

ASSIGNMENT NUMBER: A1**Revised On: 16-12-2019**

TITLE	Study of Raspberry- Pi, Beagle board, Arduino and other micro controller.
PROBLEM STATEMENT /DEFINITION	Study of Raspberry-Pi, Beagle board, Arduino and other micro controller (History & Elevation)
OBJECTIVE	<ul style="list-style-type: none">• To understand fundamentals of IoT and embedded system including essence, basic design strategy and process modeling.• To develop comprehensive approach towards building small low cost embedded IoT.
S/W PACKAGES AND HARDWARE APPARATUS USED	Raspberry-pi, Beagle board, PC with the configuration as Latest Version of 64 bit Operating Systems,Open Source Fedora-GHz. 8 G.B. RAM, 500 G.B. HDD, 15"Color Monitor, Keyboard, Mouse
REFERENCES	1. Derek Molley, Exploring Beaglebonel, Willey, ISBN: 978-1-118-935125 2. Matt Richardson and Shawn Wallace, Getting with Raspberry Pil, MAKER MEDIA, ISBN:978-93-5213-450-2
STEPS	Refer to details
INSTRUCTIONS FOR WRITING JOURNAL	<ul style="list-style-type: none">• Title• Problem Definition• Objectives• Theory• Class Diagram• Algorithm• Test cases• Program Listing(soft copy)• Output• Conclusion

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

Pre-requisite:

Basic knowledge of embedded system and IOT

Learning Objectives:

- To develop comprehensive approach towards building small low cost embedded IoT system.

Learning Outcomes:

The students will be able to

- Able to perform the connectivity with Raspberry-Pi, Beagle board, Arduino and other micro controller
- Implement an architectural design for IoT for specified requirement.

Theory:

Raspberry Pi

The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and in developing countries. The original model became far more popular than anticipated, selling outside its target market for uses such as robotics. It does not include peripherals (such as keyboards, mice and cases). However, some accessories have been included in several official and unofficial bundles. According to the Raspberry Pi Foundation, over 5 million Raspberry Pis were sold by February 2015, making it the best-selling British computer. By November 2016 they had sold 11 million units and 12.5m by March 2017, making it the third best-selling "general purpose computer" In July 2017, sales reached nearly 15 million.

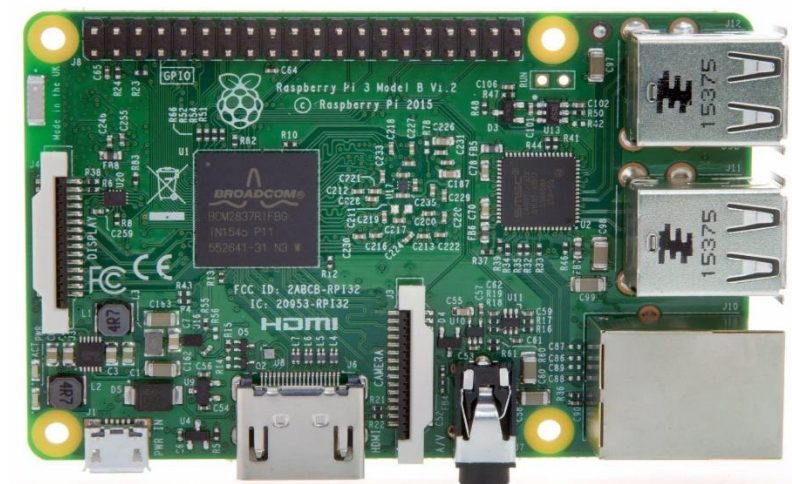


Fig: Raspberry Pi

Beagle Board:

The BeagleBoard is a low-power open-source single-board computer produced by Texas Instruments in association with Digi-Key and Newark element14. The BeagleBoard 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.

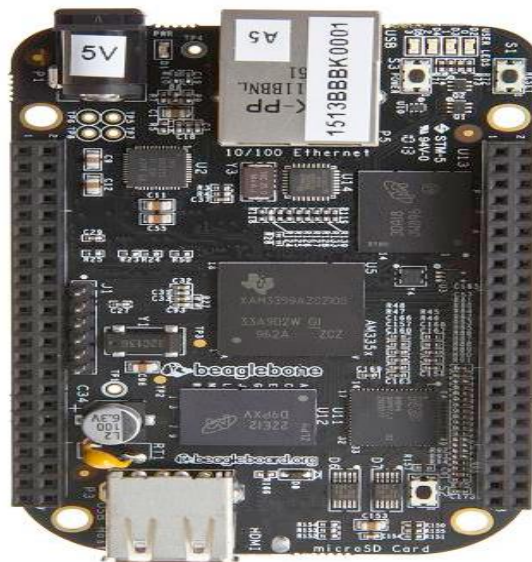


Fig: BeagleBoard

Arduino:

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

The project's products are distributed as open-source hardware and software, which are licensed under the GNU Lesser General Public License (LGPL) or the GNU General Public License (GPL), permitting the manufacture of Arduino boards and software distribution by anyone. Arduino boards are available commercially in preassembled form, or as do-it-yourself (DIY) kits. Arduino board designs use a variety of microprocessors and controllers.

The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits.

The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers.

The microcontrollers are typically programmed using a dialect of features from the programming languages C and C++. In addition to using traditional compiler toolchains, the Arduino project provides an integrated development environment (IDE) based on the Processing language project.

The Arduino project started in 2003 as a program for students at the Interaction Design Institute Ivrea in Ivrea, Italy, aiming to provide a low-cost and easy way for novices and professionals to create devices that interact with their environment using sensors and actuators.

Common examples of such devices intended for beginner hobbyists include simple robots, thermostats, and motion detectors.

The name Arduino comes from a bar in Ivrea, Italy, where some of the founders of the project used to meet. The bar was named after Arduin of Ivrea, who was the margrave of the March of Ivrea and King of Italy from 1002 to 1014.

Program structure:

A minimal Arduino C/C++ program consist of only two functions:

- `setup()`: This function is called once when a sketch starts after power-up or reset. It is used to initialize variables, input and output pin modes, and other libraries needed in the sketch.
- `loop()`: After `setup()` has been called, function `loop()` is executed repeatedly in the main program. It controls the board until the board is powered off or is reset.

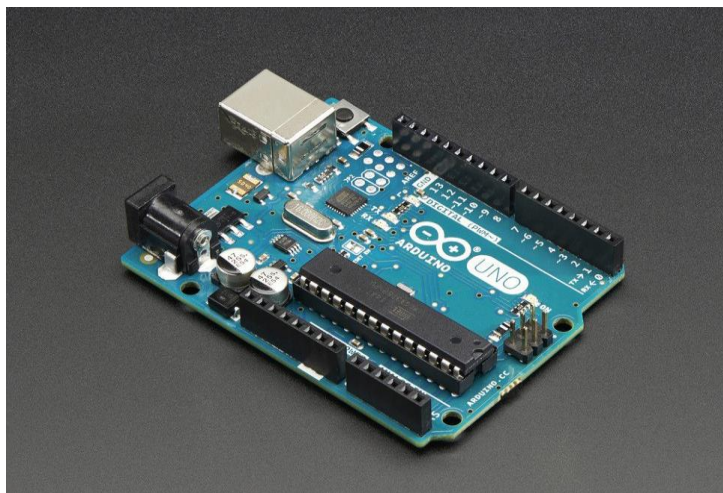


Fig: Arduino Uno SMD R3

Applications:

- Arduboy, a handheld game console based on Arduino
- Arduino Motion Control Rig.
- Arduinome, a MIDI controller device that mimics the Monome.
- ArduinoPhone, a do-it-yourself cellphone.
- Ardupilot, drone software and hardware.
- ArduSat, a cubesat based on Arduino.
- Automatic titration system based on Arduino and stepper motor.
- C-STEM Studio, a platform for hands-on integrated learning of computing, science, technology, engineering, and mathematics (C-STEM) with robotics.
- DC motor control using Arduino and H-Bridge.
- Data loggers for scientific research.
- Gameduino, an Arduino shield to create retro 2D video games.
- Homemade CNC using Arduino and DC motors with close loop control by Homofaciens.

FAQs:

1. How to establish connections?
2. Which referential actions to be used while creating tables?

Oral/Review Questions:

1. What is the Raspberry-pi?
2. Explain the need of Raspbeery-pi/ Beagle board.

ASSIGNMENT NUMBER: A2**Revised On: 16-12-2019**

TITLE	Study of different operating systems for Raspberry-Pi /Beagle board.
PROBLEM STATEMENT /DEFINITION	Study of different operating systems for Raspberry-Pi /Beagle board. Understanding the process of OS installation on Raspberry-Pi /Beagle board
OBJECTIVE	<ul style="list-style-type: none">• To Understand the different operating system for raspberry-pi/ Beagle board.• To Understand the process of installation on raspberry-pi
S/W PACKAGES AND HARDWARE APPARATUS USED	Raspberry-Pi /Beagle board. PC with the configuration as Latest Version of 64 bit Operating Systems,Open Source Fedora-GHz. 8 G.B. RAM, 500 G.B. HDD, 15"Color Monitor, Keyboard, Mouse
REFERENCES	1. Nitesh Dhanjani, Abusing the Internet of Things, O'REILLY, ISBN: 13:978-93-5313-217-1 2. Cuno Pfister, Getting Started with the Internet of Things, O'REILLY, ISBN: 13:978-93-53023-413-6
STEPS	Refer to details
INSTRUCTIONS FOR WRITING JOURNAL	<ul style="list-style-type: none">• Title• Problem Definition• Learning Objectives• Theory• Class Diagram/ER Diagram• Test cases• Program Listing• Output• Conclusion

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

Pre-requisite:

Basic knowledge of Raspberry-pi/Beagle board and OS installation.

Learning Objectives:

- To understand & implement the process of OS installation on Raspberry-Pi /Beagle board.

Learning Outcomes:

The students will be able to

- Install different operating system on Raspberry-pi

Theory:

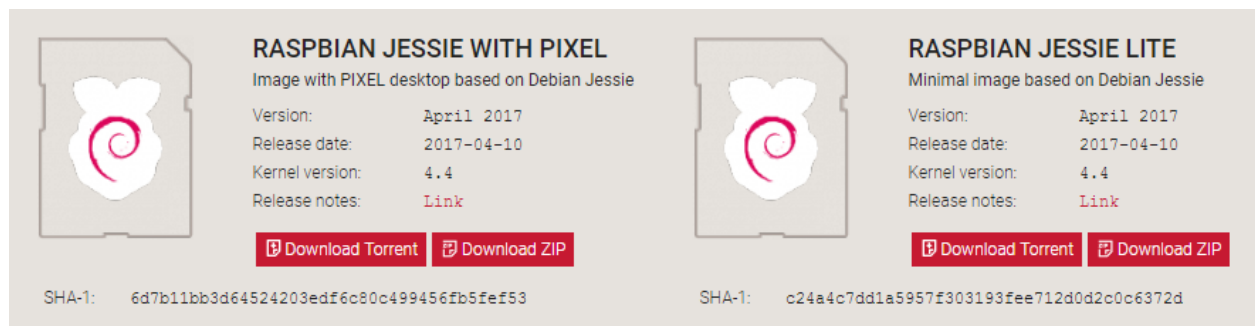
Different operating systems for Raspberry-Pi :

1. Raspbian
2. Ubuntu MATE
3. Snappy Ubuntu
4. Pidora
5. Linutop
6. SARPi
7. Arch Linux ARM
8. Gentoo Linux
9. FreeBSD
10. Kali Linux
11. RISC OS Pi

Raspbian:

Installing Raspbian on the Raspberry Pi is pretty straightforward. We'll be downloading Raspbian and writing the disc image to a microSD card, then booting the Raspberry Pi to that microSD card. For this project, you'll need a microSD card (go with at least 8 GB), a computer with a slot for it, and, of course, a Raspberry Pi and basic peripherals (a mouse, keyboard, screen, and power source). This isn't the only method for installing Raspbian (more on that in a moment), but it's a useful technique to learn because it can also be used to

install so many other operating systems on the Raspberry Pi. Once you know how to write a disc image to a microSD card, you open up a lot of options for fun Raspberry Pi projects.p 1: Download Raspbian

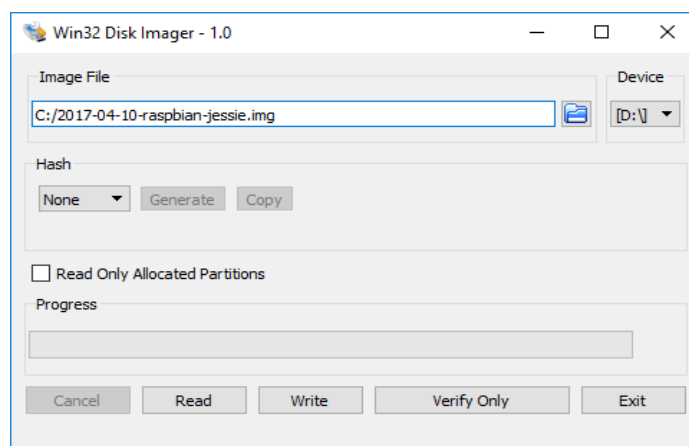


I promised to show you how to install Raspbian on the Raspberry Pi, so it's about time that we got started! First things first: hop onto your computer (Mac and PC are both fine) and download the Raspbian disc image. You can find the latest version of Raspbian <https://www.raspberrypi.org/downloads/raspbian/> Give yourself some time for this, especially if you plan to use the traditional download option rather than the torrent. It can easily take a half hour or more to download.

Step 2: Unzip the file

The Raspbian disc image is compressed, so you'll need to unzip it. The file uses the ZIP64 format, so depending on how current your built-in utilities are, you need to use certain programs to unzip it. If you have any trouble, try these programs recommended by the Raspberry Pi Foundation:

- Windows users, you'll want 7-Zip.
- Mac users, The Unarchiver is your best bet.
- Linux users will use the appropriately named Unzip. Step 3: Write the disc image to your microSD card.



Next, pop your microSD card into your computer and write the disc image to it. You'll need a specific program to do this:

- Windows users, your answer is Win32 Disk Imager.

- Mac users, you can use the disk utility that's already on your machine.
- Linux people, Etcher – which also works on Mac and Windows – is what the Raspberry Pi Foundation recommends.

The process of actually writing the image will be slightly different across these programs, but it's pretty self-explanatory no matter what you're using. Each of these programs will have you select the destination (make sure you've picked your microSD card!) and the disc image (the unzipped Raspbian file). Choose, double-check, and then hit the button to write.

Step 4: Put the microSD card in your Pi and boot up

Once the disc image has been written to the microSD card, you're ready to go! Put that sucker into your Raspberry Pi, plug in the peripherals and power source, and enjoy. The current edition to Raspbian will boot directly to the desktop. Your default credentials are username **pi** and password **raspberry**

Different operating systems for Beagle board:

1 Linux based

- 1.1 Android
- 1.2 Angstrom
- 1.3 Debian
- 1.4 Fedora
- 1.5 ArchLinux
- 1.6 Buildroot
- 1.7 Gentoo
- 1.8 Nerves Erlang/OTP
- 1.9 Sabayon
- 1.10 Ubuntu
- 1.11 Yocto

2 . Other / non-Linux

- 2.1 MINIX 3
- 2.2 Windows Embedded Compact 7
- 2.3 Windows CE 6.0
- 2.4 Windows Embedded Compact 2013

Debian:

FAQs:

1.How to install different OS with raspberr-pi/Beagle board

Oral/Review Questions:

1. List different OS for Raspberr-pi/ Beagle board

ASSIGNMENT NUMBER: A3**Revised On: 16-12-2019**

TITLE	Connectivity and configuration of Raspberry-
PROBLEM STATEMENT /DEFINITION	Study of Connectivity and configuration of Raspberry-Pi /Beagle board circuit with basic peripherals, LEDS. Understanding GPIO and its use in program.
OBJECTIVE	<ul style="list-style-type: none">• Understand the connectivity and configuration of Raspberry pi/Beagle board
S/W PACKAGES AND HARDWARE APPARATUS USED	Raspberry-pi/ Beagle board PC with the configuration as Latest Version of 64 bit Operating Systems,Open Source Fedora-GHz. 8 G.B. RAM, 500 G.B. HDD, 15"Color Monitor, Keyboard, Mouse
REFERENCES	1. Derek Molley, Exploring Beaglebonell, Willey, ISBN: 978-1-118-9351259. 2. Matt Richardson and Shawn Wallace, Getting with Raspberry Pi, MAKER MEDIA, ISBN:978-93-5213-450-210. 3. Dr. Simon Monk, —Raspberry PiCook-Bookll, O'REILLY, ISBN: 978-93-5213-389-5
STEPS	Refer to details
INSTRUCTIONS FOR WRITING JOURNAL	<ul style="list-style-type: none">• Title• Problem Definition• Learning Objectives• Theory• Class Diagram/ER Diagram• Test cases• Program Listing• Output• Conclusion

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

Pre-requisite:

Basic knowledge of configuration.

Learning Objectives:

- To understand configuration of Raspberry-pi/Beagle board circuit with basic peripherals and its use in the program.

Learning Outcomes:

The students will be able to

- Connectivity of Raspberry-pi and Implement the program

Theory:

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.
- config.txt
- The Raspberry Pi configuration file.
- Wireless
- Configuring your Pi to connect to a wireless network using the Raspberry Pi 3 and Pi Zero W's inbuilt wireless connectivity, or a USB wireless dongle.
- Wireless Access Point
- Configuring your Pi as a wireless access point using the Raspberry Pi 3 and Pi Zero W's inbuilt wireless connectivity, or a USB wireless dongle.
- Audio Config
- Switch your audio output between HDMI and the 3.5mm jack.
- Camera Config
- Installing and setting up the Raspberry Pi camera board.
- External Storage Config
- Mounting and setting up external storage on a Raspberry Pi.
- Localisation

- Setting up your Pi to work in your local language/timezone.
- Default pin configuration
- Changing the default pin states.
- Device Trees Config
- Device Trees, overlays, and parameters.
- Kernel Command line
- How to set options in the kernel command line.
- UART configuration
- How to set up the on-board UARTS.
- Firmware Warning Icons
- Description of warning icons displayed if the firmware detects issues.

raspi-config :

raspi-config is the Raspberry Pi configuration tool written and maintained by [Alex Bradbury](#). It targets Raspbian.

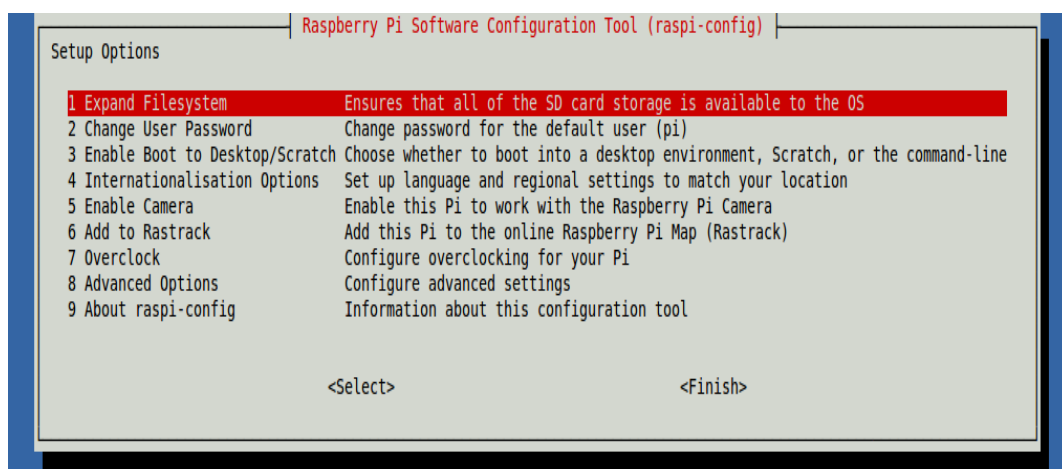
Usage

You will be shown raspi-config on first booting into Raspbian. To open the configuration tool after this, simply run the following from the command line:

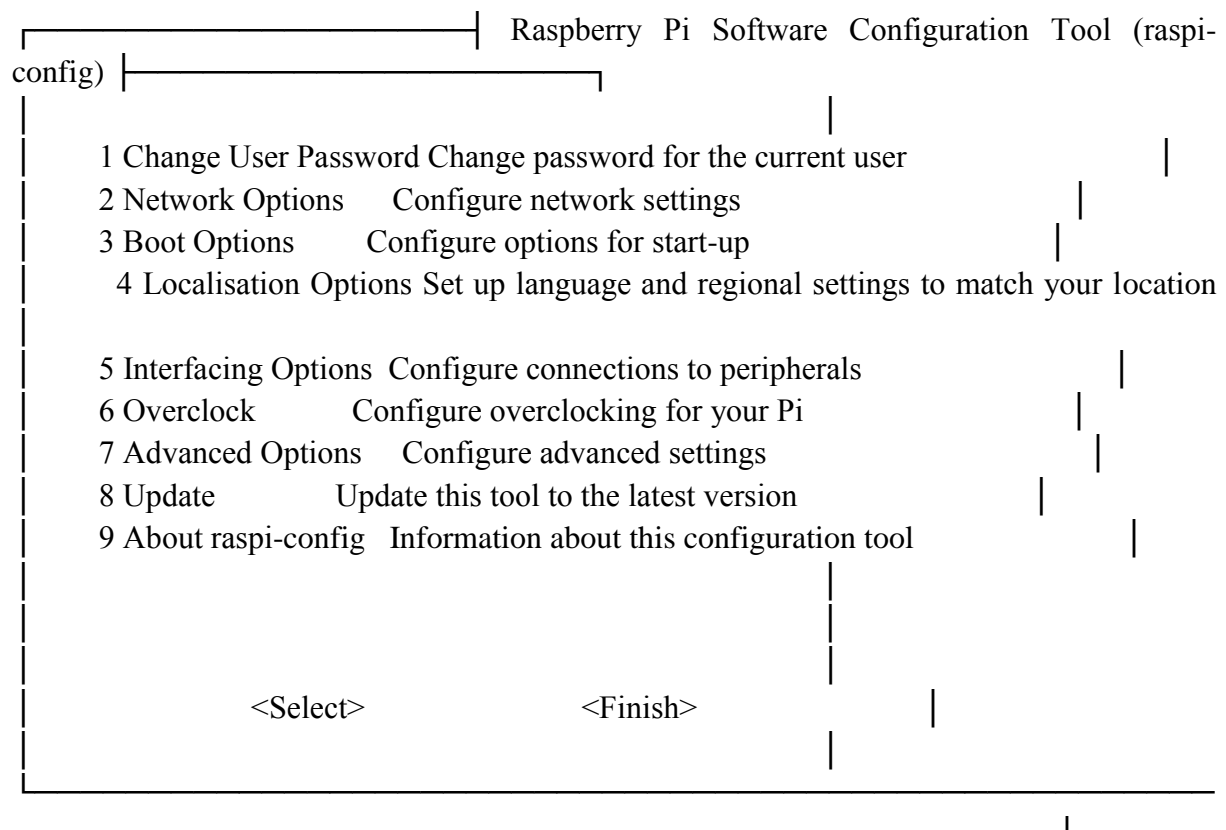
```
sudo raspi-config
```

The sudo is required because you will be changing files that you do not own as the pi user.

You should see a blue screen with options in a grey box in the centre, like so:



It has the following options available:



Moving around the menu

Use the up and down arrow keys to move the highlighted selection between the options available. Pressing the right arrow key will jump out of the Options menu and take you to the <Select> and <Finish> buttons. Pressing left will take you back to the options. Alternatively, you can use the Tab key to switch between these.

Note that in long lists of option values (like the list of timezone cities), you can also type a letter to skip to that section of the list. For example, entering L will skip you to Lisbon, just two options away from London, to save you scrolling all the way through the alphabet.

What raspi-config does

Generally speaking, raspi-config aims to provide the functionality to make the most common configuration changes. This may result in automated edits to /boot/config.txt and various standard Linux configuration files. Some options require a reboot to take effect. If you changed any of those, raspi-config will ask if you wish to reboot now when you select the <Finish> button.

Menu options

Change User Password

The default user on Raspbian is pi with the password raspberry. You can change that here. Read about other [users](#).

Network Options

From this submenu you can set the host name, your WiFi SSID, and pre-shared key, or enable/disable predictable network interface names.

Hostname

Set the visible name for this Pi on a network.

Boot Options

From here you can change what happens when your Pi boots. Use this option to change your boot preference to command line or desktop. You can choose whether boot-up waits for the network to be available, and whether the Plymouth splash screen is displayed at boot-up.

Localisation Options

The localisation submenu gives you these options to choose from: keyboard layout, time zone, locale, and WiFi country code. All options on these menus default to British or GB until you change them.

Change locale

Select a locale, for example en_GB.UTF-8 UTF-8.

Change time zone

Select your local time zone, starting with the region, e.g. Europe, then selecting a city, e.g. London. Type a letter to skip down the list to that point in the alphabet.

Change keyboard layout

This option opens another menu which allows you to select your keyboard layout. It will take a long time to display while it reads all the keyboard types. Changes usually take effect immediately, but may require a reboot.

Change WiFi Country

This option sets the country code for your WiFi network.

Interfacing Options

In this submenu there are the following options to enable/disable: Camera, SSH, VNC, SPI, I2C, Serial, 1-wire, and Remote GPIO.

Camera

Enable/disable the CSI camera interface.

SSH

Enable/disable remote command line access to your Pi using SSH.

P:F-LTL-UG/03/R1

SSH allows you to remotely access the command line of the Raspberry Pi from another computer. SSH is disabled by default. Read more about using SSH on the [SSH documentation page](#). If connecting your Pi directly to a public network, you should not enable SSH unless you have set up secure passwords for all users.

VNC

Enable/disable the RealVNC virtual network computing server.

SPI

Enable/disable SPI interfaces and automatic loading of the SPI kernel module, needed for products such as PiFace.

I2C

Enable/disable I2C interfaces and automatic loading of the I2C kernel module.

Serial

Enable/disable shell and kernel messages on the serial connection.

1-wire

Enable/disable the Dallas 1-wire interface. This is usually used for DS18B20 temperature sensors.

Overclock

It is possible to overclock your Raspberry Pi's CPU. The default is 700MHz but it can be set up to 1000MHz. The overclocking you can achieve will vary; overclocking too high may result in instability. Selecting this option shows the following warning:

Be aware that overclocking may reduce the lifetime of your Raspberry Pi. If overclocking at a certain level causes system instability, try a more modest overclock. Hold down the Shift key during boot to temporarily disable overclocking.

See http://elinux.org/RPi_Overclocking for more information.

Advanced Options

Expand Filesystem

If you have installed Raspbian using NOOBS, the filesystem will have been expanded automatically. There may be a rare occasion where this is not the case, e.g. if you have copied a smaller SD card onto a larger one. In this case, you should use this option to expand your installation to fill the whole SD card, giving you more space to use for files. You will need to reboot the Raspberry Pi to make this available. Note that there is no confirmation: selecting the option begins the partition expansion immediately. Overscan Old TV sets had a significant variation in the size of the picture they produced; some had cabinets that overlapped the screen. TV pictures were therefore given a black border so that none of the

picture was lost; this is called overscan. Modern TVs and monitors don't need the border, and the signal doesn't allow for it. If the initial text shown on the screen disappears off the edge, you need to enable overscan to bring the border back.

Any changes will take effect after a reboot. You can have greater control over the settings by editing [config.txt](#).

On some displays, particularly monitors, disabling overscan will make the picture fill the whole screen and correct the resolution. For other displays, it may be necessary to leave overscan enabled and adjust its values.

Memory split

Change the amount of memory made available to the GPU.

Audio

Force audio out through HDMI or a 3.5mm jack. Read more on the [audio configuration documentation page](#).

Resolution

Define the default HDMI/DVI video resolution to use when the system boots without a TV or monitor being connected. This can have an effect on RealVNC if the VNC option is enabled.

Pixel Doubling

Enable/disable 2x2 pixel mapping.

GL Driver

Enable/disable the experimental GL desktop graphics drivers.

GL (Full KMS)

Enable/disable the experimental OpenGL Full KMS (kernel mode setting) desktop graphics driver.

GL (Fake KMS)

Enable/disable the experimental OpenGL Fake KMS desktop graphics driver.

Legacy

Enable/disable the original legacy non-GL videocore desktop graphics driver.

Update

Update this tool to the latest version.

About raspi-config

Selecting this option shows the following text:

This tool provides a straightforward way of doing initial configuration of the Raspberry Pi. Although it can be run at any time, some of the options may have difficulties if you have heavily customised your installation.

FAQs:

1. Explain connectivity and configuration of Raspberry-pi/Beagle board with basic peripheral?

Oral/Review Questions:

ASSIGNMENT NUMBER: A4**Revised On: 16-12-2019**

TITLE	Connectivity of Raspberry Pi /Beagle board circuit with temperature sensor.
PROBLEM STATEMENT /DEFINITION	Write an application to read the environment temperature. If temperature crosses a threshold value, the application indicated user using LEDSs
OBJECTIVE	Understanding the connectivity of Raspberry Pi /Beagle board circuit with temperature sensor.
S/W PACKAGES AND HARDWARE APPARATUS USED	Raspberry pi board/ BBB , DTH-11 temperature sensor, LED Raspbian (OS), Adafruit_DTH Library
REFERENCES	1. https://www.raspberrypi.org/ 2. https://www.bbb.org/ 3. https://www.adafruit.com/
STEPS	Refer to details
INSTRUCTIONS FOR WRITING JOURNAL	<ul style="list-style-type: none">• Title• Problem Definition• Objectives• Theory• Interfacing Diagram• Algorithm• Test cases• Program Listing(soft copy)• Output• Conclusion

Aim: Connectivity of Raspberry Pi /Beagle board circuit with temperature sensor to read the environment temperature. If temperature crosses a threshold value, the application indicated user using LEDSs

Pre-requisite:

Basic knowledge of GPIO of Raspberry pi/BBB

Basic knowledge of Python programming.

Working and connections of sensors.

Learning Objectives:

- **Understanding the connectivity of Raspberry Pi /Beagle board circuit with temperature sensor.**

Learning Outcomes:

The students will be able to

- To interface temperature sensor to Raspberry pi.
- Read and analyze temperature values.
- Can perform actuation.

Theory:

Introduction:

The Raspberry Pi is a series of credit card-sized 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. It is a capable little computer which can be used in electronics projects, and for many of the things that your desktop PC does, like spreadsheets, word processing, browsing the internet, and playing games. It also plays high-definition video.

The Raspberry Pi is open hardware, with the exception of the primary chip on the Raspberry Pi, the Broadcom SoC (System on a Chip), which runs many of the main components of the board—CPU, graphics, memory, the USB controller, etc. Many of the projects made with a Raspberry Pi are open and well-documented as well and are things you can build and modify yourself.

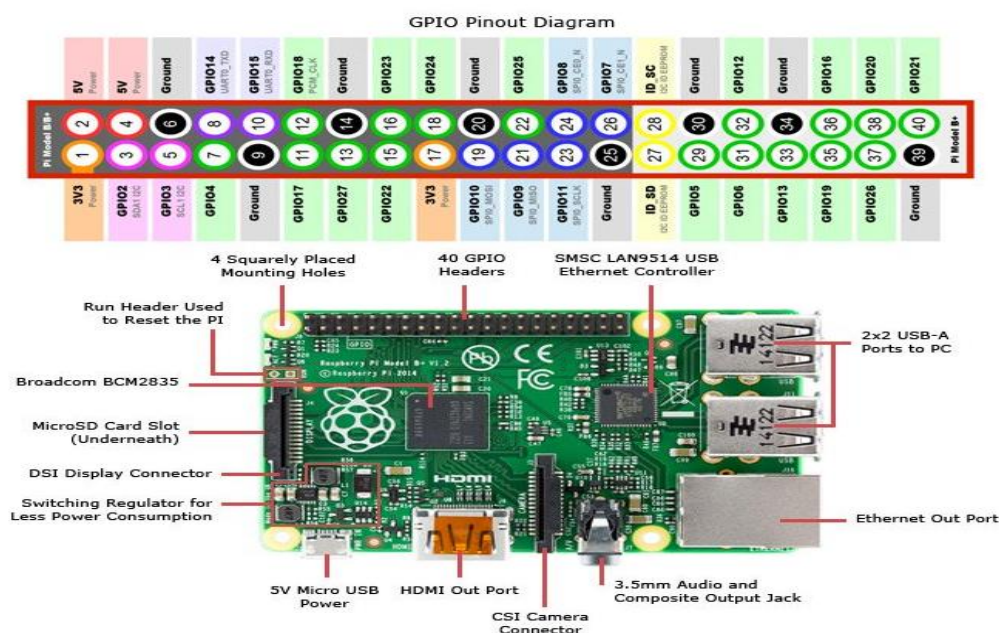
The Raspberry Pi was designed for the Linux operating system, and many Linux distributions now have a version optimized for the Raspberry Pi.

One powerful feature of the Raspberry Pi is the row of GPIO (general purpose input/output) pins along the top edge of the board. These pins are a physical interface between the Pi and the outside world. At the simplest level, you can think of them as switches that you can turn

on or off (input) or that the Pi can turn on or off (output). Of the 40 pins, 26 are GPIO pins and the others are power or ground pins (plus two ID EEPROM pins which you should not play with unless you know your stuff!)

You can program the pins to interact in amazing ways with the real world. Inputs don't have to come from a physical switch; it could be input from a sensor or a signal from another computer or device, for example. The output can also do anything, from turning on an LED to sending a signal or data to another device. If the Raspberry Pi is on a network, you can control devices that are attached to it from anywhere and those devices can send data back. Connectivity and control of physical devices over the internet is a powerful and exciting thing, and the Raspberry Pi is ideal for this.

Raspberry Pi Board with GPIO:



Technical Specification:

- Broadcom BCM2835
- 1GB RAM
- 37
- 64bit
- AR
- Mv7

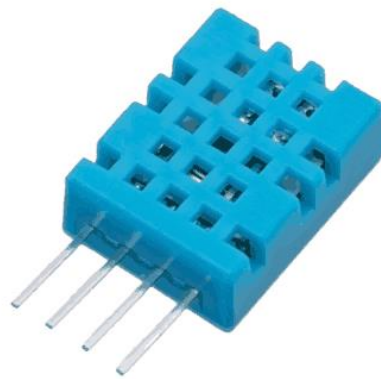
Quad Core Processor powered Single Board

- Computer running at 1.2GHz
- 1GB RAM
- BCM43143 WiFi on board
- Bluetooth Low Energy (BLE) on board
- 40pin extended GPIO
- 4 x USB 2 ports
- 4 pole Stereo output and Composite video port
- Full size HDMI

- CSI camera port for connecting the Raspberry Pi camera
- DSI display port for connecting the Raspberry Pi touch screen display
- Micro SD port for loading your operating system and storing data
- Upgraded switched Micro USB power source (now supports up to 2.4 Amps)
- Expected to have the same form factor as the Pi 2 Model B, however the LEDs will
Change position

DHT11 Sensor

Introduction



DHT11 Sensor

DHT11 is a single wire digital humidity and temperature sensor, which provides humidity and temperature values serially.

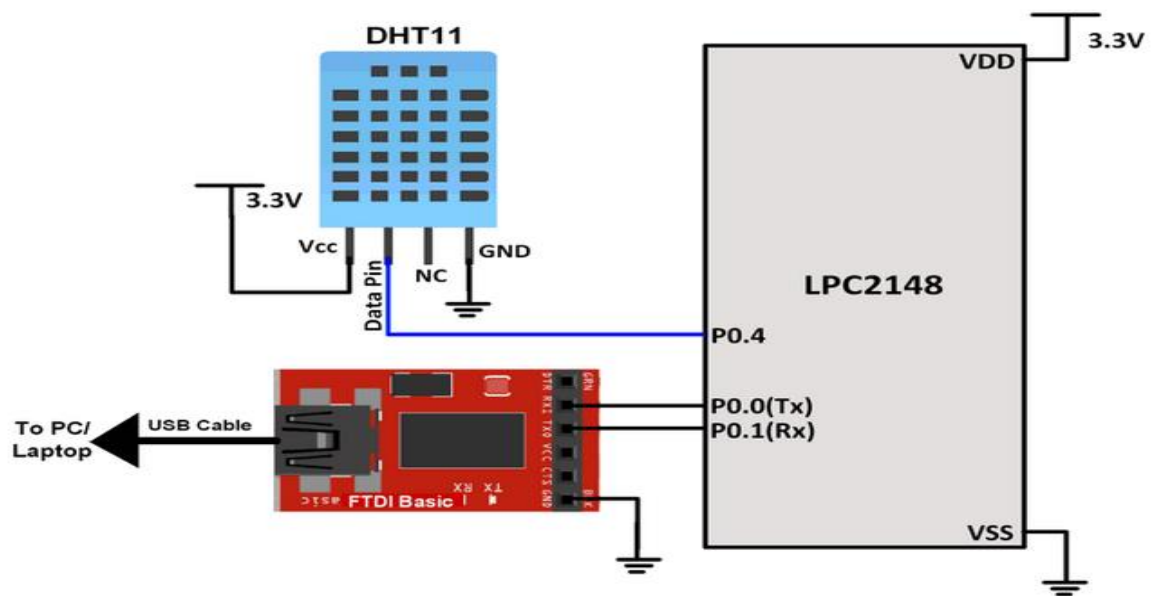
It can measure relative humidity in percentage (20 to 90% RH) and temperature in degree Celsius in the range of 0 to 50°C.

It has 4 pins of which 2 pins are used for supply, 1 is not used and the last one is used for data.

The data pin is the only pin used for communication. Pulses of different TON and TOFF are decoded as logic 1 or logic 0 or start pulse or end of the frame.

For more information about DHT11 sensor and how to use it, refer the topic [DHT11 sensor](#) in the sensors and modules topic.

Interfacing Diagram



Interfacing DHT11 Sensor with LPC2148

ASSIGNMENT NUMBER: B1**Revised On: 16-12-2019**

TITLE	Connectivity of Raspberry Pi /Beagle board circuit with InfraRed(IR) sensor.
PROBLEM STATEMENT /DEFINITION	Understanding the connectivity of Raspberry-Pi /Beagle board circuit with IR sensor. Write an application to detect obstacle and notify user using LEDs.
OBJECTIVE	Understanding the connectivity of Raspberry Pi /Beagle board circuit with IR sensor.
S/W PACKAGES AND HARDWARE APPARATUS USED	Raspberry pi board/ BBB , IR sensor, LED Raspbian (OS),
REFERENCES	1. https://www.raspberrypi.org/ 2. https://www.bbb.org/
STEPS	Refer to details
INSTRUCTIONS FOR WRITING JOURNAL	<ul style="list-style-type: none">• Title• Problem Definition• Objectives• Theory• Interfacing Diagram• Algorithm• Test cases• Program Listing(soft copy)• Output• Conclusion

Aim: Connectivity of Raspberry Pi /Beagle board circuit with IR sensor to detect obstacle and notify user using LEDs.

Pre-requisite:

Basic knowledge of GPIO of Raspberry pi/BBB

Basic knowledge of Python programming.

Working and connections of sensors.

Learning Objectives:

- Understanding the connectivity of Raspberry Pi /Beagle board circuit with IR sensor.

Learning Outcomes:

The students will be able to

- To interface IR sensor to Raspberry pi.
- Detect the obstacle with IR sensor.
- Can perform actuation.

H/W AND S/W Requirements :

Raspberry Pi/Beagle board Development Boards

PC / Monitor/Keyboard

IR (Infrared) Sensor, 1 LED, 1 Resistor (330 Ω)

Few jumper cables,1 Breadboard

Raspbian (OS), Debian LINUX and Python

Theory:

Introduction:

The Raspberry Pi is a series of credit card-sized 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. It is a capable little computer which can be used in electronics projects, and for many of the things that your desktop PC does, like spreadsheets, word processing, browsing the internet, and playing games. It also plays high-definition video.

The Raspberry Pi is open hardware, with the exception of the primary chip on the Raspberry Pi, the Broadcom SoC (System on a Chip), which runs many of the main components of the board—CPU, graphics, memory, the USB controller, etc. Many of the projects made with a

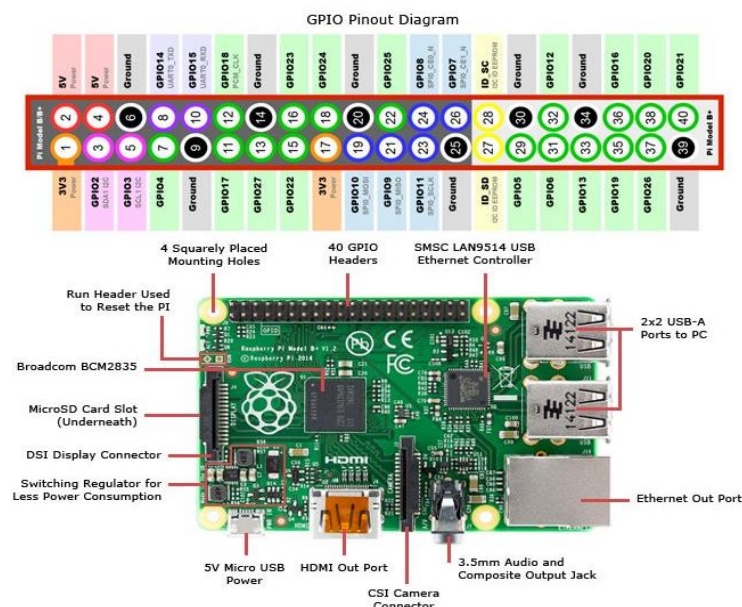
Raspberry Pi are open and well-documented as well and are things you can build and modify yourself.

The Raspberry Pi was designed for the Linux operating system, and many Linux distributions now have a version optimized for the Raspberry Pi.

One powerful feature of the Raspberry Pi is the row of GPIO (general purpose input/output) pins along the top edge of the board. These pins are a physical interface between the Pi and the outside world. At the simplest level, you can think of them as switches that you can turn on or off (input) or that the Pi can turn on or off (output). Of the 40 pins, 26 are GPIO pins and the others are power or ground pins (plus two ID EEPROM pins which you should not play with unless you know your stuff!)

You can program the pins to interact in amazing ways with the real world. Inputs don't have to come from a physical switch; it could be input from a sensor or a signal from another computer or device, for example. The output can also do anything, from turning on an LED to sending a signal or data to another device. If the Raspberry Pi is on a network, you can control devices that are attached to it from anywhere and those devices can send data back. Connectivity and control of physical devices over the internet is a powerful and exciting thing, and the Raspberry Pi is ideal for this.

Raspberry Pi Board with GPIO:



Technical Specification:

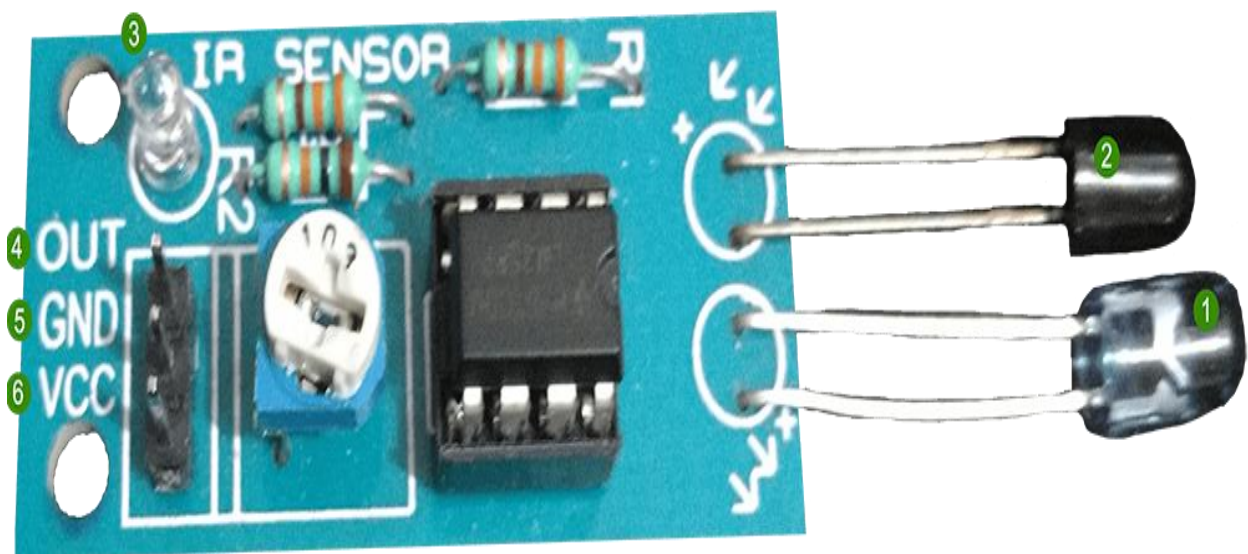
- Broadcom BCM2837 64bit ARMv7 Quad Core Processor powered Single Board
- Computer running at 1.2GHz

- 1GB RAM
- BCM43143 WiFi on board
- Bluetooth Low Energy (BLE) on board
- 40pin extended GPIO
- 4 x USB 2 ports
- 4 pole Stereo output and Composite video port
- Full size HDMI
- CSI camera port for connecting the Raspberry Pi camera
- DSI display port for connecting the Raspberry Pi touch screen display
- Micro SD port for loading your operating system and storing data
- Upgraded switched Micro USB power source (now supports up to 2.4 Amps)
- Expected to have the same form factor has the Pi 2 Model B, however the LEDs will

Change position

InfraRed (IR) 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, if comes back by bouncing on any obstacle in the way.



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

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.

Connecting IR Sensor

1. Connect GPIO 17 from the Raspberry Pi to Breadboard
2. Connect OUT pin of the sensor with the Breadboard
This will send input received from sensor to GPIO 17, which will be processed further.
3. Connect GND (any pin from board will work) with negative line on left side of the breadboard
4. Connect GND of the IR Sensor to Breadboard
5. Connect GND from Step 3 to Breadboard
6. Connect VCC of the IR Sensor to Breadboard
7. Connect 3v3 (Pin #1) to positive line on left side of the breadboard
8. Connect 3v3 (connected in Step 7) to the Breadboard

Connecting LED

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

1. Connect GPIO 4 from the board to the Breadboard
2. Connect positive point of the LED (longer pin of the LED) to the Breadboard
3. Connect negative point of the LED (smaller pin of the LED) to the Breadboard
4. Use resistor (330 Ω) to connect negative to the negative point of the LED

Python code to detect obstacle

```
from gpiozero import LED
from signal import pause
import RPi.GPIO as GPIO
import time
```

```
GPIO.setmode(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)
```

FAQs:

1. What is IR sensor?
2. How it is interfaced with Raspberry pi/Beaglebone ?

ASSIGNMENT NUMBER: B2**Revised On: 16-12-2019**

TITLE	Write an application to capture and store the image.
PROBLEM STATEMENT /DEFINITION	Understanding and connectivity of Raspberry-Pi /Beagle board with camera. Write an application to capture and store the image.
OBJECTIVE	To capture and store image using Raspberry-pi.
S/W PACKAGES AND HARDWARE APPARATUS USED	Picamera package Raspberry pi Camera module
REFERENCES	http://www.electronicwings.com/raspberry-pi/pi-camera-module-interface-with-raspberry-pi-using-python
STEPS	Refer to details
INSTRUCTIONS FOR WRITING JOURNAL	<ul style="list-style-type: none">• Title• Problem Definition• Objectives• Theory• Class Diagram/ER diagram• Test cases• Program Listing• Output• Conclusion

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

Pre-requisite:

Basic knowledge of configuration.

Learning Objectives:

- To understand configuration of Raspberry-pi/Beagle board circuit with basic peripherals and its use in the program.

Learning Outcomes:

The students will be able to

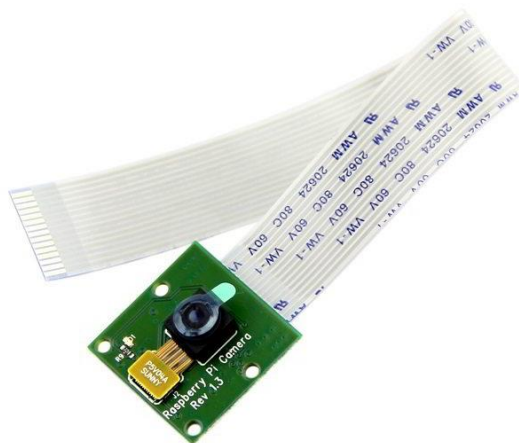
- Connectivity of Raspberry-pi and Implement the program

Theory:

Pi Camera Module Interface with Raspberry Pi using Python

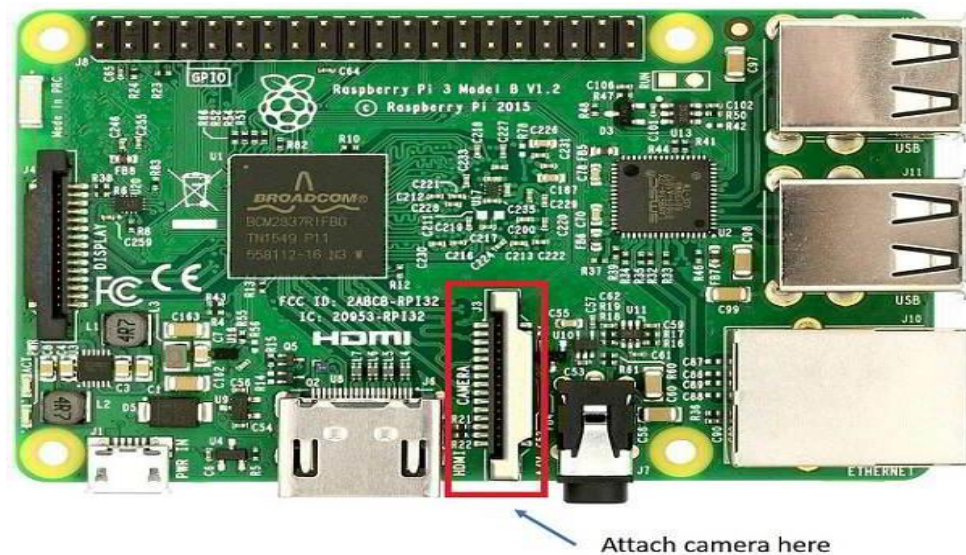
Introduction

Pi Camera module is a camera which can be used to take pictures and high definition video. Raspberry Pi Board has CSI (Camera Serial Interface) interface to which we can attach PiCamera module directly. This Pi Camera module can attach to the Raspberry Pi's CSI port using 15-pin ribbon cable.



How to attach Pi Camera to Raspberry Pi?

Connect Pi Camera to CSI interface of Raspberry Pi board as shown below,



Now, we can use Pi Camera for capturing images and videos using Raspberry Pi.

Python Program for Image Capture & Store

```
import picamera
from time import sleep

#create object for PiCamera class
camera = picamera.PiCamera()
#set resolution
camera.resolution = (1024, 768)
camera.brightness = 60
```

P:F-LTL-UG/03/R1

```
camera.start_preview()
#add text on image
camera.annotate_text = 'Hi Pi User'
sleep(5)
#store image
camera.capture('image1.jpeg')
camera.stop_preview()
```

ASSIGNMENT NUMBER: B3**Revised On: 16-12-2019**

TITLE	Write a network application for communication between two devices using Zigbee
PROBLEM STATEMENT /DEFINITION	Understanding and connectivity of Raspberry-Pi /Beagle board with a Zigbee module. Write a network application for communication between two devices using Zigbee.
OBJECTIVE	<ul style="list-style-type: none">• To understand functionalities of various single board embedded platforms fundamentals.• Develop application for Communication between more raspberrypi hardware.
S/W PACKAGES AND HARDWARE APPARATUS USED	Python, two Raspberrypi devices, Zigbee device, Latest Version of 64 bit Operating Systems, Open Source Fedora-GHz. 8 G.B. RAM, 500 G.B. HDD, 15"Color Monitor, Keyboard, Mouse
REFERENCES	<ul style="list-style-type: none">• David Easley and Jon Kleinberg, "Networks, Crowds, and Markets: Reasoning About a Highly Connected World", Cambridge University Press, 2010, ISBN:10: 0521195330• Olivier Hersent, Omar Elloumi and David Boswarthick, "The Internet of Things: Applications to the Smart Grid and Building Automation", Wiley, 2012, 9781119958345
STEPS	Refer to details
INSTRUCTIONS FOR WRITING JOURNAL	<ul style="list-style-type: none">• Title• Problem Definition• Objectives• Theory• Class Diagram/ER diagram• Test cases• Program Listing• Output• Conclusion

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

Pre-requisite:

Basic knowledge of Embedded system and IOT

Learning Objectives:

- To Develop application for Communication between more raspberrypi hardware.

Learning Outcomes:

The students will be able to

- Perform the connectivity with Raspberry-Pi, Beagle board, Arduino and other micro controller
- Implement transmitter and receiver program using python by using zigbee device

Theory:

ZigBee Communication Using Raspberry Pi

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.

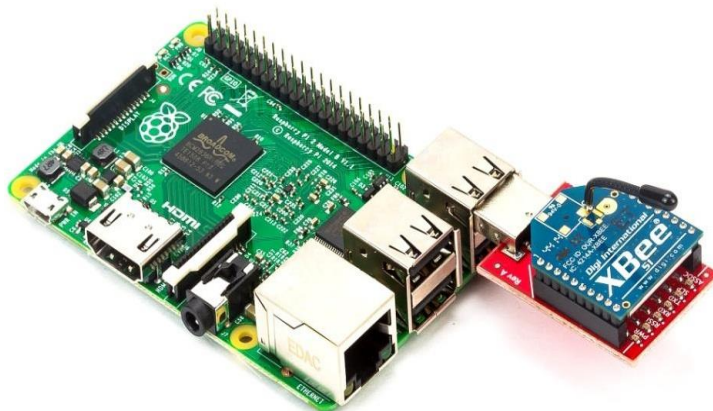


Fig:- Zigbee

Zigbee Technology

Zigbee communication is specially built for control and sensor networks on IEEE 802.15.4 standard for wireless personal area networks (WPANs), and it is the product from Zigbee alliance. This communication standard defines physical and Media Access Control (MAC) layers to handle many devices at low-data rates. These Zigbee's WPANs operate at 868 MHz, 902-928MHz and 2.4 GHz frequencies. The data rate of 250 kbps is best suited for periodic as well as intermediate two way transmission of data between sensors and controllers.

Zigbee is low-cost and low-powered mesh network widely deployed for controlling and monitoring applications where it covers 10-100 meters within the range. This communication system is less expensive and simpler than the other proprietary short-range wireless sensor networks as Bluetooth and Wi-Fi. Zigbee supports different network configurations for master to master or master to slave communications. And also, it can be operated in different modes as a result the battery power is conserved. Zigbee networks are extendable with the use of routers and allow many nodes to interconnect with each other for building a wider area network

Zigbee Architecture

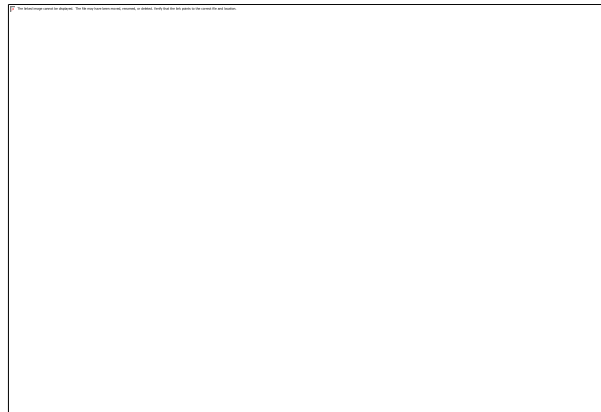


Fig:- Zigbee architecture

Zigbee system structure consists of three different types of devices such as Zigbee coordinator, Router and End device. Every Zigbee network must consist of at least one coordinator which acts as a root and bridge of the network. The coordinator is responsible for handling and storing the information while performing receiving and transmitting data operations. Zigbee routers act as intermediary devices that permit data to pass to and from through them to other devices. End devices have limited functionality to communicate with the parent nodes such that the battery power is saved as shown in the figure. The number of routers, coordinators and end devices depends on the type of network such as star, tree and mesh networks. Zigbee protocol architecture consists of a stack of various layers where IEEE 802.15.4 is defined by physical and MAC layers while this protocol is completed by accumulating Zigbee's own network and application layers.

Physical Layer: This layer does modulation and demodulation operations up on transmitting and receiving signals respectively. This layer's frequency, data rate and number of channels are given below.

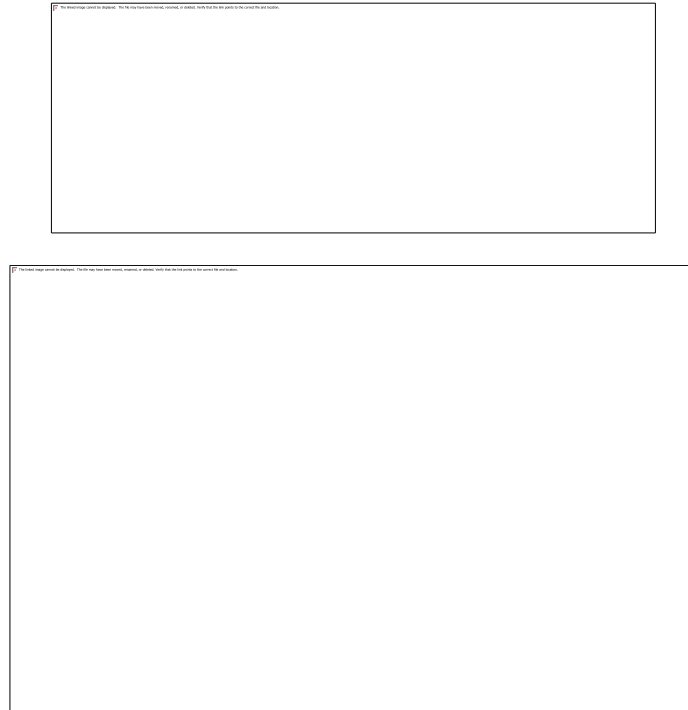


Fig :- Layers

MAC Layer: This layer is responsible for reliable transmission of data by accessing different networks with the carrier sense multiple access collision avoidance (CSMA). This also transmits the beacon frames for synchronizing communication.

Network Layer: This layer takes care of all network related operations such as network setup, end device connection and disconnection to network, routing, device configurations, etc.

Application Support Sub-Layer: This layer enables the services necessary for Zigbee device object and application objects to interface with the network layers for data managing services. This layer is responsible for matching two devices according to their services and needs.

Application Framework: It provides two types of data services as key value pair and generic message services. Generic message is a developer defined structure, whereas the key value pair is used for getting attributes within the application objects. ZDO provides an interface between application objects and APS layer in Zigbee devices. It is responsible for detecting, initiating and binding other devices to the network.

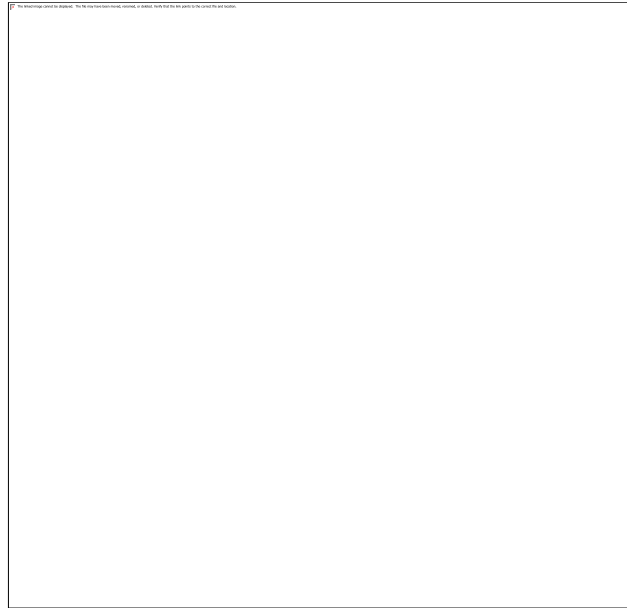


Fig- Zigbee topology

Zigbee two way data is transferred in two modes: Non-beacon mode and Beacon mode. In a beacon mode, the coordinators and routers continuously monitor active state of incoming data hence more power is consumed. In this mode, the routers and coordinators do not sleep because at any time any node can wake up and communicate. However, it requires more power supply and its overall power consumption is low because most of the devices are in an inactive state for over long periods in the network.

In a beacon mode, when there is no data communication from end devices, then the routers and coordinators enter into sleep state. Periodically this coordinator wakes up and transmits the beacons to the routers in the network. These beacon networks are work for time slots which means, they operate when the communication needed results in lower duty cycles and longer battery usage. These beacon and non-beacon modes of Zigbee can manage periodic (sensors data), intermittent (Light switches) and repetitive data types.

Program

Zigbee transmitter code:

```
Import serial
Port=serial.Serial("/dev/ttyUSB0",baudrate=9600,timeout=3.0)
While True:
X=raw_input('pass your data')
Port.write(x)
Rcv=port.read(1)
Print('received data:',rcv)
```

Zigbee receiver code:

```
Import serial
Import time
Port=serial.Serial("/dev/ttyUSB0", baudrate=9600,timeout=0.1)
P:F-LTL-UG/03/R1
```

```
While= True:
#x=raw_input("pass your input")
#port.write(x)
Rev=port.read(1)
Print(rev)
#time.sleep(1)
If(rev!=''):
F=open("log.txt",'a')
f.write(rev)
f.close()
)
```

Conclusion: - Thus the communication between two devices using Zigbee are successfully done.

ASSIGNMENT NUMBER: B4**Revised On: 16-12-2019**

TITLE	Study of different CPU frequency governors.
PROBLEM STATEMENT /DEFINITION	Write an application to change CPU frequency of Raspberry-Pi/BBB
OBJECTIVE	Understanding different CPU frequency governors.
S/W PACKAGES AND HARDWARE APPARATUS USED	Raspberry pi board/ BBB , Beagle Board
REFERENCES	1. https://www.raspberrypi.org/ 2. https://www.bbb.org/
STEPS	Refer to details
INSTRUCTIONS FOR WRITING JOURNAL	<ul style="list-style-type: none">• Title• Problem Definition• Objectives• Theory• Class Diagram/ER diagram• Test cases• Program Listing• Output• Conclusion

Aim: Understanding different CPU frequency governors **to** change CPU frequency of Raspberry-Pi/BBB

Pre-requisite:

Basic knowledge of file system of Raspbian.

Basic knowledge CPU frequency.

Learning Objectives:

- Understanding different CPU frequency governors **to** change CPU frequency of Raspberry-Pi/BBB.

Learning Outcomes:

The students will be able to

- Change CPU frequency as per load requirement.
- Can save power.

Technical Specification:

- Broadcom BCM2837 64bit ARMv7 Quad Core Processor powered Single Board
- Computer running at 1.2GHz
- 1GB RAM
- BCM43143 WiFi on board
- Bluetooth Low Energy (BLE) on board
- 40pin extended GPIO
- 4 x USB 2 ports
- 4 pole Stereo output and Composite video port
- Full size HDMI
- CSI camera port for connecting the Raspberry Pi camera
- DSI display port for connecting the Raspberry Pi touch screen display
- Micro SD port for loading your operating system and storing data
- Upgraded switched Micro USB power source (now supports up to 2.4 Amps)
- Expected to have the same form factor has the Pi 2 Model B, however the LEDs will Change position

Theory:

CPU frequency scaling on raspberry-pi

Linux kernel has driver which manages cpu freq policy.

Following option is available for that. Rasbian set powersave by default.

performance - Always use max cpu freq

powersave - Always use min cpu freq

ondemand - Change cpu freq depending on cpu load (On rasbian, it just switches min and max)

conservative - Smoothly change cpu freq depending on cpu load

userspace - Allow user space daemon to control cpufreq

Raspbian fixes cpu frequency to 700MHz by default. It would not be reasonable to keep max cpu freq for the usage which doesn't require full cpu power at all times.

To check current/min/max cpu freq please use following commands on shell.

- Current cpu freq

```
# cat /sys/devices/system/cpu/cpu0/cpufreq/cpuinfo_cur_freq
```

- Max cpu freq

```
# cat /sys/devices/system/cpu/cpu0/cpufreq/cpuinfo_max_freq
```

- Min cpu freq

```
# cat /sys/devices/system/cpu/cpu0/cpufreq/cpuinfo_min_freq
```

We can change the setting from shell at runtime via sysfs.

```
# echo ondemand > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
```

To keep this setting over reboot, There are 2 options.

1 - Put command in /etc/rc.local.

The command execution is after kernel boot. So, it slows bootup down.

2 - Enable kernel configuration.

Enable either kernel config.

```
CONFIG_CPU_FREQ_DEFAULT_GOV_ONDEMAND  
CONFIG_CPU_FREQ_DEFAULT_GOV_CONSERVATIVE
```

Though second way requires kernel build, we can use full cpu power during boot.

After all, you can see cpu freq change depending on cpu load.

ASSIGNMENT NUMBER: C1**Revised On: 16-12-2019**

TITLE	Controlling the operation of stepper motor using Raspberry Pi /Beagle board circuit.
PROBLEM STATEMENT /DEFINITION	Write an application using Raspberry-Pi /Beagle board to control the operation of stepper motor.
OBJECTIVE	Understanding the connectivity of Raspberry Pi /Beagle board circuit with stepper motor. To understand the actuation.
S/W PACKAGES AND HARDWARE APPARATUS USED	Raspberry pi board/ BBB Stepper motor Raspbian (OS),
REFERENCES	1. https://www.raspberrypi.org/ 2. https://www.bbb.org/
STEPS	Refer to details
INSTRUCTIONS FOR WRITING JOURNAL	<ul style="list-style-type: none">• Title• Problem Definition• Objectives• Theory• Class Diagram/ER diagram• Test cases• Program Listing• Output• Conclusion

Aim: To write an application using Raspberry-Pi /Beagle board to control the operation of stepper motor.

Pre-requisite:

Basic knowledge of GPIO of Raspberry pi/BBB

Basic knowledge of Python programming.

Working and connections of sensors.

Learning Objectives:

- Understanding the connectivity of Raspberry Pi /Beagle board circuit with stepper motor.
- To perform actuation using Raspberry Pi /Beagle.

Learning Outcomes:

The students will be able to

- To interface stepper motor to Raspberry pi/Beagle board.
- To control operation of stepper motor through Raspberry pi/Beagle board.
- Can perform actuation.

H/W AND S/W Requirements :

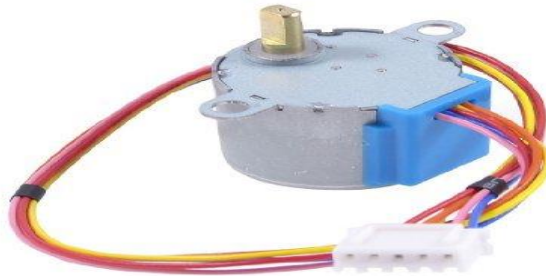
Raspberry Pi/Beagle board Development Boards

PC / Monitor/Keyboard

Raspbian (OS), Debian LINUX and Python

Theory:

A stepper motor is an electromechanical device which converts electrical pulses into discrete mechanical movements. The shaft or spindle of a stepper motor rotates in discrete step increments when electrical command pulses are applied to it in the proper sequence. The motors rotation has several direct relationships to these applied input pulses. The sequence of the applied pulses is directly related to the direction of motor shafts rotation. The speed of the motor shafts rotation is directly related to the frequency of the input pulses and the length of rotation is directly related to the number of input pulses applied. One of the most significant advantages of a stepper motor is its ability to be accurately controlled in an open loop system. Open loop control means no feedback information about position is needed. This type of control eliminates the need for expensive sensing and feedback devices such as optical encoders. Your position is known simply by keeping track of the input step pulses.



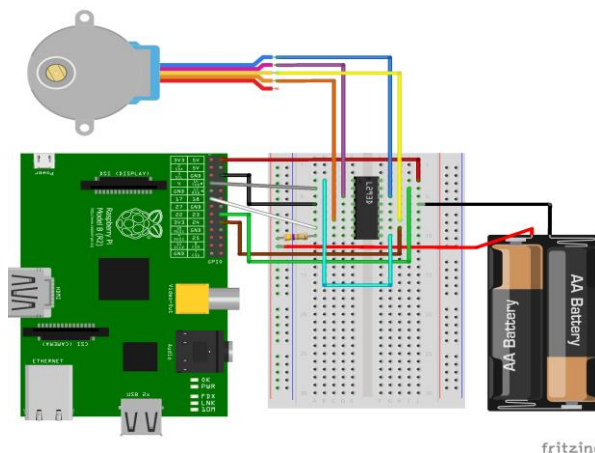
Interfacing with Raspberry Pi

The motor connects to the controller board with a pre-supplied connector. The controller board has 4+2 pins that need to be connected to the Pi header (P1).

- 5V (P1-02)
- GND (P1-06)
- Inp1 (P1-11)
- Inp2 (P1-15)
- Inp3 (P1-16)
- Inp4 (P1-18)

The P1-XX references above represent the Pi header pins used. These are defined in the Python example below in the StepPins list so if you use different pins be sure to update the Python list as well. You can use other GPIO pins if required just remember to update your Python script.

To rotate the stepper motor you provide a sequence of “high” and “low” levels to each of the 4 inputs in sequence. By setting the correct sequence of high and low levels the motor spindle will rotate. The direction can be reversed by reversing the sequence.




```

#!/usr/bin/python

# Import required libraries
import sys
import time
import RPi.GPIO as GPIO

# Use BCM GPIO references
# instead of physical pin numbers
GPIO.setmode(GPIO.BCM)

# Define GPIO signals to use
# Physical pins 11,15,16,18
# GPIO17,GPIO22,GPIO23,GPIO24
StepPins = [17,22,23,24]

# Set all pins as output
for pin in StepPins:
    print "Setup pins"
    GPIO.setup(pin,GPIO.OUT)
    GPIO.output(pin, False)

# Define advanced sequence
# as shown in manufacturers datasheet
Seq = [[1,0,0,1],
[1,0,0,0],
[1,1,0,0],
[0,1,0,0],
[0,1,1,0],
[0,0,1,0],
[0,0,1,1],
[0,0,0,1]]

StepCount = len(Seq)
StepDir = 1 # Set to 1 or 2 for clockwise
# Set to -1 or -2 for anti-clockwise

# Read wait time from command line
if len(sys.argv)>1:
    WaitTime = int(sys.argv[1])/float(1000)
else:
    WaitTime = 10/float(1000)

P:F-LTL-UG/03/R1

```

```

# Initialise variables
StepCounter = 0

# Start main loop
while True:
    print StepCounter,
    print Seq[StepCounter]

    for pin in range(0,4):
        xpin=StepPins[pin]# Get GPIO
        if Seq[StepCounter][pin]!=0:
            print " Enable GPIO %i" %(xpin)
            GPIO.output(xpin, True)
        else:
            GPIO.output(xpin, False)

    StepCounter += StepDir

# If we reach the end of the sequence
# start again
if (StepCounter>=StepCount):
    StepCounter = 0
if (StepCounter<0):
    StepCounter = StepCount+StepDir
# Wait before moving on
time.sleep(WaitTime)

```

FAQs:

1. What is actuation?
2. How stepper motor can be operated using Raspberry Pi/Beaglebone board

ASSIGNMENT NUMBER: C2**Revised On: 16-12-2019**

TITLE	Controlling the operation of a hardware simulated traffic signal using Raspberry Pi /Beagle board circuit.
PROBLEM STATEMENT /DEFINITION	Write an application using Raspberry-Pi /Beagle board to control the operation of a hardware simulated traffic signal.
OBJECTIVE	<ul style="list-style-type: none">• Understanding the controlling of devices through Raspberry Pi /Beagle board.• To understand the actuation.
S/W PACKAGES AND HARDWARE APPARATUS USED	Raspberry pi board/ BBB LED Raspbian (OS),
REFERENCES	1. https://www.raspberrypi.org/ 2. https://www.bbb.org/
STEPS	Refer to details
INSTRUCTIONS FOR WRITING JOURNAL	<ul style="list-style-type: none">• Title• Problem Definition• Objectives• Theory• Class Diagram/ER diagram• Test cases• Program Listing• Output• Conclusion

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

Pre-requisite:

Basic knowledge of GPIO of Raspberry pi/BBB

Basic knowledge of Python programming.

Working and connections of sensors/actuators.

Learning Objectives:

- Understanding the controlling of devices through Raspberry Pi /Beagle board.

Learning Outcomes:

The students will be able to

- To simulate traffic signal through LEDs.
- To control this simulated traffic signal through Raspberry Pi /Beagle board.
- Can perform actuation.

H/W AND S/W Requirements :

Raspberry Pi/Beagle board Development Boards

PC / Monitor/Keyboard

Raspbian (OS), Debian LINUX and Python

Theory:

Beagle Bone black is an open hardware, community-supported embedded computer for developers. It works with 1GHz with the Sitara™ ARM® Cortex-A8 processor. The REV C comes with Debian Linux pre installed. It has GPIO(69 max), which can be programmed using Python.

LEDs are the light emitting Diodes. LEDs have two wires. One wire is the anode (positive) and another is the cathode (negative). Push the LED leads into the breadboard, with the longer (positive) lead towards the top of the breadboard. It does not matter which way around the resistor goes. The top two connections on the BBB expansion header we are using (P8) are both GND. The other lead is connected to pin 10, which is the right-hand connector on the fifth row down.

Steps to do :

1. Connect BBB board to Machine using USB cable.

P:F-LTL-UG/03/R1

2. Install Python library to perform I/O programming on GPIOs
3. Connect Red LED to P8 pin no 10
4. Connect Yellow LED to P8 pin no 12
5. Connect Green LED to P8 pin no 14
6. Reset all 3 LEDs using GPIO.OUT command
7. Turn on Red LED using GPIO.HIGH command
8. Put some delay
9. Turn off Red LED using GPIO.LOW command
10. Repeat steps 7, 8, 9 for Yellow LED
11. Repeat steps 7, 8, 9 for Green LED
12. Go to step 7

Faqs :

Which pins are available on headers p8 and p9?

Which are the registers available with GPIOs?

ASSIGNMENT NUMBER: C3**Revised On: 16-12-2019**

TITLE	Study of lift elevator system.
PROBLEM STATEMENT /DEFINITION	Write an application using Raspberry-Pi /Beagle board to control the operation of a hardware simulated lift elevator
OBJECTIVE	Understanding lift elevator cystem
S/W PACKAGES AND HARDWARE APPARATUS USED	Raspberry pi board/ BBB , Beagle Board
REFERENCES	1. https://www.raspberrypi.org/ 2. https://www.bbb.org/
STEPS	Refer to details
INSTRUCTIONS FOR WRITING JOURNAL	<ul style="list-style-type: none">• Title• Problem Definition• Objectives• Theory• Class Diagram/ER diagram• Test cases• Program Listing• Output• Conclusion

Aim: Write an application using Raspberry-Pi /Beagle board to control the operation of a hardware simulated lift elevator

Pre-requisite:

Basic knowledge of file system of Raspbian.

Basic knowledge CPU frequency.

Learning Objectives:

- Understanding the circuit and connection of lift elevator system

Learning Outcomes:

The students will be able

- To develop lift elevator system

Theory:

Introduction:

Ever wondered how small things in our lives go unnoticed at times. In today's world of skyscrapers, crossing a hundred floors in a matter of few minutes is no big deal, thanks to the Elevators! But no, we will not be talking about how an elevator works as it is primarily guided by principles that are better understood to the mechanical engineers. The whole set-up is rather complex with many control systems and processors forming the core of the elevator scheme. In this discussion, we will deal with the role of electronics in the modern-day elevator system the display control.

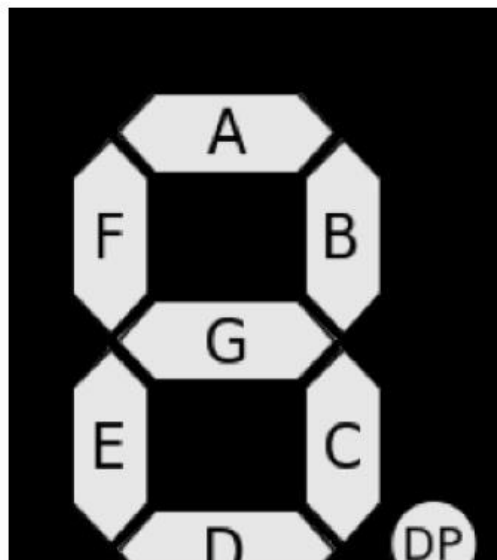
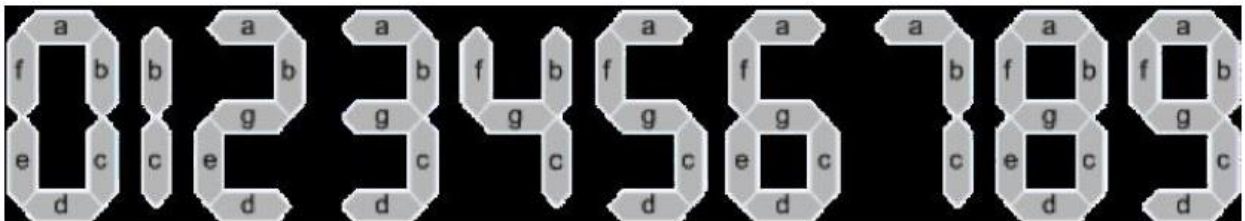


For simplicity, we will only be dealing with the fundamental circuitry or concept underlying the displays employed. The actual connections or circuits may-be larger and heavily complicated. To give you an idea, here is a block diagram you can identify with.



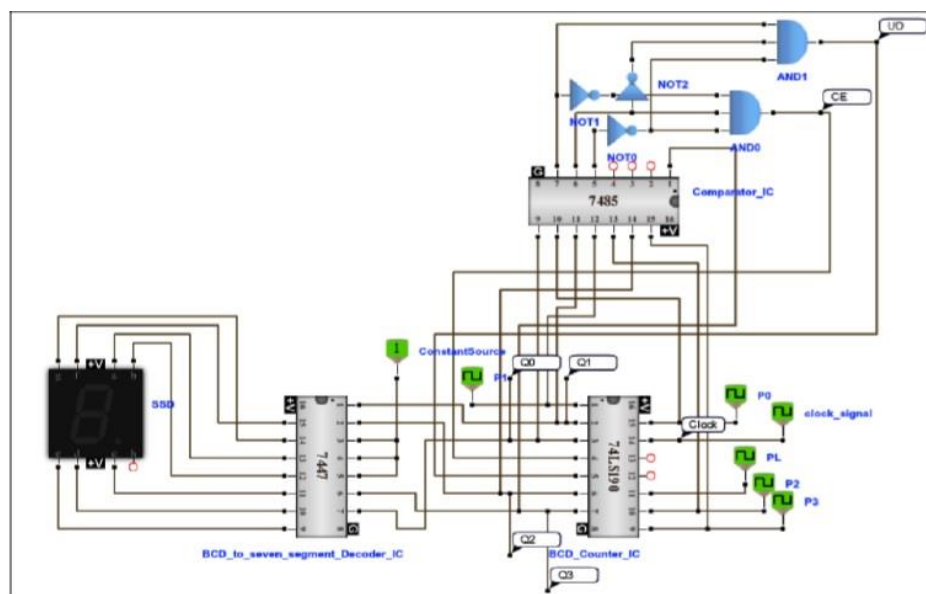
What exactly are we talking about? Here we will dig into the details behind the displays that are synchronized with the lift movement, more precisely, how does the display show "1" when we are currently at the first floor and so forth.

Coming back to the block diagram, let us first define what a keypad matrix is. You can consider it to be the same as the buttons you see in the elevator: 0-1-2-3-4-5-6-7-8-9 and so on. When you press on a particular number, say "1", a signal is sent to the processing unit that identifies the input as "1" and accordingly generates the corresponding output. So this is how we take the input from the user. The next part is to process this unit in a form that is better understood by the processor. For this we employ an encoder that converts "1" into "0001" (binary) and sends it to the next block for further processing. After the processing has been done, it is time to display the data to the user for which a device called the seven segment display or SSD is used. Now, it is the same device that displays time in a digital clock but sadly it cannot be directly connected to the core circuitry. The SSD contains a set of seven LEDs, namely a-b-c-d-e-f-g-h that light up in accordance to the input to show 0-1-2-3-4-5-6-7-8-9. Have a look:



For this the connections are made via a BCD-to-seven segment decoder that decodes the input into a format that is understood by the SSD. The format of the data output coming from

the counter will be in the form of binary coded decimal or BCD. IC 7447 is most commonly used for this decoding purpose. It accepts BCD and subsequently assigns a logic 1 to the LEDs that should be glowing in order to display that particular BCD, for example, if the input BCD is "0001" the decoder output will be a=1,b=0,c=1,d=1,e=1,f=1,g=1. This enables the SSD to deduce that the number 6 is to be displayed. This is how all the digits from 0 to 9 can be displayed using a single SSD. Coming back to the processing block, it can be explained with the help of a situational example: Suppose you enter a multi-storey building and wish to go to the 3rd floor. Your first action will be to press the button at the ground floor and wait for the lift to come down if the lift is not at the ground floor initially. Once you enter the lift, you will press the button "3" and read the display that shows: 0-1-2-3, and voila! After the lift has dropped you at the 3rd floor, it will once again go back to the ground floor i.e. now the display will be something like this: 3-2-1-0. So, how does this counting take place? It's simple, it uses a counter. Are you asking yourself how could it be that easy? Have a look at the circuit designed as:

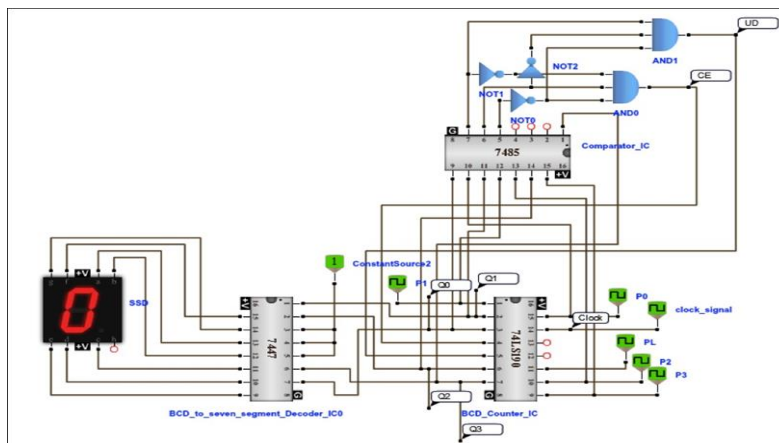


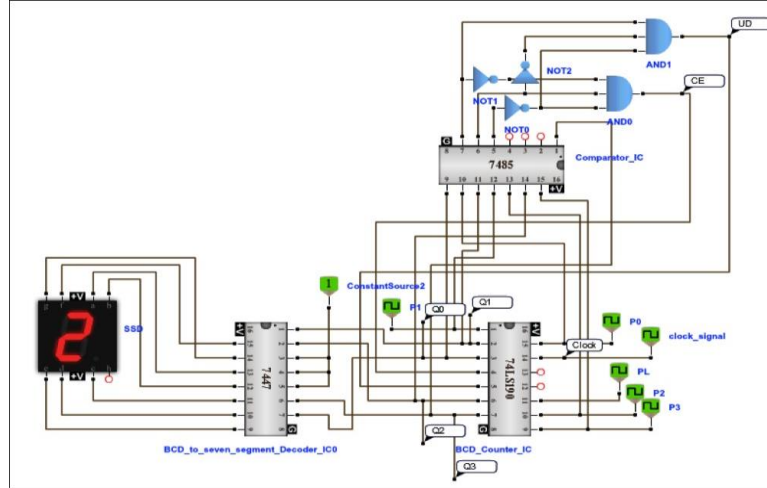
Elevator Control Circuit The circuit uses three ICs namely: 7485 which is a 4-magnitude comparator, 74190 which is an up-down decade counter and 7447 which is a BCD to seven-segment decoder IC. The interconnections made within the circuit would illustrate the working of the display system. First, the input given by the user (after binary encoding) is given as input to the decade counter from the parallel input pins P0, P1, P2, P3. The parallel load pin PL is active low and therefore is given logic 0. The output from the decade counter is passed to the magnitude comparator which is compared with the input from the P0, P1, P2, P3 pins. The comparator has three outputs viz: greater than, lesser than, equal to. The comparison may be any of these three possibilities and therefore it is essential to decide whether to keep counting up or start counting down based on the comparator output. This decision circuitry is designed as a combinational circuit with the help of logic gates. As can be seen in the circuit, the two outputs from the logic circuit are CE and UD. These are respectively connected to the CE and U/D pin of the counter IC. The CE pin is active low and is called count enable, therefore the counter counts every time this pin receives a logic '0'. The U/D pin dictates whether the counter

is in up-counter (logic '0' at U/D pin) or down-counter (logic '1' at U/D pin) mode respectively. Finally, the output at each stage from the counter is given to the decoder IC that keeps displaying the

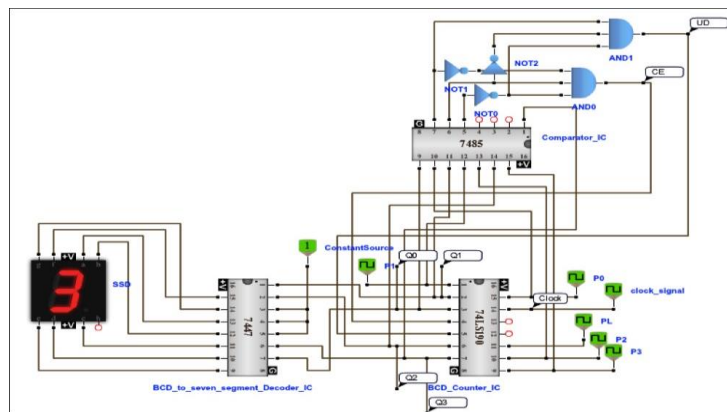
corresponding floor number in the seven segment display. Carrying on with our example, the user input here is '3' (for third floor) or '0001' (binary output from the encoder). When this '0011' is fed into this counter IC as shown, the ordinary decade counter will not count up to 9 anymore, instead it will count up to 3. The initial floor is '0' and it moves up to the 3rd floor. Then we have simulated a user input of '1' -0001- at the parallel load input. This results in the display now going to '1' akin to the movement of the elevator to a lower floor from the upper floor.

During UP count:





During DOWN count:



Therefore the lift comes back to its ground state after every cycle unless a trigger is applied to call the elevator to some other floor. Not as easily done as said, but certainly the fundamentals remain the same. It is an irrefutable fact that some simple electronics runs at the back-end of almost everything that comprises today's modern world, be it the metro that displays station name and other vital information, your trendy digital watches, the cool display with stock market updates on Wall Street, or of course the elevators!

Conclusion:

We have successfully developed an application using Beaglebone Black/ ARM Cortex A5 development board to simulate the operations of LIFT.

ASSIGNMENT NUMBER: D1**Revised On: 16-12-2019**

TITLE	Write an Client/Server application by using Raspberry-pi.
PROBLEM STATEMENT /DEFINITION	Write a server application to be deployed on Raspberry-Pi /Beagle board. Write client applications to get services from the server application.
OBJECTIVE	<ul style="list-style-type: none">• To understand functionalities of various single board embedded platforms fundamentals.• To develop client server application.
S/W PACKAGES AND HARDWARE APPARATUS USED	Laptop/Computer, Raspberry pi, Ethernet cable, Micro USB
REFERENCES	http://www.electronicwings.com/raspberry-pi/pi-camera-module-interface-with-raspberry-pi-using-python
STEPS	Refer to details
INSTRUCTIONS FOR WRITING JOURNAL	<ul style="list-style-type: none">• Title• Problem Definition• Objectives• Theory• Class Diagram/ER diagram• Test cases• Program Listing• Output• Conclusion

Aim: Write a server application to be deployed on Raspberry-Pi /Beagle board. Write client applications to get services from the server application.

Pre-requisite:

Basic knowledge of embedded system and IOT

Learning Objectives:

- To develop client applications to get services from the server application.

Learning Outcomes:

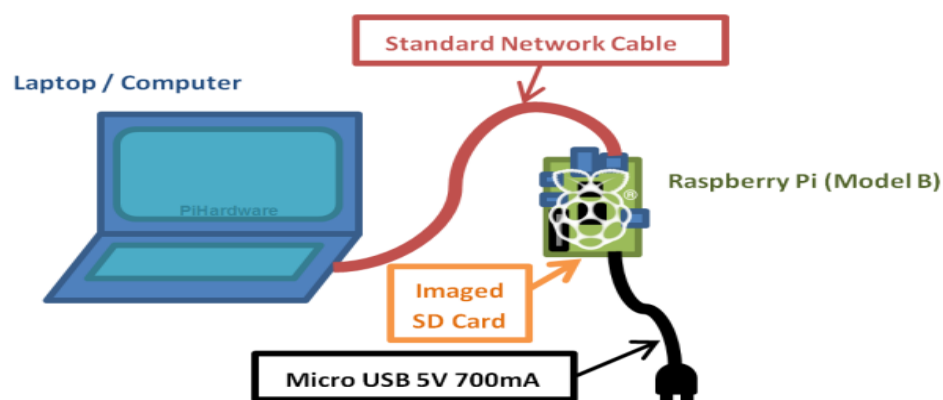
The students will be able to

- Implement an architectural design for IoT for specified requirement.
- CO2: Solve the given societal challenge using IoT

Theory:

A simple example to get started. The Raspberry Pi runs a server that waits for connection from a laptop, and expects integers from it. It multiplies each integer by 2 and sends it back. The laptop runs a client that initiates a connection, sends a bunch of positive integers that it gets back multiplied by two, and closes the connection by sending a -1. Sending a -2 causes the server to stop.

Hardware Setup



Server/Client setup

Server code

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>

void error( char *msg ) {
    perror( msg );
    exit(1);
}

int func( int a ) {
    return 2 * a;
}

void sendData( int sockfd, int x ) {
    int n;

    char buffer[32];
    sprintf( buffer, "%d\n", x );
    if ( ( n = write( sockfd, buffer, strlen(buffer) ) ) < 0 )
        error( const_cast<char *>( "ERROR writing to socket" ) );
    buffer[n] = '\0';
}

int getData( int sockfd ) {
    char buffer[32];
    int n;

    if ( ( n = read(sockfd,buffer,31) ) < 0 )
        error( const_cast<char *>( "ERROR reading from socket" ) );
    buffer[n] = '\0';
    return atoi( buffer );
}

int main(int argc, char *argv[]) {
P:F-LTL-UG/03/R1
```

```

int sockfd, newsockfd, portno = 51717, clilen;
char buffer[256];
struct sockaddr_in serv_addr, cli_addr;
int n;
int data;

printf( "using port #%%d\\n", portno );

sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd < 0)
    error( const_cast<char *>("ERROR opening socket") );
bzero((char *) &serv_addr, sizeof(serv_addr));

serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = INADDR_ANY;
serv_addr.sin_port = htons( portno );
if (bind(sockfd, (struct sockaddr *) &serv_addr,
    sizeof(serv_addr)) < 0)
    error( const_cast<char *>( "ERROR on binding" ) );
listen(sockfd,5);
clilen = sizeof(cli_addr);

//--- infinite wait on a connection ---
while ( 1 ) {
    printf( "waiting for new client...\\n" );
    if ( ( newsockfd = accept( sockfd, (struct sockaddr *) &cli_addr, (socklen_t*) &clilen) ) <
0 )
        error( const_cast<char *>("ERROR on accept") );
    printf( "opened new communication with client\\n" );
    while ( 1 ) {
        //---- wait for a number from client ---
        data = getData( newsockfd );
        printf( "got %%d\\n", data );
        if ( data < 0 )
            break;

        data = func( data );

        //--- send new data back ---
        printf( "sending back %%d\\n", data );
        sendData( newsockfd, data );
    }
    close( newsockfd );
}

```

```

    //--- if -2 sent by client, we can quit ---
    if ( data == -2 )
        break;
}
return 0;
}

```

Client code

```

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <netdb.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <arpa/inet.h>

void error(char *msg) {
    perror(msg);
    exit(0);
}

void sendData( int sockfd, int x ) {
    int n;

    char buffer[32];
    sprintf( buffer, "%d\n", x );
    if ( (n = write( sockfd, buffer, strlen(buffer) ) ) < 0 )
        error( const_cast<char *>( "ERROR writing to socket" ) );
    buffer[n] = '\0';
}

int getData( int sockfd ) {
    char buffer[32];
    int n;

    if ( (n = read(sockfd,buffer,31) ) < 0 )
        error( const_cast<char *>( "ERROR reading from socket" ) );
}

```



```

    buffer[n] = '\0';
    return atoi( buffer );
}

int main(int argc, char *argv[])
{
    int sockfd, portno = 51717, n;
    char serverIp[] = "169.254.0.2";
    struct sockaddr_in serv_addr;
    struct hostent *server;
    char buffer[256];
    int data;

    if (argc < 3) {
        // error( const_cast<char *>( "usage myClient2 hostname port\n" ) );
        printf( "contacting %s on port %d\n", serverIp, portno );
        // exit(0);
    }
    if ( ( sockfd = socket(AF_INET, SOCK_STREAM, 0) ) < 0 )
        error( const_cast<char *>( "ERROR opening socket" ) );

    if ( ( server = gethostbyname( serverIp ) ) == NULL )
        error( const_cast<char *>( "ERROR, no such host\n" ) );

    bzero( (char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    bcopy( (char *)server->h_addr, (char *)&serv_addr.sin_addr.s_addr, server->h_length);
    serv_addr.sin_port = htons(portno);
    if ( connect(sockfd,(struct sockaddr *)&serv_addr,sizeof(serv_addr)) < 0)
        error( const_cast<char *>( "ERROR connecting" ) );

    for ( n = 0; n < 10; n++ ) {
        sendData( sockfd, n );
        data = getData( sockfd );
        printf("%d -> %d\n",n, data );
    }
    sendData( sockfd, -2 );

    close( sockfd );
    return 0;
}

```

ASSIGNMENT NUMBER: D2**Revised On: 16-12-2019**

TITLE	Create a small dashboard application to be deployed on cloud.
PROBLEM STATEMENT /DEFINITION	Create a small dashboard application to be deployed on cloud. Different publisher devices can publish their information and interested application can subscribe.
OBJECTIVE	<ul style="list-style-type: none">• To develop comprehensive approach towards building small low cost embedded IoT system.• To understand different sensory inputs.
S/W PACKAGES AND HARDWARE APPARATUS USED	Cloud (ThingSpeak), client server model, controller/processor ,Python , PC with the configuration as Latest Version of 64 bit Operating Systems, Open Source Fedora-GHz. 8 G.B. RAM, 500 G.B. HDD, 15"Color Monitor, Keyboard, Mouse
REFERENCES	<ul style="list-style-type: none">• Olivier Hersent, David Boswarthick, Omar Elloumi , “The Internet of Things – Key applications and Protocols”, Wiley, 2012, ISBN:978-1-119-99435-0• Barrie Sosinsky, “Cloud Computing Bible”, Wiley-India, 2010.ISBN : 978-0-470-90356-8• Adrian McEwen, Hakim Cassimally, “Designing the Internet of Things”, Wiley, 2014, ISBN: 978-1-118-43063-7
STEPS	Refer to details
INSTRUCTIONS FOR WRITING JOURNAL	<ul style="list-style-type: none">• Title• Problem Definition• Objectives• Theory• Class Diagram/ER diagram• Test cases• Program Listing• Output• Conclusion

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

Pre-requisite:

Basic knowledge of Embedded system and IOT

Learning Objectives:

- To Develop application based on cloud.
- To understand different sensory inputs
- Understand client server model programming.

Learning Outcomes:

The students will be able to

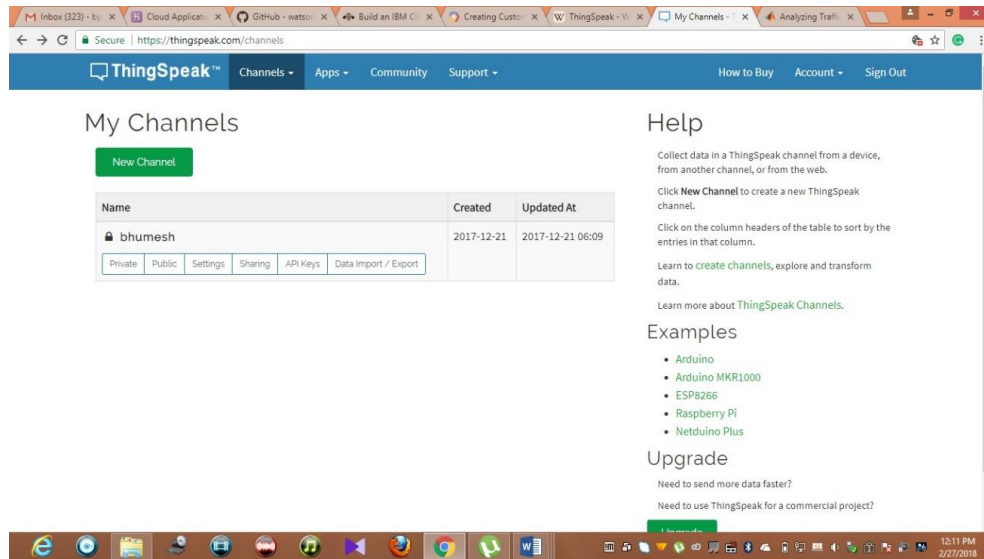
- Perform the connectivity with Raspberry-Pi, Beagle board, Arduino and other micro controller.
- Implement cloud application with the help of client server programming by using python.

Theory:

Thingspeak is an open source Internet of Things (IoT) application and API to store and retrieve data from things using the HTTP protocol over the Internet or via a Local Area Network. ThingSpeak enables the creation of sensor logging applications, location tracking applications, and a social network of things with status updates. ThingSpeak was originally launched by ioBridge in 2010 as a service in support of IoT applications. ThingSpeak has integrated support from the numerical computing software MATLAB from MathWorks, allowing ThingSpeak users to analyze and visualize uploaded data using Matlab without requiring the purchase of a Matlab license from Mathworks.

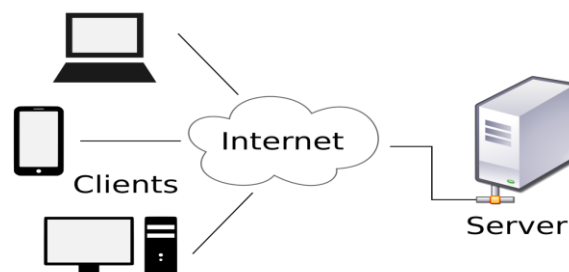
For accessing Thingspeak

Need to create account on <https://thingspeak.com/login>



client server model :

The **client-server model** is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A server host runs one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests. Examples of computer applications that use the client-server model are Email, network printing, and the World Wide Web



SETTING UP AN APACHE WEB SERVER ON A RASPBERRY PI:

Apache is a popular web server application you can install on the Raspberry Pi to allow it to serve web pages.

On its own, Apache can serve HTML files over HTTP, and with additional modules can serve dynamic web pages using scripting languages such as PHP.

INSTALL APACHE

First install the `apache2` package by typing the following command in to the Terminal:

```
sudo apt-get install apache2 -y
```

TEST THE WEB SERVER

By default, Apache puts a test HTML file in the web folder. This default web page is served when you browse to `http://localhost/` on the Pi itself, or `http://192.168.1.10` (whatever the Pi's IP address is) from another computer on the network. To find the Pi's IP address, type `hostname -I` at the command line (or read more about finding your [IP address](#)).

CHANGING THE DEFAULT WEB PAGE

This default web page is just a HTML file on the filesystem. It is located at `/var/www/html/index.html`.

Note: The directory was `/var/www` in Raspbian Wheezy but is now `/var/www/html` in Raspbian Jessie

Navigate to this directory in the Terminal and have a look at what's inside:

```
cd /var/www/html
ls -al
```

This will show you:

```
total 12
drwxr-xr-x  2 root root 4096 Jan  8 01:29 .
drwxr-xr-x 12 root root 4096 Jan  8 01:28 ..
-rw-r--r--  1 root root 177 Jan  8 01:29 index.html
```

This shows that there is one file in `/var/www/html/` called `index.html`. The `.` refers to the directory itself `/var/www/html` and the `..` refers to the parent directory `/www/`.

WHAT THE COLUMNS MEAN

1. The permissions of the file or directory
2. The number of files in the directory (or 1 if it's a file).
3. The user which owns the file or directory
4. The group which owns the file or directory
5. The file size
6. The last modification date & time

As you can see, by default the `html` directory and `index.html` file are both owned by the `root` user. In order to edit the file, you must gain rootpermissions. Change the owner to your own user with `sudo chown pi: index.html` before editing.

Try editing this file and refreshing the browser to see the web page change.

YOUR OWN WEBSITE

If you know HTML you can put your own HTML files and other assets in this directory and serve them as a website on your local network.

ADDITIONAL - INSTALL PHP

To allow your Apache server to process PHP files, you'll need to install PHP5 and the PHP5 module for Apache. Type the following command to install these:

```
sudo apt-get install php5 libapache2-mod-php5 -y
```

Now remove the `index.html` file:

```
sudo rm index.html
```

and create the file `index.php`:

```
sudo leafpad index.php
```

Note: Leafpad is a graphical editor. Alternatively, use `nano` if you're restricted to the command line

Put some PHP content in it:

```
<?php echo "hello world"; ?>
```

Now save and refresh your browser. You should see "hello world". This is not dynamic but still served by PHP. Try something dynamic:

```
<?php echo date('Y-m-d H:i:s'); ?>
```

or show your PHP info:

```
<?php phpinfo(); ?>
```

```
sudo apt-get install apache2 -y
```

```
sudo apt-get install php5 libapache2-mod-php5 -y
```

```
sudo apt-get install git-core
```

```
git clone git://git.drogon.net/wiringPi
```

```
cd wiringPi
```

```
./build
```

-----now open terminal and enter this:-----

```
$cd /var/www
```

```
$sudo nano rahul.php
```

-----this will open a empty black screen window where we have to write these instructions-----

```
*****  
*****
```

```
<html>
```

```
<head>
```

```
<meta name="viewport" content="width=device-width" />
```

```
<title>LED Control</title>
```

```
</head>
```

```
    <body>
```

```
P:F-LTL-UG/03/R1
```

```
<form method="get" action="gpio.php">
    <input type="submit" value="ON" name="on">
    <input type="submit" value="OFF" name="off">
</form>
```

</html>

```
<?php
$myfile = fopen("/home/pi/log.txt", "r") or die("Unable to open file!");
echo fread($myfile,filesize("/home/pi/log.txt"));
fclose($myfile);
?>
```

-----now press "cnt+O" to save and then "cnt+X" to exit
P:F-LTL-UG/03/R1

-----once go to /var/www library there we can find rahul.php open it and cross check it-----
-----now check ip address of our pi by giving ifconfig in terminal window-----

-----enter that ip in your mobile browser as 192.168.2.26/rahul.php-----
-----bingo here it is-----

Mail code:-

```
import RPi.GPIO as GPIO

from subprocess import call

import time

import os

import glob

import smtplib

import base64

from email.mime.image import MIMEImage

from email.mime.multipart import MIMEMultipart

import subprocess

gmail_user = "checking999mail@gmail.com"

gmail_pwd = "mail999checking"

FROM = 'checking999mail@gmail.com'

TO = ['hyd.embedded@pantechmail.com'] #must be a list

i=1

while (i):

    i=i-1

    subprocess.Popen( "fswebcam -r 1280x720 /home/pi/Downloads/pan.jpg", shell=True )
```

P:F-LTL-UG/03/R1

```

    time.sleep(1)
msg = MIMEMultipart()
    time.sleep(1)
msg['Subject'] = "testing msg send from python"
    time.sleep(1)
fp = open("/home/pi/Downloads/pan.jpg", 'rb')
    time.sleep(1)
img = MIMEImage(fp.read())
    time.sleep(1)
fp.close()
    time.sleep(1)
msg.attach(img)
    time.sleep(1)
try:
    server = smtplib.SMTP("smtp.gmail.com", 587) #or port 465 doesn't seem to work!
        print "smtp.gmail"
    server.ehlo()
        print "ehlo"
    server.starttls()
        print "starttls"
    server.login(gmail_user, gmail_pwd)
        print "reading mail & password"
    server.sendmail(FROM, TO, msg.as_string())
        print "from"
    server.close()
        print 'successfully sent the mail'
P:F-LTL-UG/03/R1

```

```
except:
    print "failed to send mail"

sudo apt-get install apache2
sudo apt-get install php5 libapache2-mod-php5
sudo apt-get install git-core
git clone git://git.drogon.net/wiringPi
cd wiringPi
./build
```

-----now open terminal and enter this:-----

```
$cd /var/www/html
```

```
$sudo leafpad gpio.php
```

-----this will open a empty black screen window where we have to write these instructions-----

```
*****
*****
```

```
<html>
```

```
<head>
```

```
<meta name="viewport" content="width=device-width" />
```

```
<title>LED Control</title>
```

```
</head>
```

```
<body>
```

WEB PAGE ON PHP BASED GPIO Control:

```
<form method="get" action="gpio2.php">
```

```
<input type="submit" value="ON" name="on">
```

```
<input type="submit" value="OFF" name="off">
```

```
</form>
```

P:F-LTL-UG/03/R1

-----now press "ctrl+s" to save and then to exit

-----once go to /var/www/html library there we can find gpio.php open it and cross check it----

-----now check ip address of our pi by giving ifconfig in terminal window-----

-----enter that ip in your mobile browser as 192.168.2.26/gpio.php-----

-----bingo here it is-----

```
import subprocess
```

```
from subprocess import call
```

```
import RPi.GPIO as GPIO
```

```
import time
```

```
GPIO.setmode(GPIO.BCM)
```

```
GPIO.setup(6, GPIO.IN, pull_up_down = GPIO.PUD_UP)
```

```
GPIO.setup(13, GPIO.IN, pull_up_down = GPIO.PUD_UP)
```

```
GPIO.setup(19, GPIO.IN, pull_up_down = GPIO.PUD_UP)
```

```
GPIO.setup(26, GPIO.IN, pull_up_down = GPIO.PUD_UP)
```

```
while True:
```

```
    if(GPIO.input(26) == 0):
```

```
        pageToOpen="en.wikipedia.org/wiki/A"
```

```
        subprocess.Popen(["midori","-a", pageToOpen])
```

```
        call(["espeak","Object found is an apple"],shell=True)
```

```
        time.sleep(5)
```

P:F-LTL-UG/03/R1

```
        print('a')
    else:
        time.sleep(0.5)
GPIO.cleanup()
```

ASSIGNMENT NUMBER: E1**Revised On: 16-12-2019**

TITLE	Develop a real time application.
PROBLEM STATEMENT /DEFINITION	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.
OBJECTIVE	<ul style="list-style-type: none">• To develop comprehensive approach towards building small low cost embedded IoT system.• To understand different sensory inputs.• To develop real time IoT based application.
S/W PACKAGES AND HARDWARE APPARATUS USED	Raspberry Pi 3 (Any other Version will be nice), Memory card 8 or 16GB running Raspbian Jessie, 5v Relays, 2n222 transistors, Diodes, Jumper Wires, Connection Blocks, LEDs to test., AC lamp to Test, Breadboard and jumper cables, 220 or 100 ohms resistor, python
REFERENCES	<ul style="list-style-type: none">• Olivier Hersent, Omar Elloumi and David Boswarthick, “The Internet of Things: Applications to the Smart Grid and Building Automation”, Wiley, 2012, 9781119958345• Olivier Hersent, David Boswarthick, Omar Elloumi , “The Internet of Things – Key applications and Protocols”, Wiley, 2012, ISBN:978-1-119-99435-0
STEPS	Refer to details
INSTRUCTIONS FOR WRITING JOURNAL	<ul style="list-style-type: none">• Title• Problem Definition• Objectives• Theory• Class Diagram/ER diagram• Test cases• Program Listing• Output• Conclusion

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.

Pre-requisite:

Basic knowledge of Embedded system and IOT

Learning Objectives:

- To Develop an IoT Based (smart) application.

Learning Outcomes:

The students will be able to

- Implement an architectural design for IoT for specified requirement
- Solve the given societal challenge using IoT

Theory:

Automatic Room Light Control

When we enter a room, as a habitual tendency, we often search for a switch to turn the light on, and if we are new to the room, we often find it difficult to locate the switch. Most of the times, many of us forget to switch off the lights while leaving the room in which we stay most of the time. This results in unnecessary power wastage. Therefore, an automatic room-light controller automatically turns on the lights when a person enters into a room, and turns off the lights when the person leaves the room. This automatic room controller can be implemented by using a simple microcontroller and wireless IR technologies.

INTRODUCTION

Home automation is the control of any or all electrical devices in our home or office. There are many different types of home automation system available. These systems are typically designed and purchased for different purposes. In

fact, one of the major problems in the area is that these different systems are neither interoperable nor interconnected. There are number of issues involve when designing a home automation system. It should also provide a user friendly

interface on the host side, so that the devices can be easily setup, monitored and controlled In smart home systems, the internet is also use to ensure remote control. For years, the internet has been widely use for the processes such as surfing on the pages, searching information, chatting, downloading and installation. By the rapid developments of new technologies, monitoring, controlling services have been started to be served along with internet as an instrument providing interaction with machinery and devices. The system can be use in several places like banks, hospital, labs and other sophisticated automated system, which dramatically reduced the hazards of unauthorized entry. The main reason to develop this system is to save time and man power along with maintaining security and convenience.

Home Appliances Control using a Remote Control:

The lights, fans can be automatically turned on/off with the help of a remote where there will be a sensor instead of going near to a switch board and putting on/off the switch. Companies like Legrand and Gold Medal already started

these kinds of control system and they are at present available i
n the market.

Home Appliances Control using DTMF:

In this method, the control of home appliances can be done even though when we are elsewhere just by using the DTMF tone generated when the user pushes mobile phone keypad buttons or when connected to a remote mobile.

Home Appliance Control Using Free Hand Gesture:

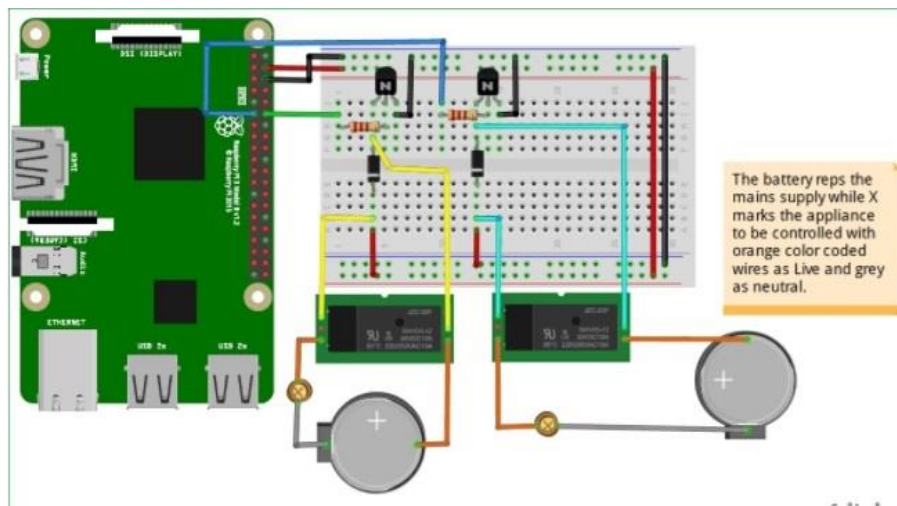
This is a type of home appliance control system where the person must be present in sight to the appliance that is needed to be controlled and a predefined gesture must be used to turn on the device and another gesture must be used by

us to turnoff the device. The performance of the proposed system is done with a hardware embedded in that particular device.

Home Appliance Control Using Internet and Radio Connection:

In this system, the control of home appliances can be done from a remote are with an option from a local server, using the Internet and radio connection. This system is accomplished by personal computers, interface cards, radio

Circuit Diagram and Explanation:



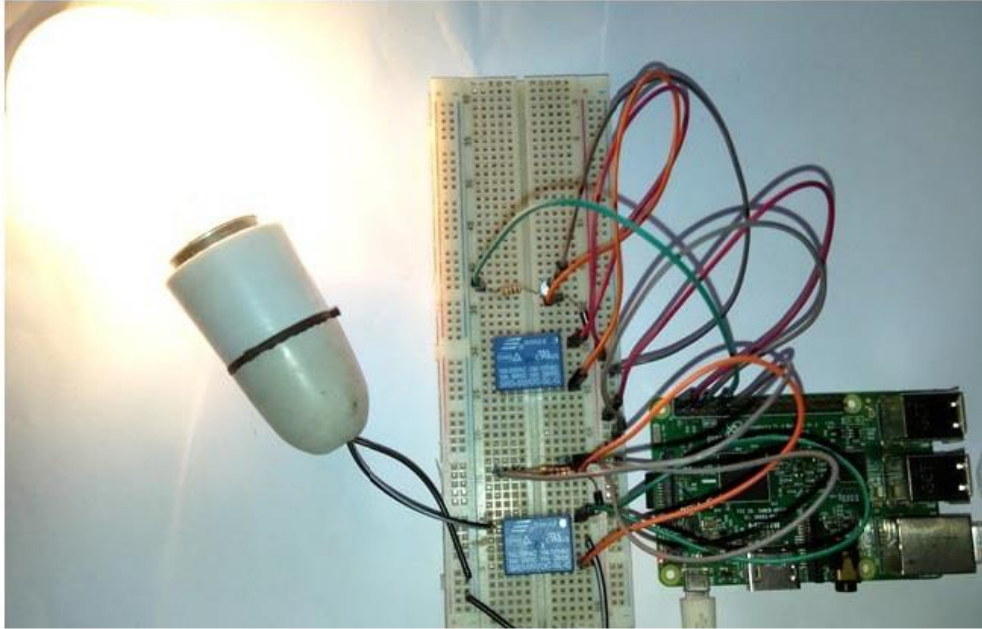
I. Hardware Requirements:

1. Raspberry Pi 3 (Any other Version will be nice)
2. Memory card 8 or 16GB running Raspbian Jessie
3. 5v Relays
4. 2n222 transistors
5. Diodes
6. Jumper Wires
7. Connection Blocks
8. LEDs to test.
9. AC lamp to Test
10. Breadboard and jumper cables
11. 220 or 100 ohms resistor

II. Software Requirements:

Asides the Raspbian Jessie operating system running on the raspberry pi, we will also be using the **WebIOPi** frame work, **notepad++** running on your PC and **filezilla** to copy files from the PC to the raspberry pi, especially the web app files.

P:F-LTL-UG/03/R1



Preparing the Raspberry Pi:

To **update the raspberry Pi** below commands and then reboot the RPi;

```
sudo apt-get update  
sudo apt-get upgrade  
sudo reboot
```

With this done, the next thing is for us to **install the webIOPi framework**.

Make sure you are in home directory using;

```
cd ~
```

Use wget to get the file from their sourceforge page;

```
wget http://sourceforge.net/projects/webiopi/files/WebIOPi-0.7.1.tar.gz
```

P:F-LTL-UG/03/R1

When download is done, extract the file and go into the directory;

```
tar xvzf WebIOPi-0.7.1.tar.gz  
cd WebIOPi-0.7.1/
```

At this point before running the setup, we need to **install a patch as this version of the WebIOPi** does not work with the raspberry pi 3 which I am using and I couldn't find a version of the WebIOPi that works expressly with the Pi 3.

Below commands are used to install patch while still in the WebIOPi directory, run;

```
wget https://raw.githubusercontent.com/doublebind/raspi/master/webiopi-pi2bplus.patch  
patch -p1 -i webiopi-pi2bplus.patch
```

Then we can run the setup installation for the WebIOPi using;

```
sudo ./setup.sh
```

Keep saying yes if asked to install any dependencies during setup installation. When done, reboot your pi;

```
sudo reboot
```

Test WebIOPi Installation:

Before jumping in to schematics and codes, With the Raspberry Pi back on, we will need to test our WebIOPi installation to be sure everything works fine as desired.

Run the command;

```
sudo webiopi -d -c /etc/webiopi/config
```

After issuing the command above on the pi, point the web browser of your computer connected to the raspberry pi to **http://raspberrypi.mshome.net:8000** or **http://thepi'sIPaddress:8000**. The system will prompt you for username and password.

Username is webiopi

Password is raspberry

This login can be removed later if desired but even your home automation system deserves some P:F-LTL-UG/03/R1

extra level of security to prevent just anyone with the IP controlling appliances and IOT devices in your home.

Building the Web Application for Raspberry Pi Home Automation:

Here we will be editing the default configuration of the WebIOPi service and add our own code to be run when called. The first thing we will do is get filezilla or anyother FTP/SCP copy software installed on our PC. You will agree with me that coding on the pi via the terminal is quite stressful, so filezilla or any other SCP software will come in handy at this stage. Before we start writing the html, css and java script codes for this **IoT Home automation Web application** and moving them to the Raspberry Pi, lets create the project folder where all our web files will be.

Make sure you are in home directory using, then create the folder, go into the created folder and create html folder in the directory:

```
cd ~  
mkdir webapp  
cd webapp  
mkdir html
```

Create a folder for scripts, CSS and images inside the html folder

```
mkdir html/css  
mkdir html/img  
mkdir html/scripts
```

Test WebIOPi Installation:

Before jumping in to schematics and codes, With the Raspberry Pi back on, we will need to test our WebIOPi installation to be sure everything works fine as desired.

Run the command;

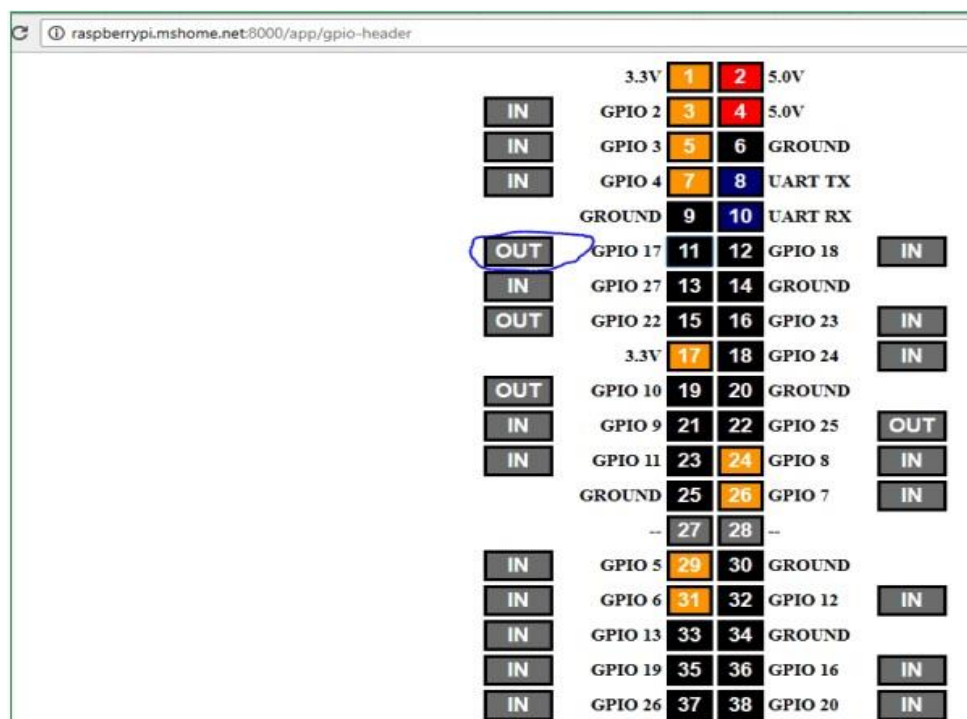
```
sudo webiopi -d -c /etc/webiopi/config
```

After issuing the command above on the pi, point the web browser of your computer connected to the raspberry pi to **http://raspberrypi.mshome.net:8000** or **http://theipi'sIPaddress:8000**. The system will prompt you for username and password.

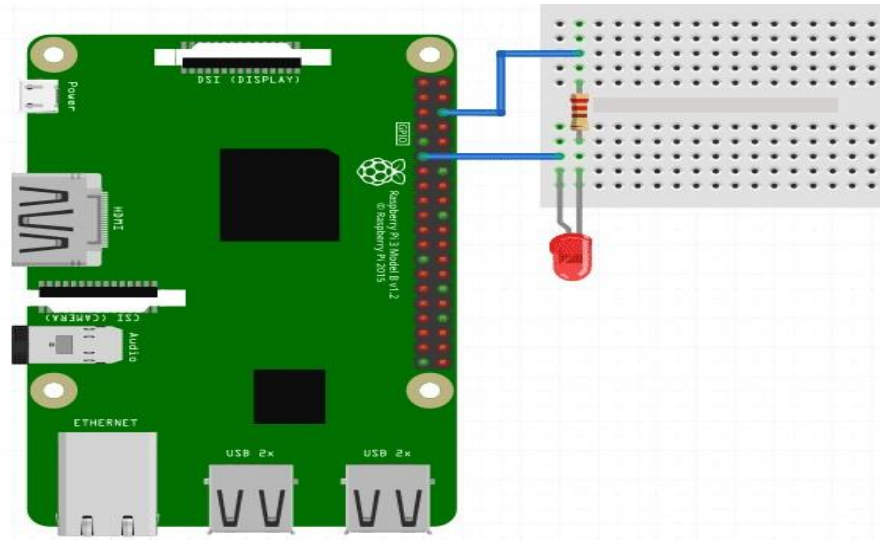
Username is webiopi

Password is raspberry

This login can be removed later if desired but even your home automation system deserves some extra level of security to prevent just anyone with the IP controlling appliances and IOT devices in your home.



With this done, connect the led to your raspberry pi as shown in the schematics below.



After the connection, go back to the webpage and **click the pin 11 button to turn on or off the LED**. This way we can control the Raspberry Pi GPIO using WebIOPi.

After the test, if everything worked as described, then we can go back to the terminal and stop the program with CTRL + C. If you have any issue with this setup, then hit me up via the comment section.

More info on Webiopi can be found on its Wiki page (<http://webiopi.trouch.com/INSTALL.html>)

With the test complete, we are then set to start the main project.

Building the Web Application for Raspberry Pi Home Automation:

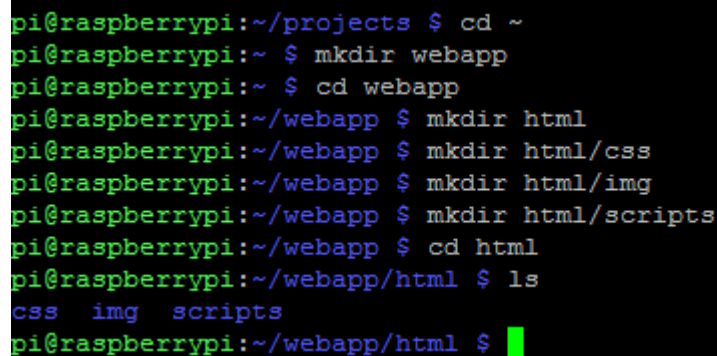
Here we will be editing the default configuration of the WebIOPi service and add our own code to be run when called. The first thing we will do is get **filezilla** or anyother FTP/SCP copy software installed on our PC. You will agree with me that coding on the pi via the terminal is quite stressful, so filezilla or any other SCP software will come in handy at this stage. Before we start writing the html, css and java script codes for this **IoT Home automation Web application** and moving them to the Raspberry Pi, lets create the project folder where all our web files will be.

Make sure you are in home directory using, then create the folder, go into the created folder and create html folder in the directory:

```
cd ~  
mkdir webapp  
cd webapp  
mkdir html
```

Create a folder for scripts, CSS and images inside the html folder

```
mkdir html/css  
mkdir html/img  
mkdir html/scripts
```

A terminal window on a Raspberry Pi showing the following commands and output:

```
pi@raspberrypi:~/projects $ cd ~  
pi@raspberrypi:~ $ mkdir webapp  
pi@raspberrypi:~ $ cd webapp  
pi@raspberrypi:~/webapp $ mkdir html  
pi@raspberrypi:~/webapp $ mkdir html/css  
pi@raspberrypi:~/webapp $ mkdir html/img  
pi@raspberrypi:~/webapp $ mkdir html/scripts  
pi@raspberrypi:~/webapp $ cd html  
pi@raspberrypi:~/webapp/html $ ls  
css  img  scripts  
pi@raspberrypi:~/webapp/html $
```

With our files created lets move to writing the codes on our PC and from then move to the Pi via filezilla.

The JavaScript Code:

The first piece of code we will write is that of the javascript. Its a simple script to communicate with the WebIOPi service.

Its important to note that for this project, our web app will consist of four buttons, which means we plan to control just four GPIO pins, although we will be controlling just two relays in our demonstration. Check the **full Video** at the end.


```

webiopi().ready(function() {
    webiopi().setFunction(17, "out");
    webiopi().setFunction(18, "out");
    webiopi().setFunction(22, "out");
    webiopi().setFunction(23, "out");
    var content, button;
    content = $("#content");
    button = webiopi().createGPIOButton(17, "Relay 1");
    content.append(button);
    button = webiopi().createGPIOButton(18, "Relay 2");
    content.append(button);
    button = webiopi().createGPIOButton(22, "Relay 3");
    content.append(button);
    button = webiopi().createGPIOButton(23, "Relay 4");
    content.append(button);
});

```

The code above runs when the WebIOPi is ready.

Below we have explained the JavaScript code:

webiopi().ready(function()): This just instructs our system to create this function and run it when the webiopi is ready.

webiopi().setFunction(23,"out"); This helps us tell the WebIOPi service to set GPIO23 as output. We Have four buttons here, you could have more of it if you are implementing more buttons.

var content, button; This line tells our system to create a variable named content and make the variable a button.

content = \$("#content"); The content variable is still going to be used across our html and css. So when we refer to #content, the WebIOPi framework creates everything associated with it.

button = webiopi().createGPIOButton(17,"Relay 1"); WebIOPi can create different kinds of buttons. The piece of code above helps us to tell the WebIOPi service to create a GPIO button that controls the GPIO pin in this case 17 labeled “Relay 1”. Same goes for the other ones.

content.append(button); Append this code to any other code for the button created either in the html file or elsewhere. More buttons can be created and will all have the same properties of this button. This is especially useful when writing the CSS or Script.

After creating the JS files, we save it and then copy it using filezilla to webapp/html/scripts if you created your files the same way I did mine.

The CSS Code:

The CSS helps us make our IoT Raspberry Pi home automation webpage look pretty.

P:F-LTL-UG/03/R1

I wanted the webpage to look like the image below and thus had to write the smarthome.css style sheet to achieve it.



WebIOPi Server Edits for Home Automation:

At this stage, we are ready to start creating our schematics and test our web app but before then we need to edit the config file of the webiopi service so its pointed to use data from our html folder instead of the config files that came with it.

To edit the configuration run the following with root permission;

```
sudo nano /etc/webiopi/config
```

Look for the http section of the config file, check under the section where you have something like, #Use doc-root to change default HTML and resource files location

Comment out anything under it using # then if your folder is setup like mine, point your doc-root to the path of your project file

```
doc-root = /home/pi/webapp/html
```

Save and exit. You can also change the port from 8000, in case you have another server running on the pi using that port. If not save and quit, as we move on.

Its Important to note that you can change the password of the WebIOPi service using the command;

```
sudo webiopi-passwd
```

It will prompt you for a new username and password. This can also be removed totally but security is important right?

Lastly run the WebIOPi service by issuing below command:

```
sudo /etc/init.d/webiopi start
```

The server status can be checked using;

```
sudo /etc/init.d/webiopi status
```

It can be stopped from running using;

```
sudo /etc/init.d/webiopi stop
```

To setup WebIOPi to run at boot, use;

```
sudo update-rc.d webiopi defaults
```

If you want to reverse and stop it from running at boot, use;

```
sudo update-rc.d webiopi remove
```

OUTPUT:-



ASSIGNMENT NUMBER: E2**Revised On: 16-12-2019**

TITLE	Develop a real time application of smart home.
PROBLEM STATEMENT /DEFINITION	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.
OBJECTIVE	To develop real time IoT based application.
S/W PACKAGES AND HARDWARE APPARATUS USED	Raspberry Pi, Pi Camera, PIR Sensor, LED, Bread Board, Resistor (1k), Connecting wires, Power supply
REFERENCES	https://circuitdigest.com/microcontroller-projects/raspberry-pi-iot-intruder-alert-system
STEPS	Refer to details
INSTRUCTIONS FOR WRITING JOURNAL	<ul style="list-style-type: none">• Title• Problem Definition• Objectives• Theory• Class Diagram/ER diagram• Test cases• Program Listing• Output• Conclusion

Aim: 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.

Pre-requisite:

Basic knowledge of embedded system and IOT

Learning Objectives:

The students will be able to

- To Develop an IoT based (smart) application.

Learning Outcomes:

The students will be able to

- Implement an architectural design for IoT for specified requirement
- Solve the given societal challenge using IoT

Theory:

Raspberry Pi is an ARM cortex based popular development board designed for Electronic Engineers and Hobbyists. With the processing speed and memory, Raspberry Pi can be used for performing different functions at a time, like a normal PC, and hence it is called Mini Computer in your palm. Here **interfacing Pi camera with Raspberry Pi** to capture the image of every visitor which has entered through the Gate or door. In this assignment, whenever any person is arrived at the Gate, he has to press a button to open the Gate, and as soon as he/she press the button, his/her **picture will be captured and saved in the system with the Date and time of the entry**. This can be very useful for **security and surveillance** purpose. This system is very useful in offices or factories where visitor entry record is maintained for visitors and attendance record is maintained for employees. This Monitoring system will digitize and automate the whole visitor entries and attendances, and there will be no need to maintain them manually. This system can be either operated by the person himself or there can be operator for pressing the button for very visitor. This is a good project for **getting started with Pi camera and interface it with Raspberry Pi**.

Working Explanation:

Working of this **Raspberry Pi Monitoring System** is simple. In this, a **Pi camera** is used to capture the images of visitors, when a push button is pressed or triggered. A **DC motor is used as a gate**. Whenever anyone wants to enter in the place then he/she needs to push the button.

After pushing the button, Raspberry Pi sends command to Pi Camera to click the picture and save it. After it, the gate is opened for a while and then gets closed again. The buzzer is used to generate sound when button pressed and LED is used for indicating that Raspberry Pi is ready to accept Push Button press, means when LED is ON, system is ready for operation.

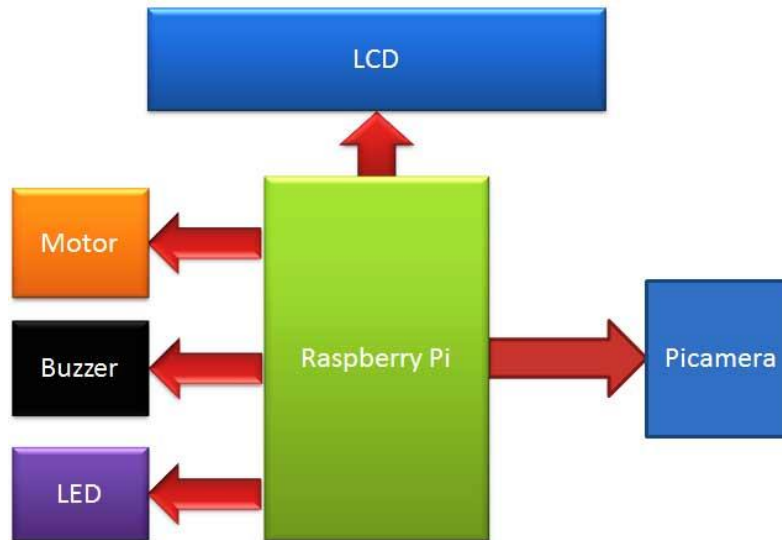


Fig:- Block diagram of interfacing PiCamera with raspberrypi

Circuit Explanation:

Circuit of this **Raspberry Pi Visitor Surveillance System** is very simple. Here a **Liquid Crystal Display (LCD)** is used for displaying Time/Date of visitor entry and some other messages. LCD is connected to Raspberry Pi in 4-bit mode. Pins of LCD namely RS, EN, D4, D5, D6, and D7 are connected to Raspberry Pi GPIO pin number 18, 23, 24, 16, 20 and 21. **Pi camera module** is connected at camera slot of the Raspberry Pi. A buzzer is connected to GPIO pin 26 of Raspberry Pi for indication purpose. LED is connected to GPIO pin 5 through a 1k resistor and a **push button** is connected to GPIO pin 19 with respect to ground, to trigger the camera and open the Gate. **DC motor (as Gate)** is connected with Raspberry Pi GPIO pin 17 and 27 through **Motor Driver IC (L293D)**. Rest of connections are shown in circuit diagram.

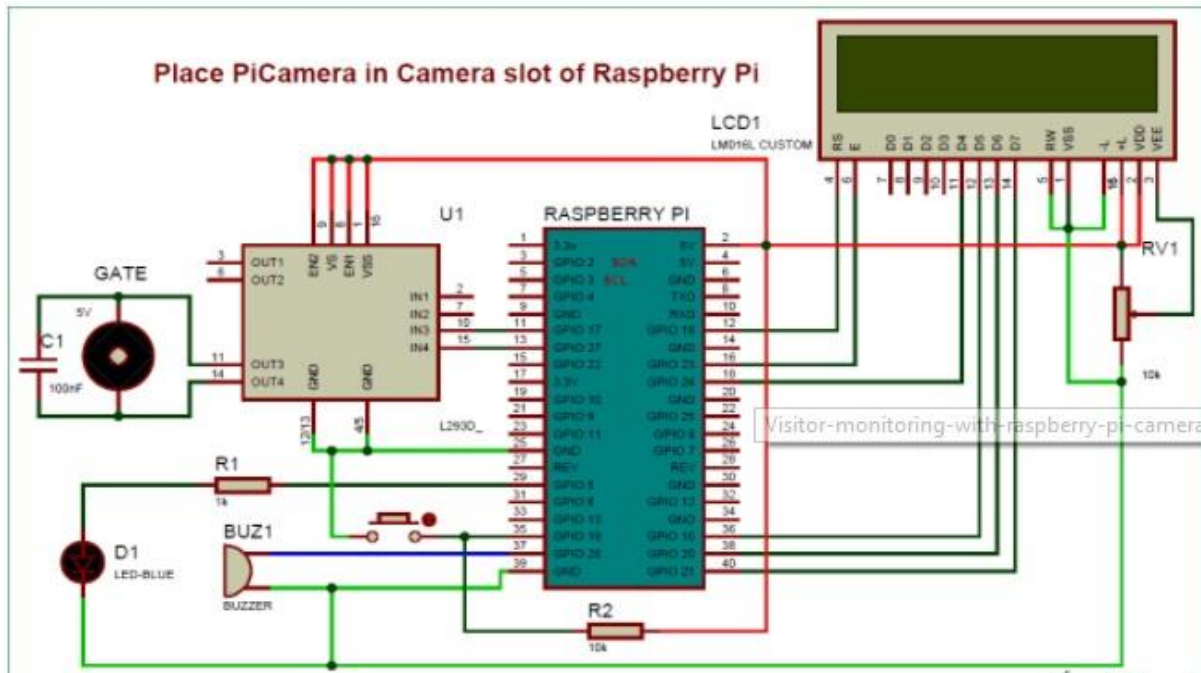


Fig :- circuit diagram

To connect the Pi Camera, insert the Ribbon cable of Pi Camera into camera slot, slightly pull up the tabs of the connector at RPi board and insert the Ribbon cable into the slot, then gently push down the tabs again to fix the ribbon cable.

Raspberry Pi Configuration and Programming Explanation:

We are using **Python language** here for the Program. Before coding, user needs to configure Raspberry Pi. You should follow below two tutorials for Getting Started with Raspberry Pi and Installing & Configuring Raspbian Jessie OS in Pi:

Circuit Description:

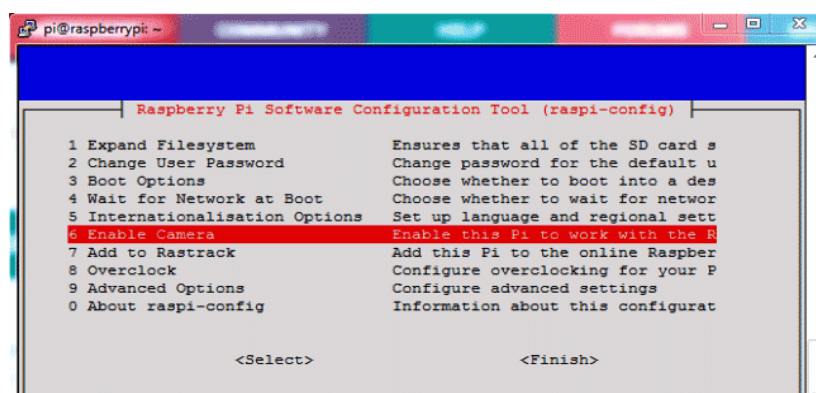
In this **Intruder Alert System**, we only need to **connect Pi Camera module and PIR sensor to Raspberry Pi 3**. Pi Camera is connected at the camera slot of the Raspberry Pi and PIR is connected to GPIO pin 18. A LED is also connected to GPIO pin 17 through a 1k resistor.


```
pi@raspberrypi: ~  
Ign http://mirrordirector.raspbian.org jessie/rpi Translation-en  
Fetched 9,278 kB in 39s (237 kB/s)  
Reading package lists... Done  
pi@raspberrypi:~ $ sudo apt-get install python-picamera  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
python-picamera is already the newest version.  
The following packages were automatically installed and are no longer required:  
  libgssapi3-heimdal libheimntlm0-heimdal libxfce4ui-1-0 pypy-upstream-doc  
  xfce-keyboard-shortcuts  
Use 'apt-get autoremove' to remove them.  
0 upgraded, 0 newly installed, 0 to remove and 39 not upgraded.  
pi@raspberrypi:~ $ sudo apt-get install python3-picamera  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
python3-picamera is already the newest version.  
The following packages were automatically installed and are no longer required:  
  libgssapi3-heimdal libheimntlm0-heimdal libxfce4ui-1-0 pypy-upstream-doc  
  xfce-keyboard-shortcuts  
Use 'apt-get autoremove' to remove them.  
0 upgraded, 0 newly installed, 0 to remove and 39 not upgraded.  
pi@raspberrypi:~ $
```

After it, user needs to enable Raspberry Pi Camera by using Raspberry Pi Software Configuration Tool (raspi-config):

```
$ sudo raspi-config
```

Then select Enable camera and Enable it.

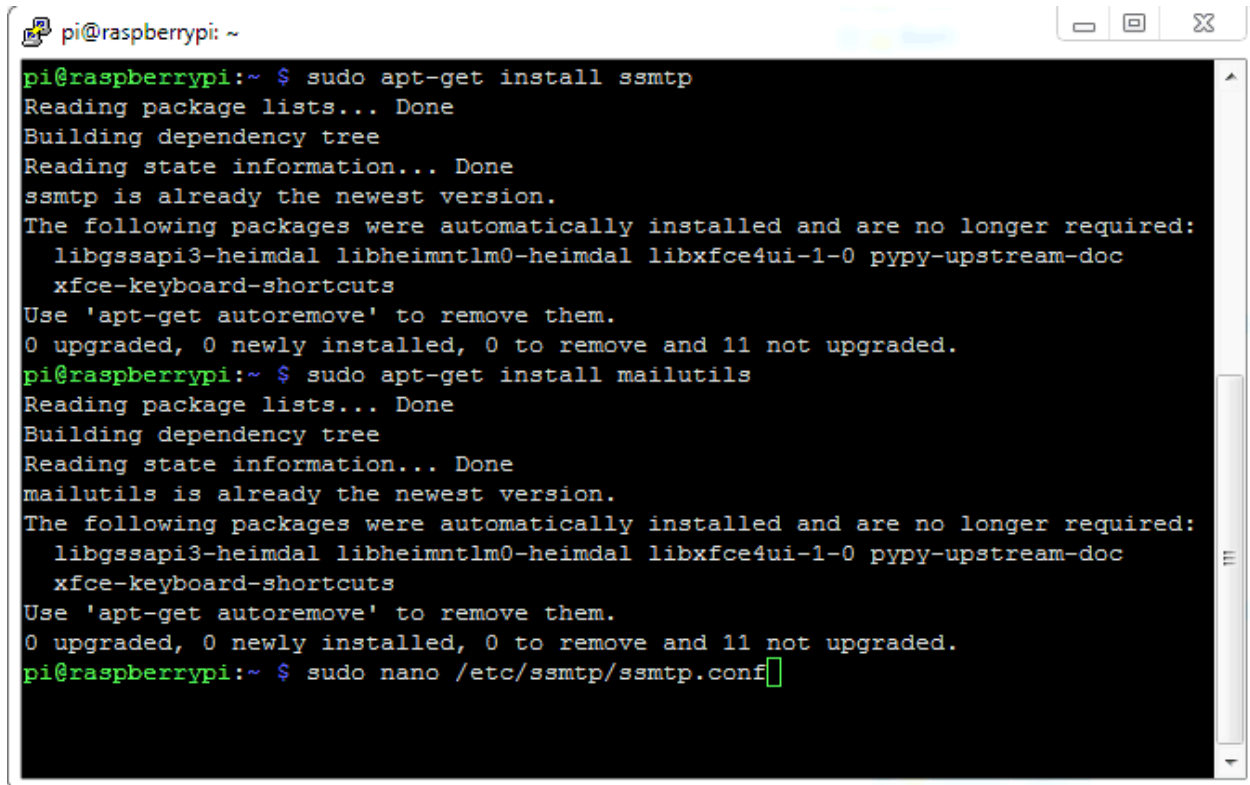


Then user needs to reboot Raspberry Pi, by issuing `sudo reboot`, so that new setting can take.
Now your Pi camera is ready to use.

Now after setting up the Pi Camera, we will install software for sending the mail. Here we are using ssmtp which is an easy and good solution for sending mail using command line or using Python Script. We need to install two Libraries for sending mails using SMTP:

```
sudo apt-get install ssmtp
```

```
sudo apt-get install mailutils
```



```
pi@raspberrypi: ~  
pi@raspberrypi:~ $ sudo apt-get install ssmtp  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
ssmtp is already the newest version.  
The following packages were automatically installed and are no longer required:  
  libgssapi3-heimdal libheimntlm0-heimdal libxfce4ui-1-0 pypy-upstream-doc  
  xfce-keyboard-shortcuts  
Use 'apt-get autoremove' to remove them.  
0 upgraded, 0 newly installed, 0 to remove and 11 not upgraded.  
pi@raspberrypi:~ $ sudo apt-get install mailutils  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
mailutils is already the newest version.  
The following packages were automatically installed and are no longer required:  
  libgssapi3-heimdal libheimntlm0-heimdal libxfce4ui-1-0 pypy-upstream-doc  
  xfce-keyboard-shortcuts  
Use 'apt-get autoremove' to remove them.  
0 upgraded, 0 newly installed, 0 to remove and 11 not upgraded.  
pi@raspberrypi:~ $ sudo nano /etc/ssmtp/ssmtp.conf
```

After installing libraries, user needs to open ssmtp.conf file and edit this configuration file as shown in the Picture below and then save the file. To save and exit the file, Press 'CTRL+x', then 'y' and then press 'enter'.

```
sudo nano /etc/ssmtp/ssmtp.conf
```

```
root=YourEmailAddress
```

```
mailhub=smtp.gmail.com:587
```

P:F-LTL-UG/03/R1

hostname=raspberrypi

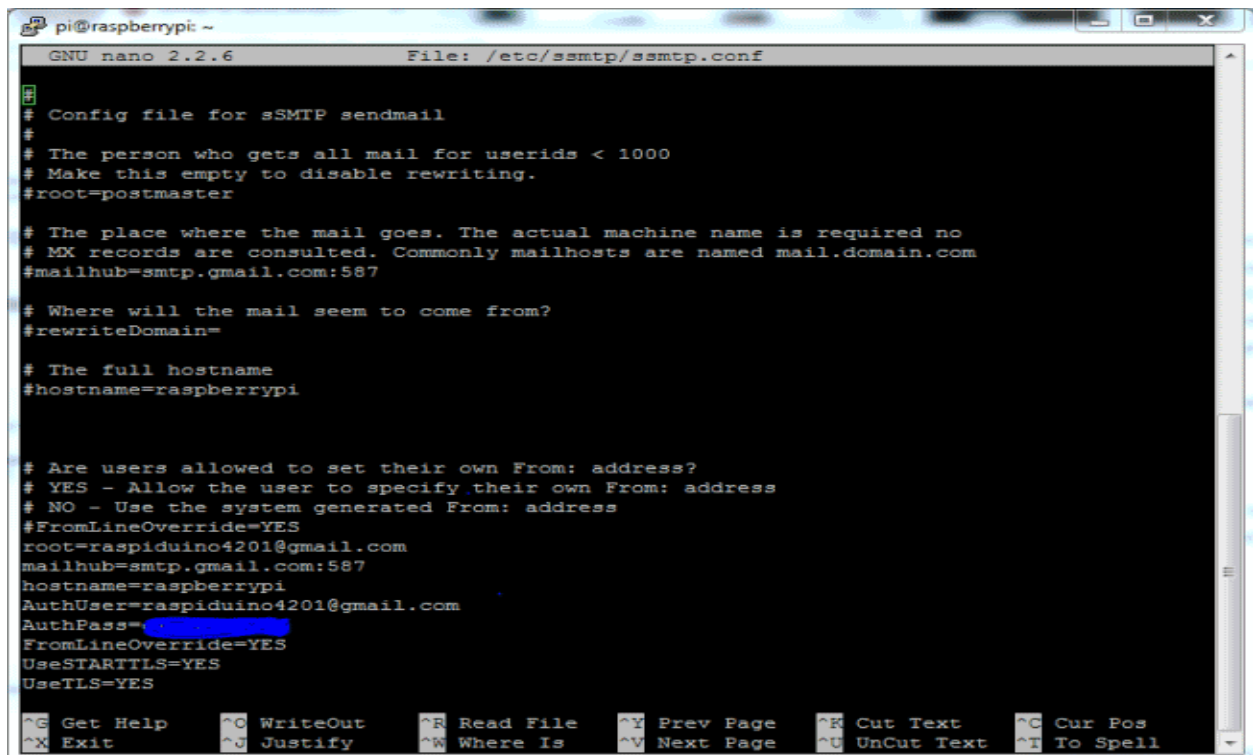
AuthUser=YourEmailAddress

AuthPass=YourEmailPassword

FromLineOverride=YES

UseSTARTTLS=YES

UseTLS=YES



```
pi@raspberrypi: ~  
GNU nano 2.2.6 File: /etc/smtplib/smtplib.conf  
# Config file for sSMTP sendmail  
#  
# The person who gets all mail for userids < 1000  
# Make this empty to disable rewriting.  
#root=postmaster  
  
# The place where the mail goes. The actual machine name is required no  
# MX records are consulted. Commonly mailhosts are named mail.domain.com  
#mailhub=smtplib.gmail.com:587  
  
# Where will the mail seem to come from?  
#rewriteDomain=  
  
# The full hostname  
#hostname=raspberrypi  
  
# Are users allowed to set their own From: address?  
# YES - Allow the user to specify their own From: address  
# NO - Use the system generated From: address  
#FromLineOverride=YES  
root=raspibuddy4201@gmail.com  
mailhub=smtplib.gmail.com:587  
hostname=raspberrypi  
AuthUser=raspibuddy4201@gmail.com  
AuthPass=  
FromLineOverride=YES  
UseSTARTTLS=YES  
UseTLS=YES  
  
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos  
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

We can also test it by sending a test mail by issuing below command, you shall get the mail on the mentioned email address if everything is working fine:

```
echo "Hello saddam" | mail -s "Testing..." saddam4201@gmail.com
```

The Python Program of this project plays a very important role to perform all the operations. First of all, we include required libraries for email, initialize variables and define pins for PIR, LED and other components. For sending simple email, smtplib is enough but if you want to send mail in cleaner way with subject line, attachment etc. then you need to use MIME (Multipurpose Internet Mail Extensions).

```
import RPi.GPIO as gpio

import picamera

import time

import smtplib

from email.MIMEMultipart import MIMEMultipart

from email.MIMEText import MIMEText

from email.MIMEBase import MIMEBase

from email import encoders

from email.mime.image import MIMEImage
```

After it, we have initialized mail and define mail address and messages:

```
fromaddr = "raspiduino4201@gmail.com"

toaddr = "saddam4201@gmail.com"

mail = MIMEMultipart()

mail['From'] = fromaddr

mail['To'] = toaddr

mail['Subject'] = "Attachment"

body = "Please find the attachment"
```

Then we have created def sendMail(data) function for sending mail:

```
def sendMail(data):  
    mail.attach(MIMEText(body, 'plain'))  
    print data  
    dat='%s.jpg'%data  
    print dat  
    attachment = open(dat, 'rb')  
    image=MIMEImage(attachment.read())  
    attachment.close()  
    mail.attach(image)  
    server = smtplib.SMTP('smtp.gmail.com', 587)  
    server.starttls()  
    server.login(fromaddr, "your password")  
    text = mail.as_string()  
    server.sendmail(fromaddr, toaddr, text)  
    server.quit()
```

Function def capture_image() is created to capture the image of intruder with time and date.

```
def capture_image():  
    data= time.strftime("%d_%b_%Y|%H:%M:%S")  
    camera.start_preview()  
    time.sleep(5)  
    print data  
P:F-LTL-UG/03/R1
```

```
camera.capture('%s.jpg'%data)

camera.stop_preview()

time.sleep(1)

sendMail(data)
```

Then we initialized the Picamera with some of its settings:

```
camera = picamera.PiCamera()

camera.rotation=180

camera.awb_mode= 'auto'

camera.brightness=55
```

And now in last, we have read PIR sensor output and when its goes high Raspberry Pi calls the capture_image() function to capture the image of intruder and send a alert message with the picture of intruder as an attachment. We have used sendmail() insdie capture_image() function for sending the mail.

```
while 1:

    if gpio.input(pir)==1:

        gpio.output(led, HIGH)

        capture_image()

        while(gpio.input(pir)==1):

            time.sleep(1)

        else:

            gpio.output(led, LOW)

            time.sleep(0.01)
```

So this how this Raspberry Pi Security System works, you can also use Ultrasonic sensor or IR sensor to detect the presence of burglar or intruder. Further check the Full Code and demonstration Video below.

Code:

```
import RPi.GPIO as gpio

import picamera

import time


import smtplib

from email.MIMEMultipart import MIMEMultipart
from email.MIMEText import MIMEText
from email.MIMEBase import MIMEBase
from email import encoders
from email.mime.image import MIMEImage


fromaddr = "raspiduino4201@gmail.com" # change the email address accordingly
toaddr = "saddam4201@gmail.com"


mail = MIMEMultipart()


mail['From'] = fromaddr
mail['To'] = toaddr


P:F-LTL-UG/03/R1
```



```
mail['Subject'] = "Attachment"
```

```
body = "Please find the attachment"
```

```
led=17
```

```
pir=18
```

```
HIGH=1
```

```
LOW=0
```

```
gpio.setwarnings(False)
```

```
gpio.setmode(gpio.BCM)
```

```
gpio.setup(led, gpio.OUT)      # initialize GPIO Pin as outputs
```

```
gpio.setup(pir, gpio.IN)      # initialize GPIO Pin as input
```

```
data=""
```

```
def sendMail(data):
```

```
    mail.attach(MIMEText(body, 'plain'))
```

```
    print data
```

```
    dat='%s.jpg'%data
```

```
    print dat
```

```
    attachment = open(dat, 'rb')
```

```
    image=MIMEImage(attachment.read())
```

```
    attachment.close()
```

```
    mail.attach(image)
```

```
    server = smtplib.SMTP('smtp.gmail.com', 587)
```

```
    server.starttls()
```

```
P:F-LTL-UG/03/R1
```

```
server.login(fromaddr, "your password")  
  
text = mail.as_string()  
  
server.sendmail(fromaddr, toaddr, text)  
  
server.quit()
```

```
def capture_image():  
  
    data= time.strftime("%d_%b_%Y|%H:%M:%S")  
  
    camera.start_preview()  
  
    time.sleep(5)  
  
    print data  
  
    camera.capture('%s.jpg'%data)  
  
    camera.stop_preview()  
  
    time.sleep(1)  
  
    sendMail(data)
```

```
gpio.output(led , 0)  
  
camera = picamera.PiCamera()  
  
camera.rotation=180  
  
camera.awb_mode= 'auto'  
  
camera.brightness=55  
  
while 1:  
  
    if gpio.input(pir)==1:  
  
        gpio.output(led, HIGH)  
  
        capture_image()  
P:F-LTL-UG/03/R1
```

```
while(gpio.input(pir)==1):  
    time.sleep(1)  
    else:  
        gpio.output(led, LOW)  
        time.sleep(0.01)
```