310258:: : Embedded Systems and Internet of Things Lab

Course Objectives: -

1. To understand functionalities of various single board embedded platforms fundamentals
2. To develop comprehensive approach towards building small low cost embedded IoT system.
3. To implement assignments based on sensory inputs.

Course Outcomes: -

1. Choose between available technologies and devices for stated IoT challenge.
2. Design the minimum system for sensor based application in IOT.
3. Develop full fledged IoT application for distributed environment.
4. Solve the problems related to primitive needs using IOT.

Assignment List

Sr No.	List of Practical's
	Group A
1.	Study of Raspberry-Pi, Beagle board, Arduino and other micro controller (History & Elevation)
2.	Study of different operating systems for Raspberry-Pi /Beagle board. Understanding the process of OS installation on Raspberry-Pi /Beagle board
3.	Study of Connectivity and configuration of Raspberry-Pi /Beagle board circuit with basic peripherals, LEDS. Understanding GPIO and its use in program.
4.	Understanding the connectivity of Raspberry-Pi /Beagle board circuit with temperature sensor. Write an application to read the environment temperature. If temperature crosses a threshold value, the application indicated user using LEDSs
	Group B
5.	Understanding the connectivity of Raspberry-Pi /Beagle board circuit with IR sensor. Write an application to detect obstacle and notify user using LEDs.
6.	Understanding and connectivity of Raspberry-Pi /Beagle board with camera. Write an application to capture and store the image.
7.	Understanding and connectivity of Raspberry-Pi /Beagle board with a Zigbee module. Write a network application for communication between two devices using Zigbee.
8.	Study of different CPU frequency governors. Write an application to change CPU frequency of Raspberry-Pi /Beagle board
	Group C

9.	Write an application using Raspberry-Pi /Beagle board to control the operation of stepper motor.
10.	Write an application using Raspberry-Pi /Beagle board to control the operation of a hardware simulated traffic signal.
11.	Write an application using Raspberry-Pi /Beagle board to control the operation of a hardware simulated lift elevator
	Group D
12.	Write a server application to be deployed on Raspberry-Pi /Beagle board. Write client applications to get services from the server application.
13.	Create a small dashboard application to be deployed on cloud. Different publisher devices can publish their information and interested application can subscribe.
14.	Create a simple web interface for Raspberry-pi/Beagle board to control the connected LEDs remotely through the interface.
	Group E
15.	Develop a Real time application like smart home with following requirements: When user enters into house the required appliances like fan, light should be switched ON. Appliances should also get controlled remotely by a suitable web interface. The objective of this application is student should construct complete Smart application in group.
16.	Develop a Real time application like a smart home with following requirements: If anyone comes at door the camera module automatically captures his image send it to the email account of user or send notification to the user. Door will open only after user‘s approval.

Experiment 1

Aim

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

Theory

Study of Raspberry Pi3

The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and in developing countries. The original model became far more popular than anticipated, selling outside of its target market for uses such as robotics. Over 5 million Raspberry Pis have been sold before February 2015, making it the best-selling British computer. By November 2016 they had sold 11 million units.

The first generation (Raspberry Pi1 Model B) was released in February 2012, followed by the simpler and cheaper Model A. In 2014, the Foundation released a board with an improved design, Raspberry Pi 1 Model B+. These boards are approximately credit-card sized and represent the standard mainline form-factor. Improved A+ and B+ models were released a year later.

A Raspberry Pi Zero with smaller size and reduced input/output (I/O) and general-purpose input/output (GPIO) capabilities was released in November 2015 for US$5. Raspberry Pi 3 Model B was released in February 2016 and has on-board WiFi, Bluetooth and USB boot capabilities. By2017, it became the newest mainline Raspberry Pi. On 28 February 2017, the Raspberry Pi Zero W was launched, which is like Raspberry Pi Zero with Wi-Fi and Bluetooth, for US$10.Processor speed ranges from 700 MHz to 1.2 GHz for the Pi 3; on-board memory ranges from 256MB to 1 GB RAM. Secure Digital (SD) cards are used to store the operating system and program memory in either SDHC or Microcosmic sizes. The boards have one to four USB ports. For video output, HDMI and composite video are supported, with a standard 3.5 mm phono jack for audio output. Lower-level output is provided by a number of GPO pins which support common protocols like I2C. The B-models have an 8P8C Ethernet port and the Pi 3 and Pi Zero W have on-board Wi-Fi 802.11n and Bluetooth. Prices range US$5 to $35.

Raspberry Pi 3 Model B

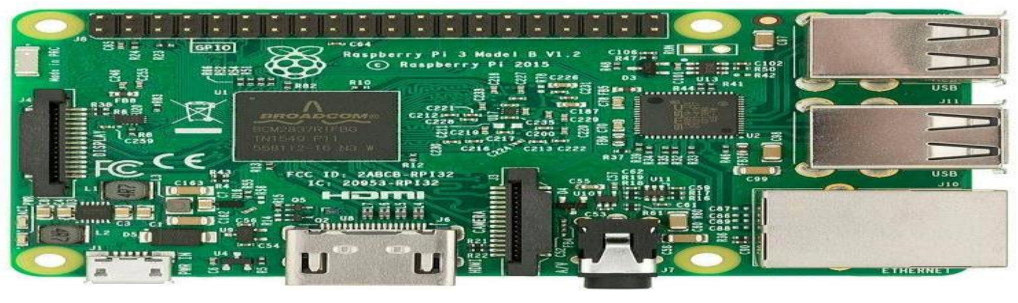


Fig.1. Raspberry Pi3 Kit

History and Elevation

In 2006, early concepts of the Raspberry Pi were based on the Atmel ATmega644 microcontroller. Its schematics and PCB layout are publicly available. Foundation trustee Eben Upton assembled a group of teachers, academics and computer enthusiasts to devise a computer to inspire children. The computer is inspired by Acorn's BBC Micro of 1981. The Model A, Model B and Model B+ names are references to the original models of the British educational BBC Micro computer, developed by Acorn Computers. The first ARM prototype version of the computer was mounted in a package the same size as a USB memory stick. It had a USB port on one end and an HDMI port on the other.

The Foundation's goal was to offer two versions, priced at US$25 and $35. They started accepting orders for the higher priced Model B on 29 February 2012, the lower cost Model A on 4 February2013. and the even lower cost (US$20) A+ on 10 November 2014.On 26 November 2015, the cheapest Raspberry Pi yet, the Raspberry Pi Zero, was launched at US$5 or £4.

Study of Beagle Board

The Beagle Board is a low-power open-source single-board computer produced by Texas Instruments in association with Digi-Key and Newark element14. The Beagle Board was also designed with open source software development in mind, and as a way of demonstrating the Texas Instrument's OMAP3530 system-on-a-chip. The board was developed by a small team of engineers as an educational board that could be used in colleges around the world to teach open source hardware and software capabilities. It is also sold to the public under the Creative Commons share-alike license. The board was designed using Cadence OrCAD for schematics and Cadence Allegro for PCB manufacturing; no simulation software was used.

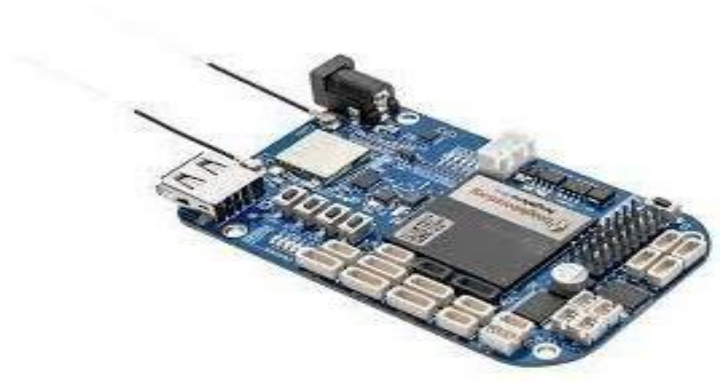


Fig.2.Beagle Bone Kit

Study of Arduino

The Arduino project started at the Interaction Design Institute Ivrea (IDII) in Ivrea, Italy. At that time, the students used a BASIC Stamp microcontroller at a cost of $100, a considerable expense for many students. In 2003 Hernando Barrage created the development platform Wiring as a Master's thesis project at IDII, under the supervision of Massimo Banzi and Casey Reas, who are known for work on the Processing language. The project goal was to create simple, low cost toolsfor creating digital projects by non-engineers. The Wiring platform consisted of a printed circuitboard (PCB) with an ATmega168 microcontroller, an IDE based on Processing and library functionsto easily program the microcontroller.In 2003, Massimo Banzi, with David Mellis, another IDIIstudent, and David Cuartielles, added support for the cheaper ATmega8 micro controller to Wiring. But instead of continuing the work on Wiring, they forked the project and renamed it Arduino.

The initial Arduino core team consisted of Massimo Banzi, David Cuartielles, Tom Igoe, GianlucaMartino, and David Mellis, but Barragán was not invited to participate.

Following the completion of the Wiring platform, lighter and less expensive versions were distributed in the open-source community.

Adafruit Industries, a New York City supplier of Arduino boards, parts, and assemblies, estimated inmid-2011 that over 300,000 official Arduinos had been commercially produced,[6] and in 2013 that700,000 official boards were in users' hands.

In October 2016, Federico Musto, Arduino's former CEO, secured a 50% ownership of thecompany. In April 2017, Wired reported that Musto had "fabricated his academic record On his company's website, personal Linked In accounts, and even on Italian business documents, Musto was until recently listed as holding a PhD from the Massachusetts Institute of Technology. In some cases, his bios also claimed an MBA from New York University." Wired reported that neither University had any record of Musto's attendance, and Musto later admitted in an interview with Wired that he had never earned those degrees.

Around that same time, Massimo Banzi announced that the Arduino foundation would be "a new beginning for Arduino." But a year later, the Foundation still hasn't been established, and the state of the project remains sun clear. The controversy surrounding Musto continued when, in July 2017, he reportedly pulled many Open source licenses, schematics, and code from the Arduino website, prompting scrutiny and outcry. In October 2017, Arduino announced its partnership with ARM Holdings (ARM). The announcement said, in part, "ARM recognized independence as a core value of Arduino without any lock-in with the ARM architecture.” Arduino intends to continue to work with all technology vendors and architectures.

Arduino Kit Fig.3.

Conclusion

Thus, we have studied history of Raspberry Pi, Beagle bone and Arduino.

Experiment 2

Title: Interfacing and Programming Arduino with PC

Aim/Objectives:

1. To study the layout of Arduino Board.
2. To Identify the microcontroller IC on Arduino (ATMEGA328-UNO-R3)
3. To study the installation and interface of Arduino IDE.
4. To learn basic command of Arduino IDE.
5. To write a simple program in Arduino IDE to interface the Arduino board with PC.

Software:

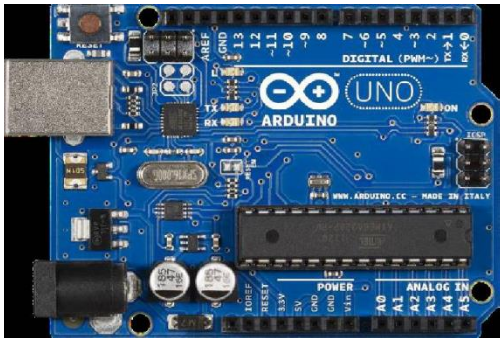
1 . Arduino IDE 1.6.9 or higher

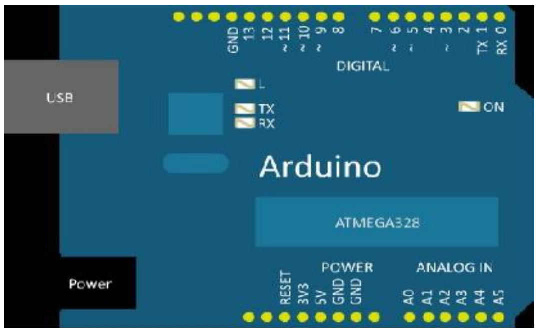
Hardware Modules:

1. Arduino Board
2. PC / Laptop

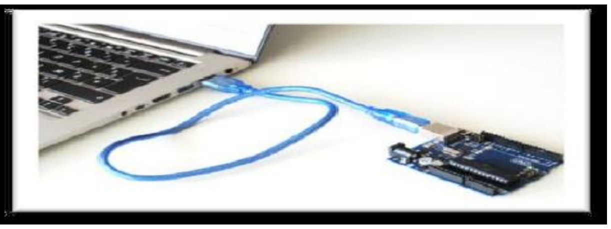
Theory:

Introduction to Arduino Board





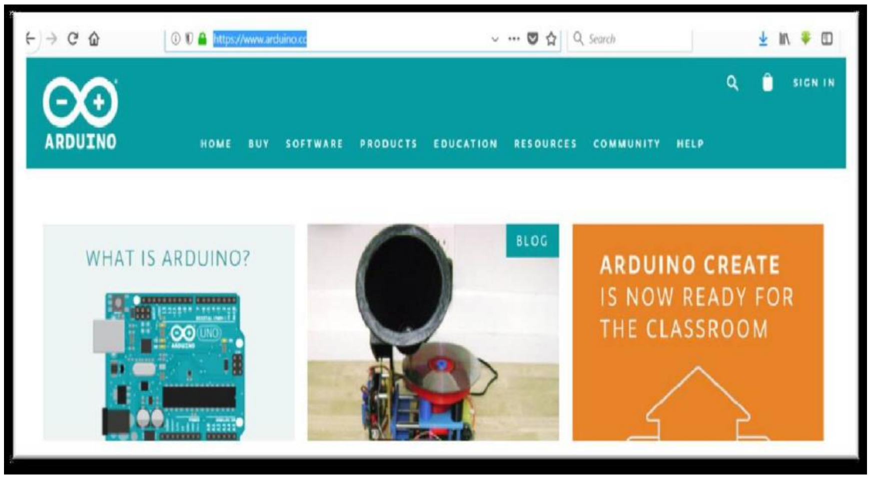
1. Arduino Board is an Open Source software and hardware.
2. The brain of Arduino Board is the microcontroller IC ATMEGA328 (3rd version) present on the bottom right corner of the board.
3. There are total 20 GPIO pins mounted on the board.
4. From these, on the top side of the board, 14 are digital pins denoted by 0, 1….13. These pins can be set as digital input or digital output pins (digital read or write).
5. The remaining 6 pins, at the bottom right side of the board are the Analog pins, denoted by A0, A1….A5. These pins are used to read the analog signal (analog read), these pins can also be used as digital input/output pins.
6. From the 14 digital pins, 6 pins are denoted by '~' sign, called as PWM pin. These pins are used to write the analog signal (analog write).
7. Arduino board can be connected to the PC using USB cable. The cable has two different connectors at its two ends. One of type-A(PC side)) and the other is type-B(Arduino side).
8. Power is given to Arduino Board by two ways, 1. Through USB cable or 2. By external adapter (when there is no need of PC/Laptop).



Introduction to Arduino IDE (Integrated Development Environment)

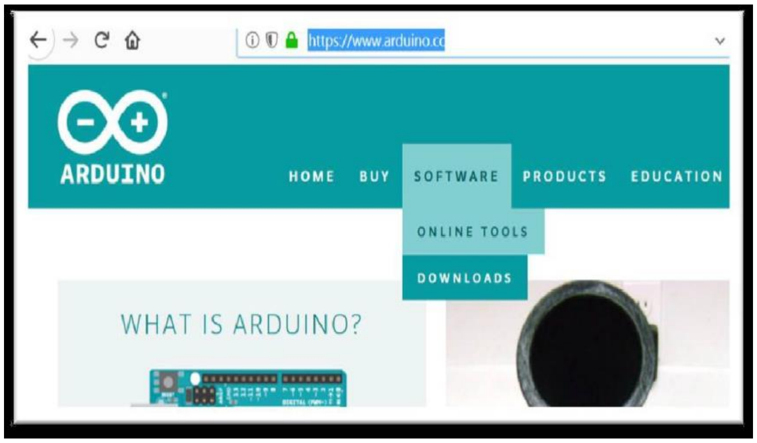
9. The latest version of Arduino IDE is Arduino-1.8.5.

10. It can be downloaded freely from the website **Arduino.cc**


https://www.arduino.cc
ARDUINO
SIGN IN
HOME BUY SOFTWARE PRODUCTS EDUCATION RESOURCES COMMUNITY HELP
WHAT IS ARDUINO?
BLOG
ARDUINO CREATE IS NOW READY FOR THE CLASSROOM

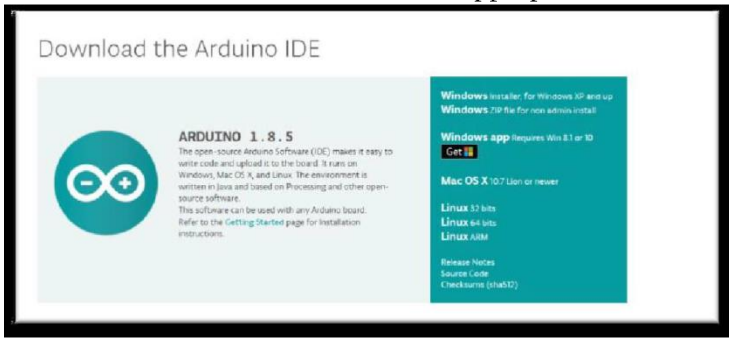
11. It is an exe file.

12. Click Software and then Downloads to download the software



13. On the next screen, scroll down to see the menu “Download the Arduino IDE”. Here, in the left side of the window, you can see the latest version of the Arduino IDE. On the right side of the window, different

options are given. To download the exe for windows operating system, click on the option, “Windows Installer, for Windows XP and UP”. For other OS, click on the appropriate OS.



Download the Arduino IDE

ARDUINO 1.8.5

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.
This software can be used with any Arduino board.
Refer to the Getting Started page for Installation instructions.

Windows Installer, for Windows XP and up
Windows ZIP file for non admin install

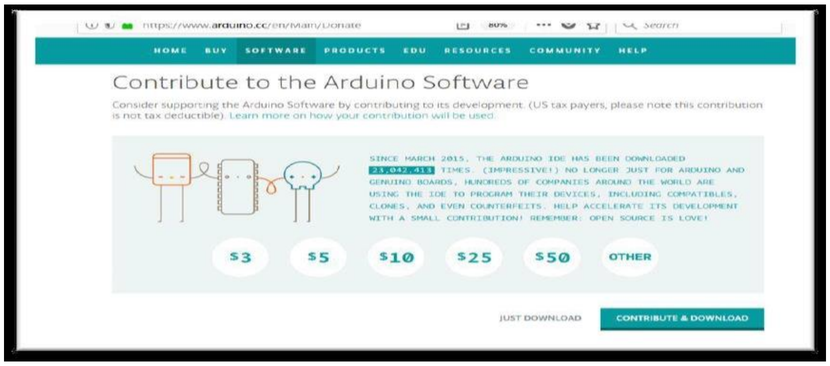
Windows app Requires Win 8.1 or 10
Get

Mac OS X 10.7 Lion or newer

Linux 32 bits
Linux 64 bits
Linux ARM

Release Notes
Source Code
Checksums (sha512)

14. On the next screen, click on the option “JUST DOWNLOAD” to download the exe.

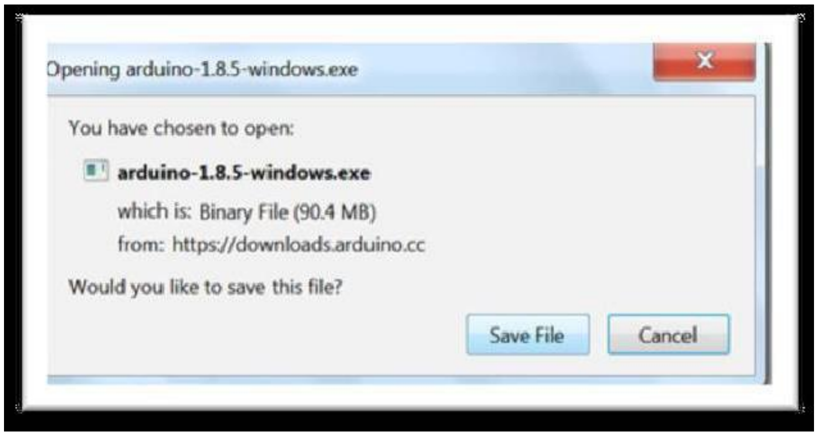

https://www.arduino.cc/en/Main/Donate
HOME BUY SOFTWARE PRODUCTS EDU RESOURCES COMMUNITY HELP

Contribute to the Arduino Software

Consider supporting the Arduino Software by contributing to its development. (US tax payers, please note this contribution is not tax deductible). Learn more on how your contribution will be used.

SINCE MARCH 2015, THE ARDUINO IDE HAS BEEN DOWNLOADED 23,042,413 TIMES. (IMPRESSIVE!) NO LONGER JUST FOR ARDUINO AND GENUINO BOARDS, HUNDREDS OF COMPANIES AROUND THE WORLD ARE USING THE IDE TO PROGRAM THEIR DEVICES, INCLUDING COMPATIBLES, CLONES, AND EVEN COUNTERFEITS. HELP ACCELERATE ITS DEVELOPMENT WITH A SMALL CONTRIBUTION! REMEMBER: OPEN SOURCE IS LOVE!

16. On the next screen, click on the “Save File” option.



Opening arduino-1.8.5-windows.exe

You have chosen to open:

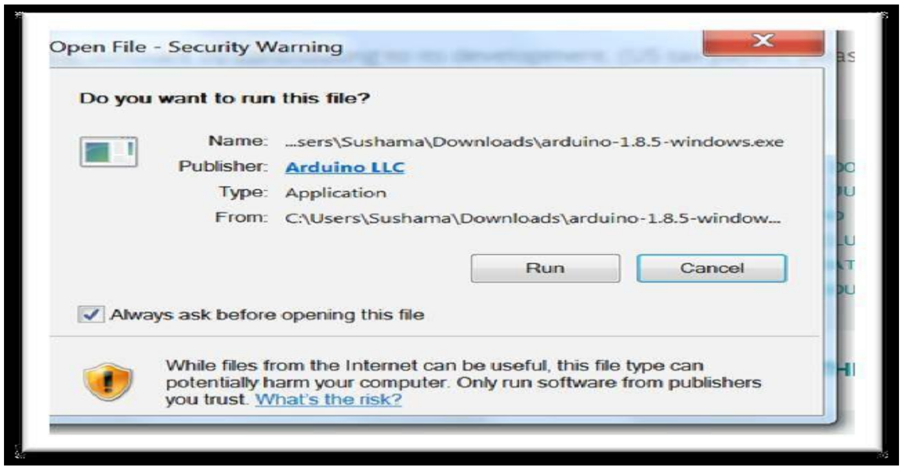
arduino-1.8.5-windows.exe

which is: Binary File (90.4 MB)

from: https://downloads.arduino.cc

Would you like to save this file?

17. After the downloading is completed, “Aduino-1.8.5-windows.exe” can be seen in the downloads folder.
18. Double click on this exe to install on the PC.
19. on the next screen, click on the “Run” option.


Open File - Security Warning

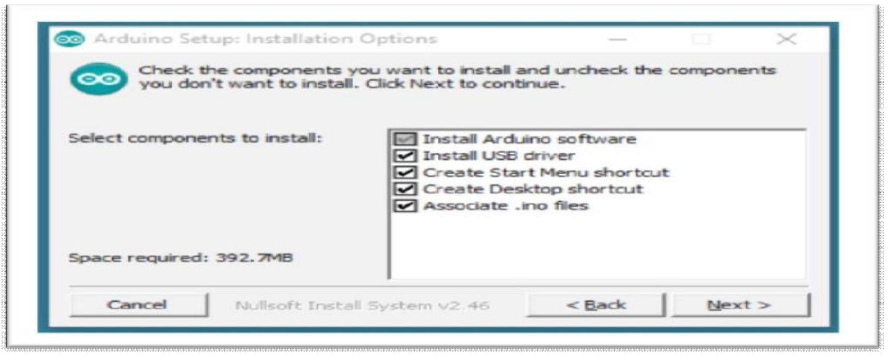
Do you want to run this file?

Name: ...sers\Sushama\Downloads\arduino-1.8.5-windows.exe
Publisher: Arduino LLC
Type: Application
From: C:\Users\Sushama\Downloads\arduino-1.8.5-window...

☑ Always ask before opening this file

While files from the Internet can be useful, this file type can potentially harm your computer. Only run software from publishers you trust. What's the risk?

20. On the next screen, tick all the options and click on next.


Arduino Setup: Installation Options

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

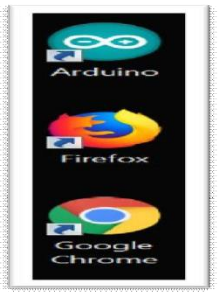
Select components to install:
- ☑ Install Arduino software
- ☑ Install USB driver
- ☑ Create Start Menu shortcut
- ☑ Create Desktop shortcut
- ☑ Associate .ino files

Space required: 392.7MB

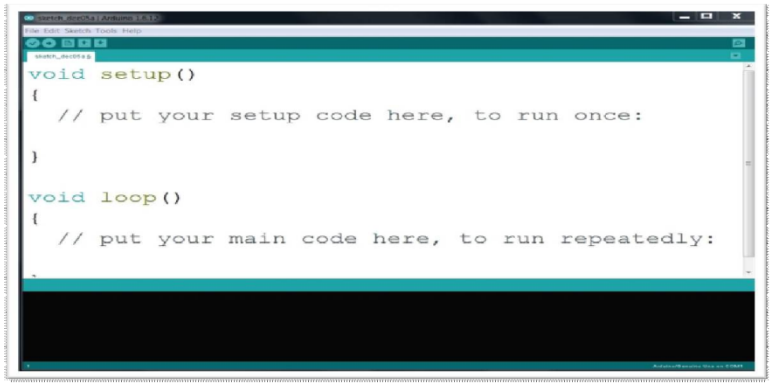
21. On the next screen, click on “Install” option.

22. After this, multiple times you will be asked to click on ‘install’ option, don’t ignore any option, otherwise installation will not be completed.

23. After completion of the installation, Arduino IDE icon will be seen on the desktop



Using Arduino IDE:



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

}

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

Arduino IDE has following options,

1. Text editor
2. Toolbar
3. Compiler
4. Serial monitor

Text Editor

1. The white color part in the IDE is called editor.
2. We can type and edit the program in this editor.
3. When we open this IDE, the two functions called void setup() and void loop() automatically appears in the editor.
4. These are the compulsory functions of Arduino IDE. If any of the function is deleted then the program will not compile.
5. In void setup() function, those instructions are written which are to be executed only once.
6. In void loop() function, those instructions are written which are to be executed repeatedly.

Toolbar

1. The toolbar is having 5 options
2. First is Compile. Using this we can compile the program.
3. Second option is Upload. This option is used to upload the program in Arduino board. But before uploading the program, ensure that
4. The program is saved and compiled
5. The Arduino board is connected to PC or Laptop using USB cable.
6. From ‘Tool’ menu, click on ‘Board’ and from boards, select the board which you are using in the pratical.
7. From ‘Tool’ menu, click on ‘Port’ and from ports, select the port to which the arduino board is connected.
8. Now click on the upload option.

9. After uploading is completed, “Done uploading” message is displayed.
10. The Third option is given to create the new file.
11. The Fourth option is given to open the saved the file.
12. And the Fifth option is given to save the file. After saving the file, the .ino extension is automatically given to the file. We need not give any extension.

Compiler

1. Arduino IDE has inbuilt compiler.
2. While compilation, the errors detected are shown in the bottom blank window.
3. When all errors are corrected, the message “Compilation done” appears.
4. To compile the program, it is not necessary to connect the arduino board to the PC or laptop.

Serial Monitor

1. In the Arduino IDE, on the right top corner, an icon for Serial monitor is shown. Clicking on this icon, the serial monitor window opens.
2. This monitor is used to display the output of the program.
3. Also from this monitor window, we give input to the program from keyboard.

Writing a program in Arduino IDE

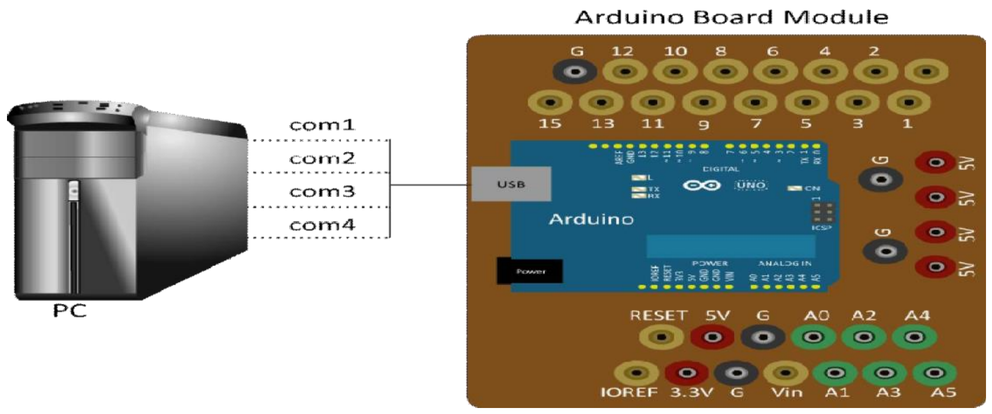
Let’s write a program to blink the inbuilt LED, that is the LED connected to pin number 13 on the Arduino board. Due to this program, the LED will be ON for 2 seconds and the LED will be OFF for 2 seconds.

1. In void setup() function, write the statement, pinMode(13, OUTPUT);
2. Here note that at the end of the statment semicolon is compulsory.
3. This statement is used to declare that the pin no 13 is used in Output mode.
4. In this function obeserv that M is written in capital letter and other alphabets are written in small letters. This notation is called camel notation. In Arduino IDE, most of the functions are written in camel notation. If they are not written in camel notation, program will not be compiled. And if written in proper notation, then its color gets changed.
5. In void loop() function, to send the digital output on pin no 13, write the statement, digitalWrite(13, HIGH). Using this statement, the high singal (5v) is sent on pin no.13.
6. To keep the LED ON for 2 seconds, write the function delay(2000). In the delay() function, the delay is given in miliseconds.
7. 1000 miliseconds= 1 second
8. Now to make the LED OFF, write the function digitalWrite(13, LOW). Using this statement, the low singal (0v) is sent on pin no.13.
9. Again to give the delay of 2 seconds, write the function delay(2000).

Safety precautions:

1. First, make all the connections as per steps given below
2. Then connect Arduino board to PC/Laptop

Schematic diagram:



Procedure:

1. Write the program as per the algorithm given below.
2. Save and compile the program.
3. Connect the Arduino board to PC/Laptop using USB cable.
4. Upload the compiled program and check the output.

Algorithm:

1. Start IDE
2. Configure the pin number ‘13’ as ‘OUTPUT’ pin
3. Make the ‘OUTPUT’ as ‘HIGH’
4. Give delay of one second
5. Make the ‘OUTPUT’ as ‘LOW’
6. Give delay of one second

Observation:

1. Observe the LED near the pin number ‘13’, it starts blinking as soon as the program is uploaded.

Applications:

Prepare a list of Applications of this practical

Student assessment:

Experiment 2

Aim

Study of different operating systems for Raspberry-Pi. Understanding the process of OS installation on Raspberry-Pi

Theory

Introduction

The Raspberry Pi is a wonderful but powerful little computer that fits the palm of your hand. Despite of its size it has enough power to run your operating system smoothly, home media center, a VPN and a lot more. The Raspberry Pi has a SD card slot for mass storage and will attempt to boot off that device from SD card when the board is powered on by 5v micro USB supply.

The Raspberry is a very capable minicomputer and moreover its very inexpensive, it is available at unbelievable price that you could not resist yourself to buy one, if you are technocrat. Latest Raspberry Pi ie. Pi 3 comes with case less computer with HDMI and analog composite video output. It comes with 4 USB port that makes it more user friendly and programmable to achieve specific goals. This Raspberry Pi has an integrated 802.11n wifi adaptor and Bluetooth 4.1.wifi and Bluetooth to make it more user friendly, you doesn’t need TP link anymore to use wifi on this kit. It runs 5v micro UBS supply. It also provide RJ 45 port to use Ethernet connection. The Raspberry Pi 3 B model excellent processing speed provided by a powerful new 1.2GHz 64-bit quad core ARMv8 CPU with four ARM cortex –A53 cores and 1 GB of RAM. It does not include a built -in hard disk or solid-state drive, but uses an SD card for booting and long-term storage. We are going to compare different operating systems available for Raspberry Pi. Many from the available lists of operating systems, each one of them are segregated based on their applications, features and specifications

Brief Discussion of Operating Systems

No matter how good and powerful the hardware of the Raspberry Pi is, without an operating system it is just a piece of silicon, fiberglass, and a few other semiconductor materials. There are several different operating systems for the Raspberry Pi, including RISC OS, Pidora, Arch Linux,and Raspbian.

1. Raspbian

Currently, Raspbian is the most popular Linux-based operating system for the Raspberry Pi. Raspbian is an open source operating system based on Debian, which has been modified specifically for the Raspberry Pi (thus the name Raspbian).Raspbian is the default free and open source operating system that often comes with the Raspberry Pi kit, Raspbian is a official operating system of Raspberry Pi foundation. Raspbian is a version of Debian which is specially designed and optimized for the Raspberry Pi hardware and the build consists of more than 35,000 Raspbian packages. Raspbian is still under active development phase with an emphasis on improving the capability, stability and performance. For a beginner it's a good place to start especially if you're starting with programming and are used to a windows based system as it bears some resemblance to Windows. Raspbian comes with Python programming language. This OS is real treat to the python programmer. Raspbian also includes a 'Pi store' so you can download free and purchasable applications such as Libre Office, Free Civ (a game). Raspbian is a operating system which proves to be very efficient for the basic operating requirements with pi. Raspbian is designed to be easy to use and is the recommended operating system for beginners to start off with their Raspberry Pi.

2. Pidora

After waiting for a long, Raspberry Pi users are finally getting an optimized version of Fedora, the Pidora, to replace the current Rasbian OS. The news caused excitement among the Raspberry Pi community, who are finally getting the opportunity to enjoy Fedora on their devices after the previous attempt to introduce Fedora Remix for Pi ended up as a failure. However, the Seneca Center for Development of Open Technology (CDOT), the authority group behind Pidora, is confident that the Raspberry Pi community would love the newly optimized OS, coupled with greater speed and most of the features of Fedora 18. The current Rasbian OS, which was a remix of the Open Source Debian OS chip based on ARMv6 would make way for Pidora, currently available for download on the CDOT website.

3. Arch Linux

Arch Linux is an excellent choice for many reasons. One of the greatest advantages of the Arch Linux distribution is its simplicity in approach and attitude. Arch gives you the ability to build your system from the ground up, including only the software you actually need. This minimizes the amount of SD card memory it takes to hold the operating system for your Raspberry Pi, leaving more space for everything else you'll be doing. On a cautionary note, Arch moves forward as technology evolves, and this can sometimes lead to documentation lagging behind. Arch has now finished it's transition to System D from the old initscripts

4. OSMC

OSMC (Open Source Media Center) is a free and open source media player based on Linux. Founded in 2014, OSMC lets you play back media from your local network, attached storage and the Internet. OSMC is the leading media center in terms of feature set and community and is based on the Kodi project. Although OSMC is derived from Linux, you don’t need to have any experience with Linux to use it up and running in the way you want. Everything is easily managed through the OSMC interface. This OS comes with over 30,000 packages from Debian repository.

5. RetroPIE

Retro Pie allows you to turn your Raspberry Pi into a retro-gaming machine. Its platform developed on the base of Raspbian, Emulation Station, Retro Pie enable you to play your favorite Arcade, home-console, and classic PC games with the minimum set-up. For technocrat users it also provides a large variety of configuration tools to customize the system as per user need and purpose. The Retro Pie SD image is built on top of Raspbian but Retro Pie can be installed on any Debian based Linux distribution. Retro Pie has the most supported and customizable operating systems out of any retro programming software for the Raspberry Pi. This OS is very useful emulation many games.

6. RISC OS

RISC OS is a British operating system originally designed by Acorn Computers Ltd in Cambridge, England, and was first released in 1987. It was specifically designed to run on the ARM chipset. It is fast, compact and efficient. RISC OS is not a version of Linux, nor is it in any way related to Windows and interestingly was developed by the original ARM team. RISC OS Pi comes with a small set of utilities and applications, It includes a browser called NetSurf, a simple text editor, a scientific calculator, and it also has two software/package managers, packman and a store. Although it’s not a modern operating system (when compared Linux, Windows and OSX) is does have number of unique features and aspects to its design.

It is available to download from RISC OS Open Website or RaspberryPi.org.

7. Firefox OS

Firefox OS (also known internally as Boot to Gecko/B2G) is an OS which is more associated with being a Linux kernel-based open-source operating system primarily designed for smart phones and tablet computers. It was primarily designed as a community based alternative system utilizing open standards and HTML5 applications, JavaScript and open web API’s. It mainly competes with Android, Windows Phone and Jolla Sailfish OS Recently Mozilla on a Raspberry Pi. This OS is based on Mozilla technology The device is affordable and flexible as it can run a number of operating systems and might therefore be a very suitable device to provide an entry level upgrade in network protection.

8. Kali Linux

Kali Linux is a Debian-based security auditing Linux distribution. It is specially designed for digital forensics and penetration testing. It is maintained and funded by Offensive Security Ltd. Kali Linux provides many pre-installed packages with numerous penetration-testing programs, like nmap (a port scanner), Wire shark (a packet analyzer), John the Ripper (a password cracker), Air crack-ng(suite for penetration-testing wireless LANs), Burp suite and OWASP ZAP (security scanners).Recently support for TFT touch screens was added. If you want to install Kali on the Raspberry Pi kit you can download it from their official download page, it is freely available there. Raspberry Pi has changed the way of programming and usability. But without operating system it is just a piece of semiconductor material. Operating system have made the Raspberry Pi more popular and user friendly. We have gone through 8 different operating system. Each operating has its own features.

Conclusion-

Thus, we have studied installation for various OS in Raspberry Pi.

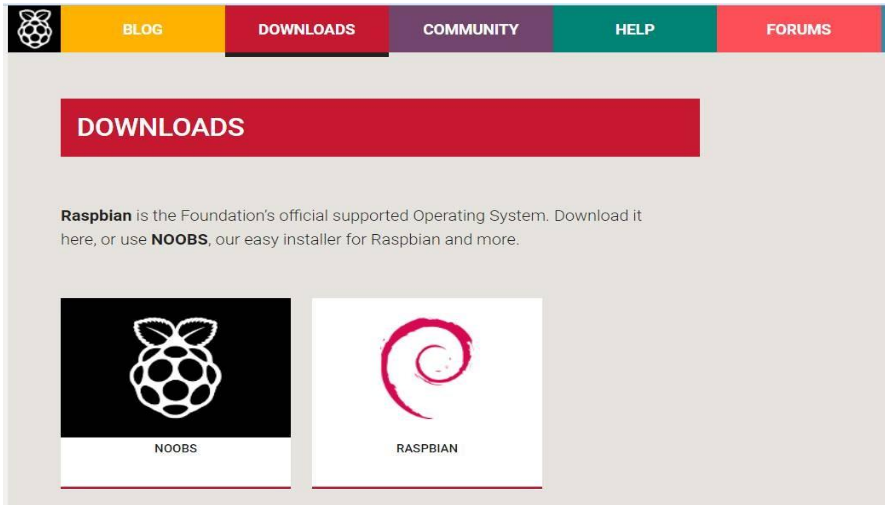
Installing OS for Raspberry-Pi-3

Aim/Objectives:

1. To understand the OS installation for Raspberry-Pi 3

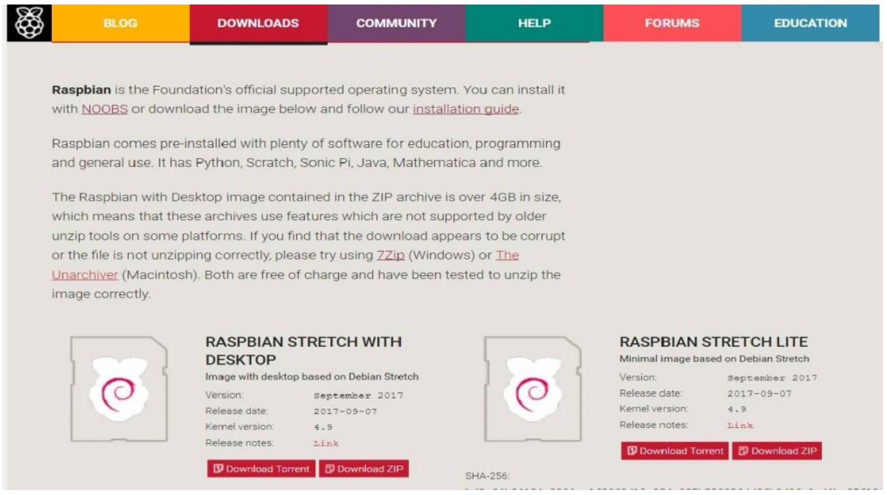
Process of OS installation on Raspberry Pi Board

1. Open the website: www.raspberrypi.org
2. Click on the "Downloads" tab
3.



Raspbian is the Foundation's official supported Operating System. Download it here, or use **NOOBS**, our easy installer for Raspbian and more.

4. Click on the "RASPBIAN" option.



Raspbian is the Foundation's official supported operating system. You can install it with NOOBS or download the image below and follow our installation guide.

Raspbian comes pre-installed with plenty of software for education, programming and general use. It has Python, Scratch, Sonic Pi, Java, Mathematica and more.

The Raspbian with Desktop image contained in the ZIP archive is over 4GB in size, which means that these archives use features which are not supported by older unzip tools on some platforms. If you find that the download appears to be corrupt or the file is not unzipping correctly, please try using 7Zip (Windows) or The Unarchiver (Macintosh). Both are free of charge and have been tested to unzip the image correctly.

RASPBIAN STRETCH WITH DESKTOP
Image with desktop based on Debian Stretch

Version:	September 2017
Release date:	2017-09-07
Kernel version:	4.9
Release notes:	Link

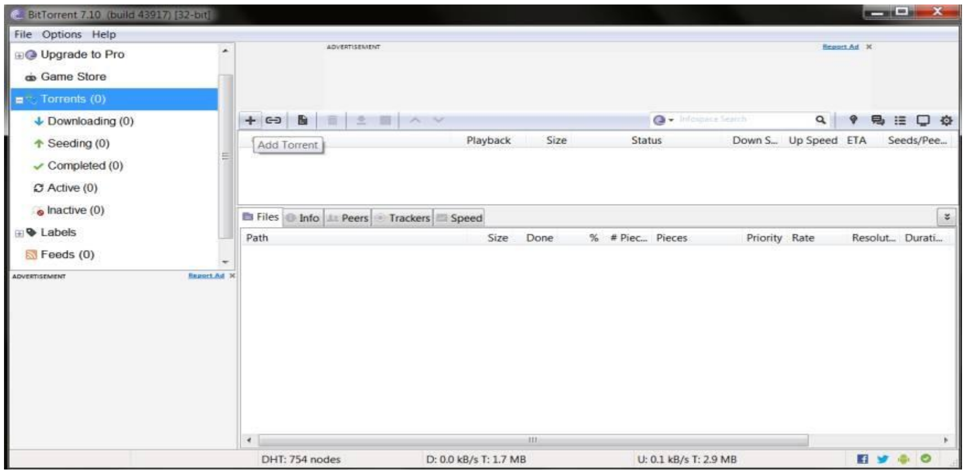
RASPBIAN STRETCH LITE
Minimal image based on Debian Stretch

Version:	September 2017
Release date:	2017-09-07
Kernel version:	4.9
Release notes:	Link

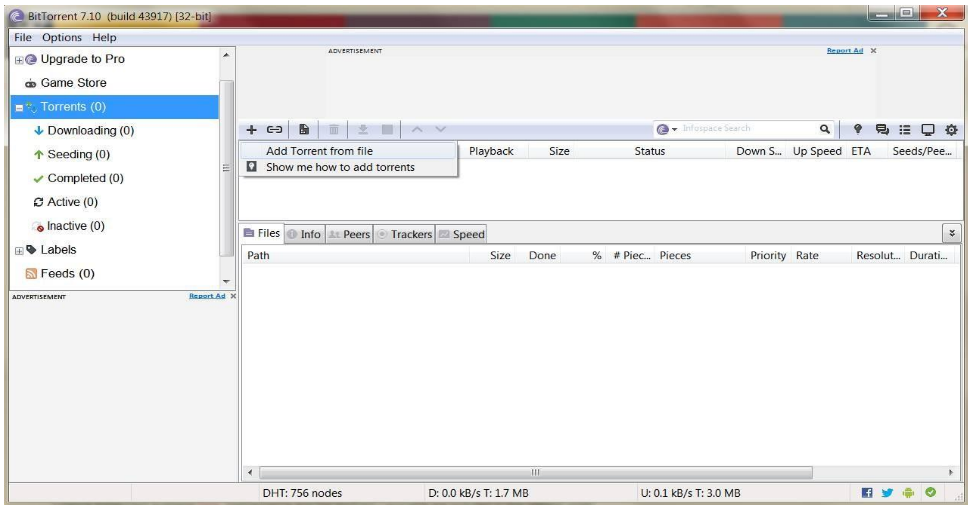
SHA-256:

5. We require "RASPBIAN STRETCH WITH DESKTOP", so under this heading, click on "Download Torrent" option.
6. A "Torrent file" is downloaded.
7. But the actual OS is present in the ZIP file of this torrent.
8. So using this "Torrent file" and the "Bit Torrent" software, we download the ZIP file of the Raspbian OS.
9. So download the "Bit Torrent" Software and install it.

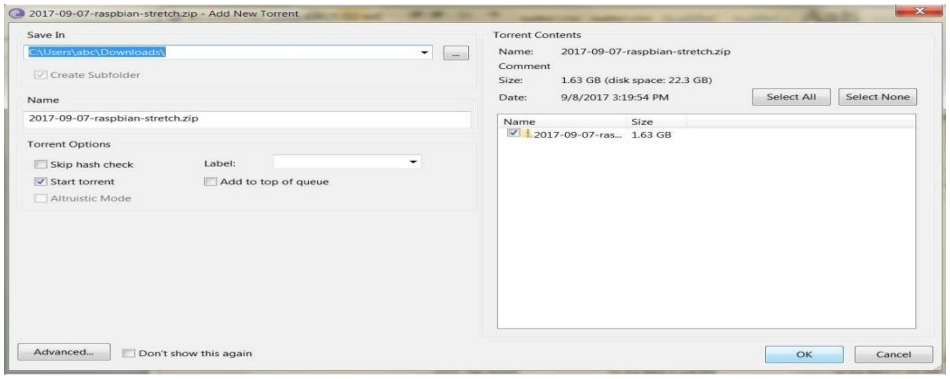
10. Now open the “Bit Torrent” software.
11. Click on the option “+” and under this click on “Add Torrent”.



12. Here select the path of downloaded “Torrent file”.


BitTorrent 7.10 (build 43917) [32-bit]
File Options Help
Upgrade to Pro
Game Store
Torrents (0)
Downloading (0)
Seeding (0)
Completed (0)
Active (0)
Inactive (0)
Labels
Feeds (0)
Add Torrent from file
Show me how to add torrents
Playback Size Status Down S... Up Speed ETA Seeds/Pee...
Files Info Peers Trackers Speed
Path Size Done % # Piec... Pieces Priority Rate Resolut... Durati...
DHT: 756 nodes
D: 0.0 kB/s T: 1.7 MB
U: 0.1 kB/s T: 3.0 MB

13. After selecting the torrent file, following window appears. Here click on OK


2017-09-07-raspbian-stretch.zip - Add New Torrent

Save In: C:\Users\abc\Downloads\

Create Subfolder

Name: 2017-09-07-raspbian-stretch.zip

Torrent Options

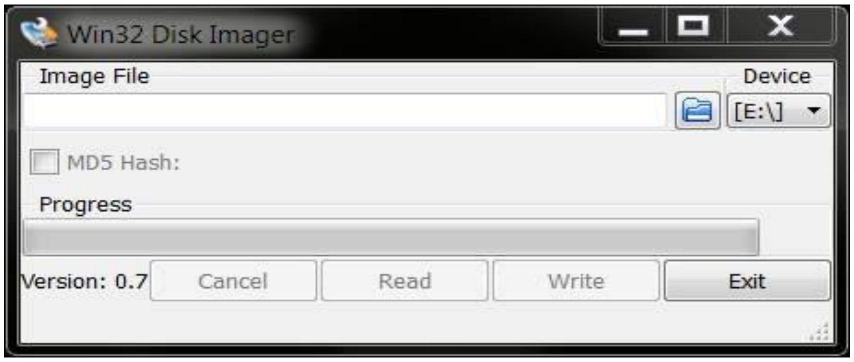
- Skip hash check
- [✓] Start torrent
- Altruistic Mode
- Label:
- Add to top of queue

Torrent Contents

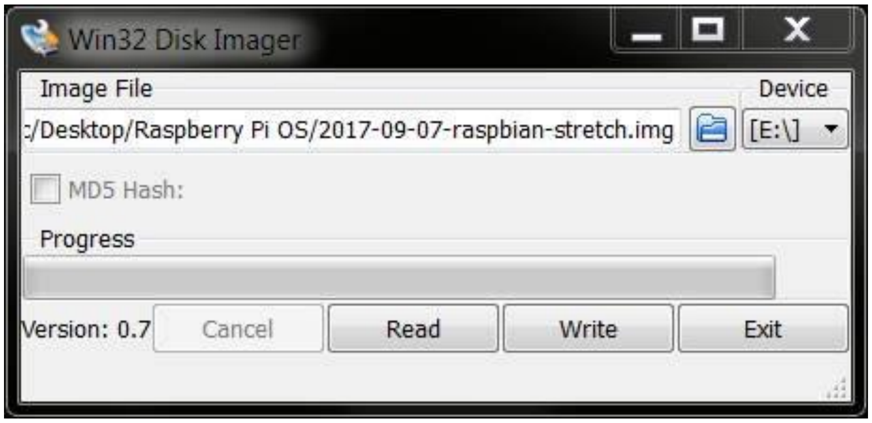
Name: 2017-09-07-raspbian-stretch.zip
Comment
Size: 1.63 GB (disk space: 22.3 GB)
Date: 9/8/2017 3:19:54 PM

Name	Size
[✓] 2017-09-07-ras...	1.63 GB

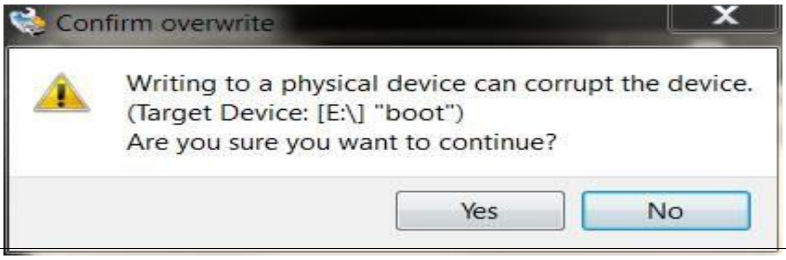
After completion of this process, we get the zip file named as “raspbian-strethc.zip”.
□ Now we have to unzip this file to get the actual disk image of the OS.
□ AS the ZIP archive of the OS is more than 4GB, we require special software named “7Zip” toUnzip the file. So download the software and install it.
14. Using this 7Zip software, unzip of the file. After this we get the required disk image of theRaspbian OS (approx. 4GB)
15. Now we have to write this disk image on SD card.
16. To write the OS on SD card, we require the software “win32 disk imager”. So download thissoftware and install it.
17. After completion of the installation, the following window appears.



18. Open the unzipped file in the “Image file” option byselecting the path from the Blue icon. The selected path is shown in the below image.
19. Now plug-in the SD card reader having SD card inside it, in the USB port of your PC.
20. Ensure that your SD card reader is having the same drive which is shown in the Device option (near the blue icon)



21. After ensuring that the “Image file path” and the “Device” are selected correctly, now click‘Write’ button to write the image on the SD card.
22. After this the following window appears.



Writing to a physical device can corrupt the device.
(Target Device: [E:\] "boot")
Are you sure you want to continue?

23. Here click ‘Yes’ and Confirm the overwrite
24. Image file will be written on SD card.
25. After the procedure is completed, it gives “Write Successful” message.
26. Congratulations! Your SD card is ready with your OS to work in the Raspberry-Pi-3 board.
27. Insert this SD card in Raspberry pi3.

28. Do the necessary connections and make the power ON. Your Raspberry-Pi starts and theDesktop of the OS is shown on the screen. Now Raspbery-Pi is ready to work on.

Experiment 3

Aim

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

Theory

Connectivity and configuration of Raspberry-Pi Guides to configure Raspberry Pi

1. **raspi-config**

   The Raspberry Pi configuration tool in Raspbian, allowing you to easily enable features such as the camera, and to change your specific settings such as keyboard layout.

2. **config.txt**

   The Raspberry Pi configuration file.

3. **Wireless**

   Configuring your Pi to connect to a wireless network using the Raspberry Pi 3 and Pi Zero W's in built wireless connectivity, or a USB wireless dongle.

4. **Wireless Access Point**

   Configuring your Pi as a wireless access point using the Raspberry Pi 3 and Pi Zero W's In built wireless connectivity, or a USB wireless dongle.

5. **Audio Config**

   Switch your audio output between HDMI and the 3.5mm jack.

6. **Camera Config**

   Installing and setting up the Raspberry Pi camera board.

7. **Externel Storage Config**

   Mounting and setting up external storage on a Raspberry Pi.

8. **Localisation**

   Setting up your Pi to work in your local language/time zone.

9. **Default pin configuration**

Changing the default pin states.

10. Device Trees Config

Device Trees, overlays, and parameters.

11. Kernel Command Line

The Linux kernel accepts a command line of parameters during boot. On the Raspberry Pi, this command line is defined in a file in the boot partition, called cmdline.txt. This is a simple text file that can be edited using any text editor, e.g. Nano.

sudo nano /boot/cmdline.txt

12. UART Configuration

The SoCs used on the Raspberry Pis have two built-in UARTs, a PL011 and a mini UART. They are implemented using different hardware blocks, so they have slightly different characteristics. However, both are 3.3V devices, which means extra care must be taken when connecting up to an RS232 or other system that utilizes different voltage levels. An adapter must be used to convert the voltage levels between the two protocols. Alternatively, 3.3V USB UART adapters can be purchased for very lowprices.

13. Screensaver

If you are using the Raspberry Pi solely on the console (no desktop GUI), you need to set the console blanking. The current setting, in seconds, can be displayed using

cat /sys/module/kernel/parameters/ console blank

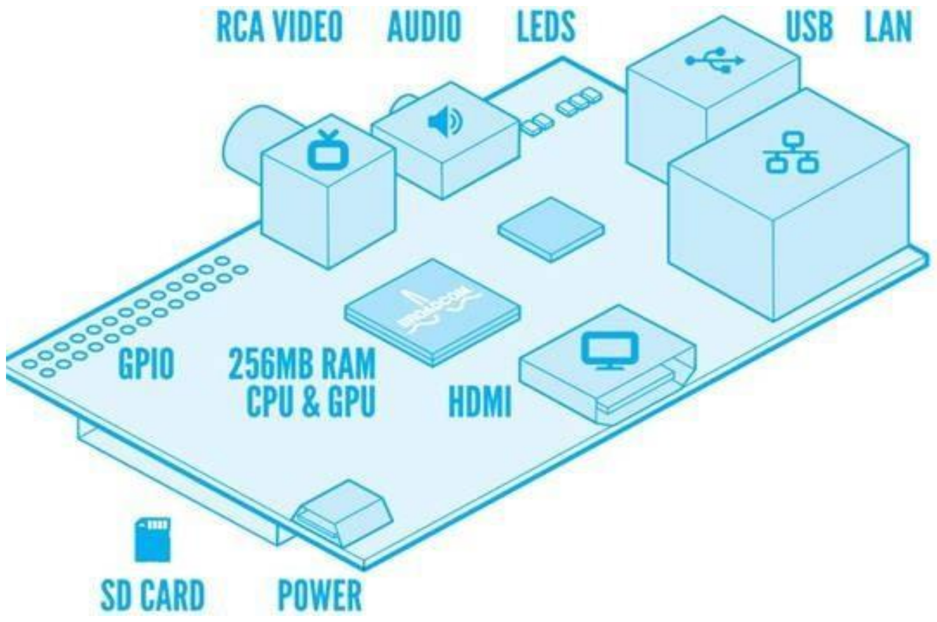
Here, console blank is a kernel parameter. In order to be permanently set, it needs to be defined on the kernel command line.

sudo nano /boot/cmdline.txt

Add console blank=0 to turn screen blanking off completely, or edit it to set the number of seconds of inactivity before the console will blank. Note the kernel command line must be a single line of text.

Connectivity of Raspberry Pi

Connectivity is truly superb for such a tiny device, especially on the B version of the Raspberry Pi. There are two USB 2.0 ports that can be used to hook up peripherals or adapters, and this can be further expanded with a powered hub. It's worth noting that both ports already share the bandwidth of a single channel to the system bus



For video, there's a full-size HDMI port, making the Raspberry Pi compatible with practically every monitor, TV and other display out there. For older displays that don't support digital connectivity, the Raspberry Pi even has an analogue composite/RCA video output, which can be used with SCART via an adapter. Stereo audio can be output over a 3.5mm jack, or you can get the full 5.1 surround sound package through the aforementioned HDMI. There are headers for further expansion, including the ability to hook up a camera or screen. Keep in mind that the micro USB port is for power rather than data. All of these ports are found at the top of the board, while the SD card reader is located at the bottom.

GPO Mode

The GPIO.BOARD option specifies that you are referring to the pins by the number of the pin the plug - i.e the numbers printed on the board (e.g. P1) and in the middle of the diagrams below.

The GPIO.BCM option means that you are referring to the pins by the "Broadcom SOC channel" number, these are the numbers after "GPIO" in the green rectangles around the outside of the belowdiagrams:

Unfortunately the BCM numbers changed between versions of the Pi1 Model B.

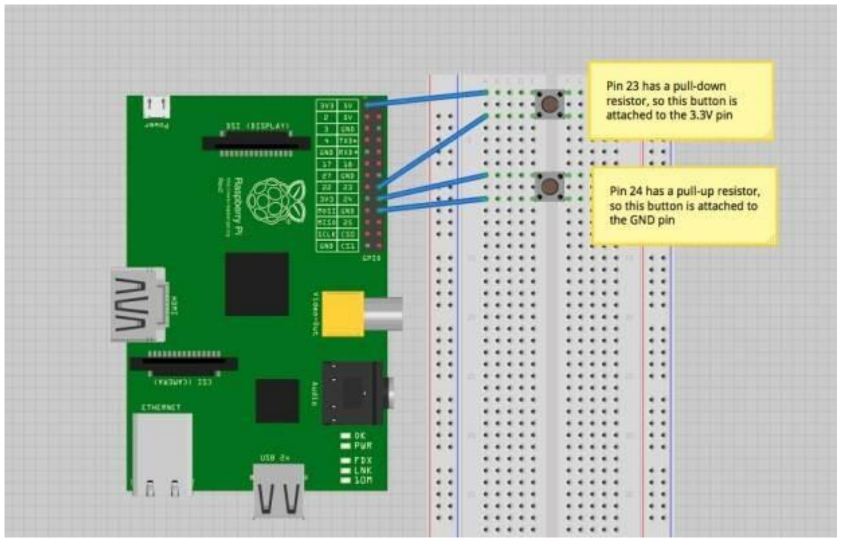
– The Model B+ uses the same numbering as the Model B r2.0, and adds new pins (board numbers

27-40).

– The Raspberry Pi Zero, Pi 2B and Pi 3B use the same numbering as the B+.

Building a Circuit

In the circuit shown below, two momentary switches are wired to GPIO pins 23 and 24 (pins 16 and 18 on the board). The switch on pin 23 is tied to 3.3V, while the switch on pin 24 is tied to ground. The reason for this is that the Raspberry Pi as internal pull-up and pull-down resistors that can be specified when the pin declarations are made.


Pin 23 has a pull-down resistor, so this button is attached to the 3.3V pin
Pin 24 has a pull-up resistor, so this button is attached to the GND pin
Raspberry Pi
GPIO

To set up these pins, write:

GPIO.setup (23, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

GPIO.setup (24, GPIO.IN, pull_up_down=GPIO.PUD_UP)

This will enable a pull-down resistor on pin 23, and a pull-up resistor on pin 24. Now, let’s check to see if we can read them. The Pi is looking for a high voltage on Pin 23 and a low voltage on Pin 24.We’ll also need to put these inside of a loop, so that it is constantly checking the pin voltage.

The code so far looks like this:

```
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)

GPIO.setup(23, GPIO.IN, pull_up_down = GPIO.PUD_DOWN)

GPIO.setup(24, GPIO.IN, pull_up_down = GPIO.PUD_UP)
```

```
while    True:
if(GPIO.input(23) ==1):
print("Button 1 pressed")
if(GPIO.input(24) ==    0):
print("Button 2 pressed")
GPIO.cleanup()
```

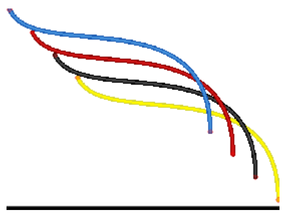
The indents in Python are important when using loops, so be sure to include them. You also must run your script as "sudo" to access the GPIO pins. The GPIO. cleanup() command at the end is necessary to reset the status of any GPIO pins when you exit the program. If you don't use this, then the GPIO pins will remain at whatever state they were last set to.

Resister

You must ALWAYS use resistors to connect LEDs up to the GPIO pins of the Raspberry Pi The Raspberry Pi can only supply a small current (about 60mA). The LEDs will want to draw more, and if allowed to they will burn out the Raspberry Pi. Therefore putting the resistors in the circuit will ensure that only this small current will flow and the Pi will not be damaged.

Resistors are a way of limiting the amount of electricity going through a circuit; specifically, they limit the amount of 'current' that is allowed to flow. The measure of resistance is called the Ohm(Ω), and the larger the resistance, the more it limits the current. The value of a resistor is marked with colored bands along the length of the resistor body.

Jumper Wires



Jumper wires are used on breadboards to 'jump' from one connection to another.

• The ones you will be using in this circuit have different connectors on each end.

• The end with the ‘pin’ will go into the Breadboard.

• The end with the piece of plastic with a hole in it will go onto the Raspberry Pi’s GPIO pins.

Conclusion

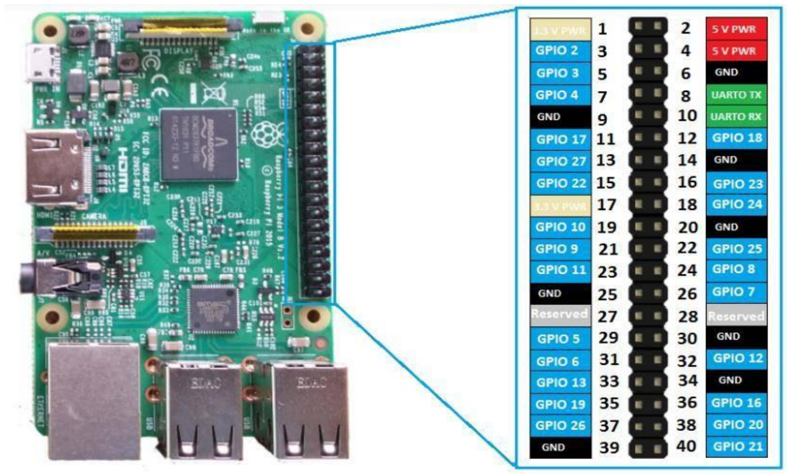
Thus, we have studied connectivity and configuration of Raspberry Pi and also use of GPIO.

Understanding GPIO pins on Raspberry Pi board and its use in program

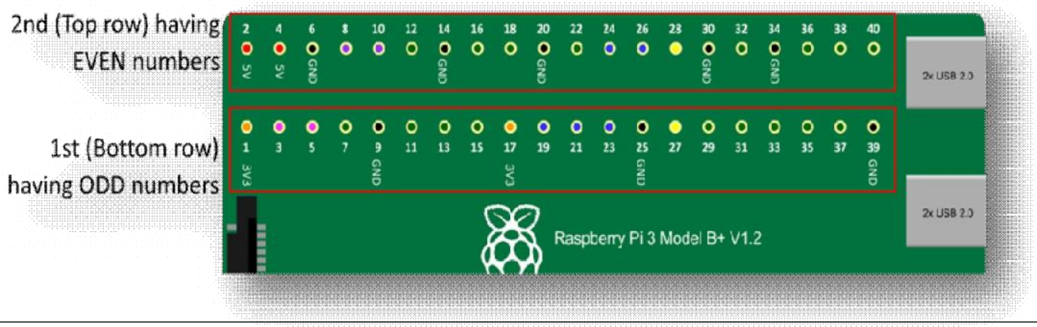
Aim/Objectives

1. To understand the GPIO pins of Raspberry-Pi 3
2 To program the GPIO pins of Raspberry-Pi 3 using Python

Introduction:

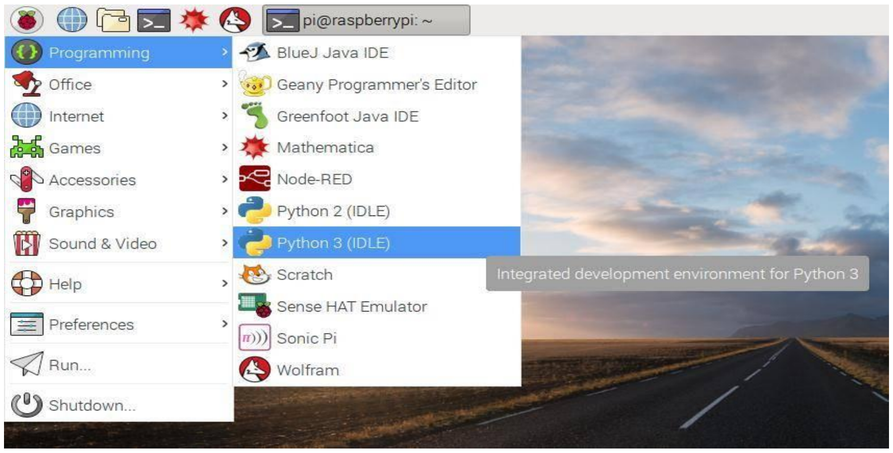


1.Raspberry Pi 3 Model B is the latest version of raspberry pi board.
2.It is released on 29 February.
3.The above figure shows the Raspberry Pi 3 Model B and It’s GPIO pins
4.General-purpose input/output (GPIO) is a generic pin on an integrated circuit orcomputer board whose behavior—including whether it is an input or outputpin—is controllable by the user at run time.



5. There are 40 pins available on board of Raspberry pi 3 model B.
6. The Pins are arranged in a 2×20 fashion as shown in the figure aboveOut of these, 26 pins are GPIO pinsAs you can observe, the numbers to the pins are given in zigzag manner.
7. The first (bottom) row starts with number ‘1’. So the pins in this row have odd numbersi.e. from 1 to 39.
8. The 2 nd (Top) row starts with number ‘2’. So the pins in this row have even numbers i.e.
9. from 2 to 40.
10.26 pins are GPIO pins,
8 pins are Ground (GND) pins,
2 pins are 5V power supply pins
2 pins are 3.3V power supply pins
2 pins are not used
11. Now if you’re coming to the Raspberry Pi as an Arduino user, you’re probably used toreferencing pins with a single, unique number.
12. In Raspberry Pi there are two different numbering schemes for referencing Pi pin

numbers:
1. Broadcom chip-specific pin numbers (BCM)
2. Physical pin numbers (BOARD)
You’re free to use either number-system.
The programs require that you declare which scheme you’re using at the very beginning of your program.
In a program, at a time, you can use only one number scheme.
Broadcom chip-specific pin numbers (BCM)
1. BCM - Broadcom pin number, commonly called "GPIO", these are the ones you probablywant to use with RPi.GPIO
2. The parameter used for this system is (GPIO.BCM).
3. This is a lower level way of working - it refers to the channel numbers on the BroadcomSOC.
4. To use this system, you have to always work with a diagram describing which channelnumber goes to which pin on the RPi board.
5. Your script could break between revisions of Raspberry Pi boards.
6. In this system26 GPIO pins are named as GPIO 01 to GPIO 26
Physical Numbering System (BOARD)
7. This system uses physical - Numbers corresponding to the pin's physical location on theheader
8. The numbers printed on the board are physical numbering system.
9. The parameter used for this system is (GPIO.BOARD).
10. The advantage of using this numbering system is that your hardware will always work,regardless of the board revision of the RPi.
11. You will not need to rewire your connector or change your code.
12. In this system
26 GPIO pins are named between 0 to 40
To use the Python IDLE IDE for programming in Raspberry-Pi use the following
Open Python 3 from the main menu:



- Or open terminal window and type the command sudo idle 3.5 and press enter
- Install all libraries required for Buzzer as given above.

1.Write the program as per algorithm given below
2.Write the program as per algorithm given below
3.Save with Ctrl + S and run with F5.
4.See output on Python Shell or Terminal Window.
Raspberry Pi GPIO programming using Python

1. The Raspberry Pi is often used in conjunction with other hardware to create interesting electronic projects.
2. The Pi3 comes with 40 GPIO pins that you can use to interface with various hardwaredevices—for both receiving data from them or for writing data to them.
3. To do this, we have to program the GPIO pins. To do this, special libraries in Pythonare used. To include these libraries in the program, the command used is ‘import’
4. This way, we can write applications to both read and also to control devices, i.e., turn them on and off, etc.
5. The default operating system used in Raspberry-Pi is Raspbian.
6. The Python package used for Raspberry Pi GPIO programming is RPi.GPIO. It is alreadyinstalled in Raspbian.
7. If you are using any other operating system, the package can be installed by using the

following command:
$ sudo pip install RPi.GPIO
There are important 8 steps in the programming of Raspberry-Pi using Python as follows
1. Import the RPi.GPIO library using the following commandimport RPi.GPIO as GPIO
2. Import the Time library using the following command
import time
3. Set numbering scheme to be used. The method used for this is GPIO.setmode(). We will use physical number scheme. So the method is written as
GPIO.setmode(GPIO.BOAD)
4. Set the pin mode as INPUT or OUTPUT using the commands
GPIO.setup(channel, GPIO.IN)

GPIO.setup(channel, GPIO.OUT)
5. Read input using following command
GPIO.input(pin no)
6. Write output using following comman
GPIO.output(pin no, state)
7. Give delay using command using following command
time.sleep(1) # delay for 1 second
8. Clean up GPIO and exit using following commands
GPIO.cleanup()
print("Exiting...")
9. You must clean up the pin set-ups before your program exits otherwise those pin settings will persist, and that might cause trouble when you use the same pins in another program.
10. The Pi 'expresses its displeasure' with a warning.
11. To clean up the entire set of pins, invoke GPIO.cleanup().
12. If you want only a few pins to be cleaned up, then the pin numbers should be provided as GPIO.cleanup (channel_list).
13. Anyway, you can suppress the warning messages by calling GPIO.setwarnings (False).
14. Save the program with proper name. The file is saved with extension '.py'.
15. The IDE named 'IDLE' used for programming is an interpreter and not a compiler. So to run the python program, we need to give the super user permission as follows.

Experiment 4

Aim

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

Theory

Infrared Sensor IR (Infrared) Sensor works by emitting infrared signal/radiation and receiving of the signal when the signal bounces back from any obstacle In other words, the IR Sensor works by continuously sending signal (in a direction) and continuously receive signal, comeback by bouncing on any obstacle in the way.

[tx_animate animation=”fadeIn” duration=”5″ delay=”.4″ inline=”no”]

[/tx_animate]

IR Sensor Fig.1

Components: IR Sensor

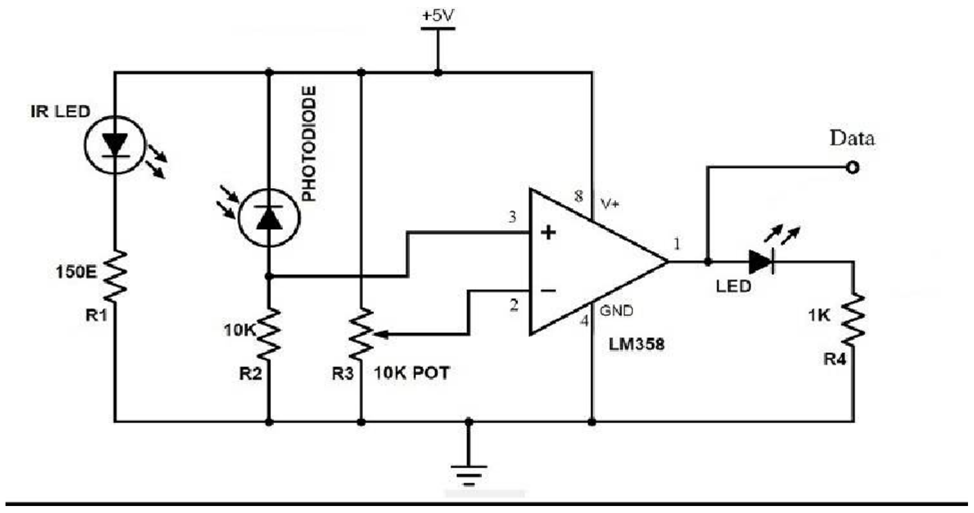
1. **Emitter**: This component continuously emits the infrared signal
2. **Receiver**: It waits for the signal which is bounced back by obstacle

3. **Indicator:** On board LED to signal if obstacle is deducted by the sensor

4. **Output**: Could be used as Input for further processing of the signal

5. **Ground**: Ground/Negative point of the circuit

6. **Voltage:** Input 3.3V



Circuit diagram of IR Sensor Fig.2

Objective

We will be creating a circuit using following components to detect obstacle.

1. Raspberry Pi 3
2. IR (Infrared) Sensor
3. 1 LED
4. 1 Resistor (330 Ω)
5. Few jumper cables
6. 1 Breadboard

Circuit: To detect obstacles

We will be creating a circuit which will turn on the LED when an obstacle is detected. And, as soon as the obstacle is removed from the way the LED will turn off. In order to achieve that follow the steps to create required circuit.

Part 1: Connecting IR Sensor

IR Sensor has 3 pins, viz VCC, GND and OUT. We will use GPIO 17 (do not get confused with pin number 17) for receiving input from the sensor.

1. Connect GPIO 17 from the Raspberry Pi to Breadboard (5a)
2. Connect OUT pin of the sensor with the Breadboard (5c)

This will send input received from sensor to GPIO 17, which will be processed further.

3. Connect GND (any pin from board will work, in this post we are using pin number 9) with negative line on left side of the breadboard
4. Connect GND of the IR Sensor to Breadboard (10c)
5. Connect GND from Step 3 to Breadboard (10a)
6. Connect VCC of the IR Sensor to Breadboard (15c)
7. Connect 3v3 (Pin #1) to positive line on left side of the breadboard
8. Connect 3v3 (connected in Step 7) to the Breadboard (15a)

Now the circuit is complete and sensor will detect the obstacle. It can be tested by putting anything in front of the IR Sensor. On-board LED will be on if obstacle is put in front of the sensor, else it will be off.

Part 2: Connecting LED

Objective is to turn on the LED when obstacle is detected.

1. Connect GPIO 4 from the board to the Breadboard(20a)
2. Connect positive point of the LED (longer pin of the LED) to the Breadboard (20c)
3. Connect negative point of the LED (smaller pin of the LED) to the Breadboard (22c)
4. Use resistor (330 Ω) to connect negative (row from Part 1: Step 3) to the negative point of the LED(22a)

Now we are ready to send signal based on the input received from IR Sensor to turn on/off the LED.

Part 3: Code to Connect IR Sensor I/P with LED status

```
from GPO zero import LED
from signal import pause
import R pi.GPO as GPO
import time
```

```
GPO.set mode(GPIO.BCM)
LED_PIN = 27
IR_PIN = 17

indicator = LED(LED_PIN)
GPIO.setup(IR_PIN, GPIO.IN)

count = 1

while True:
    got_something = GPIO.input(IR_PIN)
    if got_something:
        indicator.on()
        print("{:>3} Got something".format(count))
    else:
        indicator.off()
        print("{:>3} Nothing detected".format(count))
        count += 1
    time.sleep(0.2)
```

Part 4: Executing the code

1) Open terminal (On Pi itself or login through SSH login)
2) Navigate to the directory where the above code is saved
3) Type $ python3 ir_obstacle.py and press <enter>

On terminal it will start printing the status based on the conditions.

Conclusion:

Thus, we done connectivity of Raspberry-Pi /Beagle board circuit with IR sensor. Write an application to detect obstacle and notify user using LED's.

Understanding the connectivity of Raspberry Pi board circuit with IR sensor (Proximity)

Aim/Objectives

1. To understand the concept of Proximity sensor
2. To interface Proximity sensor with Raspberry Pi model
3 To program the Raspberry Pi model to detect the nearest object using proximity sensor and give indication through led.

Software:

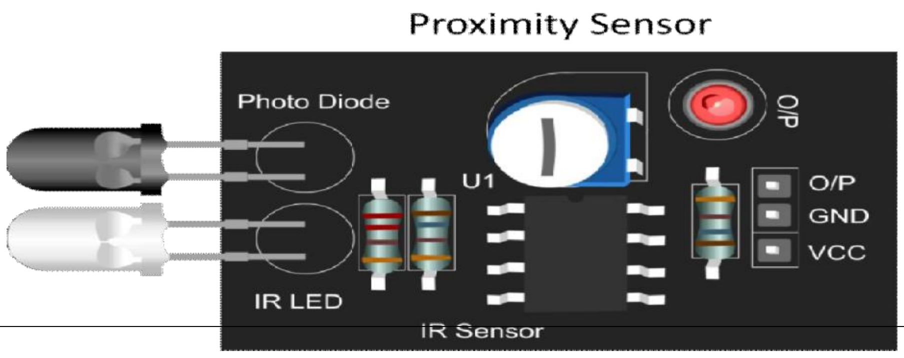
1Raspbian OS (IDLE)

Hardware Modules:

1. Raspberry Pi Board
2. Proximity sensor, Led, 330 ohm register
3 Monitor

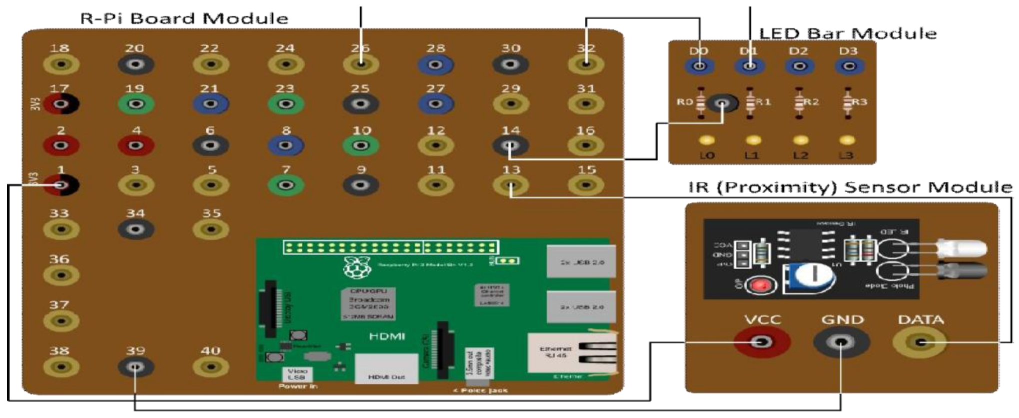
Theory:

1 Proximity IR sensor is a small board containing an IR transmitter, photodiode, IR Receiver and some processing circuitry.
2. This is a discrete sensor that senses when an object comes near to the sensor face
3. It works by detecting reflected light coming from its own infrared lights
4. By measuring the amount of reflected infrared light & it can glow Onboard led whenobject is directly front of it.
5. In Proximity, it consists of two leds, one is the transmitter (IR LED) and another isreceiver (photodiode).
6. The IR led transmits the infrared light signal which reaches till the object and deflectsback.
7.The Photo diode receives the deflected light.
8. This signal is then amplified & status of this signal is checked by the microcontroller.
9. Proximity sensor is more sensitive but it detects only object but cannot measure adistance value.
10. By using a potentiometer, we can change sensitivity accordingly.
11. When this sensor detects the object, it gives output as a digital value i.e. ‘1’ and if notdetected then the value is ‘0’



Safety precautions:
1. First, make all the connections as per steps given below
2. power supply

Interface diagram:



Steps for assembling circuit:
1. Connect the VCC pin of Proximity sensor to 3.3 V (pin) of Raspberry Pimodule
2. Connect the GND pin of Proximity sensor to GND pin of Raspberry Pimodule
3. Connect the DATA pin of Proximity sensor to pin ‘13’ of Raspberry Pi module□ Connect the D0 pin of LED bar to pin ‘28’ of Raspberry Pi module
4. Connect the GND pin of LED bar to GND pin of Raspberry Pimodule
Procedure:
5. Write the program as per the algorithm given below.
6. Save program.
7. Run code using Run module.
Algorithm:
8.Import GPIO and Time library
9.Set mode i.e. GPIO.BOARD
10.Set GPIO pin ‘13’ as Input
11.Set GPIO pin ‘28’ as Output
12.Read input from GPIO pin ‘13’
13. Store the input value in the variable ‘i’
14. If (i==1) then print the message as “Object is detected” and make the LED ON

15. If (i==0) then print the message as “No object detected” and make the LED OFF

Observation:

1. See output on Command prompt or Python shell and also check LED status.

Experiment 5

Aim:

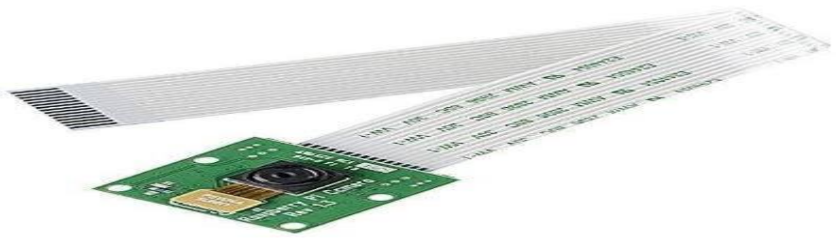
Understanding and connectivity of Raspberry-Pi /Beagle board with camera. Write an application to capture and store the image.

Theory:

Raspberry Pi Camera Module The Raspberry Pi Camera Module v2 replaced the original Camera Module in April 2016.The v2 Camera Module has a Sony IMX219 8-megapixel sensor (compared to the 5-megapixelOmniVision OV5647 sensor of the original camera).The Camera Module can be used to take high-definition video, as well as stills photographs. It’s easy to use for beginners, but has plenty to offer advanced users if you’re looking to expand your knowledge. We can also use the libraries we bundle with the camera to create effects. We can read all the gory details about IMX219 and the Exmore Rback-illuminated sensor architecture on Sony’s website, but suffice to say this is more than just are solution upgrade: it’s a leap forward in image quality, colour fidelity, and low-light performance.

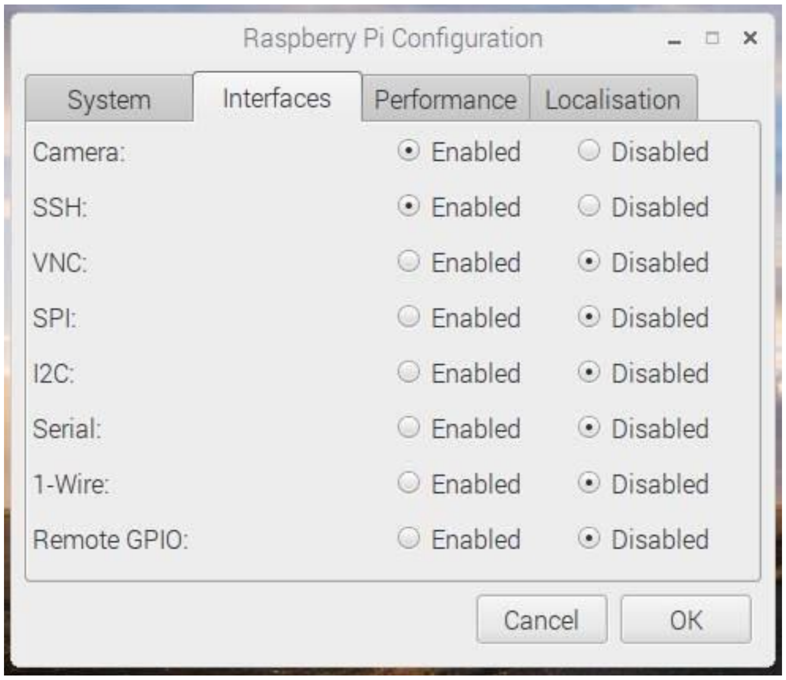
It supports 1080p30, 720p60 and VGA90 video modes, as well as still capture. It attaches via a15cm ribbon cable to the CSI port on the Raspberry Pi. The camera works with all models of Raspberry Pi 1, 2, and 3.It can be accessed through the MMAL and V4L APIs, and there are numerous third-party libraries built for it, including the Pi camera Python library. The camera module is very popular in home security applications, and in wildlife camera traps.

Pi Camera



Pi Camera Fig.1

Open Raspberry Pi Configuration and Enable the Camera



Camera:	[✓ Enabled]	Disabled
SSH:	[✓ Enabled]	Disabled
VNC:	Enabled	[✓ Disabled]
SPI:	Enabled	[✓ Disabled]
I2C:	Enabled	[✓ Disabled]
Serial:	Enabled	[✓ Disabled]
1-Wire:	Enabled	[✓ Disabled]
Remote GPIO:	Enabled	[✓ Disabled]

Camera Preview

```
from picamera import PiCamera
from time import sleep
camera=PiCamera()
camera.start_preview()
sleep(10)
camera.stop_preview()
```

Rotating the Camera

```
camera.rotation=180
camera.start_preview()
sleep(10)
camera.stop_preview()
```

Storing the image

```
from picamera import PiCamera
from time import sleep
camera=PiCamera()
camera.start_preview()
sleep(10)
camera.capture('/home/pi/Desktop/image1.jpg')
camera.stop_preview()
```

Recording the video

```
from pi camera import Pi Camera
from time import sleep
camera=Pi Camera()
camera.start_preview()
camera.start_recording('/home/pi/video.h264')
sleep(10)
camera.stop_reocrding()
camera.stop_preview()
```

Converting and Playing Video

The video format need to get converted to MP4. So install gpac.

sudo apt-get install gpac

Now convert the video to MP4:

MP4Box- fps30 -addvideo.h264 video.mp4

Conclusion

Thus, we have studied Pi Camera and also stored the images and videos using Pi Camera.

Experiment 6

Aim

Understanding and connectivity of Raspberry-Pi /Beagle board with a Zigbee module. Write a network application for communication between two devices using Zigbee.

Theory

ZigBee is a communication device used for the data transfer between the controllers, computers, systems, really anything with a serial port. As it works with low power consumption, the transmission distances is limited to 10–100 meters line-of-sight. ZigBee devices can transmit data over long distances by passing data through a mesh network of intermediate devices to reach more distant ones. ZigBee is typically used in low data rate applications that require long battery life and secure networking. Its main applications are in the field of wireless sensor network based on industries as it requires short-range low-rate wireless data transfer. The technology defined by the ZigBee specification is intended to be simpler and less expensive than other wireless networks

Here we make use of an interface of Zigbee with Raspberry Pi2 for a proper wireless communication. Raspberry Pi2 has got four USB ports, so it is better to use a Zigbee Dongle for this interface. Now we want to check the communication between the two paired ZigBee modules.

Interfacing of ZigbeeFig.1

Python script to perform Zigbee communication-

```
Import serial
#Enable USB Communication
ser=serial.Serial('/dev/ttyUSB0',9600,TIMEOUT=.5)
while True:
    ser.write('Hello User \r\n') #write a Data
    incoming=ser.readline().strip()
    print 'Received Data:'+incoming
```

Conclusion

Thus,we have done Zigbee Communication between two Raspberry Pi Devices.

Experiment 7

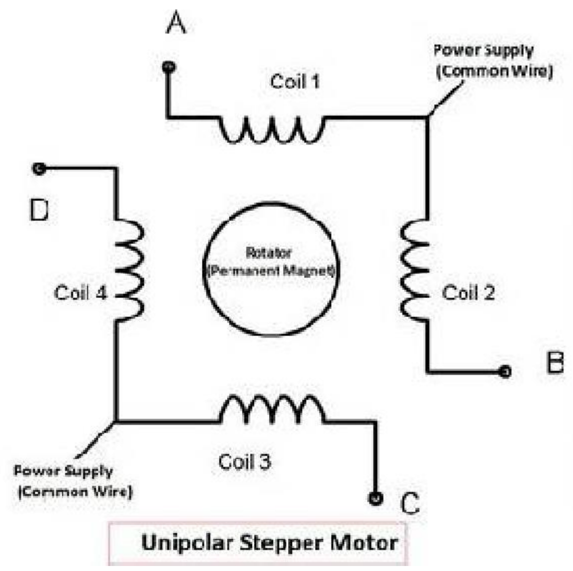
Title

Write an application using Raspberry-Pi /Beagle board to control the operation of stepper motor.

Theory

Stepper Motor

In Stepper Motor, as the name itself says, the rotation of shaft is in Step form. There are different
types of Stepper Motor; in here we will be using the most popular one that is **Unipolar Stepper Motor.** Unlike DC motor, we can rotate stepper motor to any particular angle by giving it proper instructions.

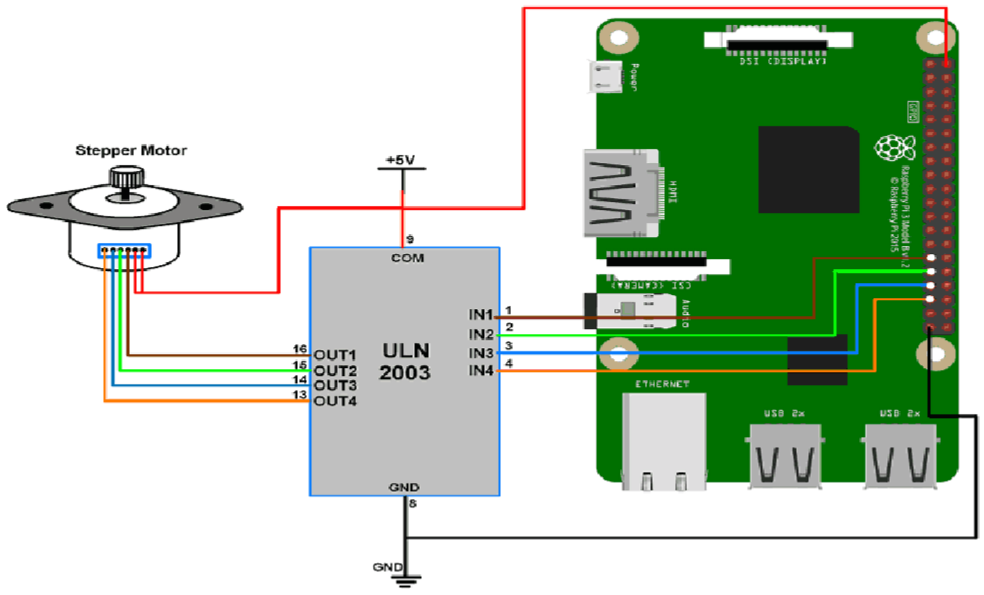

A
Coil 1
Power Supply (Common Wire)
D
Rotator (Permanent Magnet)
Coil 4
Coil 2
B
Coil 3
Power Supply (Common Wire)
C
Unipolar Stepper Motor

Unipolar Stepper motor Fig.1

To rotate this Four Stage Stepper Motor, we will deliver power pulses by using Stepper Motor Driver Circuit. The driver circuit takes logic triggers from PI. If we control the logic triggers, we control the power pulses and hence the speed of stepper motor.

There are **40 GPIO output pins in Raspberry Pi 2**. But out of 40, only 26 GPIO pins (GPIO2 to GPIO27) can be programmed. Some of these pins perform some special functions. With special GPIO put aside, we have only 17 GPIO remaining. Each of these 17 GPIO pin can deliver a maximum of **15mA**current. And the sum of currents from all GPIO Pins cannot exceed **50mA.**

There are **+5V (Pin 2 & 4) and +3.3V (Pin 1 & 17) power output pins** on the board for connecting other modules and sensors. These power rails cannot be used to drive the Stepper Motor, because we need more power to rotate it. So we have to deliver the power to Stepper Motor from another power source. My stepper motor has a voltage rating of 9V so I am using a 9v battery as my second power source. Search your stepper motor model number to know voltage rating. Depending on the rating choose the secondary source appropriately.



Sample program

Python Program

```
'
Stepper Motor interfacing with Raspberry Pi

import RPi.GPIO as GPIO
from time import sleep
import sys

#assign GPIO pins for motor
motor_channel = (29,31,33,35)
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
#for defining more than 1 GPIO channel as input/output use
GPIO.setup(motor_channel, GPIO.OUT)

motor_direction = input('select motor direction a=anticlockwise, c=clockwise: ')
while True:
try:
if(motor_direction == 'c'):
print('motor running clockwise\n')
GPIO.output(motor_channel, (GPIO.HIGH,GPIO.LOW,GPIO.LOW,GPIO.HIGH))
sleep(0.02)
GPIO.output(motor_channel, (GPIO.HIGH,GPIO.HIGH,GPIO.LOW,GPIO.LOW))
sleep(0.02)
GPIO.output(motor_channel, (GPIO.LOW,GPIO.HIGH,GPIO.HIGH,GPIO.LOW))
sleep(0.02)
GPIO.output(motor_channel, (GPIO.LOW,GPIO.LOW,GPIO.HIGH,GPIO.HIGH))
sleep(0.02)

elif(motor_direction == 'a'):
print('motor running anti-clockwise\n')
```

```
GPIO.output(motor_channel, (GPIO.HIGH,GPIO.LOW,GPIO.LOW,GPIO.HIGH))
sleep(0.02)
GPIO.output(motor_channel, (GPIO.LOW,GPIO.LOW,GPIO.HIGH,GPIO.HIGH))
sleep(0.02)
GPIO.output(motor_channel, (GPIO.LOW,GPIO.HIGH,GPIO.HIGH,GPIO.LOW))
sleep(0.02)
GPIO.output(motor_channel, (GPIO.HIGH,GPIO.HIGH,GPIO.LOW,GPIO.LOW))
sleep(0.02)


#press ctrl+c for keyboard interrupt
except KeyboardInterrupt:
#query for setting motor direction or exit
motor_direction = input('select motor direction a=anticlockwise, c=clockwise or q=exit: ')
#check for exit
if(motor_direction == 'q'):
print('motor stopped')
sys.exit(0)
```

Conclusion-

Thus, we have implemented application of stepper motors using Python with Raspberry Pi.

Experiment 8

Aim

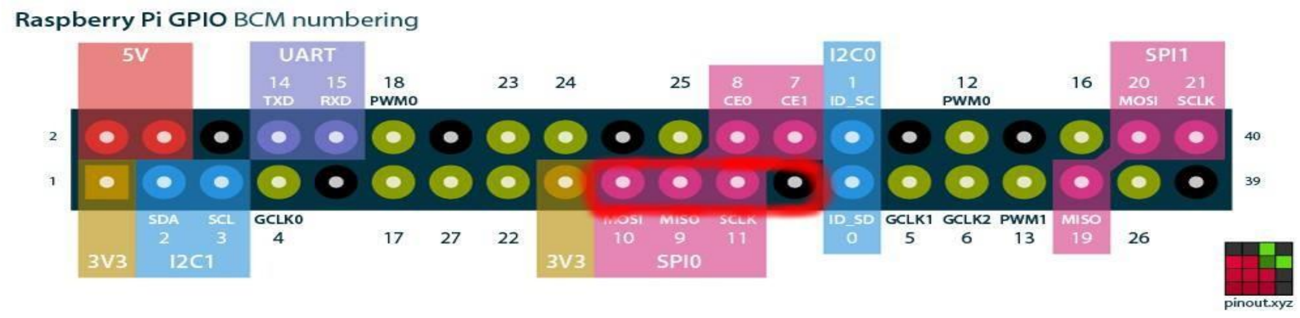
Write an application using Raspberry-Pi /Beagle board to control the operation of a hardware simulated traffic signal.

Theory

Attaching the Traffic Lights

The Low Voltage Labs traffic lights connect to the Pi using four pins. One of these needs to be ground, the other three being actual GPIO pins used to control each of the individual LEDs.

Before powering up the Pi, attach the traffic lights so that the pins connect to the GPIO pins highlighted in red:



Programming the Traffic Lights

First, you need to install a couple of extra software packages needed to allow you to download my
sample code, and to give Python access to the GPIO pins on the Pi. Enter the following at the command line:

sudo apt-get install python-dev python-rpi.gpio git

How It Works

The code for this is very simple. It starts by importing the Rpi.GPIO library, plus time which gives us a timed wait function, signal that allows us to trap the signal sent when the user tries to quit the program and sys so we can send an appropriate exit signal back to the operating system before terminating.

```
import RPi.GPIO as GPIO
import time
import signal
import sys
```

Next we put the GPIO library into “BCM” or “Broadcom” mode (so we can refer to pins by the same numbers as are labelled with in GPIO pin diagrams), and sets pins 9 (red LED), 10 (amber LED) and 11 (green LED) to be used as outputs:

# Setup

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(9,   GPIO.OUT)
GPIO.setup(10, GPIO.OUT)
GPIO.setup(11, GPIO.OUT)
```

The main part of the program will run in an infinite loop until the user exits it by stopping Python with CtrlC. It’s a good idea to add a handler function that will run whenever this happens, so that we can turn off all the lights prior to exiting (thus ensuring they’ll also be in the state we expect them to start in the next time the program is run):

```
# Turn off all lights when user ends demo
def allLightsOff(signal, frame):
        GPIO.output(9, False)
        GPIO.output(10, False)
        GPIO.output(11, False)
        GPIO.cleanup()
        Sys.exit(0)
signal.signal(signal.SIGINT, allLightsOff)
```

The main body of the code then consists of an infinite while loop that turns on the red light (pin 9), waits, turns on the amber light (pin 10), waits, then cycles through the rest of the traffic light pattern by turning the appropriate LEDs on and off:

When Control-C is pressed an interrupt signal. SIGINT is sent. This is handled by the all Lights Off function that switches all the lights off, tidies up the GPIO library state and exits cleanly back to the operating system.

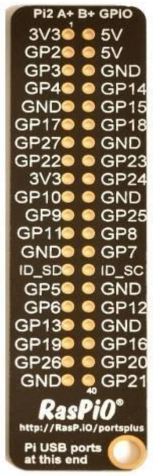
Conclusion-

Thus, we have implemented the application for traffic signals using Raspberry Pi.

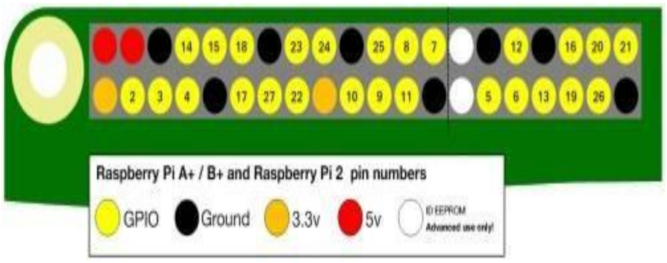
Traffic Light System Using Raspberry Pi 3 (Python)

GPIO pins

One powerfull feature of the Raspberry Pi is the row of GPIO pins along the top edge of the boadr. GPIO stands for General-Purpose Input/Output. These pins are a physical interface between the Raspberry Pi and the outside world.



If you don't have a pin label, then this can help you to identufy the pin numbers


14
15
18
23
24
25
8
7
12
16
20
21
2
3
4
17
27
22
10
9
11
5
6
13
19
26
Raspberry Pi A+ / B+ and Raspberry Pi 2 pin numbers
GPIO
Ground
3.3v
5v
ID EEPROM Advanced use only!

3.3 volts

Anything connected to these pins will always get 3.3 v of power.

5 volts

Anything connected to these pins will always get 5 v of power.

GND

Zero volts, used to complete a circuit

GPIO

These pins are for general-purpose use and can be configures as input or output pins

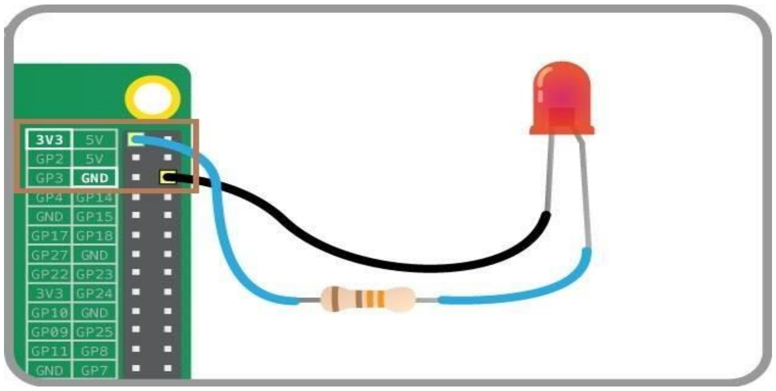
ID_SC/ID_SD/DNC

Special purpose pins.

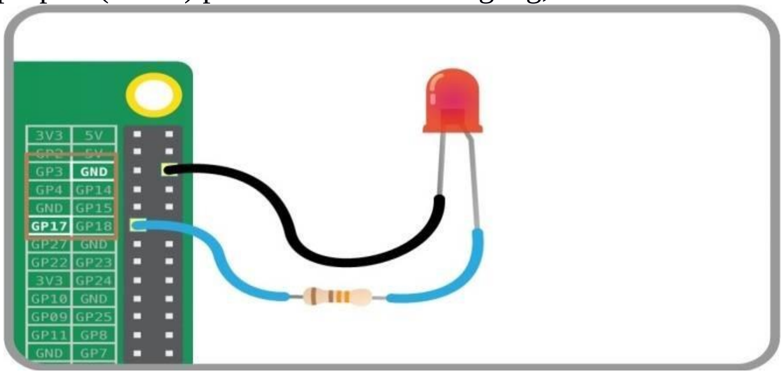
Lighting an LED

LEDs are delicate little things. if you put too much current through them they will pop. To limit the current going through the LED, you should always use a register in series with it.

Long leg an LED to the Pi 3.3v and the short leg to a GND pin. Automatically LED will turn ON



If you connect special purpose(GPIO) pin are connect to long leg, make sure to write the import code.

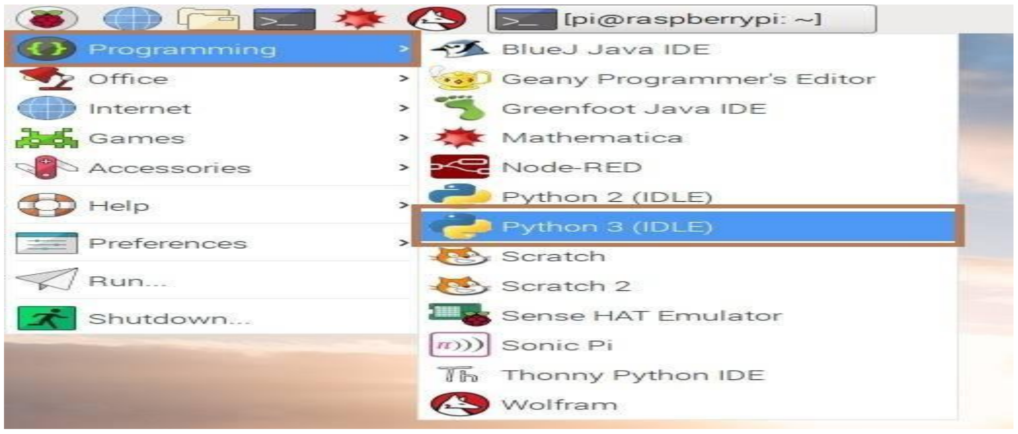


Switching an LED on and off

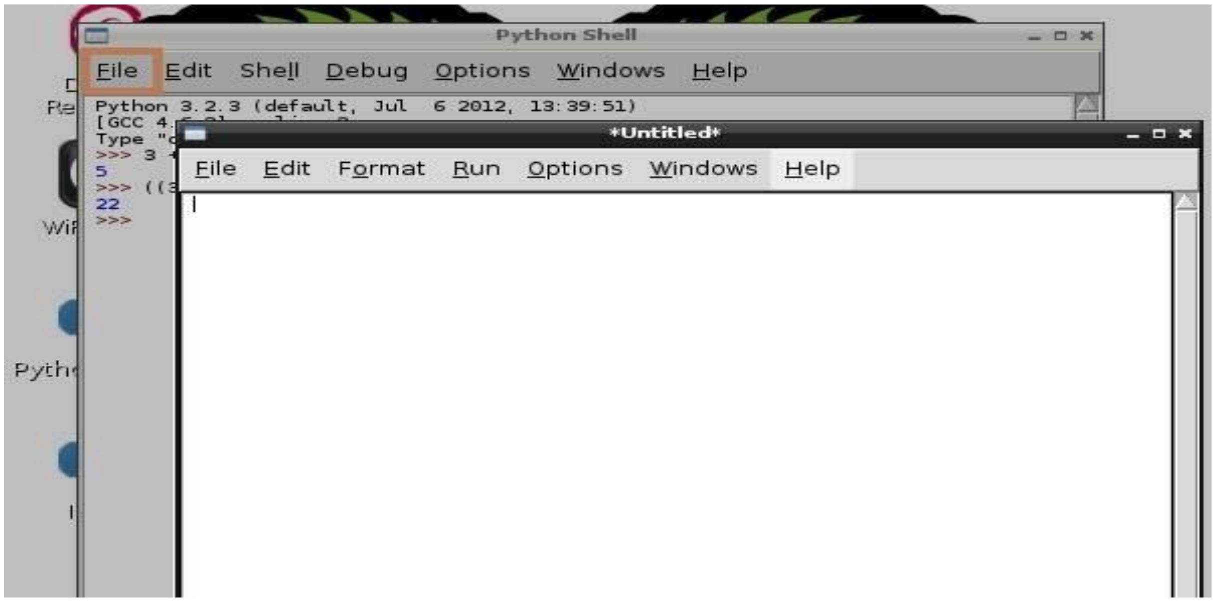
GPIO Zero is a new Python library which provides a simple interface to everyday GPIO component. It comes installed by defalut in Rasbian.

Open IDLE(*I*ntegrated **D**eve**l**opment **E**nvironment), which you can use to write and run code.

Raspbian Menu Icon >> Programming >> Python 3 (IDLE).



1. To create a new file in IDLE, You can click on File and then New File in IDLE's menu bar.
2. Create a new file by clicking File >> New File


Python Shell
File Edit Shell Debug Options Windows Help
Untitled
File Edit Format Run Options Windows Help

```
Python 3.2.3 (default, Jul  6 2012, 13:39:51)
```

1. Save the new file by clicking File >> Save. Save the file as *trffic.py*
2. You'll need the LED Class, and to tell it that the LED is on pin 17. Write the following code in your new file.

1. from gpiozero import LED
2. led = LED(17)

3. To make the LED switch on, type the following and press Enter

4. led.on()

5. To make it swith off you can type

6. led.off()

Your LED should switch on and then off again. But that's not all you can do. Similary check the Buzzer and
Button. Just import a Buzzer and Button for header file.

Making Traffic Light

We need a breadboard, three LEDs, a button, a buzzer and the necessary jumper cables and registors.

Wiring

First, you need to understand how each component is connected.

1. A push button requires 1 ground pin and 1 GPIO pin
2. An LED requires 1 ground pin and 1 GPIO pin, with a current limiting registor
3. A buzzer requies 1 ground pin and 1 GPIO pin

Place the components on the breadboard and connect them to the Raspberry Pi GPIO pins, accoring to the following diagram.

Component GPIO pin

Button	21
Red LED	25
Yellow LED	8
Green LED	7
Buzzer	15

This is the same as the Switching and LED on and off step

1. Open Pyton 3 from the main menu
2. Create a new file just save with the project name.py

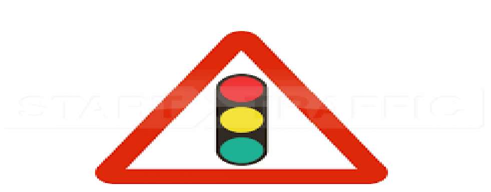
Add TrafficLight, Button and Buzzer Code

```
from gpiozero import Button, TrafficLights, Buzzer
from time import sleep
buzzer = Buzzer(15)
button = Button(21)
lights = TrafficLights(25, 8, 7)
 while True:
 button.wait_for_press()
  buzzer.on()
   light.green.on()
  sleep(1)
```

```
lights.amber.on()
 sleep(1)
  lights.red.on()
        sleep(1)
        lights.off()
        buzzer.off()
```

OUTPUT

Run your Powershell coding (F5)



Finally, we have successfully created smart traffic system using Raspberry Pi.

Experiment 9

Writ an application Using Raspberry-pi/Begalboard tp control the operaion of a hardware simulated lift elevator

Lift Elevator Simulation using Raspberry pi board

Aim/Objectives:

1. To understand the working principle of Lift Elevator
2. To interface the Lift Elevator module with Raspberry Pi model
3. To program the Raspberry Pi model to control operation of Lift Elevator module

Software:

1. Raspbian OS (IDLE)

Hardware Modules:

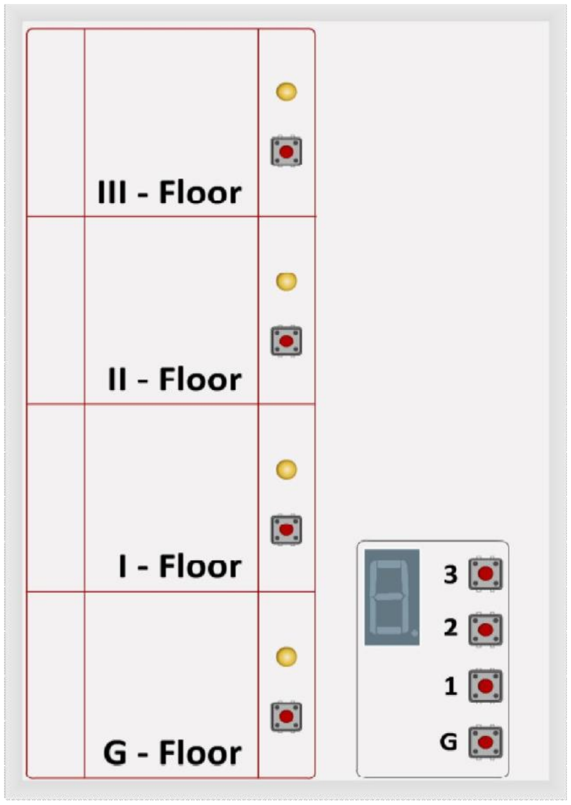
2. Raspberry Pi Board module
3. Push buttons (qty. 8)
4. Seven Segment Display (qty. 1)
5. Leds (qty. 4)
6. Monitor

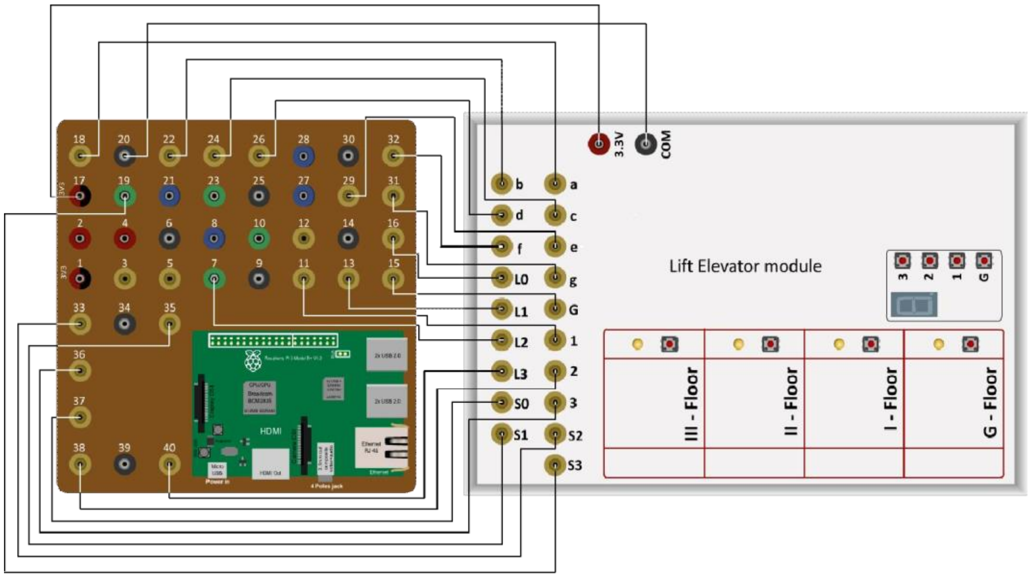
Theory:

Lift Elevator Module has two parts:

1. Moving part inside the lift and
2. Stationary part outside the lift at each floor to call the lift
3. In this simulation module, we have considered four floors of a building
4. So the Moving part contains four Push buttons. Out of these four buttons, one button is for each floorhaving floor number written below it. User has to push one of these buttons as the destination floor.
5. The moving part also contains a Seven Segment Display to indicate the current floor number when the lift is moving.
6. By pressing one of these buttons, the user indicates the destination floor.
7. At each floor, the stationary part contains a buttons for calling the lift.
8. In real life, when the lift is called by any floor, the lift starts moving towards the particular floor. When it reaches the particular floor, the lift door is opened.
9. In our module, this situation is indicated by the "LED ON" status. So the LED ON status indicates that the lift has arrived at the particular floor, its door is opened and the user is entering the lift.
10. In real life, as soon as the entering users get finished, the lift door is closed and the lift starts moving toward the destination.
11. In our module, this situation is indicated by "LED OFF" status. So the LED OFF status indicates that the lift is moving towards the destination floor.

Interface diagram:





Safety precautions:

1. First, make all the connections as per steps given below
2. power supply

Steps for assembling circuit:

1. Connect all the pins of Lift Elevator module to pins of Raspberry Pi module as shown inthe above figure.

Procedure:

2. Write the program as per the algorithm given.
3. Save the program
4. Run code using Run module.

Algorithm:

1. Import GPIO and time libraries
2. Set GPIO mode as per Board
3. Declare four Push button pins of the stationary part (outside the lift at each floor for calling the lift).
4. Declare four LED pins at each floor for detection of door close and open.
5. Declare four Push button pins of the moving part (inside the lift forselecting the destination)
6. Declare seven pins of Seven Segment Display (this indicates the current floor number of the moving lift)
7. Set the Push button pins as Input.
8. Set the seven segment display pins and LED pins as Output.
9. Store the value of each digit of seven segment display in variables.
10. In the while loop, If “BUTTON_ONE” is pressed then lift at floor 1 and LED at floor 1 get ON for 5 second then gets OFF (door close).
11. Person enters in the lift and presses the push button of any one floor inthe moving lift.
12. The Seven Segment Display displays the floor number of the destination.

Observation:

1. Observe the output on LEDs and Seven Segment Display.

```
#Interfacing Lift Elevator module with Raspberry-Pi-3

import RPi.GPIO as GPIO
import time

FloorButton0 = 37
FloorButton1 = 35
FloorButton2 = 33
FloorButton3 = 19
```

```
 LiftButton0 = 15
LiftButton1 = 11
LiftButton2 = 38
LiftButton3 = 36

#GPIO setup for the LEDs
FloorLed0 = 16
FloorLed1 = 13
FloorLed2 = 7
FloorLed3= 40

#GPIO setup for the Seven Segment Display
segAPin=18
segBPin=22
segCPin=24
segDPin=26
segEPin=29
segFPin=32
segGPin=31


 GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)


GPIO.setup(FloorButton0,  GPIO.IN)
GPIO.setup(FloorButton1,  GPIO.IN)
GPIO.setup(FloorButton2,  GPIO.IN)
GPIO.setup(FloorButton3, GPIO.IN)

 GPIO.setup(LiftButton0, GPIO.IN)
GPIO.setup(LiftButton1,  GPIO.IN)
GPIO.setup(LiftButton2,  GPIO.IN)
GPIO.setup(LiftButton3, GPIO.IN)



GPIO.setup(FloorLed0, GPIO.OUT) #Floor 1
GPIO.setup(FloorLed1, GPIO.OUT) #Floor 2
GPIO.setup(FloorLed2, GPIO.OUT) #Floor 3
GPIO.setup(FloorLed3, GPIO.OUT) #Floor 4



GPIO.setup(segAPin, GPIO.OUT)
GPIO.setup(segBPin, GPIO.OUT)
GPIO.setup(segCPin, GPIO.OUT)
GPIO.setup(segDPin, GPIO.OUT)
GPIO.setup(segEPin, GPIO.OUT)
GPIO.setup(segFPin, GPIO.OUT)
GPIO.setup(segGPin, GPIO.OUT)

digitclr=[0,0,0,0,0,0,0]
digit0=[1,1,1,1,1,1,0]
digit1=[0,1,1,0,0,0,0]
digit2=[1,1,0,1,1,0,1]
digit3=[1,1,1,1,0,0,1]


gpin=[18,22,24,26,29,32,31]
#routine to clear and then write to display
def digdisp(digit):
for x in range (0,7):
GPIO.output(gpin[x], digitclr[x])

for x in range (0,7):
GPIO.output(gpin[x], digit[x])


while True:


if (GPIO.input(FloorButton0)== True) :
GPIO.output(FloorLed0,1)

print"0"

digdisp(digit0)
time.sleep(1)
GPIO.output(FloorLed0,0)
time.sleep(3)

while True:
if(GPIO.input(LiftButton1)== True):
print'floor ONE'
digdisp(digit0)
time.sleep(1)
digdisp(digit1)
time.sleep(2)
break
```

```
 elif (GPIO.input(LiftButton2)== True):

print'floor TWO'
digdisp(digit0)
time.sleep(1)
digdisp(digit1)
time.sleep(1)
digdisp(digit2)
time.sleep(2)
break


 elif (GPIO.input(LiftButton3)== True):

 print'floor THREE'
 digdisp(digit0)
time.sleep(1)
digdisp(digit1)
time.sleep(1)
digdisp(digit2)
time.sleep(1)
digdisp(digit3)
time.sleep(2)

  break



  elif (GPIO.input(FloorButton1) == True):


 GPIO.output(FloorLed1, 1)
 print"1"

 digdisp(digit0)
time.sleep(1)
digdisp(digit1)
time.sleep(1)
time.sleep(4)

 GPIO.output(FloorLed1, 0)

while True:
 if(GPIO.input(LiftButton0)== True):
 print 'floor ZERO'
digdisp(digit0)
time.sleep(2)
break

 elif (GPIO.input(LiftButton2)== True):
 print'floor TWO'
  digdisp(digit2)
 time.sleep(2)
 break


 elif (GPIO.input(LiftButton3)== True):
print'floor THREE'
 digdisp(digit2)
time.sleep(1)
digdisp(digit3)
time.sleep(2)
break




 elif (GPIO.input(FloorButton2) == True):

 GPIO.output(FloorLed2, 1)

 print"2"

  digdisp(digit0)
time.sleep(1)
digdisp(digit1)
time.sleep(1)
digdisp(digit2)
time.sleep(1)
time.sleep(5)
GPIO.output(FloorLed2, 0)

 while True:
 if(GPIO.input(LiftButton0)== True):
 print 'floor ZERO'
digdisp(digit1)
time.sleep(1)
```

```
 digdisp(digit0)
 time.sleep(2)
 break

  elif (GPIO.input(LiftButton1)== True):
 print 'floor ONE'
  digdisp(digit1)
  time.sleep(2)
  break

  elif (GPIO.input(LiftButton3)== True):

  print'floor THREE'
  digdisp(digit3)
 time.sleep(2)
 break


  elif (GPIO.input(FloorButton3) == True):

  GPIO.output(FloorLed3, 1)

  print"3"
 digdisp(digit0)
 time.sleep(1)
 digdisp(digit1)
 time.sleep(1)
 digdisp(digit2)
 time.sleep(1)
 digdisp(digit3)
 time.sleep(6)

   GPIO.output(FloorLed3, 0)

   while True:
  if (GPIO.input(LiftButton0)== True):
 print 'floor ZERO'
  digdisp(digit2)
 time.sleep(1)
 digdisp(digit1)
 time.sleep(1)
 digdisp(digit0)
 time.sleep(2)
 break
```

```
  elif (GPIO.input(LiftButton1)== True):

  print 'Floor ONE'
 digdisp(digit2)
 time.sleep(1)
 digdisp(digit1)
 time.sleep(2)
 break

   elif (GPIO.input(LiftButton2)== True):

  print'floor TWO'
 digdisp(digit2)
  time.sleep(2)
  break



  else:
  ####       time.sleep(3)
  digdisp(digit0)
  GPIO.output(FloorLed0, 0)
  GPIO.output(FloorLed1, 0)
 GPIO.output(FloorLed2, 0)
 GPIO.output(FloorLed3, 0)


  else:
  ####       time.sleep(3)
  digdisp(digit0)
  GPIO.output(FloorLed1, 0)
  GPIO.output(FloorLed2, 0)
 GPIO.output(FloorLed3, 0)
 GPIO.output(FloorLed0, 0)
```

GESCOE, Nashik

Experiment 9

Aim

Create a small dashboard application to be deployed on cloud. Different publisher devices can publish their information and interested application can subscribe.

Theory

IoT Platforms

The IoT platforms are suites of components those help to setup and manage the internet connected devices. A person can remotely collect data, monitor and manage all internet connected devices from a single system.There are a bunch of IoT platforms available online but building an IoT solution for a company is all depend on IoT platform host and support quality.

IOT Cloud Platforms-

1. Kaa IoT Platform
2. SiteWhere: Open Platform for the Internet of Things
3. ThingSpeak: An open IoT platform with MATLAB analytics
4. DeviceHive: IoT Made Easy
5. Zetta: API-First Internet of Things Platform

Kaa-Features

1. Manage an unlimited number of connected devices
2. Set up cross-device interoperability
3. Perform A/B service testing
4. Perform real-time device monitoring
5. Perform remote device provisioning and configuration
6. Collect and analyze sensor data
7. Analyze user behaviour deliver targeted notifications
8. Create cloud services for smart products

Sitewhere-Features

1. Run any number of IoT applications on a single Site Where instance
2. Spring delivers the core configuration framework
3. Connect devices with MQTT, AMQP, Stomp, and other protocols
4. Add devices through self-registration, REST services, or in batches
5. Integrates with third-party integration frameworks such as Mule Any Point
6. Default database storage is MongoD
7. Eclipse Californium for CoAP messaging
8. InfluxDB for event data storage
9. Grafana to visualize SiteWhere data
10. HBase for non-relational data store

ThingSpeak – Features

1. Collect data in private channels
2. Share data with public channels
3. RESTful and MQTT APIs
4. MATLAB analytics and visualizations
5. Alerts
6. Event scheduling
7. App integrations
8. Worldwide community

DeviceHive – Features

1. Directly integrate with Alexa
2. Visualization dashboard of your choice
3. Customize DeviceHive behavior by running your customjavascript code.
4. It supports the Big data solutions such as ElasticSearch, ApacheSpark, Cassandra and Kafka for real-time and batch processing.
5. Connect any device via REST API, WebSockets or MQTT.
6. It comes with Apache Spark and Spark Streaming support.

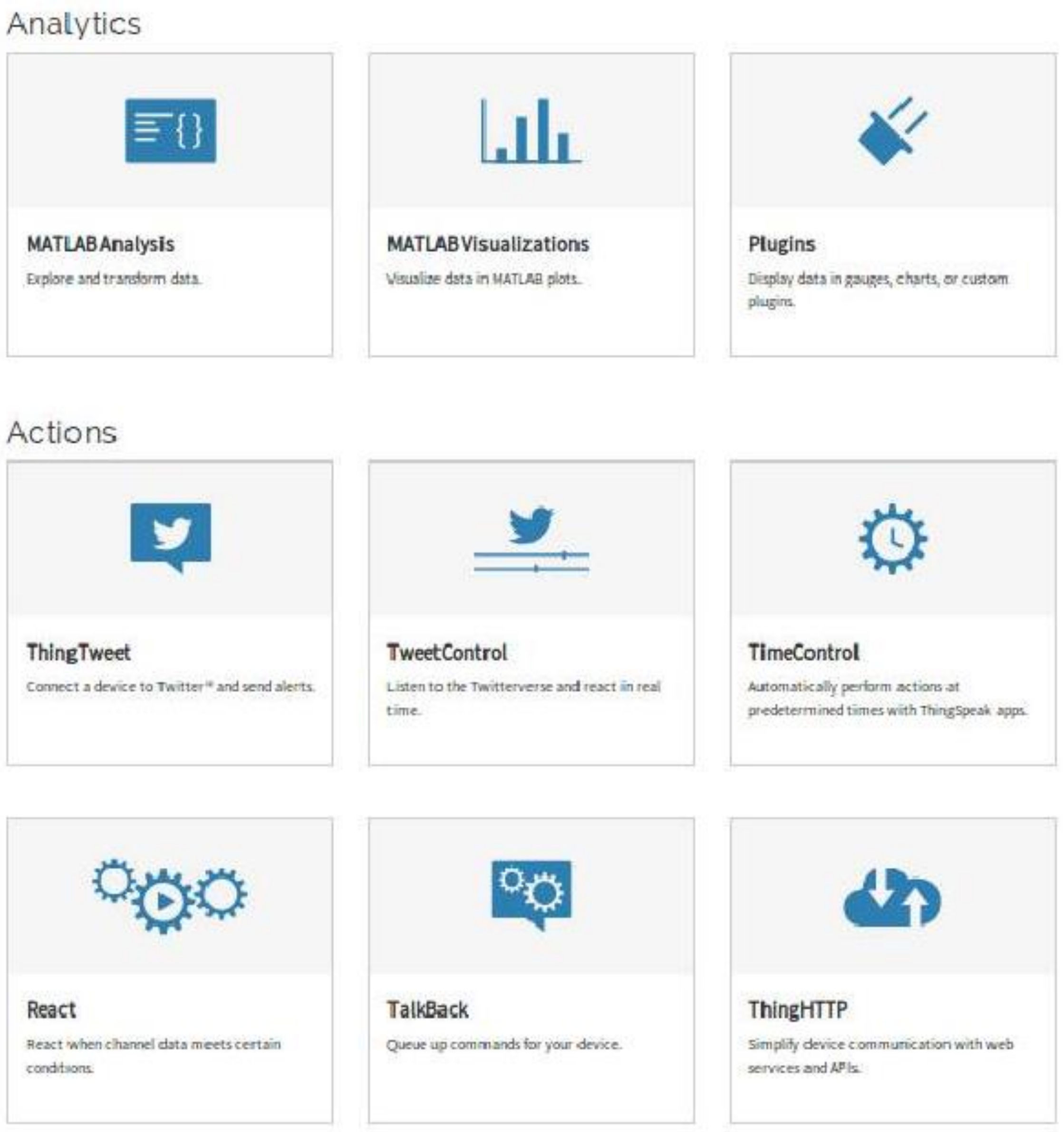
7. Supports libraries written in various programming languages,including Android and iOS Libraries

8. It allows running batch analytics and machine learning on top of your device data

Zetta – Features

1. Built around Node.js, REST, WebSockets,and a flow-based "reactive programming".
2. Supports wide range of hacker boards
3. Zetta allows you to assemble smartphone apps, device apps, and cloud apps.

ThingsSpeak Apps



Analytics

MATLAB Analysis
Explore and transform data.

MATLAB Visualizations
Visualize data in MATLAB plots.

Plugins
Display data in gauges, charts, or custom plugins.

Actions

ThingTweet
Connect a device to Twitter® and send alerts.

TweetControl
Listen to the Twitterverse and react in real time.

TimeControl
Automatically perform actions at predetermined times with ThingSpeak apps.

React
React when channel data meets certain conditions.

TalkBack
Queue up commands for your device.

ThingHTTP
Simplify device communication with web services and APIs.

nclusion

Thus, we have designed small application using ThingsSpeak.

Experiment 10

Aim

Create a simple web interface for Raspberry-Pi/Beagle board to control the connected LEDs remotely through the interface.

Theory

WiringPi

WiringPi is a PIN based GPIO access library written in C for the BCM2835 used in the Raspberry Pi. It's released under the GNU LGPLv3 license and is usable from C, C++ and RTB (BASIC) as well as many other languages with suitable wrappers

Install WiringPi

WiringPi is not included with Raspbian, so, to begin, you'll need to download and install it. That means your Pi will need a connection to the Internet – either via Ethernet or WiFi. We can do using Git to download the latest version. As long as you have Git installed, these commands should be all you need to download and install WiringPi:

```
pi@raspberrypi~$git clone git://git.drogon.net/wiringPi
pi@raspberrypi~$cdwiringPi pi@raspberrypi~/wiringPi/wiringPi$./build
```

GPIO Command Line utility

Task: Connect the LED GND to Short Pin GPIO18 to Long Pin

Remember:GPIO18 is PIN 1 in Wiring PI

GPIO Command Line utility

1. Glow the LED by Value

   gpio write 1 1

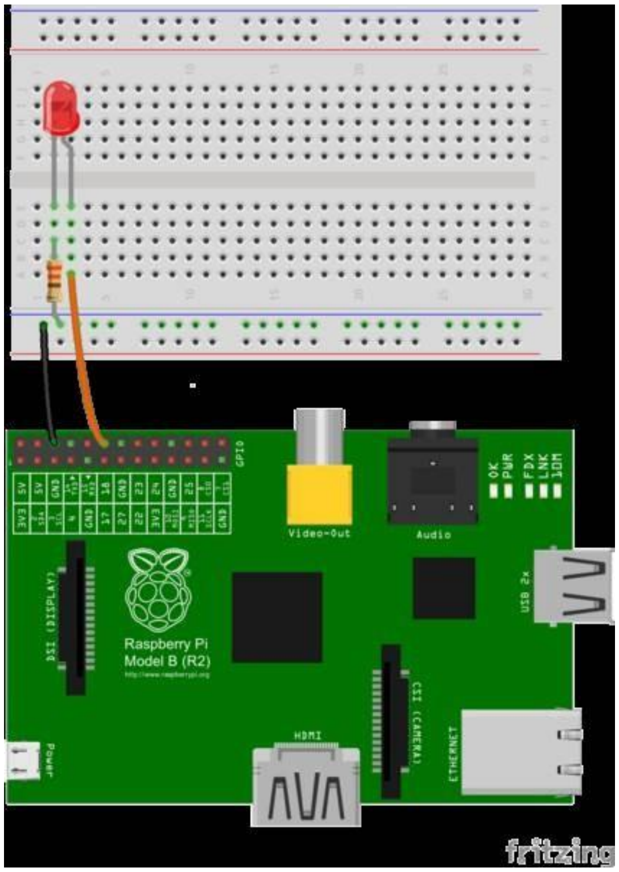
2. Off the LED by

   gpio write 1 0

Web Interface to LED

1. Create the front page using HTML which contains two buttons to put the LED in ON or OFF state.

2. Control the data input from buttons using PHP page.



Conclusion

Thus,we have created simple web interface for Raspberry-Pi/Beagle board to control the connected LEDs remotely through the interface.

Experiment 11

Aim

Develop a Real time application like smart home with following requirements: When user enters into house the required appliances like fan, light should be switched ON. Appliances should also get controlled remotely by a suitable web interface. The objective of this application is student should construct complete Smart application in group.

Theory

Basics-Send emails using Python

1. The smtplib module of Python is basically all you need to send simple emails, without any subject line or such additional information.
2. But for real emails, you do need a subject line and lots of information — maybe even pictures and attachments.
3. This is where Python's email package comes in.Keep in mind that it's not possible to send an email message using the email package alone. You need a combination of both email and smtplib.

How to send emails?

1. Set up the SMTP server and log into your account.
2. Create the MIME Multipart message object and load it with appropriate headers for From, To, and Subject fields.
3. Add your message body.
4. Send the message using the SMTP server object.

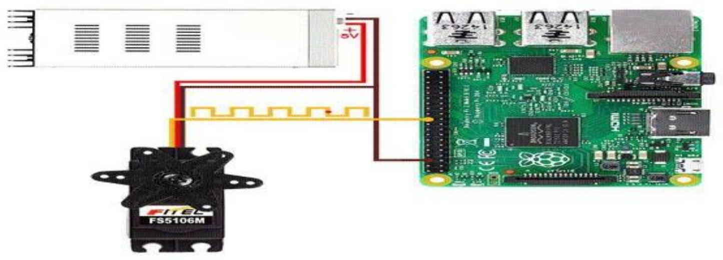
The smtplib

1. The smtplib module defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP or ESMTP listener daemon.
2. SMTP stands for Simple Mail Transfer Protocol. The smtplib modules is useful for communicating with mail servers to send mail.

3. Sending mail is done with Python's smtplib using an SMTP server.
4. Actual usage varies depending on complexity of the email and settings of the email server, the instructions here are based on sending email through Gmail

Servo Motor

1. A Servo Motor is a combination of DC motor, position control system and gears. Servos have many applications in the modern world and with that, they are available in different shapes and sizes. We will be using SG90 Servo Motor which is one of the popular and cheapest one.SG90 is a 180 degree servo. So with this servo we can position the axis from 0-180 degrees.
2. A Servo Motor mainly has three wires, one is for positive voltage, another is for ground and last one is for position setting. The Red wire is connected to power, Brown wire is connected to ground and Orange wire is connected to signal.
3. A Servo Motor is a combination of DC motor, position control system and gears. Servos have many applications in the modern world and with that, they are available in different shapes and sizes. We will be using SG90 Servo Motor which is one of the popular and cheapest one.SG90 is a 180 degree servo. So with this servo we can position the axis from 0-180 degrees.
4. A Servo Motor mainly has three wires, one is for positive voltage, another is for ground and last one is for position setting. The Red wire is connected to power, Brown wire is connected to ground and Orange wire is connected to signal.



Fig

Steps:

- Create the lock / unlock application to control the servo motor lock. Change its owner and group aswww-data.Location: /var/www/html

- Write the application to read the image and send it as email attachment to the user.

Location: /home/pi

- Write application using HTML-PHP to control theservo motor lock.

Location: /var/www/html

Conclusion

Thus, we have developed short application for Smart Home System.