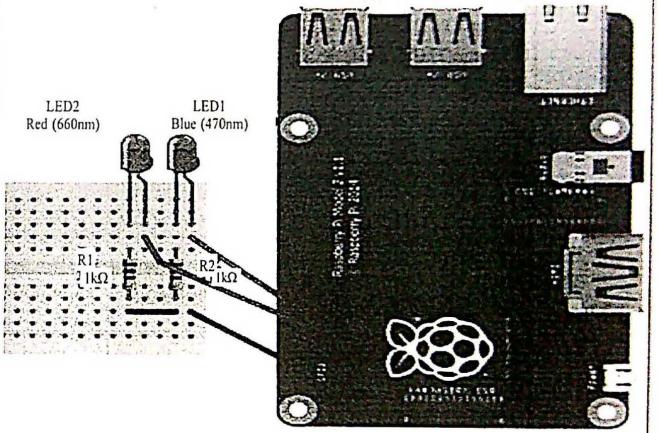
program to display calendar of given month of the year	import calendar yy=2017 mm=11 print(calendar.month(yy,mm))

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	
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

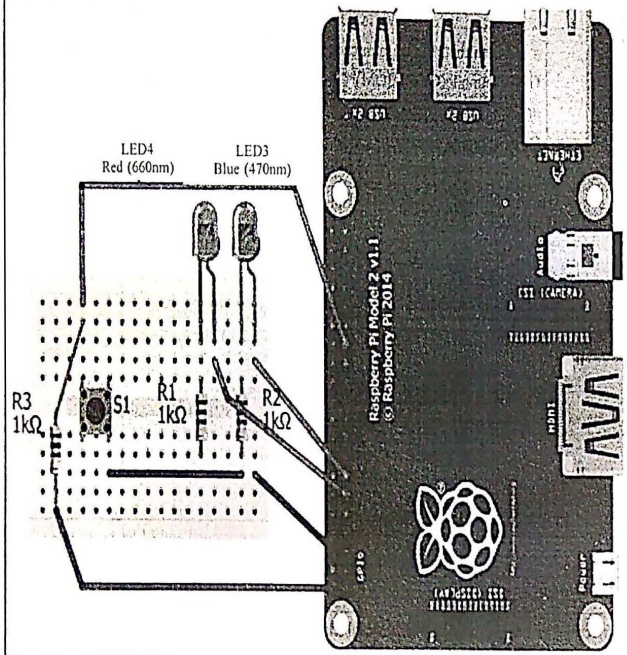
Code	**File: blink.py**
	#Access the python working environment
	#!/usr/bin/python
	#import the time module so as to switch LEDs on/off with the time elapsed
	#import the RPI.GPIO library
	import RPi.GPIO as GPIO
	#use one of the two numbering system either BOARD numbers/BCM
	# Refer to the channel numbers 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

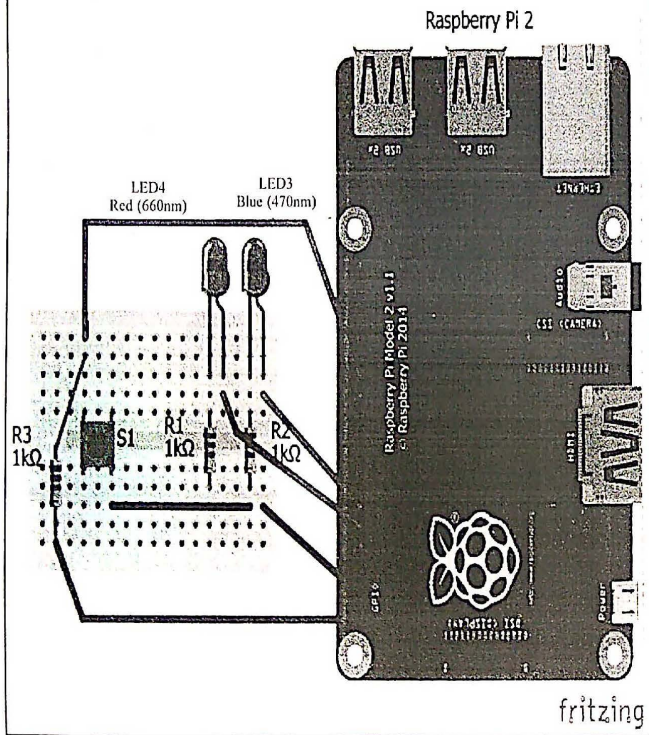


Key points	1. Create file "button.py" 2. Enter the above code 3. To run the python code "sudo python button.py"
Code	File: button.py #Access the python working environment #!/usr/bin/python #Import os module to enable interrupts from a push button import os #import the time module so as to know the time 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.

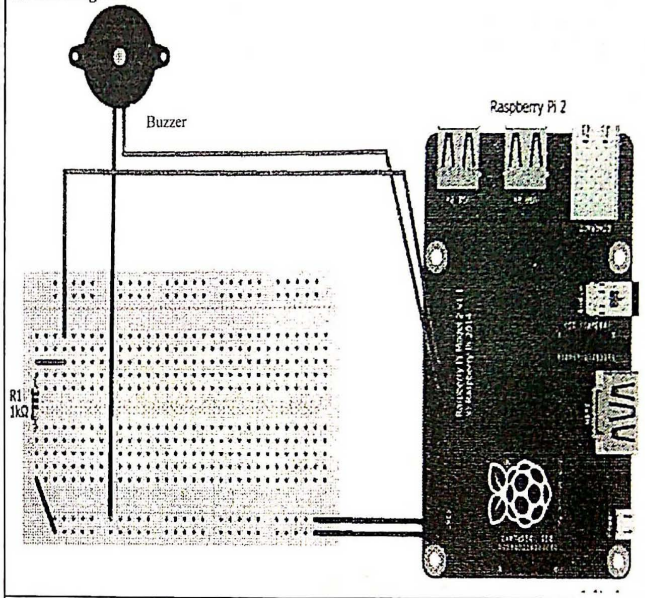
Code	**File: user_input.py**
	#Access the python working environment
	#!/usr/bin/python
	#Import os module to enable interrupts from a push button
	import os
	#import the time module so as to switch LEDs on/off with the time elapsed
	import time
	#import the RPI.GPIO library
	import RPi.GPIO as GPIO
	#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)
```

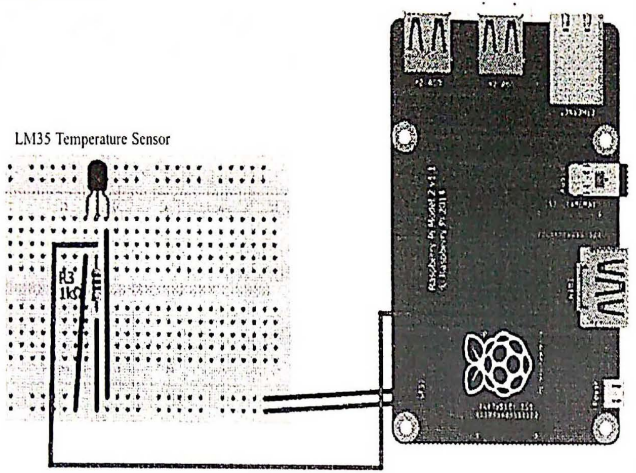


```
        #Dot Dot Dot
        GPIO.output(22,GPIO.HIGH)
        time.sleep(.1)
        GPIO.output(22,GPIO.LOW)
        time.sleep(.1)
        GPIO.output(22,GPIO.HIGH)
        time.sleep(.1)
        GPIO.output(22,GPIO.LOW)
        time.sleep(.1)
        GPIO.output(22,GPIO.HIGH)
        time.sleep(.1)
        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



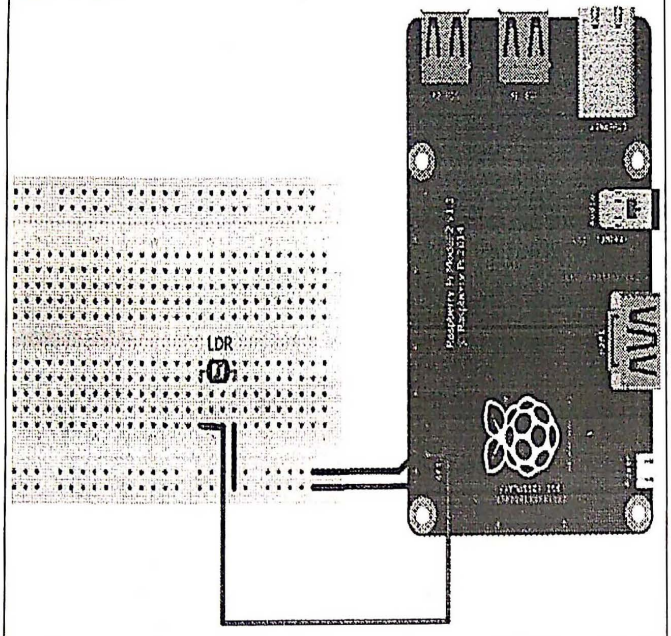
Code

```
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

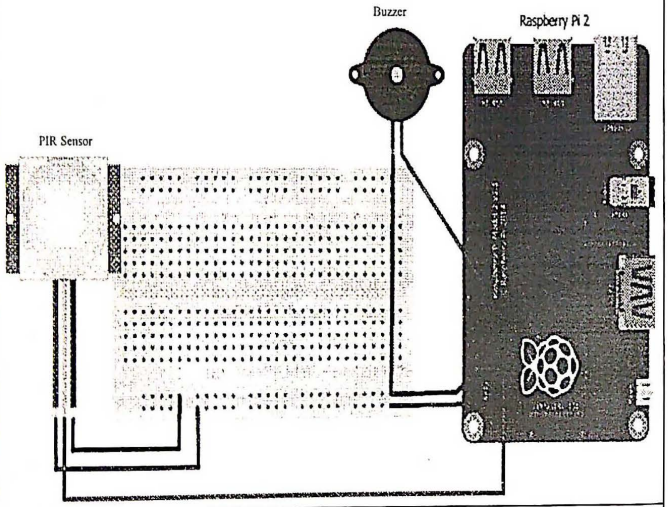


Key points	1. Create file "ldr.py" then "touch foo.txt" 2. To run the python code "sudo python ldr.py" << See what the light levels in the room are. 3. The check the file "more foo.txt" you can see your results.
Code	**File: ldr.py** #!/usr/bin/env python import os import datetime import time import 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 2 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?

FUTURE VISION BIE

By K B Hemanth Raj

Visit : https://hemanthrajhemu.github.io

Quick Links for Faster Access.

CSE 8th Semester - https://hemanthrajhemu.github.io/CSE8/

ISE 8th Semester - https://hemanthrajhemu.github.io/ISE8/

ECE 8th Semester - https://hemanthrajhemu.github.io/ECE8/

8th Semester CSE - TEXTBOOK - NOTES - QP - SCANNER & MORE

17CS81 IOT - https://hemanthrajhemu.github.io/CSE8/17SCHEME/17CS81/

17CS82 BDA - https://hemanthrajhemu.github.io/CSE8/17SCHEME/17CS82/

17CS832 UID - https://hemanthrajhemu.github.io/CSE8/17SCHEME/17CS832/

17CS834 SMS - https://hemanthrajhemu.github.io/CSE8/17SCHEME/17CS834/

8th Semester Computer Science & Engineering (CSE)

8th Semester CSE Text Books: https://hemanthrajhemu.github.io/CSE8/17SCHEME/Text-Book.html

8th Semester CSE Notes: https://hemanthrajhemu.github.io/CSE8/17SCHEME/Notes.html

8th Semester CSE Question Paper: https://hemanthrajhemu.github.io/CSE8/17SCHEME/Question-Paper.html

8th Semester CSE Scanner: https://hemanthrajhemu.github.io/CSE8/17SCHEME/Scanner.html

8th Semester CSE Question Bank: https://hemanthrajhemu.github.io/CSE8/17SCHEME/Question-Bank.html

8th Semester CSE Answer Script: https://hemanthrajhemu.github.io/CSE8/17SCHEME/Answer-Script.html

Contribution Link:

https://hemanthrajhemu.github.io/Contribution/

Stay Connected... get Updated... ask your queries...

Join Telegram to get Instant Updates:
https://telegram.me/joinchat/AAAAAFTtp8kuvCHALxuMaQ

Contact: MAIL: futurevisionbie@gmail.com

INSTAGRAM: www.instagram.com/futurevisionbie/