



**ARYA College of Engineering & Information Technology**

(Approved by AICTE & Affiliated to RTU)

SP-42, RIICO Industrial Area, Delhi Road, Kukas, Jaipur - 302028 (Raj.) India • Tel : 0141-6604555 (30 Lines)  
Toll Free No. : 1800-266-2000 • Website : [www.aryacollege.in](http://www.aryacollege.in)

# IOT Lab Manual

(Lab Code: 7CS4-21)

7<sup>th</sup> Semester, 4<sup>th</sup> Year



**Department of Computer Science Engineering**

**Session: 2023-24**

---

## TABLE OF CONTENT

S. No.	Topic/Name of Experiment	Page Number
<b>GENERAL DETAILS</b>		
1	Vision & Mission of Institute and Department	iii
2	RTU Syllabus and Marking Scheme	iv
3	Lab Outcomes and its Mapping with POs and PSOs	v-vii
4	Rubrics of Lab	viii-ix
5	Lab Conduction Plan	x
6	Lab Rotar Plan	xi
7	General Lab Instructions	xii-xiii
8	Lab Specific Safety Rules	xiii
<b>LIST OF EXPERIMENTS (AS PER RTU SYLLABUS)</b>		
1	Zero Lab	1-4
2	Introduction: Objective, scope and outcome of the course.	5
3	Start Raspberry Pi and try various Linux commands in command terminal window: ls, cd, touch, mv, rm, man, mkdir, rmdir, tar, gzip, cat, more, less, ps, sudo, cron, chown, chgrp, ping etc	6-7
4	Run some python programs on Pi like: a) Read your name and print Hello message with name b) Read two numbers and print their sum, difference, product and division. c) Word and character count of a given string. d) Area of a given shape (rectangle, triangle and circle) reading shape and appropriate values from standard input.	8-10
5	Run some python programs on Pi like: a) Print a name 'n' times, where name and n are read from standard input, using for and while loops. b) Handle Divided by Zero Exception. c) Print current time for 10 times with an interval of 10 seconds. d) Read a file line by line and print the word count of each line.	11-13
6	a) Light an LED through Python program b) Get input from two switches and switch on corresponding LEDs c) Flash an LED at a given on time and off time cycle, where the two times are taken from a file.	14-28
7	a) Flash an LED based on cron output (acts as an alarm) b) Switch on a relay at a given time using cron, where the relay's contact terminals are connected to a load. c) Get the status of a bulb at a remote place (on the LAN) through web.	29-35
8	Describe gateway-as-a-service deployment in IoT toolkit.	36-37
9	To write a program for LDR to vary the light intensity of LED using Arduino	38-39

## **INSTITUTE VISION & MISSION**

### **VISION**

To grow exponentially from a local body to a worldwide education “AGGLOMERATE ENCOMPASSING SCIENTIFIC, TECHNICAL, COMMERCIAL AND LITERARY DISCIPLINES” to fulfill the national and international requirements and to expand the knowledge horizon beyond the current FORMIDABLE BOUNDARIES.

### **MISSION**

**To become a leading Educational Institute delivering Technocrats, Administrators (Managers) and Entrepreneurs capable of transforming India to a “TECHNOLOGICAL AND ECONOMICAL SUPERPOWER” besides establishing itself as an Educational Hub of worldwide acclaim**

## **DEPARTMENT VISION & MISSION**

### **VISION**

The department vision is clearly defined and is in line with the college’s vision. The vision of the department is: To evolve as a center of academic excellence with firm determination towards modeling industry oriented technocrats, who would be leading the world in the arena of Computer science and Engineering.

### **MISSION**

This mission of the Department is concise and supports the College’s mission. The mission of the Computer Science and Engineering Department is as follows:

MD1: To provide profound understanding of fundamentals related to computer science and engineering.

MD2: To establish a continuous Industry Institute Interaction, participation and collaboration for enhancing Knowledge of various techniques, tools and skills in ICT.

MD3: To build human values, social values, entrepreneurship skills and professional ethics among the upcoming ICT technocrats.

MD4: To focus towards research and lifelong learning that will enable them to succeed as computer Engineers in society

**RTU SYLLABUS AND MARKING SCHEME**

<b>7CS4-21: IOT LAB</b>	
<b>Credit: 1</b>	<b>Max. Marks: 100 (IA:60, ETE:40)</b>
<b>0L+0T+2P</b>	<b>End Term Exam: 2 Hours</b>
<b>S. No.</b>	<b>NAME OF EXPERIMENTS</b>
1	Introduction: Objective, scope and outcome of the course.
2	Start Raspberry Pi and try various Linux commands in command terminal window: ls, cd, touch, mv, rm, man, mkdir, rmdir, tar, gzip, cat, more, less, ps, sudo, cron, chown, chgrp, ping etc
3	Run some python programs on Pi like: a) Read your name and print Hello message with name b) Read two numbers and print their sum, difference, product and division. c) Word and character count of a given string. d) Area of a given shape (rectangle, triangle and circle) reading shape and appropriate values from standard input.
4	Run some python programs on Pi like: a) Print a name 'n' times, where name and n are read from standard input, using for and while loops. b) Handle Divided by Zero Exception. c) Print current time for 10 times with an interval of 10 seconds. d) Read a file line by line and print the word count of each line.
5	a) Light an LED through Python program b) Get input from two switches and switch on corresponding LEDs c) Flash an LED at a given on time and off time cycle, where the two times are taken from a file.
6	a) Flash an LED based on cron output (acts as an alarm) b) Switch on a relay at a given time using cron, where the relay's contact terminals are connected to a load. c) Get the status of a bulb at a remote place (on the LAN) through web.
7	Describe gateway-as-a-service deployment in IoT toolkit.
8	To write a program for LDR to vary the light intensity of LED using Arduino

**EVALUATION SCHEME**

<b>I+II Mid Term Examination</b>			<b>Attendance and performance</b>			<b>End Term Examination</b>			<b>Total Marks</b>
<b>Experiment</b>	<b>Viva</b>	<b>Total</b>	<b>Attendance</b>	<b>Performance</b>	<b>Total</b>	<b>Experiment</b>	<b>Viva</b>	<b>Total</b>	
30	10	40	10	30	40	30	10	40	100

**DISTRIBUTION OF MARKS FOR EACH EXPERIMENT**

<b>Attendance</b>	<b>Record</b>	<b>Performance</b>	<b>Total</b>
2	3	5	10

**LAB OUTCOME AND ITS MAPPING WITH PO & PSO****LAB OUTCOMES**

After completion of this course, students will be able to –

<b>7CS4-21.1</b>	To define the various terminal commands used in developing IOT applications.
<b>7CS4-21.2</b>	To develop the python scripts used in IOT applications.
<b>7CS4-21.3</b>	To apply the logics of IOT for designing IOT applications
<b>7CS4-21.4</b>	To make a project to solve real life society-based problem and demonstrate following PO related capabilities: a. Improve team working skill b. Improve communication skill c. Improve ethics (i.e. plagiarism, copy others results) d. Lifelong learning attitude

**LO-PO-PSO MAPPING MATRIX OF COURSE**

<b>LO/PO/ PSO</b>	<b>PO1</b>	<b>PO2</b>	<b>PO3</b>	<b>PO4</b>	<b>PO5</b>	<b>PO6</b>	<b>PO7</b>	<b>PO8</b>	<b>PO9</b>	<b>PO10</b>	<b>PO11</b>	<b>PO12</b>	<b>PSO1</b>	<b>PSO2</b>	<b>PSO3</b>
<b>7CS4-21.1</b>	3	-	-	-	-	-	-	-	-	-	-	-	2	-	-
<b>7CS4-21.2</b>	-	3	-	-	-	-	-	-	-	-	-	-	-	-	3
<b>7CS4-21.3</b>	-	-	3	-	-	-	-	-	-	-	-	-	-	3	-
<b>7CS4-21.4</b>	-	-	3	-	3	3	3	3	3	3	3	3	3	2	3

**PROGRAM OUTCOMES (POs)**

<b>PO1</b>	<b>Engineering knowledge:</b> Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems
<b>PO2</b>	<b>Problem analysis:</b> Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

<b>PO3</b>	<b>Design/development of solutions:</b> Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
<b>PO4</b>	<b>Conduct investigations of complex problems:</b> Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
<b>PO5</b>	<b>Modern tool usage:</b> Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
<b>PO6</b>	<b>The engineer and society:</b> Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
<b>PO7</b>	<b>Environment and sustainability:</b> Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
<b>PO8</b>	<b>Ethics:</b> Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
<b>PO9</b>	<b>Individual and team work:</b> Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
<b>PO10</b>	<b>Communication:</b> Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
<b>PO11</b>	<b>Project management and finance:</b> Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
<b>PO12</b>	<b>Life-long learning:</b> Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

**PROGRAM SPECIFIC OUTCOMES (PSOs)**

<b>PSO1</b>	Students are able to model, design and implement Software and hardware communication network systems using knowledge of modern tools and techniques to serve the society as professional computer engineers.
<b>PSO2</b>	Students are able to develop software engineering system of varying complexity and Innovative products in emerging areas of Computer Engineering..

**RUBRICS FOR LAB****Laboratory Evaluation Rubrics:**

S. No.	Criteria	Sub Criteria and Marks Distribution			Outstanding (>90%)	Admirable (70-90%)	Average (40-69%)	Inadequate (<40%)
		Mid-Term	End-Team	Continues Evaluation				
A	PERFORMANCE (PO1, PO8, PO9)	Procedure Followed  M.M. 50 = 3 M.M. 75 = 4 M.M. 100 = 6	Procedure Followed  M.M. 50 = 3 M.M. 75 = 4 M.M. 100 = 6	Procedure Followed  M.M. 50 = 1 M.M. 75 = 2 M.M. 100 = 2	<ul style="list-style-type: none"> <li>• All possible system and Input/ Output variables are taken into account</li> <li>• Performance measures are properly defined</li> <li>• Experimental scenarios are very well defined</li> </ul>	<ul style="list-style-type: none"> <li>• Most of the system and Input/ Output variables are taken into account</li> <li>• Most of the Performance measures are properly defined</li> <li>• Experimental scenarios are defined correctly</li> </ul>	<ul style="list-style-type: none"> <li>• Some of the system and Input/ Output variables are taken into account</li> <li>• Some of the Performance measures are properly defined</li> <li>• Experimental scenarios are defined but not sufficient</li> </ul>	<ul style="list-style-type: none"> <li>• System and Input/ Output variables are not defined</li> <li>• Performance measures are not properly defined</li> <li>• Experimental scenarios not defined</li> </ul>
		Individual/Team Work  M.M. 50 = 3 M.M. 75 = 4 M.M. 100 = 6	Individual/Team Work  M.M. 50 = 3 M.M. 75 = 4 M.M. 100 = 6	Individual/Team Work  M.M. 50 = 1 M.M. 75 = 2 M.M. 100 = 2	<ul style="list-style-type: none"> <li>• Coordination among the group members in performing the experiment was excellent</li> </ul>	<ul style="list-style-type: none"> <li>• Coordination among the group members in performing the experiment was good</li> </ul>	<ul style="list-style-type: none"> <li>• Coordination among the group members in performing the experiment was average</li> </ul>	<ul style="list-style-type: none"> <li>• Coordination among the group members in performing the experiment was very poor</li> </ul>
		Precision in data collection  M.M. 50 = 3 M.M. 75 = 4 M.M. 100 = 6	Precision in data collection  M.M. 50 = 3 M.M. 75 = 4 M.M. 100 = 6	Precision in data collection  M.M. 50 = 2 M.M. 75 = 2 M.M. 100 = 4	<ul style="list-style-type: none"> <li>• Data collected is correct in size and from the experiment performed</li> </ul>	<ul style="list-style-type: none"> <li>• Data collected is appropriate in size and but not from proper sources.</li> </ul>	<ul style="list-style-type: none"> <li>• Data collected is not so appropriate in size and but from proper sources.</li> </ul>	<ul style="list-style-type: none"> <li>• Data collected is neither appropriate in size and nor from proper sources</li> </ul>
B	LAB RECORD/WITTEN WORK (PO1, PO8, PO10)	NA	NA	Timing of Evaluation of Experiment  M.M. 50 = 3 M.M. 75 = 4 M.M. 100 = 6	<ul style="list-style-type: none"> <li>• On the Same Date of Performance</li> </ul>	<ul style="list-style-type: none"> <li>• On the Next Turn from Performance</li> </ul>	<ul style="list-style-type: none"> <li>• Before Dead Line</li> </ul>	<ul style="list-style-type: none"> <li>• On the Dead Line</li> </ul>
		Data Analysis  M.M. 50 = 3 M.M. 75 = 5 M.M. 100 = 6	Data Analysis  M.M. 50 = 3 M.M. 75 = 5 M.M. 100 = 6	Data Analysis  M.M. 50 = 2 M.M. 75 = 3 M.M. 100 = 4	<ul style="list-style-type: none"> <li>• Data collected is exhaustively analyzed &amp; appropriate features are selected</li> </ul>	<ul style="list-style-type: none"> <li>• Data collected is analyzed &amp; but appropriate features are not selected</li> </ul>	<ul style="list-style-type: none"> <li>• Data collected is not analyzed properly. • Features selected are not appropriate</li> </ul>	<ul style="list-style-type: none"> <li>• Data collected is not analyzed &amp; the features are not selected</li> </ul>

		Results and Discussion  M.M. 50 = 3 M.M. 75 = 5 M.M. 100 = 6	Results and Discussion  M.M. 50 = 3 M.M. 75 = 5 M.M. 100 = 6	Results and Discussion  M.M. 50 = 2 M.M. 75 = 3 M.M. 100 = 4	<ul style="list-style-type: none"> <li>• All results are very well presented with all variables</li> <li>• Well prepared neat diagrams/plots/ tables for all performance measured</li> <li>• Discussed critically behavior of the system with reference to performance measures</li> <li>• Very well discussed pros n cons of outcome</li> </ul>	<ul style="list-style-type: none"> <li>• All results presented but not all variables mentioned</li> <li>• Prepared diagrams /plots/ tables for all performance measured but not so neat</li> <li>• Discussed behavior of the system with reference to performance measures but not critical</li> <li>• Discussed pros n cons of outcome in brief</li> </ul>	<ul style="list-style-type: none"> <li>• Partial results are included</li> <li>• Prepared diagrams /plots/ tables partially for the performance measures</li> <li>• Behavior of the system with reference to performance measures has been superficially presented</li> <li>• Discussed pros n cons of outcome but not so relevant</li> </ul>	<ul style="list-style-type: none"> <li>• Results are included but not as per experimental scenarios</li> <li>• No proper diagrams /plots/ tables are prepared</li> <li>• Behavior of the system with reference to performance measures has not been presented</li> <li>• Did not discuss pros n cons of outcome</li> </ul>
C	VIVA (PO1, PO10)	Way of presentation  M.M. 50 = 2.5 M.M. 75 = 4 M.M. 100 = 5	Way of presentation  M.M. 50 = 2.5 M.M. 75 = 4 M.M. 100 = 5	Way of presentation  M.M. 50 = 2 M.M. 75 = 3 M.M. 100 = 4	<ul style="list-style-type: none"> <li>• Presentation was very good</li> </ul>	<ul style="list-style-type: none"> <li>• Presentation was good</li> </ul>	<ul style="list-style-type: none"> <li>• Presentation was satisfactory</li> </ul>	<ul style="list-style-type: none"> <li>• Presentation was poor</li> </ul>
		Concept Explanation  M.M. 50 = 2.5 M.M. 75 = 4 M.M. 100 = 5	Concept Explanation  M.M. 50 = 2.5 M.M. 75 = 4 M.M. 100 = 5	Concept Explanation  M.M. 50 = 2 M.M. 75 = 3 M.M. 100 = 4	<ul style="list-style-type: none"> <li>• Conceptual explanation was excellent</li> </ul>	<ul style="list-style-type: none"> <li>• Conceptual explanation was good</li> </ul>	<ul style="list-style-type: none"> <li>• Conceptual explanation was somewhat good</li> </ul>	<ul style="list-style-type: none"> <li>• Conceptual explanation was Poor</li> </ul>
D	ATTENDANCE	NA	NA	Attendance  M.M. 50 = 5 M.M. 75 = 8 M.M. 100 = 10	<ul style="list-style-type: none"> <li>• Present more than 90% of lab sessions</li> </ul>	<ul style="list-style-type: none"> <li>• Present more than 75% of lab sessions</li> </ul>	<ul style="list-style-type: none"> <li>• Present more than 60% of lab sessions</li> </ul>	<ul style="list-style-type: none"> <li>• Present in less than 60% lab sessions</li> </ul>

## LAB CONDUCTION PLAN

**Total number of Experiments - 8**

**Total number of turns required-8**

**Number of turns required for: -**

Experiment Number	Scheduled Week
Experiment -1	Week 1
Experiment -2	Week 2
Experiment -3	Week 3
Experiment -4	Week 4
Experiment -5	Week 5
<b>I Mid Term</b>	<b>Week 6</b>
Experiment -6	Week 7
Experiment -7	Week 8
Experiment -8	Week 9
<b>II Mid Term</b>	<b>Week 10</b>

### DISTRIBUTION OF LAB HOURS

S. No.	Activity	Distribution of Lab Hours	
		Time (180 minute)	Time (120 minute)
1	Attendance	5	5
2	Explanation of Experiment & Logic	30	30
3	Performing the Experiment	60	30
4	File Checking	40	20
5	Viva/Quiz	30	20
6	Solving of Queries	15	15

**LAB ROTAR PLAN****ROTOR-1**

<b>Ex. No.</b>	<b>NAME OF EXPERIMENTS</b>
1	Introduction: Objective, scope and outcome of the course.
2	Start Raspberry Pi and try various Linux commands in command terminal window: ls, cd, touch, mv, rm, man, mkdir, rmdir, tar, gzip, cat, more, less, ps, sudo, cron, chown, chgrp, ping etc
3	Run some python programs on Pi like: a) Read your name and print Hello message with name b) Read two numbers and print their sum, difference, product and division. c) Word and character count of a given string. d) Area of a given shape (rectangle, triangle and circle) reading shape and appropriate values from standard input.
4	Run some python programs on Pi like: a) Print a name 'n' times, where name and n are read from standard input, using for and while loops. b) Handle Divided by Zero Exception. c) Print current time for 10 times with an interval of 10 seconds. d) Read a file line by line and print the word count of each line.
5	a) Light an LED through Python program b) Get input from two switches and switch on corresponding LEDs c) Flash an LED at a given on time and off time cycle, where the two times are taken from a file.

**ROTOR-2**

<b>Ex. No.</b>	<b>NAME OF EXPERIMENTS</b>
6	a) Flash an LED based on cron output (acts as an alarm) b) Switch on a relay at a given time using cron, where the relay's contact terminals are connected to a load. c) Get the status of a bulb at a remote place (on the LAN) through web.
7	Describe gateway-as-a-service deployment in IoT toolkit.
8	To write a program for LDR to vary the light intensity of LED using Arduino

## GENERAL LAB INSTRUCTIONS

### **DO'S**

1. Enter the lab on time and leave at proper time.
2. Wait for the previous class to leave before the next class enters.
3. Keep the bag outside in the respective racks.
4. Utilize lab hours in the corresponding.
5. Turn off the machine before leaving the lab unless a member of lab staff has specifically told you not to do so.
6. Leave the labs at least as nice as you found them.
7. If you notice a problem with a piece of equipment (e.g., a computer doesn't respond) or the room in general (e.g., cooling, heating, lighting) please report it to lab staff immediately. Do not attempt to fix the problem yourself.

### **DON'TS**

1. Don't abuse the equipment.
2. Do not adjust the heat or air conditioners. If you feel the temperature is not properly set, inform lab staff; we will attempt to maintain a balance that is healthy for people and machines.
3. Do not attempt to reboot a computer. Report problems to lab staff.
4. Do not remove or modify any software or file without permission.
5. Do not remove printers and machines from the network without being explicitly told to do so by lab staff.
6. Don't monopolize equipment. If you're going to be away from your machine for more than 10 or 15 minutes, log out before leaving. This is both for the security of your account, and to ensure that others are able to use the lab resources while you are not.
7. Don't use internet, internet chat of any kind in your regular lab schedule.
8. Do not download or upload of MP3, JPG or MPEG files.
9. No games are allowed in the lab sessions.
10. No hardware including USB drives can be connected or disconnected in the labs without prior permission of the lab in-charge.

11. No food or drink is allowed in the lab or near any of the equipment. Aside from the fact that it leaves a mess and attracts pests, spilling anything on a keyboard or other piece of computer equipment could cause permanent, irreparable, and costly damage. (and in fact *has*) If you need to eat or drink, take a break and do so in the canteen.
12. Don't bring any external material in the lab, except your lab record, copy and books.
13. Don't bring the mobile phones in the lab. If necessary, then keep them in silence mode.
14. Please be considerate of those around you, especially in terms of noise level. While labs are a natural place for conversations of all types, kindly keep the volume turned down.
15. If you are having problems or questions, please go to either the faculty, lab in-charge or the lab supporting staff. They will help you. We need your full support and cooperation for smooth functioning of the lab.

### **LAB SPECIFIC SAFETY RULES**

#### **Before entering in the lab**

1. All the students are supposed to prepare the theory regarding the next experiment/ Program.
2. Students are supposed to bring their lab records as per their lab schedule.
3. Previous experiment/program should be written in the lab record.
4. If applicable trace paper/graph paper must be pasted in lab record with proper labeling.
5. All the students must follow the instructions, failing which he/she may not be allowed in the lab.

#### **While working in the lab**

1. Adhere to experimental schedule as instructed by the lab in-charge/faculty.
2. Get the previously performed experiment/ program signed by the faculty/ lab in charge.
3. Get the output of current experiment/program checked by the faculty/ lab in charge in the lab copy.
4. Each student should work on his/her assigned computer at each turn of the lab.

5. Take responsibility of valuable accessories.

## Zero Lab

---

### The Internet of Things (IoT)

The term IoT, or Internet of Things, refers to the collective network of connected devices and the technology that facilitates communication between devices and the cloud, as well as between the devices themselves. Thanks to the advent of inexpensive computer chips and high bandwidth telecommunication, we now have billions of devices connected to the internet. This means everyday devices like toothbrushes, vacuums, cars, and machines can use sensors to collect data and respond intelligently to users.

The Internet of Things integrates everyday “things” with the internet. Computer Engineers have been adding sensors and processors to everyday objects since the 90s. However, progress was initially slow because the chips were big and bulky. Low power computer chips called RFID tags were first used to track expensive equipment. As computing devices shrank in size, these chips also became smaller, faster, and smarter over time.

Some common day-to-day examples could be:

- Temperatures in refrigeration or food heating units in the food and beverage industry.
- Assistance with the control of temperature and humidity levels.
- Detection of gas and dust levels.
- Monitoring of water levels and herd locations for agricultural purposes.
- Different applications in the automotive, aviation and nautical sectors such as the sensing of tyre pressures for trucking fleets.

### How does IoT work?

A typical IoT system works through the real-time collection and exchange of data. An IoT system has three components:

#### Smart devices

This is a device, like a television, security camera, or exercise equipment that has been given computing capabilities. It collects data from its environment, user inputs, or usage patterns and communicates data over the internet to and from its IoT application.

#### IoT application

An IoT application is a collection of services and software that integrates data received from various IoT devices. It uses machine learning or artificial intelligence (AI) technology to analyse this data and make informed decisions. These decisions are communicated back to the IoT device and the IoT device then responds intelligently to inputs.

**A graphical user interfaces**

The IoT device or fleet of devices can be managed through a graphical user interface. Common examples include a mobile application or website that can be used to register and control smart devices.

**What is Industrial IoT?**

Industrial IoT (IIoT) refers to smart devices used in manufacturing, retail, health, and other enterprises to create business efficiencies. Industrial devices, from sensors to equipment, give business owners detailed, real-time data that can be used to improve business processes. They provide insights on supply chain management, logistics, human resource, and production – decreasing costs and increasing revenue streams.

Let's look at existing smart industrial systems in different verticals:

**Manufacturing**

Enterprise IoT in manufacturing uses predictive maintenance to reduce unplanned downtime and wearable technology to improve worker safety. IoT applications can predict machine failure before it happens, reducing production downtime. Wearables in helmets and wristbands, as well as computer vision cameras, are used to warn workers about potential hazards.

**Automobile**

Sensor-driven analytics and robotics increase efficiency in automobile manufacturing and maintenance. For example, industrial sensors are used to provide 3D real-time images of internal vehicle components. Diagnostics and troubleshooting can be done much faster while the IoT system orders replacement parts automatically.

**Logistics and transport**

Commercial and Industrial IoT devices can help with supply chain management, including inventory management, vendor relationships, fleet management, and scheduled maintenance. Shipping companies use Industrial IoT applications to keep track of assets and optimize fuel consumption on shipping routes. The technology is especially useful for tight temperature control in refrigerated containers. Supply chain managers make informed predictions through smart routing and rerouting algorithms.

**Retail**

Amazon is driving innovation in automation and human-machine collaboration in retail. Amazon facilities make use of internet-connected robots for tracking, locating, sorting, and moving products.

**What are IoT technologies?**

Technologies used in IoT systems may include:

**Edge computing**

Edge computing refers to the technology used to make smart devices do more than just send or receive data to their IoT platform. It increases the computing power at the edges of an IoT network, reducing communication latency and improving response time.

**Cloud computing**

Cloud technology is used for remote data storage and IoT device management – making the data accessible to multiple devices in the network.

**Machine learning**

Machine learning refers to the software and algorithms used to process data and make real-time decisions based on that data. These machine learning algorithms can be deployed in the cloud or at the edge.

**Most Popular IoT Devices****1. Google Home Voice Controller**

Google Home voice controller is one of the most popular IoT devices out there today. It provides voice-enabled services like alarms, lights, thermostats, volume control and lots more.

**2. Amazon Echo Plus Voice Controller**

Amazon Echo Plus voice controller is another popular and reliable IoT device on the market. It provides voice-enabled services like answering phone calls, setting timers and alarms, checking the weather, and lots more.

**3. August Doorbell Cam**

August Doorbell Cam is an IoT device that allows you to answer your door from any remote location. It constantly captures motion changes and suspicious activity in your doorstep.

#### **4. August Smart Lock**

August Smart Lock is a proven and reliable security IoT device that helps users to manage their doors from any remote location. It helps keep thieves away and provides an extra layer of security for your home.

#### **5. Foobot**

Foobot is an IoT device that can accurately measure indoor pollution. It helps to improve the air quality in houses, cafes, workplaces, and other indoor public spaces.

## EXPERIMENT-1

---

### OBJECTIVE

Introduction: Objective, scope and outcome of the course.

### THEORY

The Internet of Things (IoT) describes the network of physical objects— “things”—that are embedded with sensors, software, and other technologies for the purpose of connecting and exchanging data with other devices and systems over the internet. These devices range from ordinary household objects to sophisticated industrial tools. With more than 7 billion connected IoT devices today.

### Industrial IoT

Industrial IoT (IIoT) refers to the application of IoT technology in industrial settings, especially with respect to instrumentation and control of sensors and devices that engage cloud technologies. Refer to this for a good example of IIoT

### SCOPE

The advancements in Artificial Intelligence and Machine Learning have made the automation of IoT devices easy. Basically, AI and ML programs are combined with IoT devices to give them proper automation. Due to this, IoT has also expanded its area of application in various sectors.

### OUTCOMES

1. Explain the definition and usage of the term “Internet of Things” in different contexts
2. Understand the key components that make up an IoT system
3. Differentiate between the levels of the IoT stack and be familiar with the key technologies and protocols employed at each layer of the stack
4. Apply the knowledge and skills acquired during the course to build and test a complete, working IoT system involving prototyping, programming and data analysis
5. Understand where the IoT concept fits within the broader ICT industry and possible future trends
6. Appreciate the role of big data, cloud computing and data analytics in a typical IoT system

## EXPERIMENT-2

---

### OBJECTIVE

Start Raspberry Pi and try various Linux commands in command terminal window:

ls, cd, touch, mv, rm, man, mkdir, rmdir, tar, gzip, cat, more, less, ps, sudo, cron, chown, chgrp, ping etc

### COMMANDS: -

1) **touch**: Create a new file or update its timestamp.

- **Syntax**: touch [OPTION]...[FILE]
- **Example**: Create empty files called 'file1' and 'file2'
  - \$ touch file1 file2

2) **cat**: Concatenate files and print to stdout.

- **Syntax**: cat [OPTION]...[FILE]
- **Example**: Create file1 with entered content
  - \$ cat > file1
  - Hello
  - ^D

3) **cp**: Copy files

- **Syntax**: cp [OPTION]source destination
- **Example**: Copies the contents from file1 to file2 and contents of file1 is retained
  - \$ cp file1 file2

4) **mv**: Move files or rename files

- **Syntax**: mv [OPTION]source destination
- **Example**: Create empty files called 'file1' and 'file2'
  - \$ mv file1 file2

5) **rm**: Remove files and directories

- **Syntax**: rm [OPTION]...[FILE]
- **Example**: Delete file1
  - \$ rm file1

6) **mkdir**: Make directory

- **Syntax**: mkdir [OPTION] directory
- **Example**: Create directory called dir1
  - \$ mkdir dir1

**7) rmdir:** Remove a directory

- **Syntax:** rmdir [OPTION] directory
- **Example:** Create empty files called 'file1' and 'file2'
  - \$ rmdir dir1

**8) cd:** Change directory

- **Syntax:** cd [OPTION] directory
- **Example:** Change working directory to dir1
  - \$ cd dir1

**9) pwd:** Print the present working directory

- **Syntax:** pwd [OPTION]
- **Example:** Print 'dir1' if a current working directory is dir1
  - \$ pwd

**10) ls or list**

- **Syntax:** \$ ls
- **Example:** The `ls` command prints out the contents of a directory

**11) more**

- more is one of the oldest terminal pagers in the UNIX ecosystem. Originally, more could only scroll down, but now we can use it to scroll up one screen-full at a time, and scroll down either one line or one screen-full:
- **more filename.txt**

**12) chown**

- chown [OPTIONS] USER [: GROUP] FILE(s)
- USER is the user name or the user ID (UID) of the new owner. GROUP is the name of the new group or the group ID (GID). FILE(s) is the name of one or more files, directories or links. Numeric IDs should be prefixed with the + symbol.

### EXPERIMENT-3

---

#### OBJECTIVE

Run some python programs on Pi like:

**(a) Read your name and print Hello message with name**

#### PROGRAM

```
person = input('Enter your name: ')
print('Hello', person)
```

#### OUTPUT

Enter your name: RAM

Hello RAM

**(b) Read two numbers and print their sum, difference, product and division.**

#### PROGRAM

```
# Python program to perform Addition Subtraction Multiplication
# and Division of two numbers
```

```
num1 = int(input("Enter First Number: "))
num2 = int(input("Enter Second Number: "))

print("Enter which operation would you like to perform?")
ch = input("Enter any of these char for specific operation +,-,*,/: ")

result = 0
if ch == '+':
    result = num1 + num2
elif ch == '-':
    result = num1 - num2
elif ch == '*':
    result = num1 * num2
elif ch == '/':
    result = num1 / num2
else:
    print("Input character is not recognized!")

print(num1, ch, num2, ":", result)
```

### **OUTPUT 1: ADDITION**

Enter First Number: 100

Enter Second Number: 5

Enter which operation would you like to perform?

Enter any of these char for specific operation +,-,\*,/:

100 + 5 : 105

### **OUTPUT 2: DIVISION**

Enter First Number: 20

Enter Second Number: 5

Enter which operation would you like to perform?

Enter any of these char for specific operation +,-,\*,/:

20 / 5 : 4.0

### **OUTPUT 3: SUBTRACTION**

Enter First Number: 8

Enter Second Number: 7

Enter which operation would you like to perform?

Enter any of these char for specific operation +,-,\*,/:

8 - 7 : 1

### **OUTPUT 4: MULTIPLICATION**

Enter First Number: 6

Enter Second Number: 8

Enter which operation would you like to perform?

Enter any of these char for specific operation +,-,\*,/:

6 \* 8 : 48

**(c) Word and character count of a given string****PROGRAM**

```
def char_frequency(str1):  
    dict = {}  
    for n in str1:  
        keys = dict.keys()  
        if n in keys:  
            dict[n] += 1  
        else:  
            dict[n] = 1  
    return dict  
print(char_frequency('google.com'))
```

**OUTPUT:**

```
{'o': 3, '!': 1, 'g': 2, 'l': 1, 'e': 1, 'c': 1, 'm': 1}
```

**(d) Area of a given shape (rectangle, triangle and circle) reading shape and appropriate Values.****PROGRAM**

```
from standard input  
# Python Program to find the area of triangle  
a = 5  
b = 6  
c = 7  
# Uncomment below to take inputs from the user  
# a = float(input('Enter first side: '))  
# b = float(input('Enter second side: '))  
# c = float(input('Enter third side: '))  
# calculate the semi-perimeter  
s = (a + b + c) / 2  
# calculate the area  
area = (s*(s-a)*(s-b)*(s-c)) ** 0.5  
print('The area of the triangle is %0.2f %area)
```

**OUTPUT**

The area of the triangle is 14.70

## EXPERIMENT-4

---

### OBJECTIVE

Run some python programs on Pi like:

**a) Print a name 'n' times, where name and n are read from standard input, using for and while loops.**

### PROGRAM

```
def string_print(n):  
    print("THE STRING IS 'Money Heist'")  
    print("The string will be printed", n ,"times")  
    for i in range(n):  
        print("Money Heist")  
#input function  
string_print(5)
```

### OUTPUT

```
THE STRING IS 'Money Heist'  
The string will be printed 5 times  
Money Heist  
Money Heist  
Money Heist  
Money Heist  
Money Heist
```

**b) Handle Divided by Zero Exception**

### PROGRAM

```
n=int(input("Enter the value of n:"))  
d=int(input("Enter the value of d:"))  
c=int(input("Enter the value of c:"))  
try:
```

```
q=n/(d-c)
print("Quotient:",q)
except ZeroDivisionError:
print("Division by Zero!")
```

**OUTPUT**

Enter the value of n:2  
Enter the value of d:5  
Enter the value of c:5  
Division by Zero!

**( C ) Print current time for 10 times to the current time.**

**PROGRAM**

```
Import datetime
x= datetime.datetime.now()
y = x + datetime.timedelta(0,10)
print(x.time())
print(y.time())
```

**OUTPUT**

17:23:46.918031  
17:23:56.918031

**(d) Read a file line by line and print the word count of each line.**

**PROGRAM**

```
L = ["Geeks\n", "for\n", "Geeks\n"]
# writing to file
file1 = open('myfile.txt', 'w')
file1.writelines(L)
```

```
file1.close()
```

```
# Using readlines()
```

```
file1 = open('myfile.txt', 'r')
```

```
Lines = file1.readlines()
```

```
count = 0
```

```
# Strips the newline character
```

```
for line in Lines:
```

```
count += 1
```

```
print("Line{}: {}".format(count, line.strip()))
```

## **OUTPUT**

Line1: Geeks

Line2: for

Line3: Geeks

## EXPERIMENT-5

---

### OBJECTIVE

- (a) Light an LED through Python program.

### SOLUTION

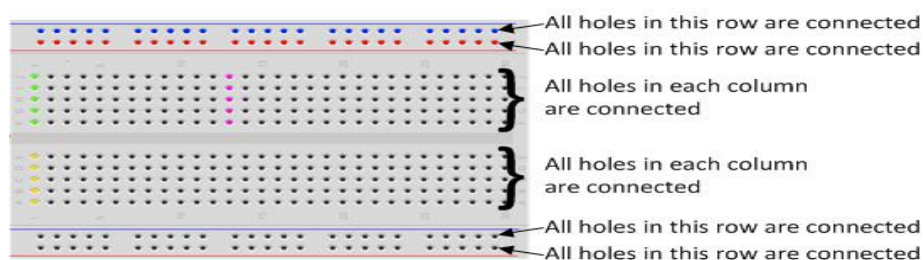
One of the biggest selling points of the Raspberry Pi is its GPIO, or General-Purpose Input/output ports. They are the little pins sticking out of the circuit board and allow you to plug various devices into your Raspberry Pi. With a little programming, you can then control them or detect what they are doing.

In this tutorial I am going to show you how to light an LED. In addition to your Raspberry Pi running Raspbian, what you will need is:

- **A Breadboard**
- **An LED**
- **A 330 ohm resistor**
- **The Breadboard:**

The breadboard is a way of connecting electronic components to each other without having to solder them together. They are often used to test a circuit design before creating a Printed Circuit Board (PCB).

- The holes on the breadboard are connected in a pattern.



With the breadboard in the CamJam EduKit, the top row of holes are all connected together – marked with red dots. And so are the second row of holes – marked with blue dots. The same goes for the two rows of holes at the bottom of the breadboard.

In the middle, the columns of wires are connected together with a break in the middle. So, for example, all the green holes marked are connected together, but they are not connected to the

yellow holes, nor the purple ones. Therefore, any wire you poke into the green holes will be connected to other wires poked into the other green holes.

### The LED

When you pick up the LED, you will notice that one leg is longer than the other. The longer leg (known as the ‘anode’), is always connected to the positive supply of the circuit. The shorter leg (known as the ‘cathode’) is connected to the negative side of the power supply, known as ‘ground’.

LEDs will only work if power is supplied the correct way round (i.e. if the ‘polarity’ is correct). You will not break the LEDs if you connect them the wrong way round – they will just not light. If you find that they do not light in your circuit, it may be because they have been connected the wrong way around.



LED stands for Light Emitting Diode, and glows when electricity is passed through it.

### The Resistor

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



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

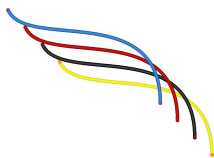
You will be using a 330 $\Omega$  resistor. You can identify the 330 $\Omega$  resistors by the colour bands along the body. The colour coding will depend on how many bands are on the resistors supplied:

- If there are four colour bands, they will be Orange, Orange, Brown, and then Gold.
- If there are five bands, then the colours will be Orange, Orange, Black, Black, Brown.

It does not matter which way round you connect the resistors. Current flows in both ways through them.

### Jumper Wires:

Jumper wires are used on breadboards to ‘jump’ from one connection to another. The ones you will be using in this circuit have different connectors on each end. The end with the ‘pin’ will go into the Breadboard. The end with the piece of plastic with a hole in it will go onto the Raspberry Pi’s GPIO pins.



### The Raspberry Pi's GPIO Pins :

**GPIO** stands for **General Purpose Input Output**. It is a way the Raspberry Pi can control and monitor the outside world by being connected to electronic circuits. The Raspberry Pi is able to control LEDs, turning them on or off, or motors, or many other things. It is also able to detect whether a switch has been pressed, or temperature, or light. In the CamJam EduKit you will learn to control LEDs and a buzzer, and detect when a button has been pressed. The diagram below left shows the pin layout for a Raspberry Pi Models A and B (Rev 2 - the original Rev 1 Pi is slightly different), looking at the Raspberry Pi with the pins in the top right corner. The new 40 pin Raspberry Pi’s shares exactly the same layout of pins for the top 13 rows of GPIO pins.

### Explanation:

So, what is happening in the code? Let’s go through it a line at a time:

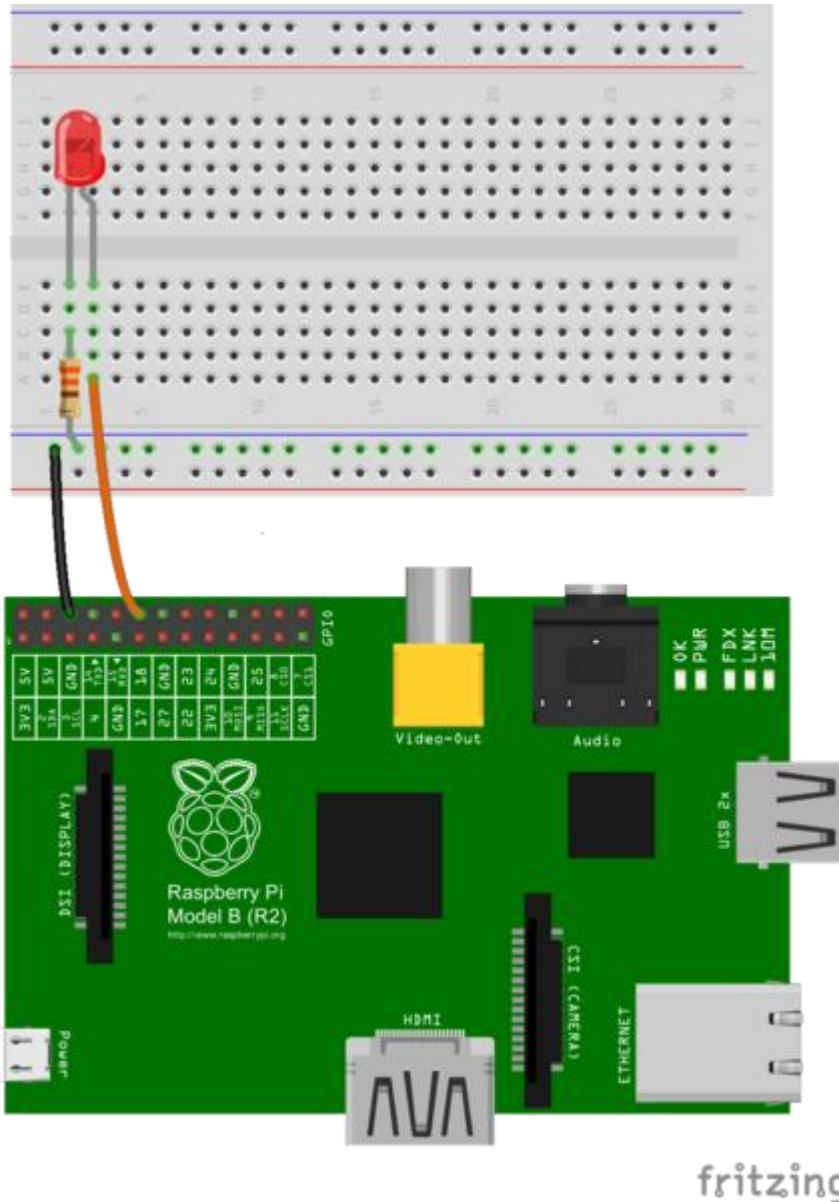
<pre>import RPi.GPIO as GPIO</pre>	<p>The first line tells the Python interpreter (the thing that runs the Python code) that it will be using a ‘library’ that will tell it how to work with the Raspberry Pi’s GPIO pins. A ‘library’ gives a programming language extra commands that can be used to do something different that it previously did not know how to do. This is like adding</p>
------------------------------------	---

	a new channel to your TV so you can watch something different.
<code>import time</code>	Imports the Time library so that we can pause the script later on.
<code>GPIO.setmode(GPIO.BCM)</code>	Each pin on the Raspberry Pi has several different names, so you need to tell the program which naming convention is to be used.
<code>GPIO.setwarnings(False)</code>	This tells Python not to print GPIO warning messages to the screen.
<code>GPIO.setup(18,GPIO.OUT)</code>	This line tells the Python interpreter that pin 18 is going to be used for outputting information, which means you are going to be able to turn the pin 'on' and 'off'.
<code>print "LED on"</code>	This line prints some information to the terminal.
<code>GPIO.output(18,GPIO.HIGH)</code>	This turns the GPIO pin 'on'. What this actually means is that the pin is made to provide power of 3.3volts. This is enough to turn the LED in our circuit on.
<code>time.sleep(1)</code>	Pauses the Python program for 1 second
<code>print "LED off"</code>	This line prints some information to the terminal.
<code>GPIO.output(18,GPIO.LOW)</code>	This turns the GPIO pin 'off', meaning that the pin is no longer supplying any power.

### Building the Circuit:

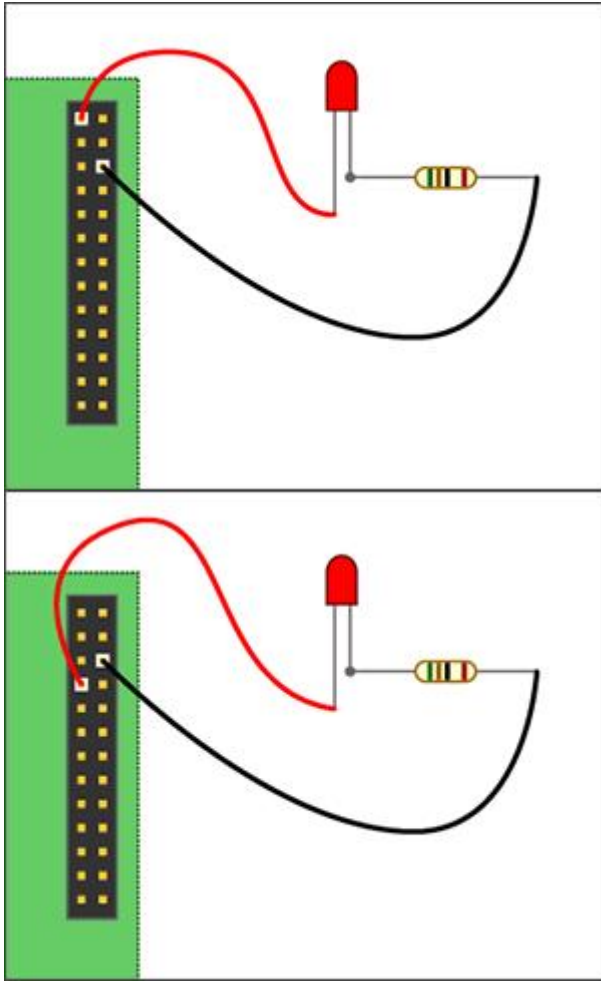
The circuit consists of a power supply (the Raspberry Pi), an LED that lights when the power is applied, and a resistor to limit the current that can flow through the circuit.

You will be using one of the 'ground' (GND) pins to act like the 'negative' or 0 volt ends of a battery. The 'positive' end of the battery will be provided by a GPIO pin. Here we will be using pin 18. When they are 'taken high', which means it outputs 3.3 volts, the LED will light. Now take a look at the circuit diagram below.



You should turn your Raspberry Pi off for the next bit, just in case you accidentally short something out.

- Use one of the jumper wires to connect a ground pin to the rail, marked with blue, on the breadboard. The female end goes on the Raspberry Pi's pin, and the male end goes into a hole on the breadboard.
- Then connect the resistor from the same row on the breadboard to a column on the breadboard, as shown above.
- Next, push the LED's legs into the breadboard, with the long leg (with the kink) on the right.
- Lastly, complete the circuit by connecting pin 18 to the right hand leg of the LED. This is shown here with the orange wire.



### The Code:

You are now ready to write some code to switch the LED on. Turn on your Raspberry Pi and open the terminal window.

Create a new text file “LED.py” by typing the following:

```
nano LED.py
```

Type in the following code:

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(18,GPIO.OUT)
print "LED on"
GPIO.output(18,GPIO.HIGH)
time.sleep(1)
print "LED off"
GPIO.output(18,GPIO.LOW)
```

Once you have typed all the code and checked it, save and exit the text editor with “Ctrl + x” then “y” then “enter”.

### Running the Code:

To run this code type:

```
sudo python LED.py
```

You will see the LED turn on for a second and then turn off.

If your code does not run and an error is reported, edit the code again using nano LED.py.

### Explanation:

So, what is happening in the code? Let's go through it a line at a time:

<code>import RPi.GPIO as GPIO</code>	The first line tells the Python interpreter (the thing that runs the Python code) that it will be using a 'library' that will tell it how to work with the Raspberry Pi's GPIO pins. A 'library' gives a programming language extra commands that can be used to do something different that it previously did not know how to do. This is like adding a new channel to your TV so you can watch something different.
<code>import time</code>	Imports the Time library so that we can pause the script later on.
<code>GPIO.setmode(GPIO.BCM)</code>	Each pin on the Raspberry Pi has several different names, so you need to tell the program which naming convention is to be used.
<code>GPIO.setwarnings(False)</code>	This tells Python not to print GPIO warning messages to the screen.
<code>GPIO.setup(18,GPIO.OUT)</code>	This line tells the Python interpreter that pin 18 is going to be used for outputting information, which means you are going to be able to turn the pin 'on' and 'off'.
<code>print "LED on"</code>	This line prints some information to the terminal.
<code>GPIO.output(18,GPIO.HIGH)</code>	This turns the GPIO pin 'on'. What this actually means is that the pin is made to provide power of 3.3volts. This is enough

	to turn the LED in our circuit on.
<code>time.sleep(1)</code>	Pauses the Python program for 1 second
<code>print "LED off"</code>	This line prints some information to the terminal.
<code>GPIO.output(18,GPIO.LOW)</code>	This turns the GPIO pin 'off', meaning that the pin is no longer supplying any power.

### (b) Get input from two switches and switch on corresponding LEDs Controlling an LED by a Button.

#### SOLUTION:

##### Introduction:

In this lesson, we will learn how to turn an LED on or off by a button.

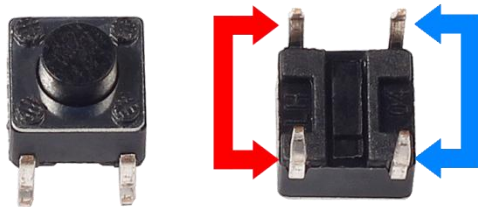
##### Components:

- 1\* Raspberry Pi
- 1\* Breadboard
- 1\* LED
- 1\* Button
- 1\* Resistor (220Ω)
- Jumper wires

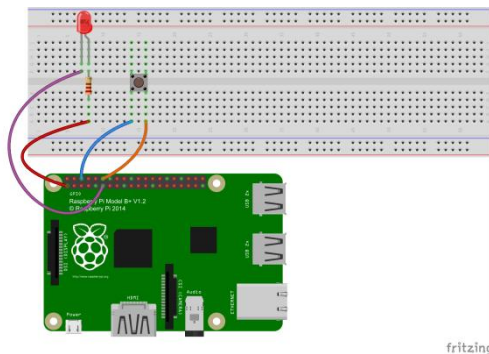
##### Principle:

##### Button

Buttons are a common component used to control electronic devices. They are usually used as switches to connect or disconnect circuits. Although buttons come in a variety of sizes and shapes, the one used here is a 6mm mini-button as shown in the following pictures. Pins pointed out by the arrows of same color are meant to be connected.



When the button is pressed, the pins pointed by the blue arrow will connect to the pins pointed by the red arrow (see the above figure), thus closing the circuit, as shown in the following diagrams.

**Experimental Procedures:****Step 1: Build the circuit****Step 2: Change directory**

**cd /home/pi/Sunfounder SuperKit Python code for RaspberryPi/**  
**and write the program in Python**

```
#!/usr/bin/env python
import RPi.GPIO as GPIO
import time

LedPin = 11    # pin11 --- led
BtnPin = 12    # pin12 --- button

Led_status = 1

def setup():
    GPIO.setmode(GPIO.BOARD)      # Numbers GPIOs by physical location
    GPIO.setup(LedPin, GPIO.OUT)   # Set LedPin's mode is output
    GPIO.setup(BtnPin, GPIO.IN, pull_up_down=GPIO.PUD_UP) # Set BtnPin's
mode is input, and pull up to high level(3.3V)
    GPIO.output(LedPin, GPIO.HIGH) # Set LedPin high(+3.3V) to off led

def swLed(ev=None):
    global Led_status
    Led_status = not Led_status
    GPIO.output(LedPin, Led_status) # switch led status(on-->off; off-->on)
    if Led_status == 1:
        print 'led off...'
    else:
        print '...led on'

def loop():
    GPIO.add_event_detect(BtnPin, GPIO.FALLING, callback=swLed,
bouncetime=200) # wait for falling and set bouncetime to prevent the callback function from
being called multiple times when the button is pressed
    while True:
        time.sleep(1) # Don't do anything

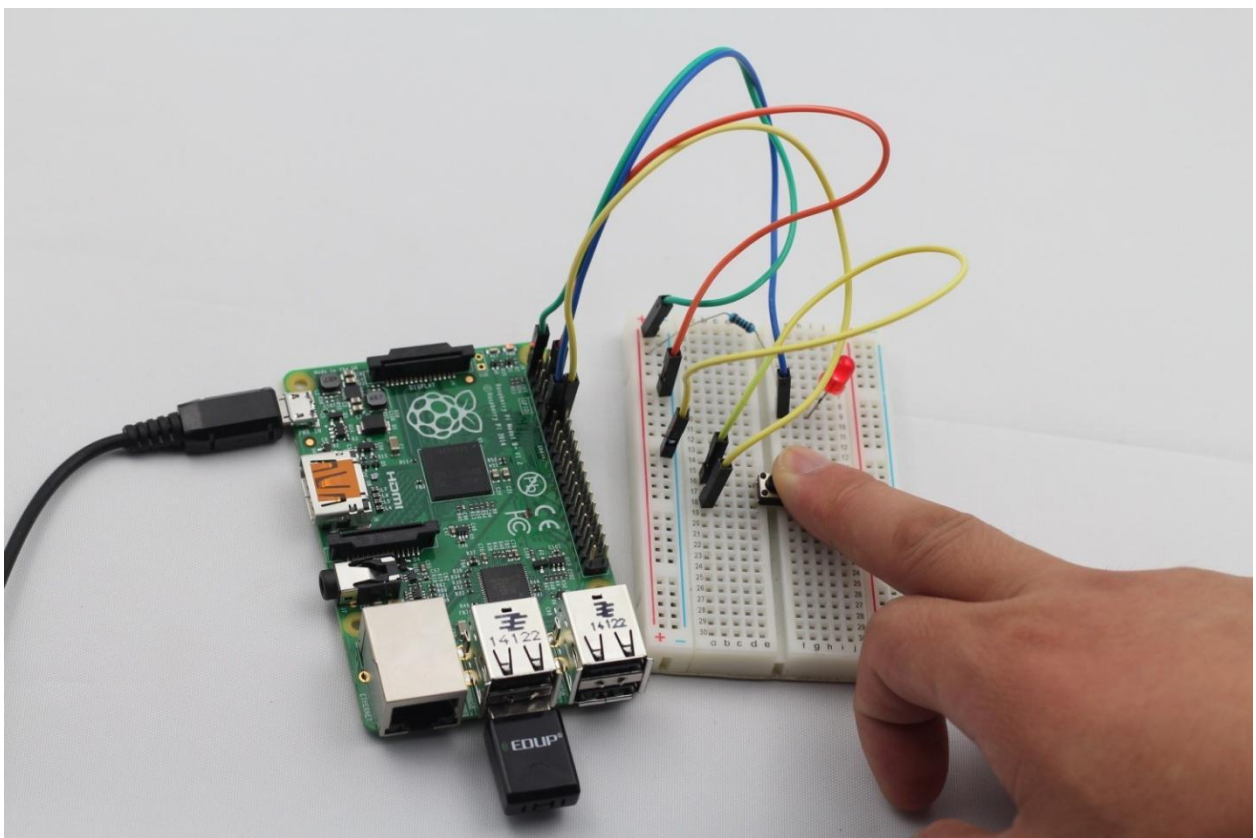
def destroy():
    GPIO.output(LedPin, GPIO.HIGH) # led off
```

```
GPIO.cleanup()          # Release resource

if __name__ == '__main__': # Program start from here
    setup()
    try:
        loop()
    except KeyboardInterrupt: # When 'Ctrl+C' is pressed, the child program destroy() will be
        executed.
        destroy()
```

**Step 3: Run****sudo python 02\_btnAndLed.py**

Now, press the button, and the LED will light up; press the button again, and the LED will go out. At the same time, the state of the LED will be printed on the screen.



**(c ) Flash an LED at a given on time and off time cycle, where the two times are taken from a file.**

**SOLUTION:**

**Components required**

- One led
- 100 ohm resistor
- Jumper cables

**Raspberry Pi GPIO Specifications:**

- Output Voltage : 3.3V
- Maximum Output Current : 16mA per pin with total current from all pins not exceeding 50mA

For controlling a Led using Raspberry Pi, both python and the GPIO library is needed.

**Installing Python GPIO Library:**

**Note: Python and GPIO library are preinstalled if you are using Raspbian.**

- Make sure your Raspberry Pi is connected to the internet using either a LAN cable or a WiFi adapter.
- Open the terminal by double clicking the LXTerminal icon
- Type the following command to download the GPIO library as a tarball

```
wget http://raspberrypi-gpio-python.googlecode.com/files/RPi.GPIO-0.4.1a.tar.gz
```

- Unzip the tarball

```
tar zxvf RPi.GPIO-0.4.1a.tar.gz
```

- Change the directory to unzipped location

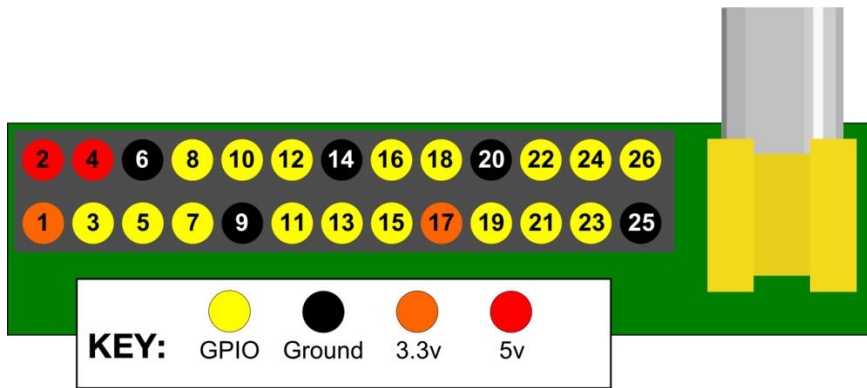
```
cd RPi.GPIO-0.4.1a
```

- Install the GPIO library in python

```
sudo python setup.py install
```

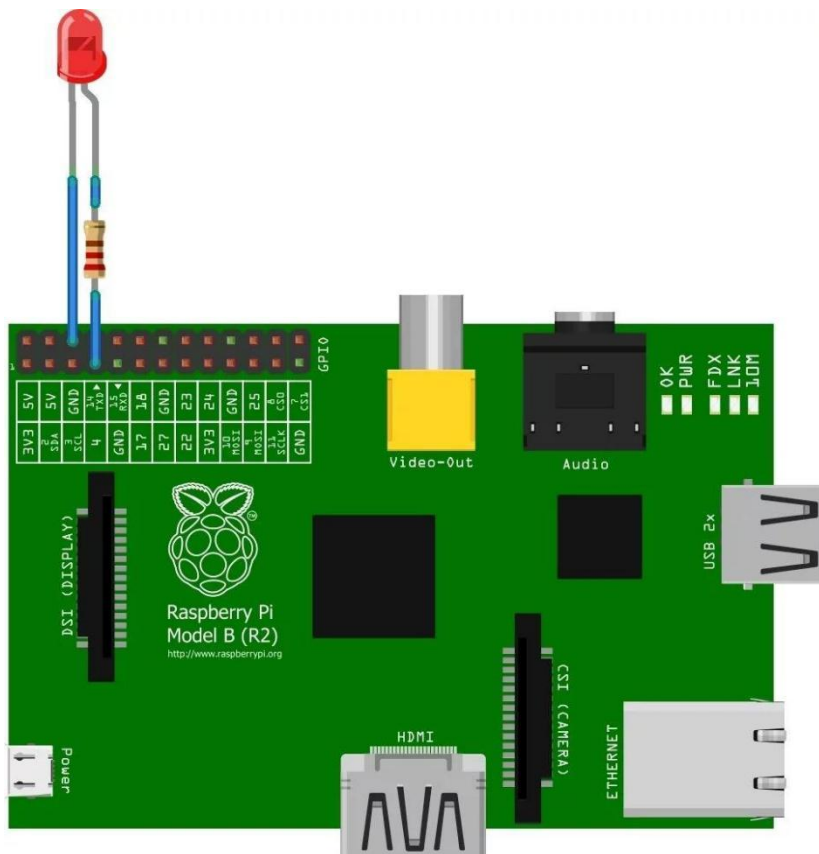
**Raspberry Pi GPIO Pin Out:**

Raspberry Pi has 17 GPIO pins out of 26 pins



### Circuit Diagram:

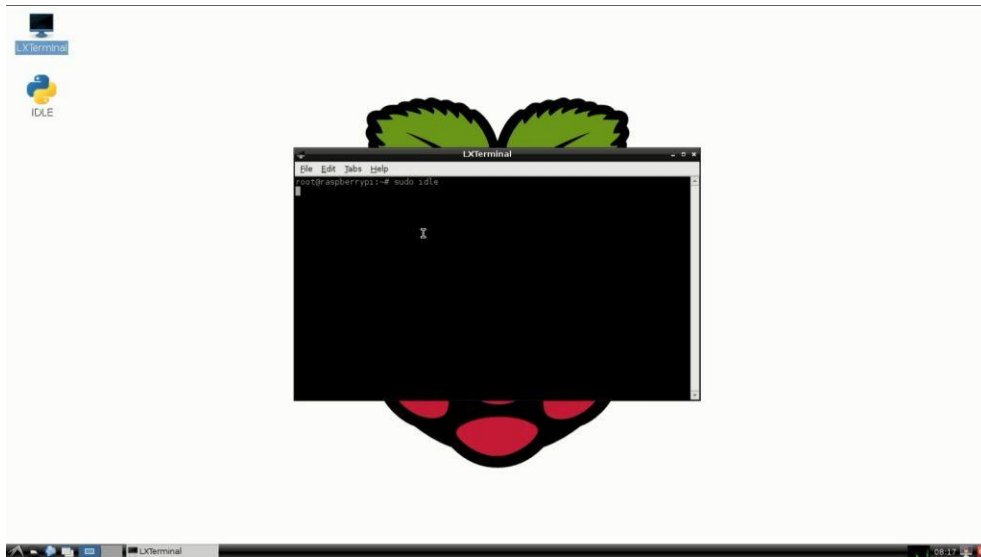
- Connect the Led to 6 (ground) and 11 (gpio) with a 100Ω resistor in series



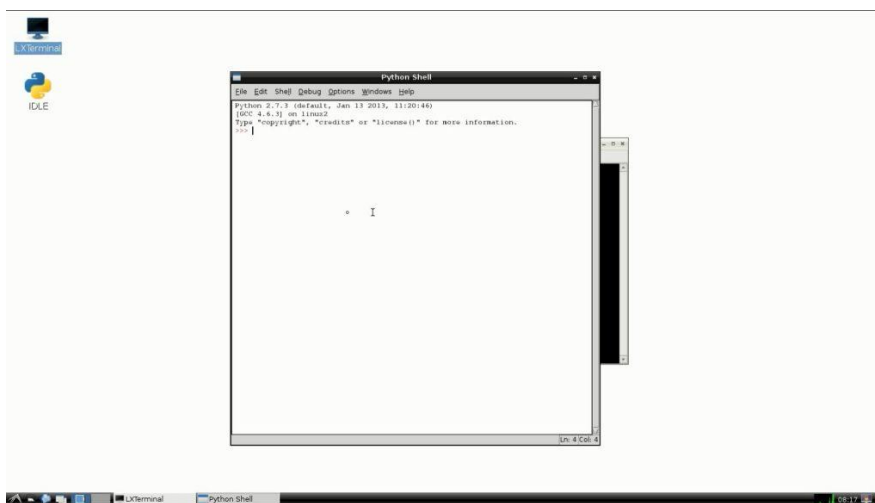
### Python Programming:

- Open Terminal
- Launch IDLE IDE by typing

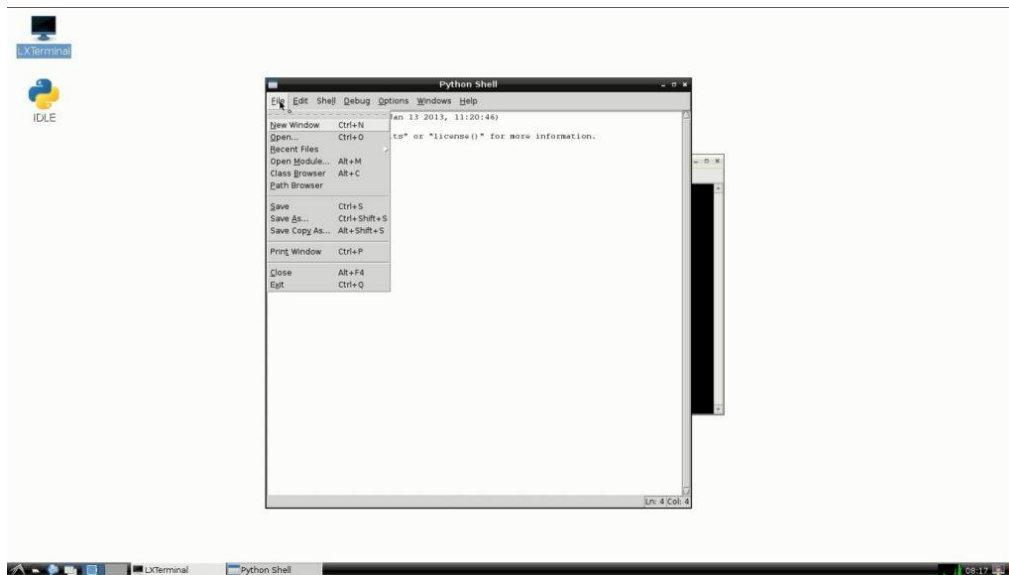
sudo idle



This launches IDLE with superuser privileges which is necessary to execute scripts for controlling the GPIO pins

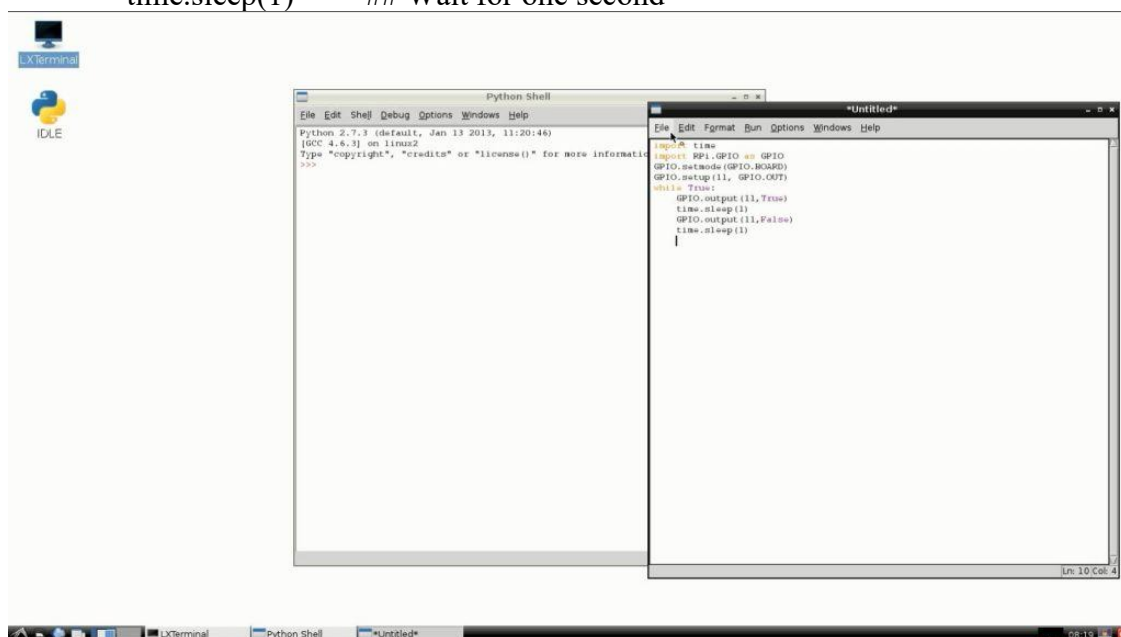


- After the IDLE launches, open a new window by FILE>OPEN or Ctrl+N

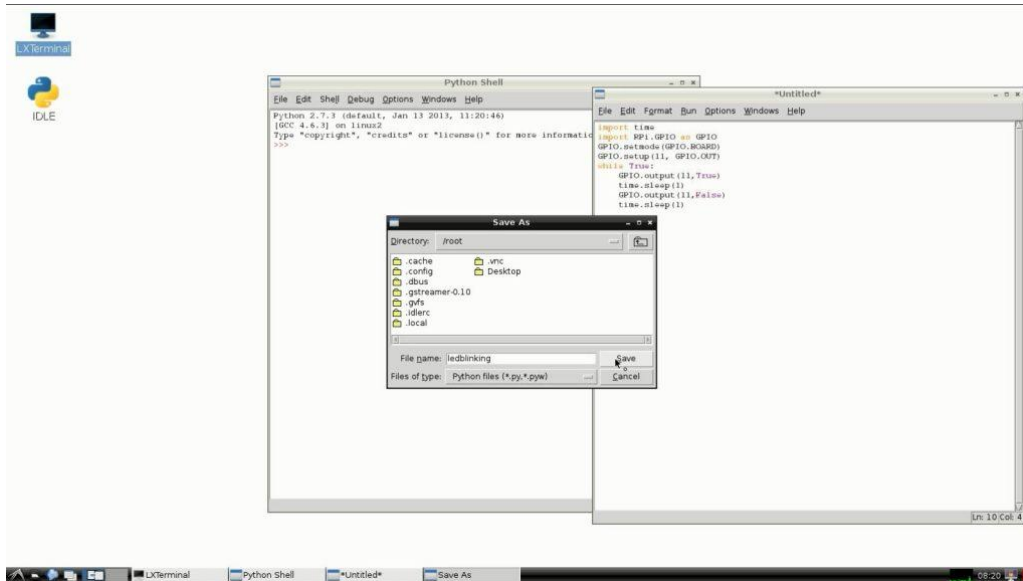


- Type the code below in the window

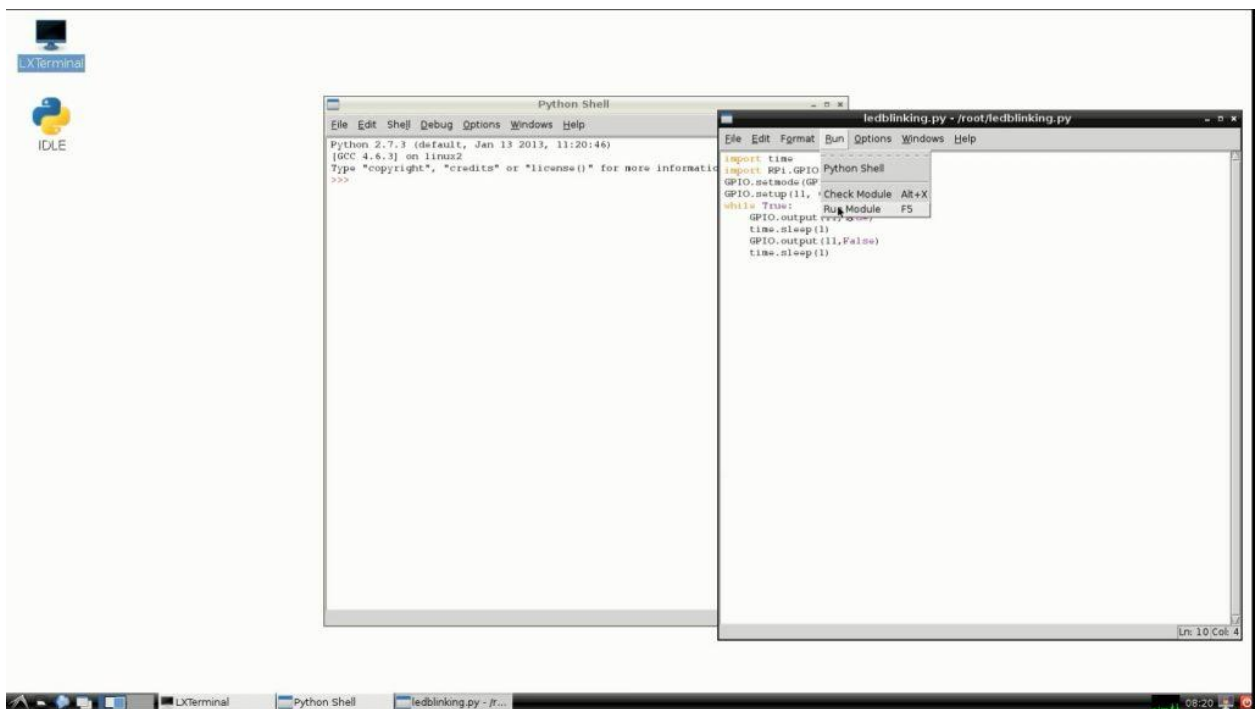
```
import time
import RPi.GPIO as GPIO    ## Import GPIO library
GPIO.setmode(GPIO.BOARD)   ## Use board pin numbering
GPIO.setup(11, GPIO.OUT)   ## Setup GPIO Pin 11 to OUT
while True:
    GPIO.output(11,True)    ## Turn on Led
    time.sleep(1)           ## Wait for one second
    GPIO.output(11,False)   ## Turn off Led
    time.sleep(1)           ## Wait for one second
```



- Save the code by FILE>SAVE or Ctrl+S



- To run your code RUN>RUN or Ctrl+F5



- The Led will start blinking now.

## EXPERIMENT-6

---

### OBJECTIVE

**A) Flash an LED based on cron output (acts as an alarm)**

### PROGRAM

Python code: blink.py

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

```
import time
```

```
LedPin = 11 # pin11
```

```
def setup():
```

```
    GPIO.setmode(GPIO.BOARD) # Numbers GPIOs by physical location
```

```
    GPIO.setup(LedPin, GPIO.OUT) # Set LedPin's mode is output
```

```
    GPIO.output(LedPin, GPIO.HIGH) # Turn ON led
```

```
def blink():
```

```
    while True:
```

```
        GPIO.output(LedPin, GPIO.HIGH) # led on
```

```
        time.sleep(1)
```

```
        GPIO.output(LedPin, GPIO.LOW) # led off
```

```
        time.sleep(1)
```

```
def destroy():
```

```
    GPIO.output(LedPin, GPIO.LOW) # led off
```

```
    GPIO.cleanup() # Release resource
```

```
if __name__ == '__main__': # Program start from here
```

```
    setup()
```

```
    try:
```

```
        blink()
```

```
    except KeyboardInterrupt: # When 'Ctrl+C' is pressed, the child program destroy()
        will be executed.
```

```
    destroy()
```

**B) Switch on a relay at a given time using cron, where the relay's contact terminals are**

**connected to a load.**

## PROGRAM

Python Program:

```
# Python 2.7 on raspberry pi
# Script to trigger thermostat control on heating
# Reads room temperatures from Prodservr DONE - from Thermostat01.py
# All heating control logic is on prodserver. This just reads on/off commands

# Dependencies
# sudo pip install max7219 - 7 seg 8 digit display driver
# sudo pip install bs4 - beautiful soup web scraper
# sudo pip install requests - python http library for GET and POST
# sudo apt-get install python-blinkt blinkt led library

# GPIO PINS
# MAX7129 7 seg display
# DIN = GPIO10(MOSI)
# CS = GPIO8(SPI CE0)
# CLK = GPIO11(SPI CLK)
# Blinkt uses GPIO23,24
Pin Constants
PIN_RELAY = 17
PIN_BOOST_DETECT = 4

import blinkt
blinkt.set_clear_on_exit(False)
blinkt.set_all(0,0,0)
blinkt.set_brightness(0.1)
blinkt.show()
# blinkt Timer leds 0,1
# blinkt Heat leds 2,3
# blinkt Boost leds 4,5

import sys
# 7 seg display set up
import max7219.led as led

device = led.sevensegment()
```

```
device.brightness(1)
import requests
from bs4 import BeautifulSoup

fp.write(logstring)
fp.close()
print logstring

def do_7segdisplay(status):
    "display on 7 seg"
    now=status[0]
    segnumber = 10000*now.hour + 100*now.minute + float(status[5])
    device.write_number(0,segnumber,decimalPlaces=2,zeroPad=True)

def do_timer(status):
    "Deals with timer status led"
    if(status[3]): # timer on
        r,g,b = (0,250,0)
    else:
        r,g,b = (0,0,0)
    blinkt.set_pixel(0,r,g,b)
    #blinkt.set_pixel(1,r,g,b)
    blinkt.show()
    return

def do_heat(status):
    "Deals with Heat status led and switches relay"
    if(status[4]): # heat on
        r,g,b = (200,000,000)
        GPIO.output(PIN_RELAY, GPIO.LOW)
    else:
        r,g,b = (0,0,0)
        GPIO.output(PIN_RELAY, GPIO.HIGH)

    blinkt.set_pixel(2,r,g,b)
    #blinkt.set_pixel(3,r,g,b)
    blinkt.show()
    return
def do_boost():
```

```
"No function as yet"  
# MAIN PROGRAMME  
status = get_status(server, status)  
do_log(status)  
do_7segdisplay(status)  
do_timer(status)  
do_heat(status)
```

**( c ) Get the status of a bulb at a remote place (on the LAN) through web.**

## **THEORY**

One function that makes the Internet of Things the Internet of Things is remote control of devices, being able to trigger action from a remote location. This is otherwise known as **remote configuration**. Whether it's a home, an office, or an industrial site, connected devices rely on some form of automation through sensors or machines and need a mechanism to control their operation remotely.

### **Hardware Requirements**

For this demonstration, we will use the following hardware:

Raspberry Pi Model B+ (with Raspbian OS)

1 LED and 1 490 Ohm Resistor

### **Software Requirements**

We will use the following sdks and tools

RPi GPIO , GPIO Python library for Raspbian OS

PubNub Python SDK

PubNub Javascript SDK

## **Web Application**

The web interface will look something like this:



This is a very simple web page with a visual indicator for the device and a button to toggle the on/off state of LED.

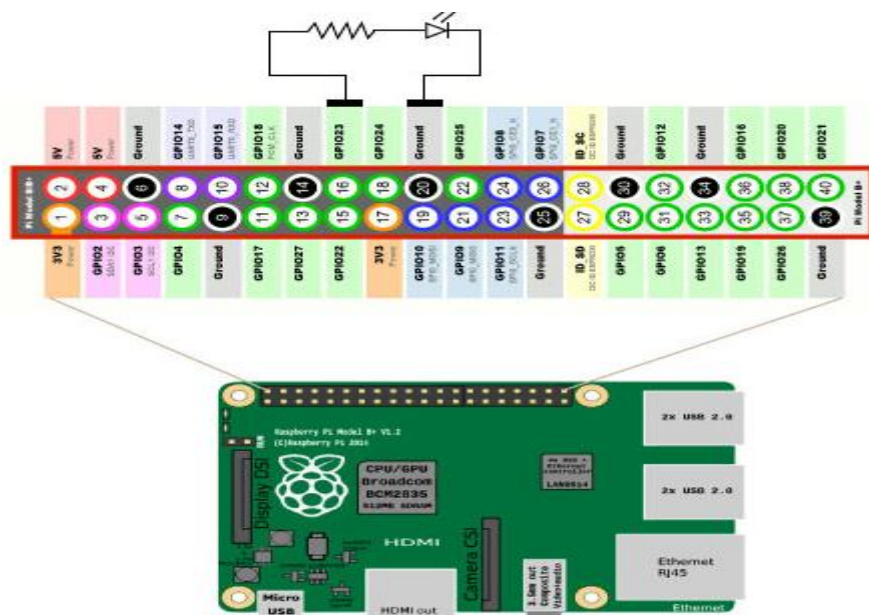
Behind the scenes, we have the PubNub Javascript API that performs two operations upon receiving certain events.

Sends a request message to toggle the state of the device.

Receives response with the current state of the device.

## Raspberry Pi and LED

Here is the schematic for the raspberry Pi connections to be used in this application.



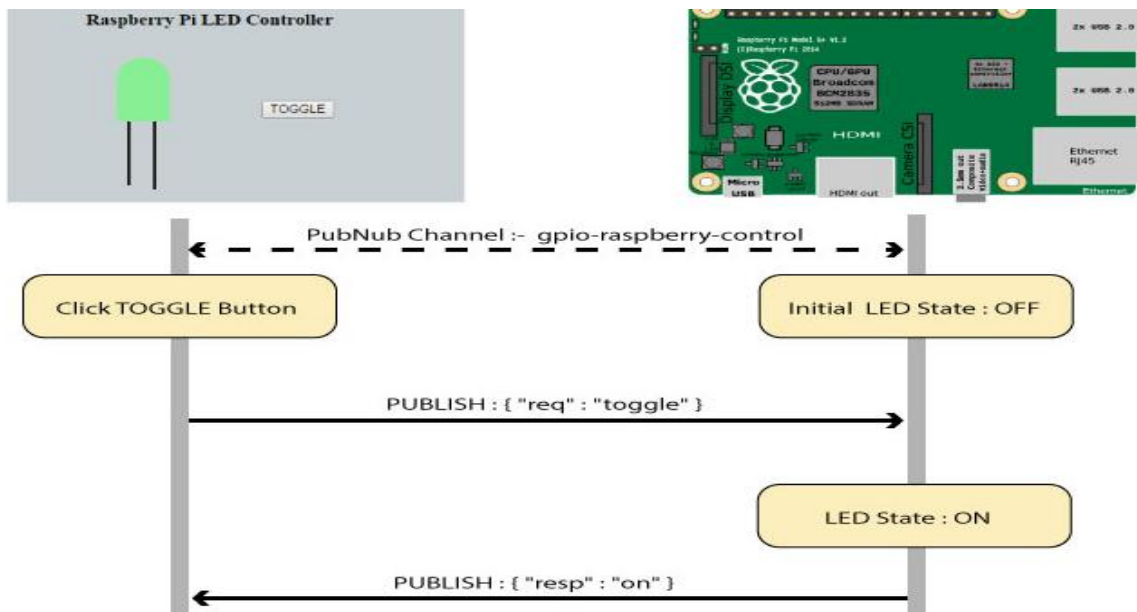
The python code for driving the LED is executed as part of the PubNub callback on 'gpio-raspberry-control' channel

```
glow = False

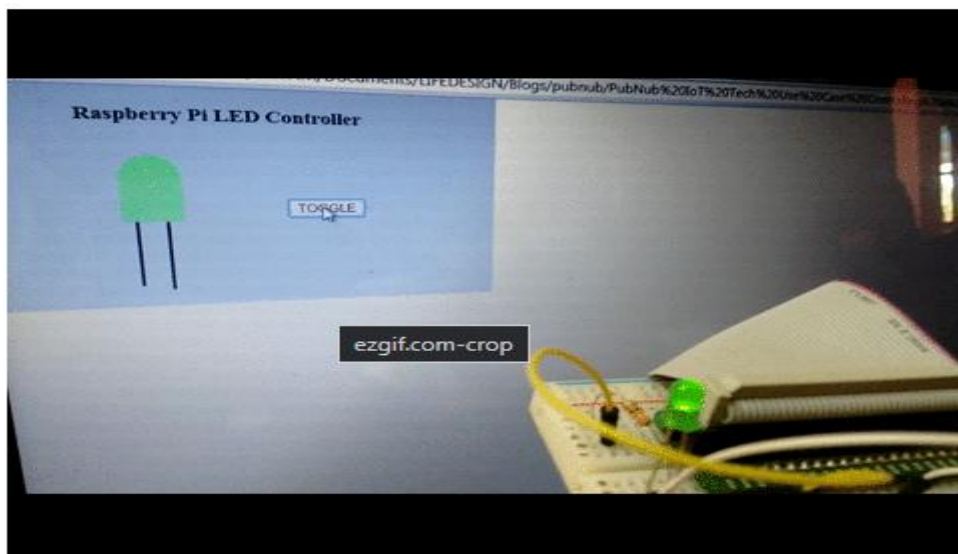
#PubNub Channel Subscribe Callback
def gpioCallback(msg,channel):
    global glow
    respstring = ""
    command = msg
    print "Command is : " + str(command)
    if('req' in command):
        if(command['req'] == 'toggle'):
            if(glow):
                glow = False;
            respstring = 'off'
        else:
            glow = True
            respstring = 'on'
    GPIO.output(16, glow)
    respmsg = {"resp" : respstring }
    pubnub.publish(pubnubChannelName, respmsg)
```

When a toggle request is received, the application checks the current state of the GPIO driving pin of the LED, toggles its state and then sends the new state back to the web application as a response message.

The exchange of messages between the web application and Raspberry Pi can be visualized as follows.



And here is how the entire system works:



## EXPERIMENT-7

---

### OBJECTIVE

Describe gateway-as-a-service deployment in IoT toolkit.

### THEORY

The Internet of Things (IoT) is set to occupy a substantial component of future Internet. The IoT connects sensors and devices that record physical observations to applications and services of the Internet. As a successor to technologies such as RFID and Wireless Sensor Networks (WSN), the IoT has stumbled into vertical silos of proprietary systems, providing little or no interoperability with similar systems. As the IoT represents future state of the Internet, an intelligent and scalable architecture is required to provide connectivity between these silos, enabling discovery of physical sensors and interpretation of messages between things. This paper proposes a gateway and Semantic Web enabled IoT architecture to provide interoperability between systems using established communication and data standards. The Semantic Gateway as Service (SGS) allows translation between messaging protocols such as XMPP, CoAP and MQTT via a multi-protocol proxy architecture. Utilization of broadly accepted specifications such as W3C's Semantic Sensor Network (SSN) ontology for semantic annotations of sensor data provide semantic interoperability between messages and support semantic reasoning to obtain higher-level actionable knowledge from low-level sensor data.

While developing IoT solutions we come across common tasks of connecting <things> to the cloud. For devices that are not connected to the cloud directly, gateways are used. Such gateway can be a system on chip (SoC) device: cable modem, set top box, home or industrial automation gateway, smart phone, home entertainment system, laptop or PC. All these gateway should have some sort of interface (BLE/ZigBee/etc radios, adapter, digital or analog inputs) that can connect to the actual device: lamp, controller, sensor, appliance. Semantic Gateway as Service (SGS) The heart of the semantic IoT architecture is the SGS, which bridges low level raw sensor information with knowledge centric application services by facilitating interoperability at messaging protocol and data modeling level. The description below is complemented by Open Source code available at <https://github.com/chheplo/node-sgs> which is further being enhanced and evaluated in the context of CityPulse, a large multi-institutional EU FP7 supported project along with an effort for additional community engagement and development.

### Semantic Gateway as Service (SGS)

The heart of the semantic IoT architecture is the SGS, which bridges low level raw sensor information with knowledge centric application services by facilitating interoperability at messaging protocol and data modelling level. The description below is complemented by Open Source code available at <https://github.com/chheplo/node-sgs> which is further being enhanced and evaluated in the context of City Pulse, a large multi-institutional EU FP7 supported project along with an effort for additional community engagement and

development.

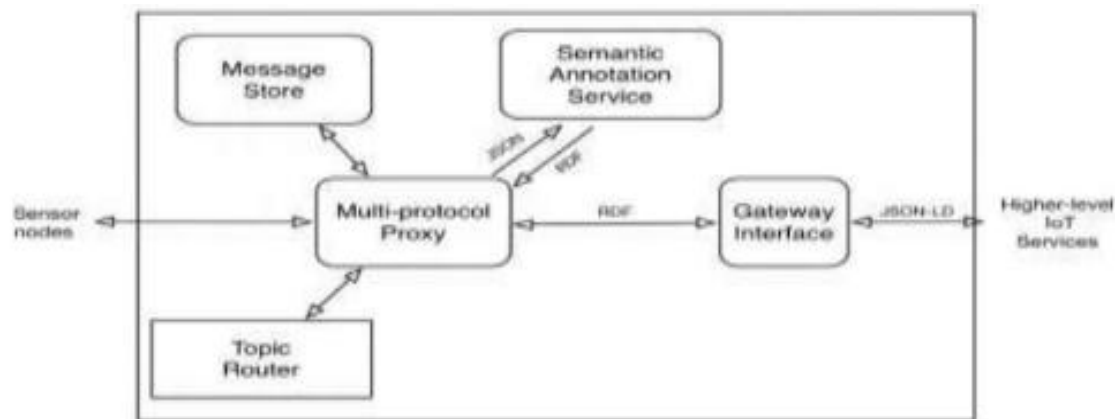


Fig.

### Gateway as a service architecture

The SGS has three core components as described in Figure:

- (1) multi-protocol proxy,
- (2) semantic annotation service,
- (3) Gateway service interface.

The SGS also has components for required capabilities such as message store and topics router, which assist multi-protocol proxy and gateway service interface in translation between messaging protocol. At a high level, SGS architecture connects external sink nodes to the gateway component using primitive client agents, which support MQTT, XMPP or CoAP.

In contrast, the gateway service interface connects cloud services or other SGSs via REST or pub sub protocol. Before raw sensor data is forwarded from proxy to gateway interface, it is annotated using SSN and domain specific ontologies. Although the semantically annotated data is in RDF format at the multi-protocol proxy, the gateway interface converts the data into JSON, specifically linked data (JSON-LD) format to support RESTful protocols.

**EXPERIMENT-8****OBJECTIVE**

To write a program for LDR to vary the light intensity of LED using Arduino

**ALGORITHM:**

STEP1: Start the program.

STEP2: Start →Arduino 1.88[IDE].

STEP3: Enter the coding in Arduino software.

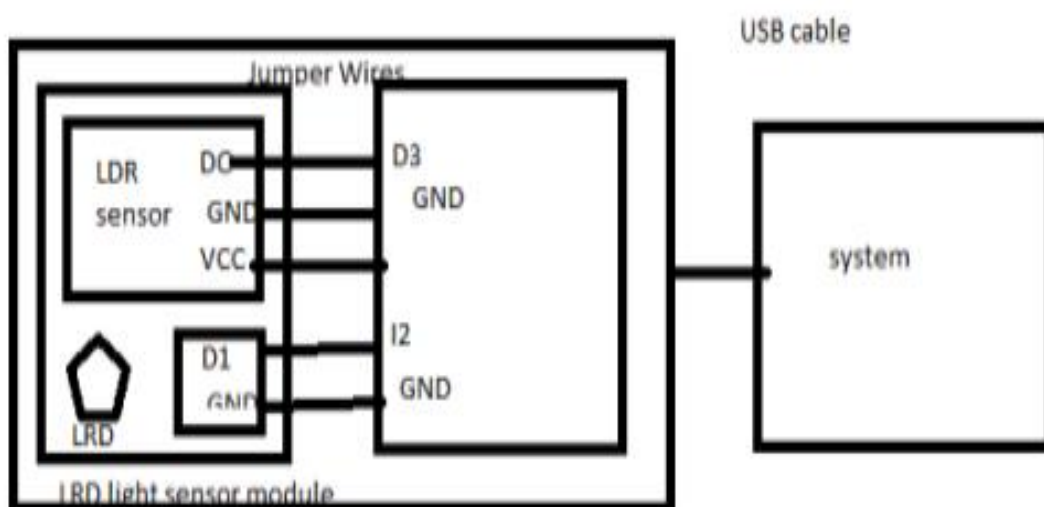
STEP4:Compile the coding in the Arduino software.

STEP5: From LDR light sensor module, connect VCC to power supply 5V and connect to digital pin D3 and connect GND to ground gnd using jumper wires to arduino board.

STEP6: For LED, connect D to digital pin D2 and connect GND to ground GND using jumper wires to arduino board.

STEP7: Show the variance of lights intensity in LED we use LDR light sensor module.

STEP8: Stop the process.

**BLOCK DIAGRAM**

**CODING:**

```
const int ldr_pin = 3; const int  
led_pin = 2; void setup() {  
  pinMode(ldr_pin, INPUT);  
  pinMode(led_pin, OUTPUT);  
  Serial.begin(9600);  
}  
void loop() {  
  if ( digitalRead( ldr_pin ) == 1) {  
    digitalWrite(led_pin, HIGH);  
  }  
  else {  
    digitalWrite(led_pin , LOW);  
  }  
  Serial.println(digitalRead( ldr_pin ));  
  delay(100);  
}
```

**OUTPUT:**