

Low Cost LoRa based Forest Fire Detection and Implementation

Submitted in partial fulfilment of the requirements for the award of

Bachelor of engineering degree in

ELECTRONICS AND COMMUNICATION ENGINEERING

By

GURALA SRI HARSHAD CHOWDARY(37130153)

GANGIREDDYGARI GOPINATH REDDY(37130137)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

SCHOOL OF ELECTRICAL AND ELECTRONICS



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119



**SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**



**Accredited with Grade “A” by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119**

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of **GURALA SRI HARSHAD CHOWDARY (37130153)** and **GANGIREDDYGARI GOPINATH REDDY(37130137)** who carried out the project entitled “**LOW COST LoRa BASED FOREST FIRE DETECTION AND IMPLEMENTATION**” under our supervision from .

**INTERNAL GUIDE
Dr. P. KaviPriya**

**Head of the Department
(DR. T. RAVI)**

Submitted for Viva voce Examination held on _____

INTERNAL EXAMINER

(Name in Capital Letters with
Signature)

EXTERNAL EXAMINER

(Name in capital Letters with
Signature)

DECLARATION

We **GURALA SRI HARSHAD CHOWDARY(37130153)** and **GANGIREDDYGARI GOPINATH REDDY(37130137)** hereby declare that the Project Report entitled “**LOW COST LoRa BASED FOREST FIRE DETECTION AND IMPLEMENTATION**” done by us under the guidance of “**Dr. P. Kavi Priya**”, **Associative Professor** Department of Electronics and Communication Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **ELECTRONICS AND COMMUNICATION ENGINEERING**.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

1.

2.

ACKNOWLEDGEMENT

We are pleased to acknowledge my sincere thanks to Board of Management of SATHYABAMA for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

We convey our thanks to **Dr. N. M. NANDHITHA M.E.,P.hD., Prof. & Dean, School of Electrical and Electronics Engineering** and **Dr. T. Ravi M.E.,P.hD., Head of the Department, Dept. of ELECTRONICS AND COMMUNICATION ENGINEERING** for providing me necessary support and details at the right time during the progressive reviews.

We would like to express my sincere and deep sense of gratitude to my Project Guide Dr. P. Kavi Priya Associate Professor for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my projectwork.

We wish to express my thanks to all Teaching and Non-teaching staff members of the Department of **Electronics and Communication Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

This project presents a fire detection and alert system based on IOT. Here a specific environment is monitored 24x7 and the user is alerted in case of any fatal situation. This can be implemented using a nodemcu and a number of sensors for detecting different physical parameters that can go high during a fire related accident. Arduino is an IOT based controller board. Here two parameters are being monitored continuously temperature and presence of smoke. For sensing temperature LM35 temperature sensor is used. For sensing presence of smoke a gas sensor called MQ6 is used, this sensor can detect and measure any carbon based gas smoke produces CO₂ this makes it possible to detect smoke using MQ6. Also a PIR sensor is used for detecting presence of any human. Both gas and temperature sensor is analog so cannot be connected to a digital pin but an analog pin that is input to an ADC (analog to digital converter). Also forest area can be monitored through a camera using this camera fire can be detected using image processing. For this we can run a python program from PC and using opencv library fire can be detected. This can be used as a contingency system if the sensors malfunction. If any fire is detected then a water sprinkler will be turned on to prevent fire also an alert is sent to concerned personal for further action

TABLE OF CONTENT

CHAPTER NO	TITLE	PAGE NO
1	Introduction	1
2	Literature Review	4
3	Aim & Scope	7
	3.1. Existing System	8
	3.2. Proposed System	9
	3.3. Blockdiagram	10
	3.3.1. Block diagram explanation	11
4	Requirements	12
	4.1. Python IDE	12
	4.1.1. Introducing python	13
	4.1.2. python and networking	14
	4.2. Arduino Software(IDE)	16
5	Materials and Methods	22
	5.1. Arduino	22
	5.2. LoRa(Long range)	33
	5.2.1. introduction	33
	5.2.2. working	34
	5.2.3. Why LoRa	35
	5.2.4. Lora WAN Specifications	41
	5.3. Gas Sensor	47
	5.4. Temperature Sensor	53
	5.5. Passive Infrared Sensor(PIR)	62
	5.6. Pump	67
	5.7. LCD (Liquid Crystal Display)	68
	5.8. Buzzer	72

6	Result	75
7	Conclusion	78
8	References	79

LIST OF FIGURES

FIG NO	TITLE	PAGE NO
3.3.1	Block diagram of Sensor Node	6
3.3.2	Block diagram of Master Node	7
5.1.1	Arduino Board	33
5.2.1	LoRa RF click	36
5.2.2	Working process of LoRa	37
5.2.3.1	Graph of LoRa	38
5.2.3.2	LoRa compliments	39
5.2.3.3	Countries using LoRa	40
5.2.3.4	LoRa simplified GO-TO network	40
5.2.3.5	LoRa Network architecture	40
5.3.1	Gas sensor	43
5.3.2	Construction of Gas sensor	44
5.3.4	MQ4 Module pinout	47
5.3.5	MQ4 sensor circuit diagram	47
5.4.1	Temperature sensor	50
5.5.1	PIR sensor	57
5.5.2	PIR Working	58
5.5.3	PIR circuit diagram	58
5.6	Pump	62
5.7	LCD	67
5.8	Buzzer	70
6.1	Transmitter section	74
6.2	Receiver section	75
6.3	Demonstration	75
6.4	output display	76
6.5	Geotagged Photo	76

CHAPTER 1

INTRODUCTION

Nearly 30% of world's land area has been covered by forest which accounts for nearly four billion hectares. Due to the dense nature of forest, unseen condition fire can spread over large area rapidly spoils the entire ecological system and endangers wild animal. This fire causes greater havoc in the environment. Recently forest fire occurs frequently due to the drought conditions. Fire in the world is increasing in recent times. This fire in the forests, farmlands and industries are owed to either natural or man-made disasters. Forest fires results from natural causes such as lightening or manmade causes such as burning of unwanted materials results in a slow combustion of saw dust and leaves. Every year the occurrence of forest fire has been increased enormously causes greater destruction to floras and faunas. Forest fire monitoring and detection system plays a significant role in preserving natural resources and safe guards humans. This attracted so many researchers to develop a significant solution to this problem. In order to sense fire at beginning and for early alarm automatic fire detection system has been proposed. But these techniques detects the fire at the scenario that it has spreaded over vast area makes it difficult to control and extinguish the fire completely. The outcome of such a scenario destructs the environment and atmosphere causes irremediable damage to the ecology. Further it has a negative impact on the pattern changes in weather and global warming. It is critical to perceive the fire and its location quickly and also alerting the fire units. According to National Institute of Space Research (INPE) 76,000 fires were happened in Brazilian Amazon. During January-October 2019 nearly 906,000 hectares of land were burnt completely. Destruction had occurred within short interval of time. Due to the feeding of fuel by ignitable materials, the fire at the central spot has increased drastically and spreads faster over large areas. Therefore timely detection of forest fire is necessary before the fire spreads over large area. A cost effective fire fighting and monitoring system is essential. IoT describes a system that uses wireless and wired Internet connections to interconnect different things in the physical world together with the sensors through the internet in which things in the physical world and sensors are connected to the Internet. Diverse local area networks like RFID, NFC, Wi-Fi, Bluetooth, and Zigbee can be exploited by these sensors. Also sensors can be interconnected to GSM, GPRS, 3G, and LTE wide area networks. The Internet of Things will connect all

physical objects with the living things. The industrial equipment was interconnected with the internet of things in the earlier version. Now we can interconnect inanimate and living things with the help of IOT. Today, the vision of IoT vision has stretched that it has connected daily objects with industrial equipment. Location, vibration, motion and temperature conditions are monitored by sensors International Journal of Future Generation Communication and Networking Vol. 13, No. 1, (2020), pp. 1430-1435 1431 ISSN: 2233-7857 IJFGCN Copyright © 2020 SERSC [15]. In IoT, understand the output signal from the sensors. Since the mobile networks supports the persistent connection of smart devices better quality of service offered to the customers will be achieved by IOT. In this project, IOT based early warning fire detection framework senses the fire as quickly as possible and save valuable lives. Detection of fire based on color information results in false prediction. Here flame and flame colored objects were distinguished based on color and motion features. Raspberry Pi is used because of its high processing speed at low cost. Several sensors are used to collect the data and these data's were transferred to Raspberry Pi. GSM module alerts the fire monitoring station through SMS.

2. Related Work Md Saifudaullah Bin Bahrudin et.al presented a fire monitoring system that detects fire by the smoke that has generated by the fire. Camera is used to capture the fire image when a fire incident happens. The system remotely sends the image of the fired room on the web page and alerts the firefighter through SMS. Sowah et. al. implements fuzzy logic based fire detection system for vehicle .Temperature, flame and smoke sensors are used to sense fire. In this system air-conditioning system is used for extinguishing fire within 20 seconds when fire occurs. Wen-bing Hang, Jim-wenpeg [10]- A novel method to identify the flame is proposed. The pixels in the Flame calculated using Hue Intensity Saturation color model.HSI rules were used for segmenting fire regions. Fire aliases can be avoided by seperating the pixels that have low intensity and low saturation in segmented fire regions. Degrees of fire flames have been measured by binary counter images results in false positive and false negatives. This method detects fire flames from test videos within a second. K.Angayarkkani, N.Radhakrishnan presented a system using Artificial Neural Intelligence to recognize the forest fire.RGB images are transformed to XYZ color space and anisotropic diffusion is used to identify fire zones. The pixel values of color space in segmented fire regions are using radial basis function neural network. T.qiu, Y.Yan, G.Lu several methods for edge detection have been used to access flame edge identification. Edges obtained from non trivial

images do not resemble notable phenomenon. Turgey, Hasan Demirel – classified the pixels in the flame based on generic rule. YCbCr color space is used for flame pixel classification. This method segments the flame region yields 99.9% accurate flame pixel classification rate. Zhao J, Zhang Z, Han S, Yuan Z proposed a fire monitoring system using cameras. Features extracted from Gaussian mixture model were trained using Support Vector Machine (SVM). False prediction occurs in case of red objects in analyzed image.

3. Methods and Material In this proposed system first the camera captures continuous frames and sends to the controller. Fire is detected by an image processing based real time flame detection algorithm.

Step 1: Frame differential method is used take out running pixels and portions that are detected. $F(x, y, k) =$

Step 2: HIS color model is obtained and if it satisfies the condition $0 \leq H \leq 60$ $0 \leq S \leq 0.2$ $127 \leq I \leq 255$

International Journal of Future Generation Communication and Networking Vol. 13, No. 1, (2020), pp. 1430-1435 1432 ISSN: 2233-7857 IJFGCN Copyright © 2020 SERSC

Then the regions are taken as flame regions

Step 3: The next step is foreground accumulation of images. In this step, time taken by the foreground images gives the value of pixels display in the same pixel portion at the time of succeeding time window is formed. Block image processing is helps to identify the flame motion features. Individual image of the video sequences is divided into 8X8 resolution blocks. Then sum of all pixel values such that the summed up values satisfies the condition $H(x, y, t) > T$.

Step 4: if half of whole pixel vales satisfies above condition then it is considered as flame block..

Steps of The Proposed Technique After the detection of flame, a mail is sent to the security and the nearest fire department with an attachment of the photo through GSM module. Smoke sensors are employed to detect smoke that caused due to the fire incident.

Step 6: In final step after detection of fire the Sprinkler motor sprays the water to prevent the fire spreads over to the nearest area.

International Journal of Future Generation Communication and Networking Vol. 13, No. 1, (2020), pp. 1430-1435 1433 ISSN: 2233-7857 IJFGCN Copyright © 2020 SERSC

Fig 2. Block Diagram of The New Method

Fig 3. Raspberry Pi Smoke Sensor

PYTHON Python is an elevated level programming language for broadly useful programming. Python is profoundly decipherable. It gives clear programming on both little and enormous scopes. Python includes a powerful kind framework and programmed memory the executives. It guides numerous including object arranged, basic, practical and procedural, and has an enormous and extensive standard library. Python mediators are accessible for some OS. IDLE can be expand in a way such

that Python's Integrated Development & Learning Environment. Coded in 100% Python, utilizing the GUI toolbox cross-stage works for the most part the equivalent on Windows, UNIX, and Mac OS X. 4. Results and Discussion The algorithm is implemented using python language. This language executes each and every line present in the program & recognize the inputs from camera, sensors and creates the respective output according program.

CHAPTER 2

LITERATURE REVIEW

1. Forest Fire Alerting System With GPS Co-ordinates Using IoT

In the advancing world, it is very crucial to protect our environment. Many incidents of man-made and natural disasters were happening around the world. Forest fires are one such catastrophe for environment. Once the fire inside deep forest starts, it burns and destroys everything and spreads everywhere within the forest. Fire spreads on hot days destroys trees and grasses due to drought conditions peaks in a forest region. Such forest fires disasters should be curbed in order to protect fauna and flora habitats in the forest.

We have tested in forest like conditions, but real hardship which we may face is during implementation in large area in real time.

GPS used here is very low ending with small antenna which it unreliable.

2. Forest Fire Detection System Reliability Test Using Wireless Sensor Network and Open MTC Communication Platform

Machine to Machine (M2M) communication system has started gaining its real world momentum by the introduction of Internet and mobile technology into this system. Several works have come up to use this integrated system into many different vertical solutions and tried to bind one solution platform for many monolithic systems. One recent innovation is to use HTTP REST architecture as communication platform. This work tackles the first step of implementing Open MTC as M2M and IoT communication platform. Involved sensors are carbon monoxide gas concentration, temperature, and humidity, combined with Zigbee and Arduino microcontroller to make up 2 Device Application components (or DAs, as proposed by ETSI M2M standard) connected to Open MTC GSCL.

Suggestions for further researches are: to conduct further reliability test by measuring overall transmission time from sensor to GSCL (over the Internet). Where the number of sensors or gateways is still not sufficient for load test, further research is also possible to develop a testing tool to simulate connections from sensors to gateways and platforms.

3. Smart Forests: fire detection service

Smart Forest is a concept derived from the Internet of Things (IoT) and defines sections of a forest where remote sensing is applied to collect data about environmental conditions. One of the main objectives of Smart Forests is to detect wild fire at early stages. However, the required technology for such monitoring usually demands a complex and expensive sensor and network infrastructure and requires central processing capabilities for analyzing data from several thousands of sensors. The goal of this work is to propose a solution focused on Edge Computing, using the concept of Mobile Hubs (M-Hubs).

In this case, the mobile hub that detected fire could notify close mobile hubs about the current event - besides notifying the Smart Forest Server -, to better control the incident.

4. Detecting and Reporting Forest Fire through Deployment of Three Dimensional Multi-sink Wireless Sensor Network

Forest fires are common in the winter months across the world. An apt forest fire detection technique is required to check the spreading of fire and reduce the degree of loss thereby. In this paper we have proposed an energy efficient forest fire detection and reporting technique through three dimensional deployment of multi-sink wireless sensor network. The scheme may be used in event driven, time driven or a proposed hybrid mode. The results of the proposed scheme have been compared with 3 dimensional versions of some of the very well-known routing protocols for sensor networks.

As future scope, we have planned for real node deployment in the terrains of Mussoorie-Dehradun belt. By embedding our code in the nodes, we would like to measure the effectiveness of the proposed scheme. Moreover, we would like to calculate the deviations in the result obtained between simulation & real time environment for the same set of network parameters.

The results indicate a significant correlation between the weather and the trending sales categories. Also, the results obtained through both the methods were found nearly similar.

5. An IoT based Weather Information Prototype Using WeMos

The Internet of Things (IOT) describes the interconnection of devices and people through the traditional internet and social networks for various day-to-day applications like weather monitoring, healthcare systems, smart cities, irrigation field, and smart lifestyle. IOT is the new revolution of today's internet world which monitors live streaming of the entire world's status like temperature, humidity, thunderstorm, earthquake, floods etc. that can stagger an alarm to human life. This paper proposes a low-cost weather monitoring system which retrieves the weather condition of any location from the cloud database management system and shows the output on an OLED display.

Efficiency can be improved by using more accurate GPS receivers. There has to be a provision for intimating crew about any sudden attacks or natural disaster warnings from monitoring section.

CHAPTER 3

AIM & SCOPE

3.1 EXISTING SYSTEM

- Fire alarm systems with buzzer
- Fire alert systems with no continuous monitoring system highly expensive monitoring systems

DRAWBACKS

Current systems are local hence the alert is not conveyed to authorities far away. Also local system could be damaged in the fire

3.2 PROPOSED SYSTEM:

- Arduino UNO based fire monitoring system
- LoRa based alerting system
- Sensor based monitoring
- LoRa can be used for monitoring and alerting

ADVANTAGES

- both local and global fire alert is provided
- very low cost implementation
- also camera based detection is provided

APPLICATIONS

- remote forest fire detection
- remote forest fire prevention
- man less fire fighting
- automated fire department

2.3. BLOCK DIAGRAM

Sensor node

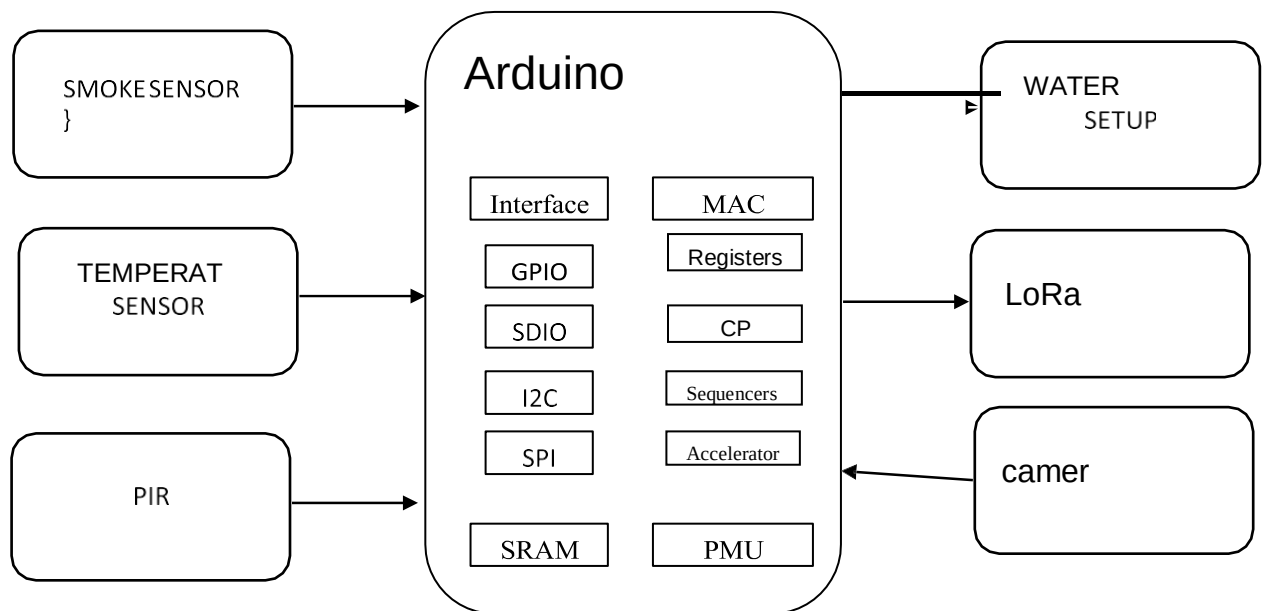


Fig.2.3.1. Block diagram of sensor node

Master node

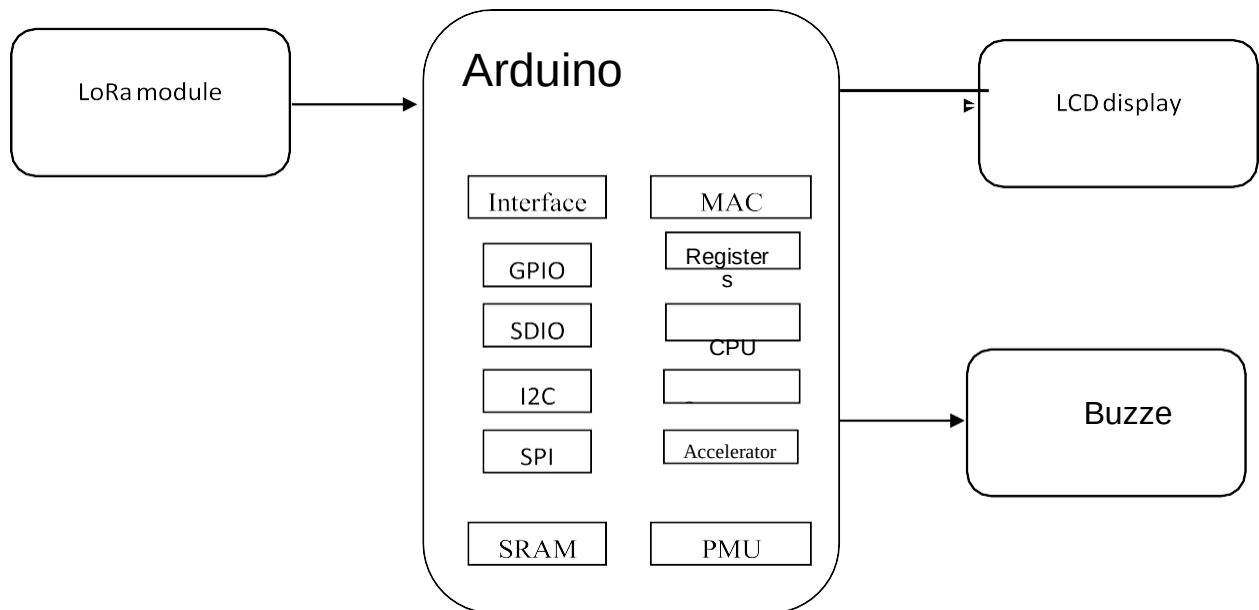


Fig.2.3.2. Block diagram of Sensor node

2.3.1 BLOCK DIAGRAM EXPLANATION

- arduino is the central part of the whole system
- Every other component is connected to it as shown in above diagram
- LM35 and MQ6 is connected to nodemcu through built in ADC in arduino uno
- PIR sensor is connected to a digital pin
- It will go high if any human is present
- Also forest area can be monitored through a camera using this camera fire can be detected using image processing

CHAPTER 4

REQUIREMENTS

4.1 Arduino Software (IDE)

- **Writing Sketches**

- File
- Edit
- Sketch
- Tools
- Help

- Sketchbook
- Tabs, Multiple Files, and Compilation Uploading
- Libraries
- Third-Party Hardware
 - Serial Monitor
- Preferences
- Language

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them.

Writing Sketches

Programs written using Arduino Software (IDE) are called sketches.

These sketches are written in the text editor and are saved with the file

extension .ino. The editor has features for cutting/pasting and for searching/replacing text.

Edit

- **Undo/Redo** Goes back of one or more steps you did while editing; when you go back, you may go forward with Redo.
- **Cut** Removes the selected text from the editor and places it into the clipboard
- **Copy** Duplicates the selected text in the editor and places it into the clipboard.
- **Copy for Forum** Copies the code of your sketch to the clipboard in a form suitable for posting to the forum, complete with syntax coloring.
- **Copy as HTML** Copies the code of your sketch to the clipboard as HTML, suitable for embedding in web pages.
- **Paste** Puts the contents of the clipboard at the cursor position, in the editor.
- **Select All** Selects and highlights the whole content of the editor.
- **Comment/Uncomment** Puts or removes the // comment marker at the beginning of each selected line.
 - **Increase/Decrease Indent** Adds or subtracts a space at the beginning of space at the beginning.
- **Find** Opens the Find and Replace window where you can specify text to search inside the current sketch according to several options.
- **Find Next** Highlights the next occurrence - if any - of the string specified as the search item in the Find window, relative to the cursor position.
- **Find Previous** Highlights the previous occurrence - if any - of the string specified as the search item in the Find window relative to the cursor position.

Sketch

- **Verify/Compile**

- Checks your sketch for errors compiling it; it will report memory usage for code and variables in the console area.
- Upload
- Compiles and loads the binary file onto the configured board through the Upload
- Using Programmer This will overwrite the bootloader on the board; you will need to use Tools > Burn Bootloader to restore it and be able to Upload to USB serial port again. However, it allows you to use the full capacity of the Flash memory for your sketch. Please note that this command will NOT burn the fuses. To do so a Tools - > Burn Bootloader command must be executed.
- Export
 - Compiled Binary Saves a .hex file that may be kept as archive or sent to the board using other tools.
- Show
- Sketch Opens the current sketch folder. Include
 - Adds a library to your sketch by inserting #include statements at the start of your code. For more details, see [libraries](#) below. Additionally, from this menu item you can access the Library Manager and import new libraries from .zip files.
- Add
 - Adds a source file to the sketch (it will be copied from its current location). The new file appears in a new tab in the sketch window. Files can be removed from the sketch using the tab menu accessible clicking on the small triangle icon below the serial monitor one on the right side o the toolbar.

Tools

- Auto
 - Format This formats your code nicely: i.e. indents it so that opening and closing curly braces line up, and that the statements inside curly braces are indented more.
- Burn Bootloader The items in this menu allow you to burn [a bootloader](#) onto the microcontroller on an Arduino board. This is not required for normal use of an Arduino or Genuino board but is useful if you purchase a new ATmega microcontroller (which normally come without a bootloader). Ensure that you've selected the correct board from the Boards menu before burning the bootloader on the target board. This command also set the right fuses.

Help

- Here you find easy access to a number of documents that come with the Arduino Software (IDE). You have access to Getting Started, Reference, this guide to the IDE and other documents locally, without an internet connection. The documents are a local copy of the online ones and may link back to our online website.
- Find in Reference This is the only interactive function of the Help menu: it directly selects the relevant page in the local copy of the Reference for the function or command under the cursor.

Sketchbook

The Arduino Software (IDE) uses the concept of a sketchbook: a standard place to store your programs (or sketches). The sketches in your sketchbook can be opened from the File > Sketchbook menu or from the Open button on the toolbar. The first time you run the Arduino software, it

will automatically create a directory for your sketchbook. You can view or change the location of the sketchbook location from with the Preferences dialog.

device in the ports section of the Windows Device Manager. On Linux, it should

b

e `/dev/ttyACMx` , `/dev/ttyUSBx` or similar. Once you've selected the correct serial port and board, press the upload button in the toolbar or select the Upload item from the Sketch menu. Current Arduino boards will reset automatically and begin the upload. With older boards (pre-Diecimila) that lack auto-reset, you'll need to press the reset button on the board just before starting the upload. On most boards, you'll see the RX and TX LEDs blink as the sketch is uploaded. The Arduino Software (IDE) will display a message when the upload is complete, or show an error.

When you upload a sketch, you're using the Arduino bootloader, a small program that has been loaded on to the microcontroller on your board. It allows you to upload code without using any additional hardware. The bootloader is active for a few seconds when the board resets; then it starts whichever sketch was most recently uploaded to the microcontroller. The bootloader will blink the on-board (pin 13) LED when it starts (i.e. when the board resets).

Libraries

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from the Sketch

> Import Library menu. This will insert one or more `#include` statements at the top of the sketch and compile

the library with your sketch. Because libraries are uploaded to the board with

There is a list of libraries in the reference. Some libraries are included with the Arduino software. Others can be downloaded from a variety of sources or through the Library Manager. Starting with version 1.0.5 of the IDE, you do can import a library from a zip file and use it in an open sketch. See these instructions for installing a third- party library.

To write your own library, see this [tutorial](#).

Third-Party Hardware

Support for third-party hardware can be added to the hardware directory of your sketchbook directory. Platforms installed there may include board definitions (which appear in the board menu), core libraries, bootloaders, and programmer definitions. To install, create the hardware directory, then unzip the third-party platform into its own sub-directory. (Don't use "arduino" as the sub- directory name or you'll override the built-in Arduino platform.) To uninstall, simply delete its directory.

For details on creating packages for third-party hardware, see the [Arduino IDE 1.5 3rd party](#)

[Hardware specification](#).

Serial Monitor

software to use the selected language. If your operating system language is not supported, the Arduino Software (IDE) will default to English.

You can return the software to its default setting of selecting its language based on your operating system by selecting System Default from the Editor Language drop-down. This setting will take effect when you restart the Arduino Software (IDE). Similarly, after changing your operating system's settings, you must restart the Arduino Software (IDE) to update it to the new default language.

Boards

The board selection has two effects: it sets the parameters (e.g. CPU speed and baud rate) used when compiling and uploading sketches; and sets the file and fuse settings used by the burn bootloader command. Some of the board definitions differ only in the latter, so even if you've been uploading successfully with a particular selection you'll want to check it before burning the bootloader. You can find a comparison table between the various boards [here](#).

Arduino Software (IDE) includes the built in support for the boards in the following list, all based on the AVR Core. The [Boards Manager](#) included in the standard installation allows to add support for the growing number of new boards based on different cores like Arduino Due, Arduino Zero, Edison, Galileo and so on.

- Arduino Yùn
 - An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
- Arduino/Genuino Uno
 - An ATmega328 running at 16 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.
- Arduino Diecimila or Duemilanove w/ ATmega168
 - An ATmega168 running at 16 MHz with auto-reset.
- Arduino Nano w/ ATmega328
 - An ATmega328 running at 16 MHz with auto-reset. Has eight analog inputs.
- Arduino/Genuino Mega 2560
 - An ATmega2560 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.
- Arduino Mega
 - An ATmega1280 running at 16 MHz with auto-reset, 16 Analog

In, 54 Digital I/O and 15 PWM.

- Arduino Mega ADK
 - An ATmega2560 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.
- Arduino Leonardo
 - An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
- Arduino/Genuino Micro
 - An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
- Arduino Esplora
 - An ATmega32u4 running at 16 MHz with aut reset.
- Arduino Mini w/ ATmega328
 - An ATmega328 running at 16 MHz with auto-reset, 8 Analog In, 14 Digital I/O and 6 PWM. Arduino Ethernet
 - Equivalent to Arduino UNO with an Ethernet shield: An ATmega328 running at 16 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.
- Arduino Fio
 - An ATmega328 running at 8 MHz with auto-reset. Equivalent to Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328, 6 Analog In, 14 Digital I/O and 6 PWM. Arduino BT w/ ATmega328
 - ATmega328 running at 16 MHz. The bootloader burned (4 KB) includes codes to initialize the on-board bluetooth module, 6 Analog In, 14 Digital I/O and 6 PWM.. LilyPad Arduino USB
 - An ATmega32u4 running at 8 MHz with auto-reset, 4 Analog In, 9 Digital I/O and 4 PWM. LilyPad Arduino
 - An ATmega168 or ATmega132 running at 8 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.
- Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328
 - An ATmega328 running at 16 MHz with auto-reset. Equivalent to

Arduino Duemilanove or Nano w/ ATmega328; 6 Analog In, 14 Digital I/O and 6 PWM. ArduinoNG or older w/ ATmega168

- An ATmega168 running at 16 MHz without auto-reset. Compilation and upload is equivalent to Arduino Diecimila or Duemilanove w/ ATmega168, but the bootloader burned has a slower timeout (and blinks the pin 13 LED three times on reset); 6 Analog In, 14 Digital I/O and 6 PWM.

Robot Control An ATmega328 running at 16 MHz with auto-

Digital I/O and 2 PWM.

4.2 Python IDE

An Introduction to Python

The Raspberry Pi gets the first half of its name from a long-standing tradition of using fruit Tangerine to more recognizably modern brands including Apple and BlackBerry—but the second half comes courtesy of the Python programming language.

4.2.1 Introducing Python

Flexible and powerful, Python was originally developed in the late 1980s at the National Research Institute for Mathematics and Computer Science by Guido van Rossum as a successor to the ABC language. Since its introduction, Python has grown in popularity thanks to what is seen as a clear and expressive syntax developed with a focus on ensuring that code is readable. Python is a high-level language. This means that Python code is written in largely recognizable English, providing the Pi

With commands in a manner that is quick to learn and easy to follow. This is in marked contrast to low-level languages, like assembler, which are closer to how

the computer “thinks” but almost impossible for a human to follow without experience. The high-level nature and clear syntax of Python make it a valuable tool for anyone who wants to learn to program. It is also the language that is recommended by the Raspberry Pi Foundation for those looking to progress from the simple Scratch

.Example: Hello World

As you learned in Chapter 10, “An Introduction to Scratch”, the easiest way to learn a new programming language is to create a project that prints “Hello World!” on the screen. In Scratch, you just had to drag and drop bricks of prewritten code, but in Python, you need to write this program entirely by hand. A Python project is, at heart, nothing more than a text file containing written instructions for the computer to follow. This file can be created using any text editor. For example, if you enjoy working at the console or in a terminal window, you can use nano; or if you prefer a graphical user interface (GUI), you can use Leafpad. Another alternative is to use an integrated development environment (IDE) such as IDLE, which provides Python-specific functionality that’s missing from a standard text editor, including syntax checking, debugging facilities and the ability to run your program without having to leave the editor. This chapter gives you instructions on how to create Python files using IDLE, but of course, the IDE program that you choose to use for programming is up to you.

The chapter also includes instructions for running your created files directly from the terminal, which can be used in conjunction any

Figure 11-1), so anything you type in the initial window will be immediately executed. To open a new Python project which can be executed later, click on the File menu and choose New Window to open a blank

file. The IDLE Python Shell window.

It’s good practice to start all Python programs with a line known as a shebang, which gets its name from the # and ! characters at the beginning of the line. This line tells

the operating system where it should look for the Python files. Although this is not entirely necessary for programs that will be run from within IDLE or will call Python explicitly at the terminal, it is required for programs that are run directly by calling the program's filename. To ensure the program runs regardless of where the Python executable is installed, the first line of your program should read as follows: `#!/usr/bin/env python`.

This line tells the operating system to look at the `$PATH` environment variable—which is where Linux stores the location of files that can be executed as programs—for the location of Python, which should work on any Linux distribution used on the Pi. The `$PATH` variable contains a list of directories where executable files are stored, and is used to find programs when you type their name at the console or in a terminal window. To achieve the goal of printing out a message, you should use Python's `print` command. As its name suggests, this command prints text to an output device—by default, to the console or terminal window from which the program is being executed. Its usage is simple: any text following the word `print` and placed between quotation marks will be printed to the standard output device. Enter the following line in your new project:

```
print "Hello, World!"
```

The final program should look like this: `#!/usr/bin/env python`
`print "Hello, World!"`

If you're creating the example program in IDLE rather than a plain text editor, you'll notice that the text is multicolored, (where colours are represented as differing shades of grey in the print edition). This is a feature known as syntax highlighting, and is a feature of IDEs and the more-advanced text editing tools. Syntax highlighting changes the colour of sections of the text function, in order to make the program a print command or forgetting to comment out a remark. For this short example, syntax highlighting isn't necessary—but in larger programs, it can be an invaluable tool for finding errors. Syntax highlighting in IDLE

Before you run your program, save it as `helloworld.py` using the File menu. If you're using IDLE, the file will be given the extension `.py` automatically. If you're using a text editor, be sure to type `.py` at the end of the filename (not `.txt`) when you save it. This extension indicates that the file contains Python code—although Python is clever enough to run the program even

if it's saved with a different file extension. How you run the file will depend on whether you're using IDLE or a text editor. In IDLE, simply choose Run Module from the Run menu, or press the F5 key on the keyboard. This will switch IDLE back to the Python shell window and run the program. You should then see the message `Hello, World!` appear on screen in blue. If not, check your syntax—in particular, check that you have quotation marks at both the beginning and end of the message on the print line.

Running `helloworld.py` in IDLE

If you created the `helloworld.py` program in a text editor, you'll need to open a terminal window from the Accessories menu on the desktop. If you saved the file anywhere except your home directory, you'll also have to use the `cd` command to change to that directory (see Chapter 2, "Linux System Administration"). Once you're in the right directory, you can run your program by typing the following: `python helloworld.py`. This tells the operating system to run Python and then load the `helloworld.py` file for execution. Unlike the Python shell in IDLE, Python will quit when it reaches the end of the file and return you to the terminal. The result, however, is the same: the message `Hello, World!` is printed to the standard output.

Running `helloworld.py` at the terminal

Making Python Programs Executable Normally, the only way to run a Python program is to tell the Python software to open the file. With the shebang line at the top of the file, however, it's possible to execute the file directly without having to call Python first. This can be a useful way of making your own tools that can be executed at the terminal: once copied into a location in the system's `$PATH` environment variable, the Python program can be called

simply by typing its name. First, you need to tell Linux that the Python file should be marked as executable—an attribute that means the file is a program. To protect the system from malware being downloaded from the Internet this attribute isn't automatically set, since only files that are marked as executable will run. To make the `helloworld.py` file executable, use the `chmod` command (described in detail in Chapter 2, "Linux System Administration") by typing the following: `chmod +x helloworld.py` Now try running the program directly by typing the following:

```
./helloworld.py
```

Despite the fact that you didn't call the Python program, the `helloworld.py` program should run just the same as if you'd typed `python helloworld.py`. The program can only be run by calling it with its full location—`/home/pi/helloworld.py`—or from the current directory by using `./` as the location. To make the file accessible in the same way as any other terminal command, it needs to be copied to `/usr/local/bin` with the following command: `sudo cp helloworld.py /usr/local/bin/`

The `sudo` prefix is required because, for security reasons, non-privileged users cannot write to the `/usr/local/bin` directory. With the `helloworld.py` file located in `/usr/local/bin`, which is included in the `$PATH` variable, it can be executed from any directory by simply typing its name. Try changing to a different directory, and then run the program by typing the following: `helloworld.py` To make your custom-made programs seem more like native utilities, you can rename them to remove the

`.py` file extension. To change the `helloworld.py` program in this way, just type the following line at the terminal as a single line: `sudo mv`

```
/usr/local/bin/helloworld.py
```

`/usr/local/bin/helloworld` Once renamed, the program can be run simply by typing `helloworld` at the terminal or console.

4.1.2 Python and Networking

So far, you have learned how Python can be used to create standalone

programs, but the language can also be used to create programs that communicate with the outside world over a computer's network connection. This next example, written by Tom Hudson, offers a brief glimpse of these possibilities with a tool for monitoring the users connected to an Internet Relay Chat (IRC) channel. As usual, line: `import sys, socket, time` You used the `sys` and `time` modules previously in the Raspberry Snake program, but you have not yet used `socket`. The `socket` module provides Python with the ability to open, close, read from and write to network sockets—giving Python programs rudimentary networking capabilities. It's the `socket` module that provides this example with its ability to connect to a remote IRC server.

There are some constants needed for this program to operate. Constants are like variables in that they can have values assigned to them—but unlike variables, the value in a constant shouldn't change. To help differentiate a constant from a variable,

it's good practice to use all-capital letters for their names—that way it's easy to see at glance whether a particular section of the code is using a constant or a variable. Type the following two lines into the program:

```
RPL_NAMREPLY = '353'  
RPL_ENDOFNAMES = '366'
```

These are IRC status codes, provided by the server to indicate when particular operations have completed. These are used by the program to know when it has received the required list of names from the IRC server. Next, set up the variables for the server connection by entering the

Following lines:

```
irc = {
```

```

'host'          :
'chat.freenode.net',
'port' : 6667,
'channel'      :
'#raspiuserguide
', 'namesinterval'
: 5
}

```

final line. The host variable should be set to the fully-qualified domain name (FQDN) of the IRC server to which the program will connect. In this example, chat.freenode.net is used, but if you want to customise the program to use a different server, change the domain name here. The port variable tells the program which network port IRC is running on, which will usually be 6667. The channel variable tells Python which channel to join in order to monitor the users, while namesinterval controls how long the program waits to refresh the list of users, measured in seconds. Set up a second dict to store the user-specific variables by typing in the following lines:

```

user = {
'nick'   :   'botnick',
'username'      :
'botuser', 'hostname'
:          'localhost',
'servername' :
'localhost',
'realname' : 'Raspberry Pi Names Bot'
}

```

As with irc, all these variables are stored within a dict called user to make it clear which variables pertain to which section. The nick variable should be set to the IRC nickname the program will use. Don't use your usual nickname if you're planning to connect to the IRC server at the same time; instead, try appending -bot to the end of your name to

make it clear that the user is a program rather than a real person. Do the same with username, and fill in the realname variable with a descriptive message about whom the bot belongs to. The hostname and servername variables can be left set to localhost, or altered to match your Internet address. The socket module requires the user to create a socket object. This object provides network connectivity to the rest of the program. Create the socket object by typing in the following line:

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Next, you need to tell the program to try connecting to the IRC server specified in the variables at the start of the program. Type the following lines:

```
print 'Connecting to %(host)s:%(port)s...' % irc
try:s.connect((irc['host'], irc['port']))
except socket.error:
    print 'Error connecting to IRC server'
    %(host)s:%(port)s' % irc
    sys.exit(1)
```

The try and except commands are included in this code for error handling. If the system fails to connect to the server—because the Pi isn't connected to the Internet, for example, or because the server is down for maintenance—the program will print an error message and gracefully exit. The s.connect line tells the socket module to try connecting to the IRC server, using the host and port variables held in the irc dict. If the program doesn't quit from the exception, it has successfully connected to the IRC server. Before you can get a list of names in a channel, however, you need to identify yourself to the server and issue some commands using the send function of the socket module. Type the following lines into the program:

```

s.send('NICK %(nick)s\r\n' % user)
s.send('USER          %(username)s
%(hostname)s
%(servername)s :%(realname)s\r\n'
%      user)      s.send('JOIN
%(channel)s\r\n'   %      irc)
s.send('NAMES %(channel)s\r\n' %
irc)

```

The send function works in almost exactly the same way as the print function, except that instead of printing to the standardoutput—usually the terminal window or console—it sends the output through the network connection. In this case, the program is sending strings of text to the IRC server and telling it to register the program using the nickname held in the nick variable and the user details held in the username, hostname, servername and realname variables. Next, the program sends the command to join the channel specified in the channel variable, and finally, it sends the command to receive the list of users in that channel.

Although this example is tailored to IRC, the same basic principle can be used to issue commands to any network service—with modifications, this program could be used to list the files on an FTP server, or unread emails on a POP3 server. Receiving data from the socket is a little more complicated `read_buffer =` “Note that there are two single quotes after the equals sign, not one double quote.

Next, create an empty list, which will be used to store the names of users, by typing the following line: `names = []`

The list data type is the same as you used to store the locations in the Raspberry Snake game. Unlike a normal variable, it can store multiple values—in this case, the names of users present in the IRC channel. The next step is to create an infinite loop, during which the program will continuously query the server for user names and print them to the screen.

Start the loop by typing:

```
while True:
```

```
    read_buffer += s.recv(1024)
```

The first line of the loop, following `while True:`, tells the `socket` module to receive 1,024 bytes (1 KB) of data from the IRCserver and place it into the `read_buffer` variable. Because the `+=` operator is used, rather than just `=`, the received data will be appended to anything already in the buffer. The value of 1024 bytes is more or less arbitrary. The next step is to split the buffer into individual lines of text, using the following program lines:

```
lines = read_buffer.split('\r\n')  
  
read_buffer = lines.pop();
```

The first line sets the `lines` variable to a full line of text from the receive buffer, by using the `split` function to find end of line characters—signified by `\r\n`. These characters only occur at the end of a line, so when the buffer has been split in this way you know that `lines` contains only full-line responses from the server. The `pop` instruction in the second line makes sure that only full lines are removed from the `read_buffer`: because responses from the server are read in 1 KB chunks, it's likely that at any given time the buffer will contain only fractions of

a line. When that's the case, the fraction is left in the buffer ready to receive the remainder of the line the next time the loop runs and the next 1 KB chunk is received from the server. At this point, the `lines` variable contains a list of full responses—full lines—received from the server. Type the following to process these lines and find the names of channel participants:

```
for line in lines:
```

This runs through every line found in the `lines` variable, and looks for the numerical IRC response code provided by the server. Although there are plenty of different response codes, this program is only interested in the two defined as constants at the start of the program: 353, which means a list of names follows, and 366, which means the list has ended. The `if` statement looks for the first of these responses, and then uses the `split` function to retrieve these names and add them to the `names` list. Now, the `names` list contains all the names received from the server in response to the program's query. This may not be all the names, however: until the 366 response, which signals the end of the member names, is received, the list is incomplete.

CHAPTER 5

MATERIALS AND METHODS

Hardware Requirements

- Arduino Board
- LM35 – TemperatureSensor
- MQ6 gas sensor
- PIR sensor
- Waterpump
- Lora Module

5.1 ARDUINO BOARD

The Arduino Uno is an open-sourcemicrocontroller board based on the MicrochipATmega328P microcontroller and developed byArduino.cc. The board is 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 board has 14 Digital pins, 6 Analog pins, and programmable with the Arduino IDE (Integrated Development Environment) via a type B USB cable. It can be powered by the USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts. It is also similar to the Arduino Nano and Leonardo. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming

ARDUINO:

setup(): A function present in every Arduino sketch. Run once before the loop() function. Often used to set pinmode to input or output. The setup() function looks like:

```
void setup( ){//code goes here}
```

loop(): A function present in every single Arduino sketch. This code happens over and over again. The loop() is where (almost) everything happens. The one exception to this is setup() and variable declaration. ModKit uses another type of loop called “forever()” which executes over Serial. The loop() function looks like:

```
void loop( ) {  
  
    //code goes here
```

Input: A pin mode that intakes information.

Output: A pin mode that sends information.

HIGH: Electrical signal present (5V for Uno). Also ON or True in boolean logic.

LOW: No electrical signal present (0V). Also OFF or False in boolean logic.

Digital Read: Get a HIGH or LOW reading from a pin already declared as an input. **Digital Write:** Assign a HIGH or LOW value to a pin already declared as an output.

Analog Read: Get a value between or including 0 (LOW) and 1023 (HIGH). This allows you to get readings from analog sensors or interfaces that have more than two states.

analogWrite: Assign a value between or including 0 (LOW) and 255 (HIGH). This allows you to set output to a PWM value instead of just HIGH or LOW.

PWM: Stands for Pulse-Width Modulation, a method of emulating an analog signal through a digital pin. A value between or including 0 and 255. Used with **analogWrite**.

These boards below use the same micro-controller, just in a different package. The Lilypad is designed for use with conductive thread instead of wire and the Arduino

Mini is simply a smaller package without the USB, Barrel Jack and Power Outs.

It depends on what you want to do with it really. There are two different purposes outlined above for the voltage divider, we will go over both.

If you wish to use the voltage divider as a sensor reading device first you need to know the maximum voltage allowed by the analog inputs you are using to read the signal. On an Arduino this is 5V. So, already we know the maximum value we need for V_{out} . The V_{in} is simply the amount of voltage already present on the circuit before it reaches the first resistor. You should be able to find the maximum voltage your sensor outputs by looking on the Datasheet; this is the maximum amount of voltage your sensor will let through given the voltage in of your circuit. Now we have exactly one variable left, the value of the second resistor. Solve for R_2 and you will have all the components of your voltage divider figured out! We solve for R_1 's highest value because a smaller resistor will simply give us a smaller signal which will be readable by our analog inputs.

Powering an analog Reference is exactly the same as reading a sensor except you have to calculate for the Voltage Out value you want to use as the analog Reference.

All of the electrical signals that the Arduino works with are either Analog or Digital. It is extremely important to understand the difference between these two types of signal and how to manipulate the information these signals represent.

DIGITAL

An electronic signal transmitted as binary code that can be either the presence or absence of current, high and low voltages or short pulses at a particular frequency.

Humans perceive the world in analog, but robots, computers and circuits use Digital. A digital signal is a signal that has only two states. These states can vary depending on the signal, but simply defined the states are ON or OFF, never in between.

In the world of Arduino, Digital signals are used for everything with the

exception of Analog Input. Depending on the voltage of the Arduino the ON or HIGH of the Digital signal will be equal to the system voltage, while the OFF or LOW signal will always equal 0V. This is a fancy way of saying that on a 5V Arduino the HIGH signals will be a little under 5V and on a 3.3V Arduino the HIGH signals will be a little under 3.3V.

To receive or send Digital signals the Arduino uses Digital pins # 0 - # 13. You may also setup your Analog In pins to act as Digital pins. To set up Analog In pins as Digital pins use the command:

```
□ pinMode(pinNumber, value);
```

Where pinNumber is an Analog pin (A0 – A5) and value is either INPUT or OUTPUT. To setup Digital pins use the same command but reference a Digital pin for pinNumber instead of an Analog In pin. Digital pins default as input, so really you only need to set them to OUTPUT in pinMode. To read these pins use the command:

```
□ digitalWrite(pinNumber);
```

Where pinNumber is the Digital pin to which the Digital component is connected. The digitalWrite command will return either a HIGH or a LOW signal. To send a Digital signal to a pin uses the command:

```
□ digitalWrite(pinNumber, value);
```

Where pinNumber is the number of the pin sending the signal and value is either HIGH or LOW. The Arduino also has the capability to output a Digital signal that acts as an Analog signal;

this signal is called Pulse Width Modulation (PWM). Digital Pins # 3, # 5, # 6, # 9, # 10 and #11 have PWM capabilities. To output a PWM signal use the command:

```
□ analogWrite(pinNumber, value);
```

Where pinNumber is a Digital Pin with PWM capabilities and value is a number

between 0 (0%) and 255 (100%). For more information on PWM see the PWM worksheets or S.I.K. circuit 12.

THINGS TO REMEMBER ABOUT DIGITAL:

- Digital Input/Output uses the Digital pins, but Analog In pins can be used as Digital. To receive a Digital signal use:
`digitalRead(pinNumber);`
- To send a Digital signal use:
`digitalWrite(pinNumber, value);` Digital Input and Output are always either HIGH or LOW

All of the electrical signals that the Arduino works with are either Analog or Digital. It is extremely important to understand the difference between these two types of signal and how to manipulate the information these signals represent.

ANALOG

Humans perceive the world in analog. Everything we see and hear is a continuous transmission of information to our senses. The temperatures we perceive are never 100% hot or 100% cold, they are constantly changing between our ranges of acceptable temperatures. (And if they are out of our range of acceptable temperatures then what are we doing there?) This continuous stream is what defines analog data. Digital information, the complementary concept to Analog, estimates analog data using only ones and zeros.

In the world of Arduino an Analog signal is simply a signal that can be HIGH (on), LOW (off) or anything in between these two states. This means an Analog signal has a voltage value that can be anything between 0V and 5V (unless you mess with the Analog Reference pin).

Analog allows you to send output or receive input about devices that run at percentages as well as on and off. The Arduino does this by sampling the voltage

signal sent to these pins and comparing it to a voltage reference signal (5V). Depending on the voltage of the Analog signal when compared to the Analog Reference signal the Arduino then assigns a numerical value to the signal somewhere between 0 (0%) and 1023 (100%). The digital system of the Arduino can then use this number in calculations and sketches.

To receive Analog Input the Arduino uses Analog pins # 0 - # 5. These pins are designed for use with components that output Analog information and can be used for Analog Input. There is no setup necessary, and to read them use the command:

```
□ analogRead(pinNumber);
```

Where pinNumber is the Analog In pin to which the the Analog component is connected. The analogRead command will return a number including or between 0 and 1023.

The Arduino also has the capability to output a digital signal that acts as an Analog signal; this signal is called Pulse Width Modulation (PWM). Digital Pins # 3, # 5, # 6, # 9, # 10 and #11 have PWM capabilities. To output a PWM signal use the command:

```
□ analogWrite(pinNumber, value);
```

Where pinNumber is a Digital Pin with PWM capabilities and value is a number between 0 (0%) and 255 (100%). On the Arduino UNO PWM pins are signified by a ~ sign. For more information on PWM see the PWM worksheets or S.I.K. circuit 12.

THINGS TO REMEMBER ABOUT ANALOG:

- Analog Input uses the Analog In pins, Analog Output uses the PWM pins To receive an Analog signal use:
analogRead(pinNumber);
- To send a PWM signal use: analogWrite(pinNumber, value);
- Analog Input values range from 0 to 1023 (1024 values because it

uses 10 bits, 210) PWM Output values range from 0 to 255 (256 values because it uses 8 bits, 28)

All of the electrical signals that the Arduino works with are either input or output. It is extremely important to understand the difference between these two types of signal and how to manipulate the information these signals represent.

OUTPUT SIGNALS

Output to the Arduino pins is always Digital, however there are two different types of Digital Output; regular Digital Output and Pulse Width Modulation Output (PWM). Output is only possible with Digital pins # 0 - # 13. The Digital pins are preset as Output pins, so

unless the pin was used as an Input in the same sketch, there is no reason to use the pinMode

command to set the pin as an Output. Should a situation arise where it is necessary to reset a Digital pin to Output from Input use the command:

```
□ pinMode(pinNumber, OUTPUT);
```

Where pinNumber is the Digital pin number set as Output. To send a Digital Output signal use the command:

```
□ digitalWrite(pinNumber, value);
```

Where pinNumber is the Digital pin that is outputting the signal and value is the signal. When outputting a Digital signal value can be either HIGH (On) or LOW (Off).

```
□ analogWrite(pinNumber, value);
```

Where pinNumber is a Digital Pin with PWM capabilities and value is a number between 0 (0%) and 255 (100%). For more information on PWM see the PWM

worksheets or S.I.K.

Output can be sent to many different devices, but it is up to the user to figure out which kind of Output signal is needed, hook up the hardware and then type the correct code to properly use these signals.

THINGS TO REMEMBER ABOUT OUTPUT:

- Output is always Digital
- There are two kinds of Output: regular Digital or PWM (Pulse Width Modulation) To send an Output signal use

All of the electrical signals that the Arduino works with are either input or output. It is extremely important to understand the difference between these two types of signal and how to manipulate the information these signals represent.

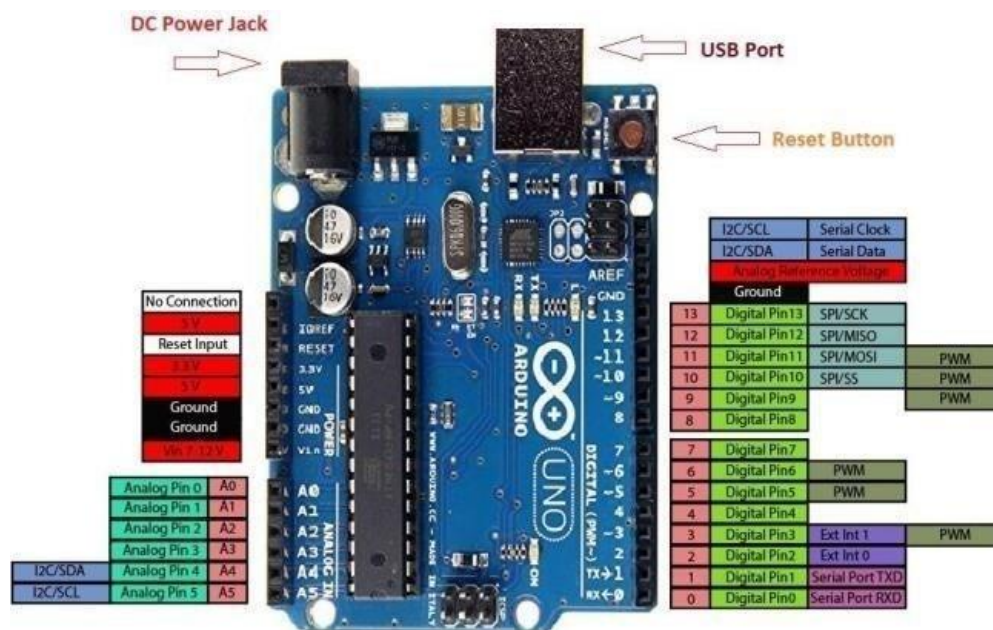


Fig.4.1 Arduino Board

Analog Input enters your Arduino through the Analog In pins # 0 - # 5. These signals originate from analog sensors and interface devices. These analog sensors and devices use voltage levels to communicate their information instead of a simple yes (HIGH) or no (LOW). For this reason you cannot use a digital pin as an input pin for these devices. Analog Input pins are used only for receiving Analog signals. It is only possible to read the Analog Input pins so there is no command necessary in the setup() function to prepare these pins for input. To read the Analog Input pins use the command:

AnalogRead (pinNumber);

Where pinNumber is the Analog Input pin number. This function will return an Analog Input reading between 0 and 1023. A reading of zero corresponds to 0 Volts and a reading of 1023 corresponds to 5 Volts. These voltage values are emitted by the analog sensors and interfaces. If you have an Analog Input that could exceed Vcc + .5V you may change the voltage that 1023 corresponds to by using the Aref pin. This pin sets the maximum voltage parameter your Analog Input pins can read. The Aref pin's preset value is 5V.

Digital Input can enter your Arduino through any of the Digital Pins # 0 - # 13. Digital Input signals are either HIGH (On, 5V) or LOW (Off, 0V). Because the Digital pins can be used either as input or output you will need to prepare the Arduino to use these pins as inputs in your

setup()function. To do this types the

command:

pinMode(pinNumber, INPUT);

inside the curly brackets of the setup() function where pinNumber is the Digital pin number you wish to declare as an input. You can change the pinMode in the loop()function if you need to switch a pin back and forth between input and output, but it is usually set in the setup()function and left untouched in the loop()function. To read the Digital pins set as inputs

use the command:

Input can come from many different devices, but each device's signal will be either Analog or Digital, it is up to the user to figure out which kind of input is needed, hook up the hardware and then type the correct code to properly use these signals.

X1:

- DE-9 serial connector

Used to connect computer (or other devices) using RS-232 standard. Needs a serial cable, with at least 4 pins connected: 2, 3, 4 and 5. Works only when JP0 is set to 2-3 position.

DC1:

- 2.1 mm. power jack

Used to connect external power source. Centre positive. Voltage Regulator Works with regulated +7 to +20 volts DC (9v. to 12v. is recommended). It is possible to alternatively connect external power using 9v. pin or 5v. pin. (see POWER PINOUT)

ICSP:

2x3 pin header Used to program Atmega with bootloader. The number 1 on both sides

of the board indicates cable pin1 position. Used to upload sketches on Atmega ICs without bootloader (available only in Arduino IDE versions 0011 and 0012).

JP0

3 pins jumper When in position 2-3, this jumper enables serial connection (through X1 connector) to/from computer/devices. Use this as default position. When in position

1-2, it disables serial communication, and enables external pull-down resistors on pin0 (RX) and pin1 (TX). Use this only to prevent noise on RX (that seems incoming data to Atmega), that sometimes makes sketch not starting. When removing this jumper, serial communication is disabled, and pin0 and pin1 work as a normal (floating) digital pin. Useful when more digital pins are needed, but only when serial communication is not necessary. External pull- down/pull-up resistor is required.

JP4

2 pins jumper When in position 1-2, this jumper enables auto reset feature, useful when uploading a sketch to Arduino, resetting Atmega automatically. It makes unnecessary to press reset button (S1) when uploading sketches. Be sure that computer COM Port speed is set to 19200bps otherwise auto reset will not work properly. If removed, disables auto reset feature. Very useful to prevent undesired Atmega reset when using sketches that needs serial communication. Auto reset works with DTR pulse on serial pin4. Sometimes Arduino senses a DTR pulse when connecting X1 (serial connector) and some softwares sends a DTR pulse when it starts or when it closes, that makes Atmega reset when not desired.

S1

Tactile button This button resets Atmega, to restart uploaded sketch or to prepare Arduino to receive a sketch through serial connector (when auto reset is not active).

LEDS

Indicative leds POWER led Turns on when Arduino is powered through DC1, +9v. pin or +5v. pin. RX led Blinks when receiving data from computer/device through serial connection. TX led Blinks when sending data to computer/device through serial connection. L led This led is connected to digital pin13 with a current limiter resistor (that doesn't affect pin13). Useful to test sketches. It is normal to blink when

bootloading too.

POWER PINOUT

6 pin header

RST pin

Makes Atmega reset when connected to GND. Useful for Shield Boards, or to connect external reset.

NC pin

This pin is not connected in Arduino S3v3. Arduino Diecimila has a 3.3 volts pin in the same position.

+9v pin

When Arduino DC1 is powered (with battery or DC adaptor), this pin is used as Vout, with the same voltage supplied on DC1 (see DC1), minus 0,7 volts. The total supplied current depends on external power source capacity. When Arduino DC1 is not powered, +9v. pin can

be used as Vin, connecting it to an external regulated power source (+7 to +20 volts) and connecting 0v. pin to external power source GND. In this case,

+5v pin

It can be used as Vout, supplying +5 volts. +5v.pin When Arduino DC1 is powered (with battery or DC adaptor), +5v. pin supplies +5 volts as a Vout pin. The total supplied current depends on Voltage Regulator (7805 supplies up to 1A). This applies only to +5v. pin:

Atmega in/out pins only supplies max. 40mA on each pin. When Arduino DC1 is not powered, this pin can be used as Vin, connecting it to a regulated +5v. and connecting 0v. pin to power source GND. In this case, +9v.pin is inactive. 0v. pin (GND) Two 0v. pins between

+5v. and +9v. / One

0v. pin

Beside AREF pin. When Arduino DC1 is powered, 0v.pin supplies 0 volts reference (GND) for +5v. pin and +9v. pin. When DC1 is not powered, and Arduino is powered through

+5v.pin or +9v. pin, 0v. pin must be used as GND reference, connecting it to the external power source GND.

GND pin

see 0v. pin (GND).

AREF pin

The AREF can be set to AVcc (default), internal 2.56 volts (Atmega8), internal 1.1 volts (Atmega168), or external AREF. In case of AVcc or internal AREF, AREF pin can be used to attach an external capacitor to decouple the signal, for better noise performance. In case of external AREF, AREF pin is used to attach the external reference voltage. Remember that it is necessary to change the fuses (wiring.c file), and re-upload sketch, before connecting external voltage to AREF

SOFTWARE TIPS

When bootloading an Atmega8 chip with Arduino 0010, there is a command (-i800) that makes bootloader delay 10 minutes. So, if you need to use bootloader, use command line

instead of IDE, removing “-i800” command and adding “-F” command, or use Arduino 0007 IDE. To upload sketches Arduino 0010 works fine.

ARDUINO S3v3 NEW FEATURES

Full compatible with Shield Boards (Version 2 is the only Arduino Board not compatible with Shield Boards because of ICSP header wrong position, and tall components);

- ☐ AVcc LP filter to reduce noise level on
- ☐ ADC; auto reset feature;
- ☐ auto reset enable/disable jumper, to avoid not desired
- ☐ resetting; arduinoDiecimila compatible reset pin;
- ☐ pin13 onboard led, with current limiter
- ☐ resistor; TX and RX onboard leds;
- ☐ power led with appropriate current limiter resistor (less 20mA of consumption);
- ☐ jumper to disable serial communication and to enable RX external pull down resistor,
to avoid “RX floating error”. This feature allows to use digital pin0 and pin1 as a normal pin, when serial communication is not needed;
- ☐ all similar components (diodes, transistors, leds, capacitors) has the same board orientation (to makes easier to mount with less mistakes);
- ☐ no wires between pads, more space between wires, larger wires, larger pads (better for etching, soldering and drilling, with no short circuits, soldering bridges or open wires in corrosion);
- ☐ only 3 wire bridges;
- ☐ electrolytic capacitor (in serial to TTL circuit) changed to bipolar type (to avoid inverted voltage problem when serial cable is not connected); All jumpers are right angle type, to allow Shield Boards use.

5.2. LoRa (Long Range)

LoRa WAN is a low power wide area network technology (LP-WAN) designed for Internet of Things (IoT) and smart sensor applications.



Fig.5.2.1 LoRa RF click

5.2.1WORKING

In LoRa technology, a message transmitted by any device can be received by single or multiple gateways. The received messages will be forwarded to the central network for processing. Smart server architecture will handle these messages to each related applications.

LoRa alliance is an open, non-profit organization of members that standardize,

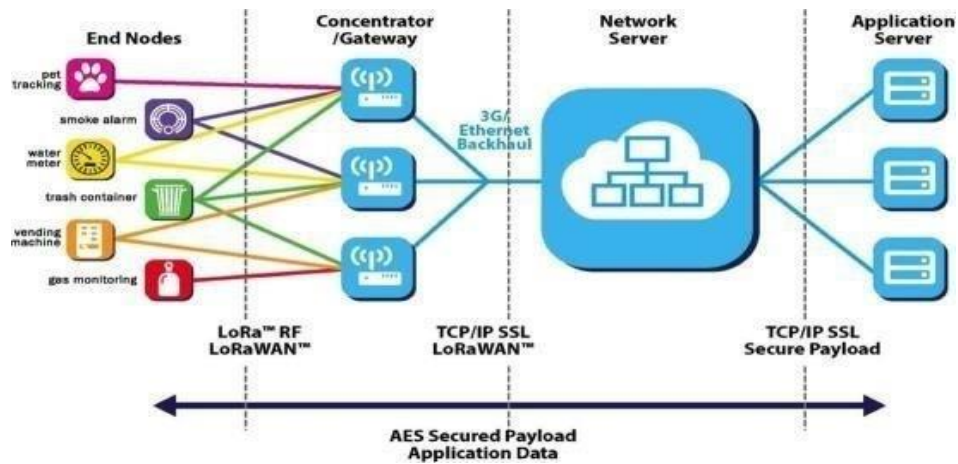


Fig.5.2.2.Working process of LoRa

5.2.2. Why LoRa®?

LoRa Fills a Technology Gap

LoRa Technology has revolutionized IoT by enabling data communication over a long range while using very little power. When connected to a non-cellular LoRaWAN network, LoRa devices accommodate a vast range of IoT applications by transmitting packets with important information. LoRaWAN fills the technology gap of Cellular and Wi-Fi/BLE based networks that require either high bandwidth or high power, or have a limited range or inability to penetrate deep indoor environments.

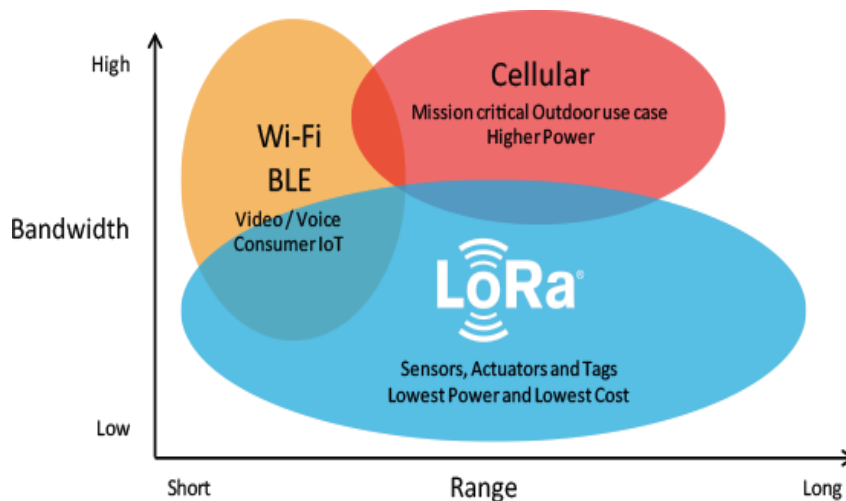


Fig.5.2.2.LoRa graph

LoRa Complements Wi-Fi, Bluetooth and Cellular

Like Wi-Fi, LoRaWAN operates in the unlicensed band and supports indoor applications; like Cellular, LoRa Technology is highly secure from end devices to the application server, and is suitable for outdoor applications. LoRa devices and the LoRaWAN protocol combine these features of Wi-Fi and Cellular networks to offer an efficient, flexible and economical connectivity solution ideal for IoT applications whether indoor or outdoor and installed in public, private or hybrid networks. Simple sensor data can fuel analytics platforms, such as those for artificial intelligence and machine learning. These require data diversity which is made possible by low-cost LoRa-enabled sensors.

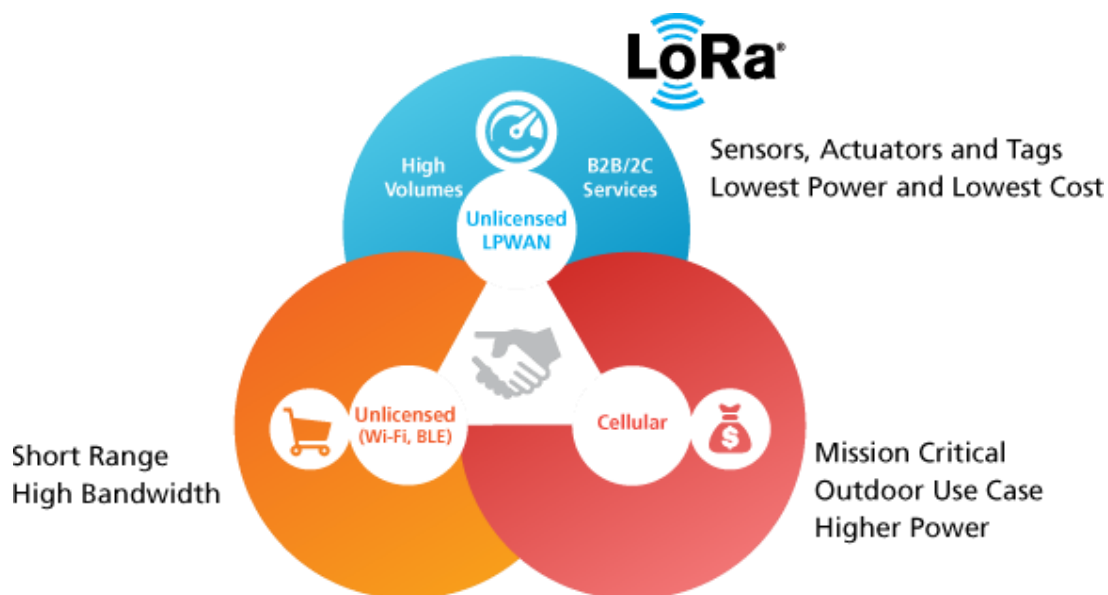


Fig.5.2.3.2.LoRa compliments

Network Availability near You

LoRa devices operate on the LoRaWAN open standard which is backed by the [LoRa Alliance®](https://www.lo-ralliance.org/), an open, non-profit association committed to driving its adoption worldwide.

The LoRa Alliance's ecosystem of over 500 members includes over 100 public network operators with service in 58 countries and growing. Public, private and hybrid LoRaWAN networks are globally available and accessible in over

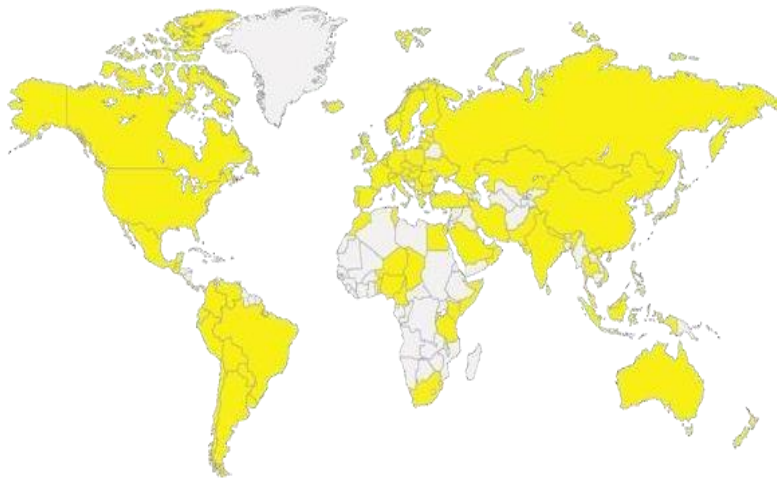


Fig.5.2.3.3.Countries using LoRa

A Rapidly Scaling Ecosystem

The ecosystem supporting LoRa and LoRaWAN includes a comprehensive collection of network operators, hardware manufacturers, software designers, service providers, universities, and industry associations that play a key role in creating and enabling devices, networks and applications. The ecosystem is continuously growing — and continues to grow at an impressive rate in conjunction with widespread adoption of LoRa Technology.



Fig.5.2.3. LoRa Network architecture

Simplified Go-To-Market

Semtech's LoRa Technology enables innovators to overcome the age-old challenges that come with launching an IoT solution. The ecosystem supporting LoRa Technology is all-inclusive, providing end-to-end integration from silicon to services. As an open platform that operates on the unlicensed band, LoRa Technology is flexible for various business models to create profitability. The standardized LoRaWAN protocol is interoperable, enabling solutions to scale, bundle and evolve. Semtech and its ecosystem partners are committed to simplifying and accelerating the path for IoT innovators to bring products to market and demonstrate real-world use cases. A great place for innovators to start is in the LoRa Developer Portal – the online forum for IoT and M2M technologists to connect, learn and share.

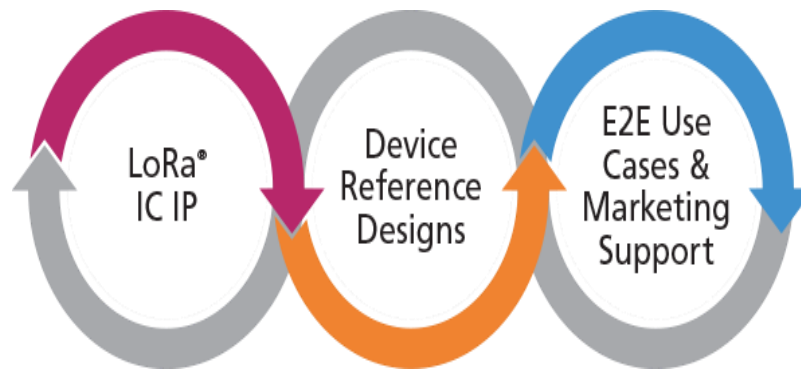


Fig.5.2.3.4.LoRa simplified GO_TO network

What is a LoRa Gateway?

LoRa gateway is a sophisticated software architecture designed to carry out communication in LoRa network terminology. It acts as a smart hub between end devices and application network server. In future, it will be preinstalled on all

IoT enabled hardware to support wide range of applications.

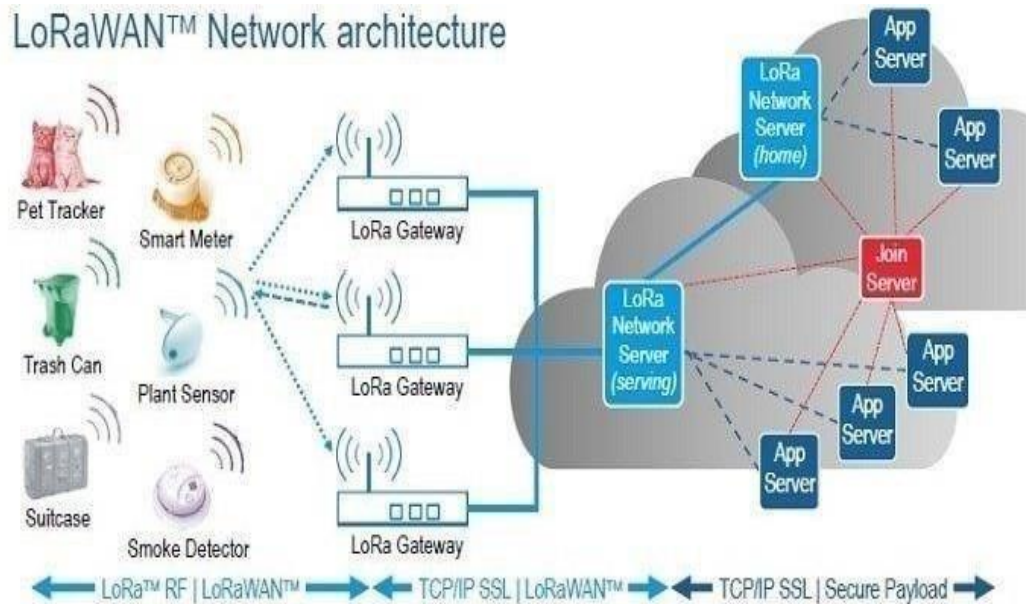


Fig.5.2.3.5.LoRa network architecture

5.2.4. LoRa WAN Specification

Application and future of LoRa WAN technology

Smart City: LoRa WAN will be inevitable technology in future smart city applications together with Internet of Things like:

- Smart lighting
- Air quality and pollution monitoring
- Smart parking and vehicle management
- Facilities and infrastructure management
- Fire detection and management
- Waste management

Industrial Applications: LoRa WAN is suitable for wide range of industrial applications.

- Radiation and
- leak detection
- Smartsensor technologyItem location and
- □ tracking
- Shipping and transportati on

Smart home applications: In future, billions of smart devices and home appliances will be connected to internet.

Enhanced home security

Home automation for IoT enables smart appliances

Healthcare: LoRa is one of the best solutions for connecting healthcare devices efficiently

- Health monitoring devices and management
- Wearable technology

Agriculture: LoRa technology can be used in smart agriculture and farming applications.

- Smart farming and livestock management
- Temperature and moisture monitoring Water level sensors and irrigation control

5.3. Gas Sensor

Ideal sensor for use to detect the presence of a dangerous LPG leak in your car or in a service station, storage tank environment. This unit can be easily incorporated into an alarm unit, to sound an alarm or give a visual indication of the LPG concentration. The sensor has excellent sensitivity combined with a quick response time. The sensor can also sense iso-butane, propane, LNG and cigarette smoke.



Fig.5.3.1.Gas sensor

A Typical human nose has 400 types of scent receptors enabling us to smell about 1 trillion different odours. But still many of us do not have the capacity to identify the type or concentration of gas present in our atmosphere. This is where Sensors comes in, there are many [types of sensors](#) to measure different parameters and a Gas sensor is one which comes handy in applications where we have to detect the variation in the concentration of toxic gases in order to maintain the system safe and avoid/caution any unexpected threats. There are various gas sensors to detect gases like oxygen, Carbon Dioxide, Nitrogen, methane etc. They can also be commonly found in devices that are used to detect the leakage of the harmful gases, monitor the air quality in industries and offices etc.

In this article, we will learn more about gas sensors, their construction, types, working and how they can be used to measure the required type and concentration of Gas in our atmosphere. There are [many types of Gas sensors](#) but the MQ type gas sensors are commonly used and widely popular so will

focus more on these types of sensors for this article.

Introduction

A gas sensor is a device which detects the presence or concentration of gases in the atmosphere. Based on the concentration of the gas the sensor produces a corresponding potential difference by changing the resistance of the material inside the sensor, which can be measured as output voltage. Based on this voltage value the type and concentration of the gas can be estimated.



Fig.5.3.2.gas sensor architecture

The type of gas the sensor could detect depends on the sensing material present inside the sensor. Normally these sensors are available as modules with comparators as shown above. These comparators can be set for a particular threshold value of gas concentration. When the concentration of the gas exceeds this threshold the digital pin goes high. The analog pin can be used to measure the concentration of the gas.

Construction

All Of the above-listed types, the most commonly used gas sensor is the Metal oxide semiconductor based gas sensor. All Gas sensors will consist of a

sensing element which comprises of the following parts.

1. Gas sensing
2. Heater Coil
3. Electrode line
4. Tubular ceramic
5. Electrode

The below image illustrates the parts present in a metal oxide gas sensor

The purpose of each of these elements is as below:

Gas sensing layer: It is the main component in the sensor which can be used to sense the variation in the concentration of the gases and generate the change in electrical resistance. The gas sensing layer is basically a chemiresistor which changes its resistance value based on the

The concentration of particular gas in the environment. Here the sensing element is made up of a Tin Dioxide (SnO_2) which is, in general, has excess electrons (donor element). So whenever toxic gases are being detected the resistance of the element changes and the current flown through it varies which represents the change in concentration of the gases.

Heater coil: The purpose of the heater coil is to burn-in the sensing element so that the sensitivity and efficiency of the sensing element increases. It is made of Nickel- Chromium which has a high melting point so that it can stay heated up without getting melted.

Electrode line: As the sensing element produces a very small current when the gas is detected it is more important to maintain the efficiency of carrying those small currents. So Platinum wires come into play where it helps in moving the electrons efficiently.

Electrode: It is a junction where the output of the sensing layer is connected to the Electrode line. So that the output current can flow to the required terminal. An electrode here is made of Gold (Au –Aurum) which is a very good conductor.

Tubular ceramic: In between the Heater coil and Gas sensing layer, the tubular ceramic exists which is made of Aluminum oxide (Al_2O_3). As it has high melting point, it helps in maintaining the burn-in (preheating) of the sensing layer which gives the high sensitivity for the sensing layer to get efficient output current.

Mesh over the sensing element: In order to protect the sensing elements and the setup, a metal mesh is used over it, which is also used to avoid/hold the dust particles entering into the mesh and prevent damaging the gas sensing layer from corrosive particles.

How to use a Gas sensor?

A basic gas sensor has 6 terminals in which 4 terminals (A, A, B, B) acts input or output and the remaining 2 terminals (H, H) are for heating the coil. Of these 4 terminals, 2 terminals from each side can be used as either input or output (these terminals are reversible as shown in the circuit diagram) and vice versa.

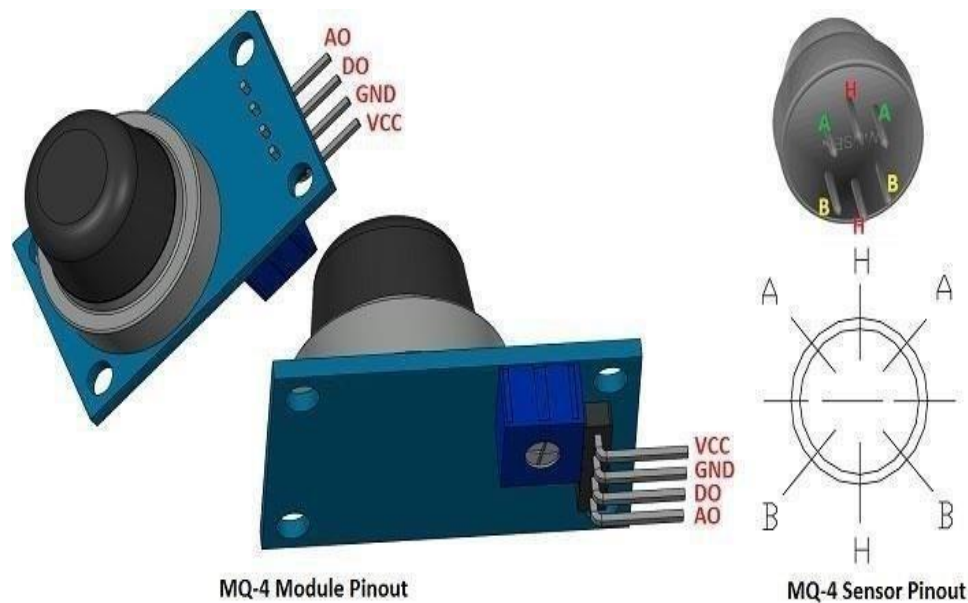


Fig.5.3.4.MQ4 module pinout

These sensors are normally available as modules (shown right), these modules consist of the gas sensor and a [comparator IC](#). Now let's see the pin description of the gas sensor module which we will generally use with an Arduino. The gas sensor module basically consists of 4 terminals

As discussed earlier the output of a gas sensor alone will be very small (in mV) so an external circuit has to be used in order to get a digital high low output from the sensor. For this purpose, a comparator ([LM393](#)), [adjustable potentiometer](#), [some resistors](#) and [capacitors](#) are used.

The purpose of LM393 is to get the output from the sensor, compare it with a reference voltage and display whether the output is logically high or not. Whereas the purpose of the potentiometer is to set the required threshold value of the gas above which the digital output pin should go high.

The below diagram shows the basic circuit diagram of a gas sensor in a gas sensor module

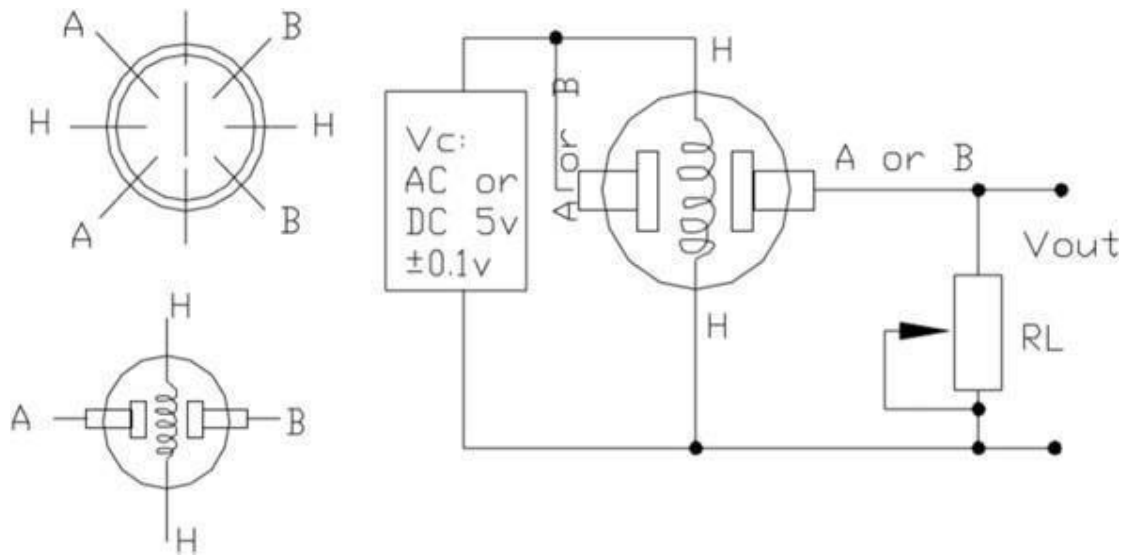


Fig.5.3.5.MQ4 sensor circuit diagram

Here A and B are the input and output terminals (these are reversible - means any of the paired terminals can be used as input or output) and H is the Heater coil terminal. The purpose of the variable resistor is to adjust the output voltage and to maintain high sensitivity.

If no input voltage is applied to the heater coil, then the output current will be very less (which is negligible or approximately 0). When sufficient voltage is applied to the input terminal and heater coil, the sensing layer wakes up and is ready to sense any combustible gases nearby it. Initially let's assume that there is no toxic gas near the sensor, so the resistance of the layer doesn't change and the output current and voltage are also unchanged and are negligible (approximately 0).

Now let's assume that there is some toxic gas nearby. As the heater coil is pre-heated it is now easy to detect any combustible gases. When the sensing layer interacts with the gases, the resistance of the material varies and the current flowing through the circuit also varies. This change in variation can be then observed at the load resistance (R_L).

The value of load resistance (R_L) can be anywhere from $10\text{K}\Omega$ to $47\text{K}\Omega$. The

exact value of the load resistance can be selected by calibrating with the known concentration of the gas. If low load resistance is selected then the circuit has less sensitivity and if high load resistance is selected then the circuit has high sensitivity.

Working

The ability of a Gas sensor to detect gases depends on the chemiresistor to conduct current. The most commonly used chemiresistor is Tin Dioxide (SnO_2) which is an n-type semiconductor that has free electrons (also called as donor). Normally the atmosphere will contain more oxygen than combustible gases. The

attracting the free electrons (black color) inside the SnO_2 and preventing it from having free electrons to conduct current.

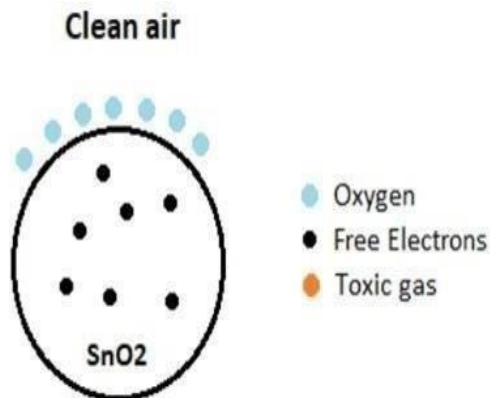


Fig.5.3.6.Clean air

When the sensor is placed in the toxic or combustible gases environment, this reducing gas (orange color) reacts with the adsorbed oxygen particles and breaks the chemical bond between oxygen and free electrons thus releasing the free electrons. As the free electrons are back to its initial position they can now conduct current, this conduction will be proportional to the amount of free electrons available in SnO_2 , if the gas is highly toxic more free electrons will be available.

Types of Gas sensors

Gas sensors are typically classified into various types based on the type of the sensing element it is built with. Below is the classification of the various types of gas sensors based on the sensing element that are generally used in various applications:

Features

- High Sensitivity
- Detection Range: 100 - 10,000 ppm iso-butane Fast Response Time: <10s
- Heater Voltage: 5.0V
- Dimensions: 18mm Diameter, 17mm High excluding pins, Pins - 6mm High

Applications

- ☐ Used in industries to monitor the concentration of the
- ☐ toxic Used in households to detect an emergency incidents.
- ☐ Used at oil rig locations to monitor the concentration of the gases
- ☐ those are Used at hotels to avoid customers from smoking.
- ☐ Used in air quality check at offices.
- ☐ Used in air conditioners to monitor the
- ☐ CO2 I. Used in detecting fire.
- ☐ Used to check concentration of gases
- ☐ in . Breath analyzer.

5.4. LM35 - Precision Centigrade Temperature Sensor

A temperature sensor is a device, usually an RTD (resistance temperature detector) or a thermocouple, that collects the data about temperature from a particular source and converts the data

into understandable form for a device or an observer. Temperature sensors are used

in many applications like HV and AC system environmental controls, food processing units, medical devices, chemical handling and automotive under the hood monitoring and controlling systems, etc.

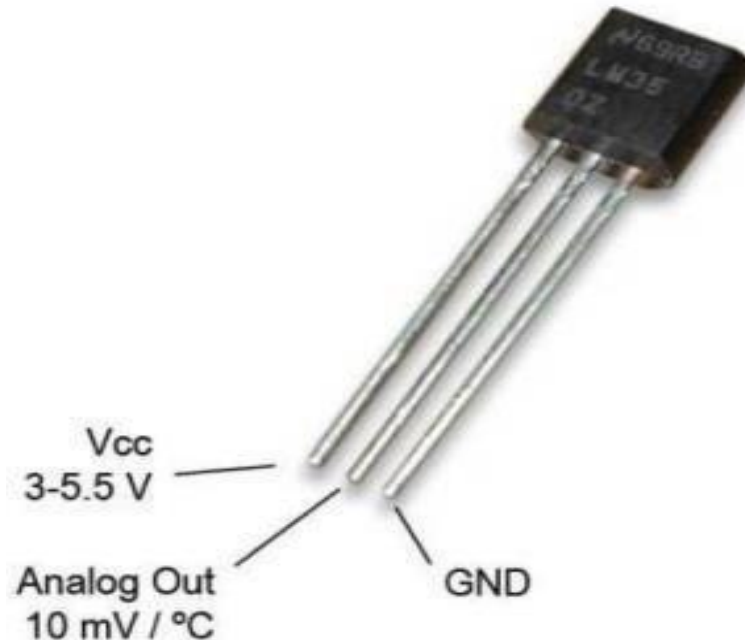


Fig.5.4.1. Temperature sensor

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 **thus** has an

advantage over linear temperature sensors calibrated in ° Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 does not require any external calibration or trimming to provide typical accuracies of

$\pm 1/4^{\circ}\text{C}$ at room temperature and $\pm 3/4^{\circ}\text{C}$ over a full -55 to $+150^{\circ}\text{C}$ temperature range. Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration

make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 μA from its supply, it has very low self-heating, less than 0.1°C in still air. The LM35 is rated to operate over a -55° to $+150^\circ\text{C}$ temperature range, while the LM35C is rated for a -40° to

$+110^\circ\text{C}$ range (-10° with improved accuracy). The LM35 series is available packaged in hermetic TO-46 transistor packages, while the LM35C, LM35CA, and LM35D are also available in the plastic TO-92 transistor package.

Operation of LM35:

- The LM35 can be connected easily in the same way as other integrated circuit temperature sensors. It can be stuck or established to a surface and its temperature will be within around the range of 0.01°C of the surface temperature.
- This presumes that the ambient air temperature is just about the same as the surface temperature; if the air temperature were much higher or lower than the surface temperature, the actual temperature of the LM35 die would be at an intermediate temperature between the surface temperature and the air temperature.

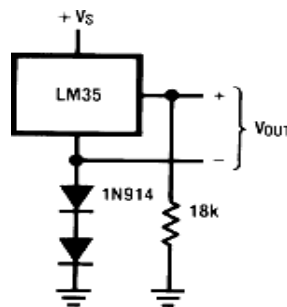


Fig.5.4.2.circuit diagram of LM35 sensor

The temperature sensors have well known applications in environmental and process control and also in test, measurement and communications. A digital temperature is a sensor, which provides 9- bit temperature readings. Digital temperature sensors offer excellent precise accuracy, these are designed to read from 0°C to 70°C and it is possible to achieve $\pm 0.5^\circ\text{C}$ accuracy. These sensor completely aligned with digital temperature readings in degree Celsius

The LM35 comes in many different packages, including the following.

- TO-92 plastic transistor-like package, TO-46 metal can transistor-like package
- How Do You Use An LM35? (Electrical Connections)
- Here is a commonly used circuit. For connections refer to the picture above.
- In this circuit, parameter values commonly used are:
 - $V_c = 4 \text{ to } 30\text{v}$
 - 5v or 12 v are typical values used. $R_a = V_c / 10^{-6}$
 - Actually, it can range from 80 KW to 600 KW , but most just use 80 KW.

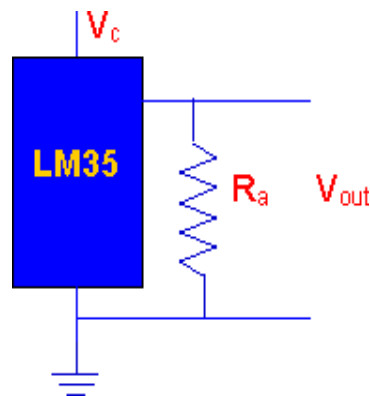
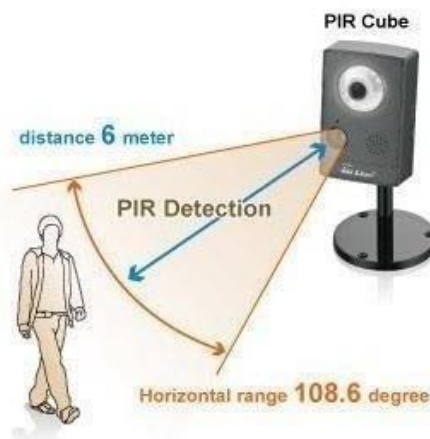


Fig.5.4.3.operation of LM35 sensor



Fig.5.5.1. PIR sensor

IR radiation is the section of the electromagnetic spectrum that has wavelengths smaller than microwaves and longer than the visible light wavelengths. The infrared region is from $0.75\mu\text{m}$ to $1000\mu\text{m}$ and, the IR waves are too small to be seen with the human eyes. If the wavelength region is from $0.75\mu\text{m}$ to $3\mu\text{m}$ – it is called as near infrared; the region from $3\mu\text{m}$ to $6\mu\text{m}$ is called as mid infrared; and, if the region is higher than $6\mu\text{m}$, then it is called as far infrared.



These radiations have been broadly used in various sensors and electronic gadgets. Right from the TV remotes to the complicated devices like night-vision equipment's use IR waves.

PIR Working

The PIR sensors are more complicated than the other sensors as they

consists of two slots. These slots are made of a special material which is sensitive to IR. The Fresnel lens is used to see that the two slots of the PIR can see out past some distance. When the sensor is inactive, then the two slots sense the same amount of IR. The ambient amount radiates from the outdoors, walls or room, etc.

When a human body or any animal passes by, then it intercepts the first slot of the PIR sensor. This causes a positive differential change between the two bisects. When a human body leaves the sensing area, the sensor generates a negative differential change between the two bisects. The infrared sensor itself is housed in a hermetically sealed metal to improve humidity/temperature/noise/immunity. There is a window which is made of typically coated silicon material to protect the sensing element.

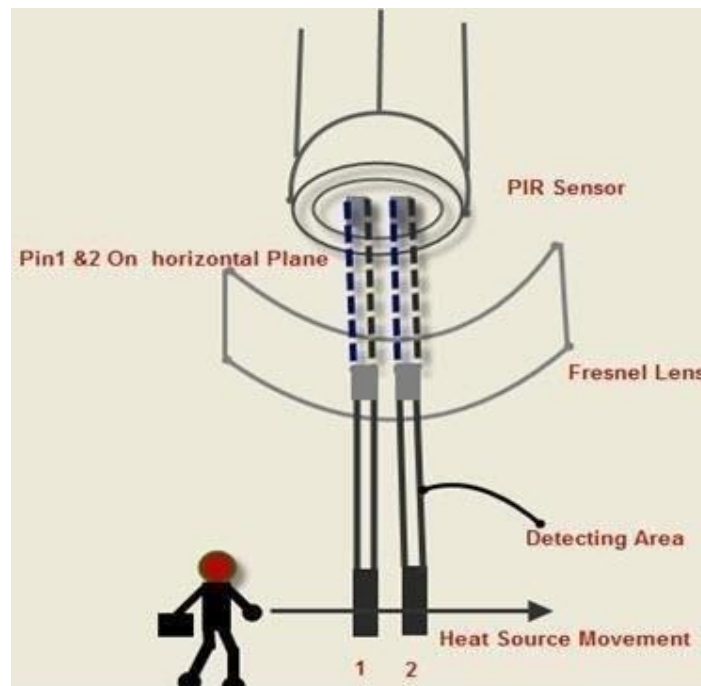


Fig.5.5.2 PIR working

A Motion Detection Circuit Using PIR Sensor

In the above segment, we have learned the pin outs of a PIR sensor; now let's move on to study a simple application of the PIR sensor. The below diagram depicts [a motion detector PIR sensor circuit](#). In the presence of a human IR energy or radiation, the infrared sensor detects the energy and immediately converts it into minute electrical pulses, enough to activate the [transistor BC547](#) into conduction and to make its collector go low.

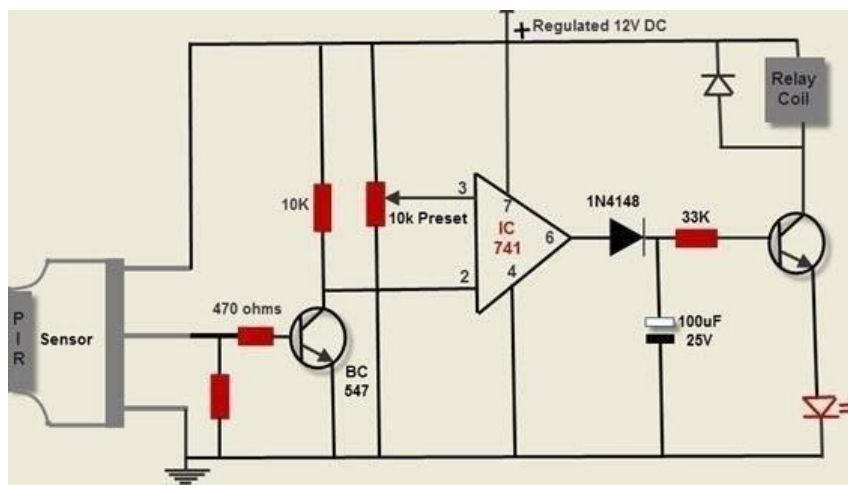


Fig.5.5.2. PIR working

As a comparator, the IC741 is set up –which consists of 8 pins. Wherein the pin3 is allocated as the reference input, while the Pin2 as the sensing input. When the collector terminal of the transistor goes low, then the potential pin2 of the IC becomes lower than the potential pin3. Immediately it makes the output of the IC high, triggering the relay driver consisting of another transistor and relay. The relay triggers and switches on the alarm device, which is connected to the circuit.

The capacitor 100uF/25V makes sure that the relay remains on even after the passive infrared sensor is turned off possibly due the exit of the radiation source. The PIR sensor device must be properly enclosed in a Fresnel lens cover to ensure that its

efficiency is sufficiently enhanced.

Pin configuration

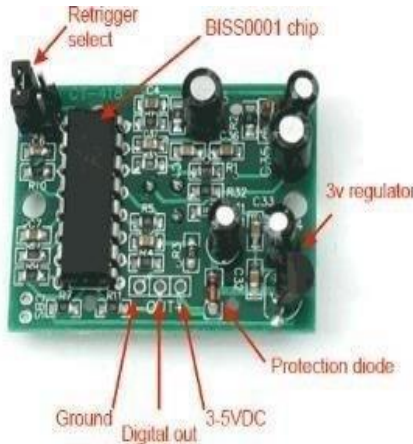


Fig5.5.3.PIR configuration

The Passive infrared sensors consist of three pins as indicated in the diagram shown above.

- Pin1 corresponds to the drain terminal of the device, which should be connected to the positive supply 5V DC.
- Pin2 corresponds to the source terminal of the device, which should be connected to the ground terminal via a 100K or 47K resistor. The Pin2 is the output pin of the sensor, and the detected IR signal is carried forward to an amplifier from the pin 2 of the sensor.
- Pin3 of the sensor is connected to the ground.

Applications of PIR Sensors

- Multi Apartment Complexes Common staircases
- For Basement or Covered Parking Area Shopping Malls
- For garden lights

Features

1. Complete with PIR, Motion Detection.
2. Dual Element Sensor with Low Noise and High voltage
3. Supply Voltage – 5V.
4. Delay Time Adjustable.
5. Standard TTL Output.

5.6. PUMP

This DC 3-6 V Mini Micro Submersible Water Pump is a low cost, small size Submersible Pump Motor which can be operated from a 2.5 ~ 6V power supply. It can take up to 120 liters per hour with a very low current consumption of 220mA. Just connect tube pipe to the motor outlet, submerge it in water and power it.

Make sure that the water level is always higher than the motor. The dry run may damage the motor due to heating and it will also produce noise.



Fig.5.6.Pump

Specification:

- DC 3v to 6v submersible pump
- micro mini submersible water pump 3v to 6v DC water pump for DIY
- DC pump for HOBBY kit

Mini Submersible Pump Motor:

This is a low cost, small size Submersible Pump Motor which can be operated from a 2.5 ~ 6V power supply. It can take up to 120 liters per hour with very low current consumption of 220mA. Just connect tube pipe to the motor outlet, submerge it in water and power it. Make sure that the water level is always higher than the motor. Dry run may damage the motor due to heating and it will also produce noise.

Specifications:

- Operating Voltage : 2.5 ~ 6V Operating Current : 130 ~ 220mA Flow Rate : 80 ~ 120 L/H Maximum Lift : 40 ~ 110 mm
- Continuous Working Life : 500 hours Driving Mode : DC, Magnetic Driving Material : Engineering Plastic
- Outlet Outside Diameter : 7.5 mm Outlet Inside Diameter : 5 mm

5.7 LCD (Liquid Crystal Display)

LCD (Liquid Crystal Display) screen is an electronic display module and find a wide range of applications. A 16×2 LCD display is very basic module and is very commonly used

in various devices and circuits. These modules are preferred over seven segments and other multi segment LEDs. The reasons being: LCDs are economical; easily programmable; have no limitation of displaying special & even custom characters (unlike in seven segments), animations and so on.

A 16×2 LCD means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5×7 pixel matrix. This LCD has two registers, namely, Command and Data.

The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD. Click to learn more about internal structure of a LCD.

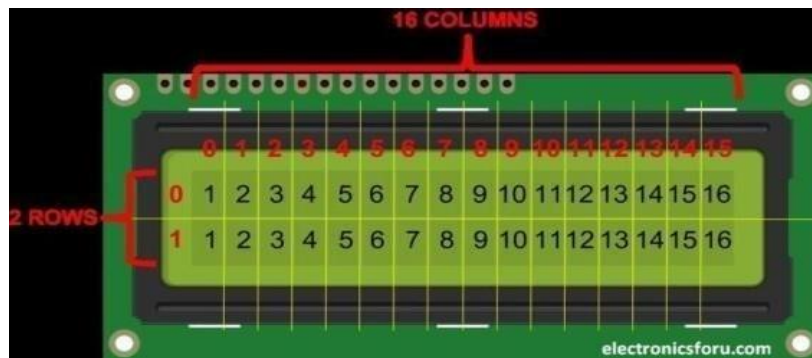


Fig.5.7.liquid crystal display

How LCDs are constructed?

The basic structure of LCD should be controlled by changing the applied . We must use a polarized light.

Liquid crystal should able be to control both of the operation to transmit or can also able to change the polarized light.

As mentioned above that we need to take two polarized glass pieces filter in the making of the liquid crystal. The glass which does not have a polarized film on the surface of it must be rubbed with a special polymer which will create microscopic grooves on the surface of the polarized glass filter. The grooves must be in the same direction of the polarized film. Now we have to add a coating of pneumatic liquid phase crystal on one of the polarized filter of the polarized glass. The microscopic channel cause the first layer molecule to align with filter orientation. When the right angle

appears at the first layer piece, we should add a second piece of glass with the polarized film. The first filter will be naturally polarized as the light strikes it at the starting stage.

Thus the light travels through each layer and guided on the next with the help of molecule. The molecule tends to change its plane of vibration of the light in order to match their angle. When the light reaches to the far end of the liquid crystal substance, it vibrates at the same angle as that of the final layer of the molecule vibrates. The light is allowed to enter into the device only if the second layer of the polarized glass matches with the final layer of the molecule.

Working

The principle behind the LCD's is that when an electrical current is applied to the liquid crystal molecule, the molecule tends to untwist. This causes the angle of light which is passing through the molecule of the polarized glass and also cause a change in the angle of the top polarizing filter. As a result a little light is allowed to pass the polarized glass through a particular area of the LCD. Thus that particular area will become dark compared to other. The LCD works on the principle of blocking light. While constructing the LCD's, a reflected mirror is arranged at the back. An electrode plane is made of indium-tin oxide which is kept on top and a polarized glass with a polarizing film is also added on the bottom of the device. The complete region of the LCD has to be enclosed by a common electrode and above it should be the liquid crystal matter.

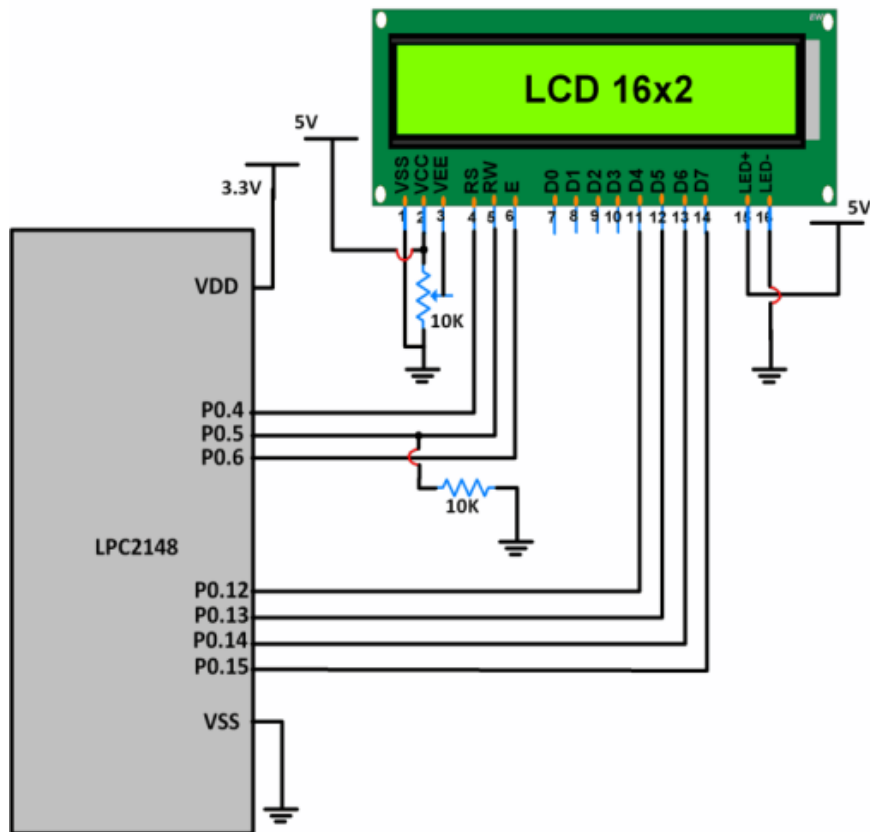


Fig.5.6.2.LCD circuit diagram

Next comes to the second piece of glass with an electrode in the form of the rectangle on the bottom and, on top, another polarizing film. It must be considered that both the pieces are kept at right angles. When there is no current, the light passes through the front of the LCD it will be reflected by the mirror and bounced back. As the electrode is connected to a battery the current from it will cause the liquid crystals between the common-plane electrode and the electrode shaped like a rectangle to untwist. Thus the light is blocked from passing through. That particular rectangular area appears blank.

5.8. Buzzer

A buzzer or beeper is an [audio](#) signalling device, which may be [mechanical](#), [electromechanical](#), or [piezoelectric](#) (piezo for short). Typical uses of buzzers and beepers include [alarm devices](#), [timers](#), and confirmation of user input such as a mouse click or keystroke



Fig.5.8.Buzzer

How to use a Buzzer

A buzzer is a small yet efficient component to add sound features to our project/system. It is very small and compact 2-pin structure hence can be easily used on breadboard, Perf Board and even on PCBs which makes this a widely used component in most electronic applications.

There are two types of buzzers that are commonly available. The one shown here is a simple buzzer which when powered will make a Continuous Beeeeeeppp sound,

the other type is called a readymade buzzer which will look bulkier than this and will produce a Beep. Beep. Beep. Sound due to the internal oscillating circuit present inside it. But, the one shown here is most widely used because it can be customised with help of other circuits to fit easily in our application.

This buzzer can be used by simply powering it using a DC power supply

ranging from 4V to 9V. A simple 9V battery can also be used, but it is recommended to use a regulated +5V or +6V DC supply. The buzzer is normally associated with a switching circuit to turn ON or turn OFF the buzzer at required time and require interval.

Types

Electromechanical

Early devices were based on an electromechanical system identical to an electric bell without the metal gong. Similarly, a relay may be connected to interrupt its own actuating current, causing the contacts to buzz. Often these units were anchored to a wall or ceiling to use it as a sounding board. The word "buzzer" comes from the rasping noise that electromechanical buzzers made.

Mechanical

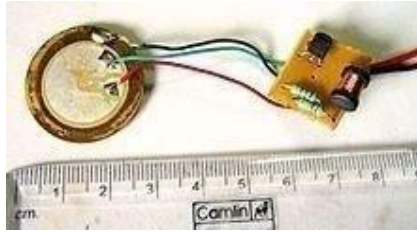
A joy buzzer is an example of a purely mechanical buzzer and they require drivers. Other examples of them are doorbells.

Piezoelectric



Fig.5.8.2.piezoelectric diode

A piezoelectric element may be driven by an oscillating electronic circuit or other audio signal source, driven with a piezoelectric audio amplifier. Sounds commonly used to indicate that a button has been pressed are a click, a ring or a beep.



Interior of a readymade loudspeaker, showing a piezoelectric-disk-beeper (With 3 electrodes ... including 1 feedback-electrode (the central, small electrode joined with red wire in this photo), and an oscillator to self-drive the buzzer.

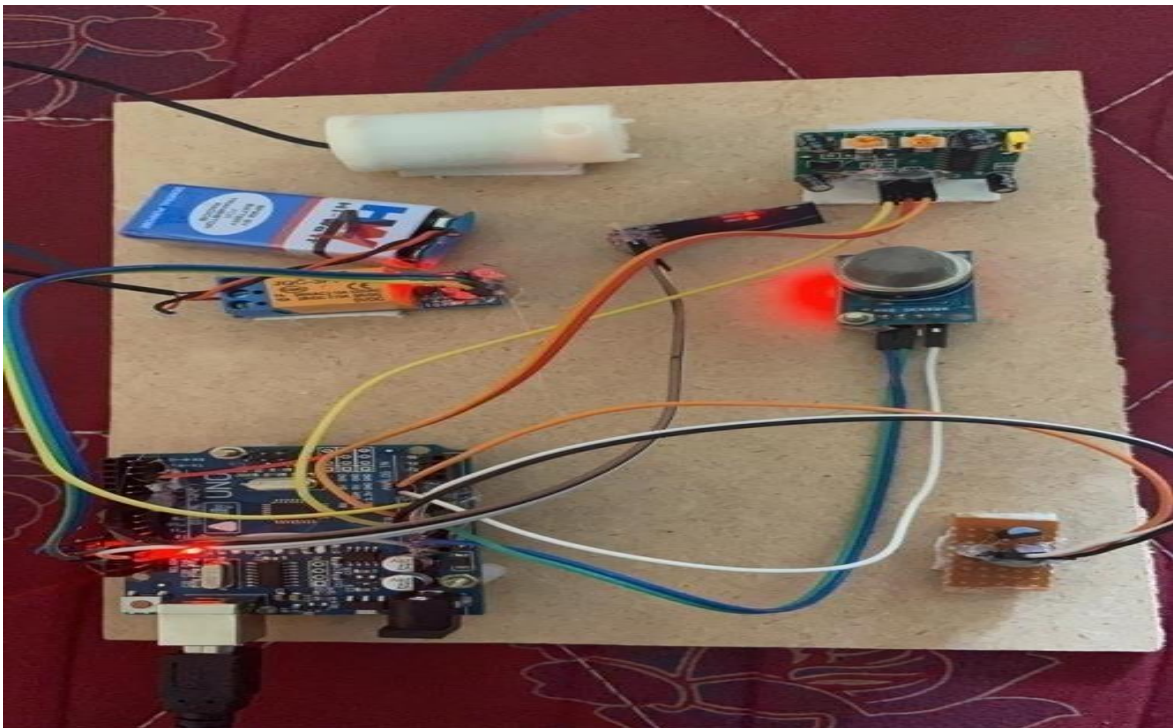
A piezoelectric buzzer/beeper also depends on acoustic cavity resonance or Helmholtz resonance to produce an audible beep

CHAPTER 6

RESULT

The principle of operation is quite simple. various sensors are used to detect the fire using lora. The transmitter section receives the detection of fire and it passes the information to the receiver section. At the same time PIR sensor detects whether any human being is sensed in the forest. by using the pump it can cover some area of the forest through water .lora in the transitter section communicates to the receiver section lora .we can able to figure the inciden where it occurred. by using this type of technique we can able to solve the issue of forest fire happening in different places

Fig.6.1.1.Transmitter section



The Transmitter Section consists of Arduino Board and LoRA, Gas Sensor, Temperature Sensor, Passive Infrared Sensor, Pump. These help to find out whether fire is detected or not. If any fire is detected by using the pump we can stop the fire.

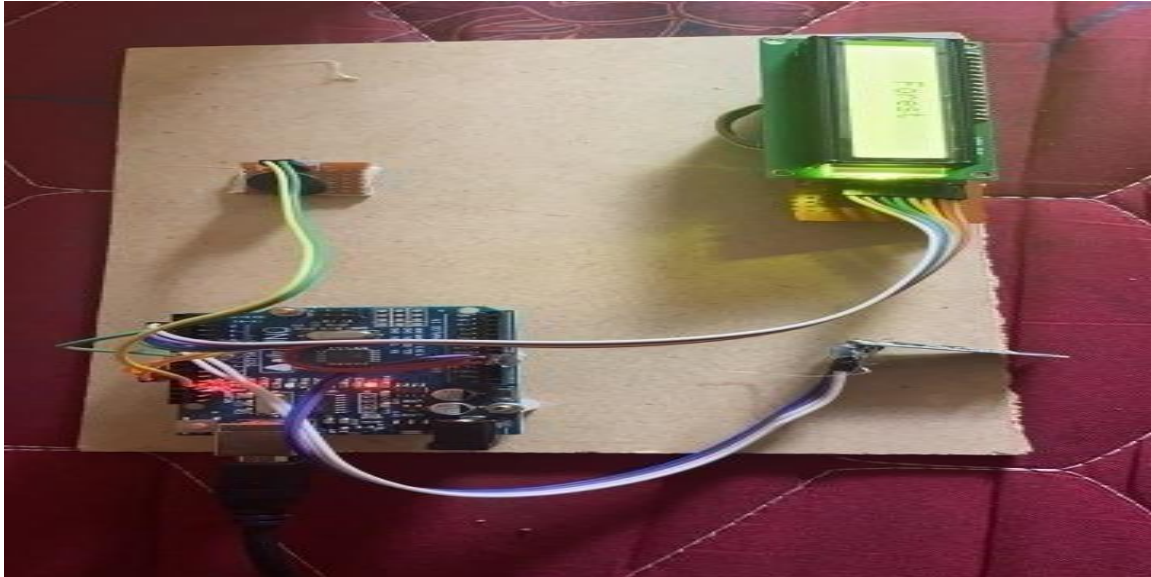


Fig.6.2.Receiver section

The Receiver Section Consists of A LoRA Receiver, Arduino Ide, Buzzer and a alarm. This helps us to prevent the fire this alerts us by bringing the buzzer. With this buzzer we can avoid forest fire.

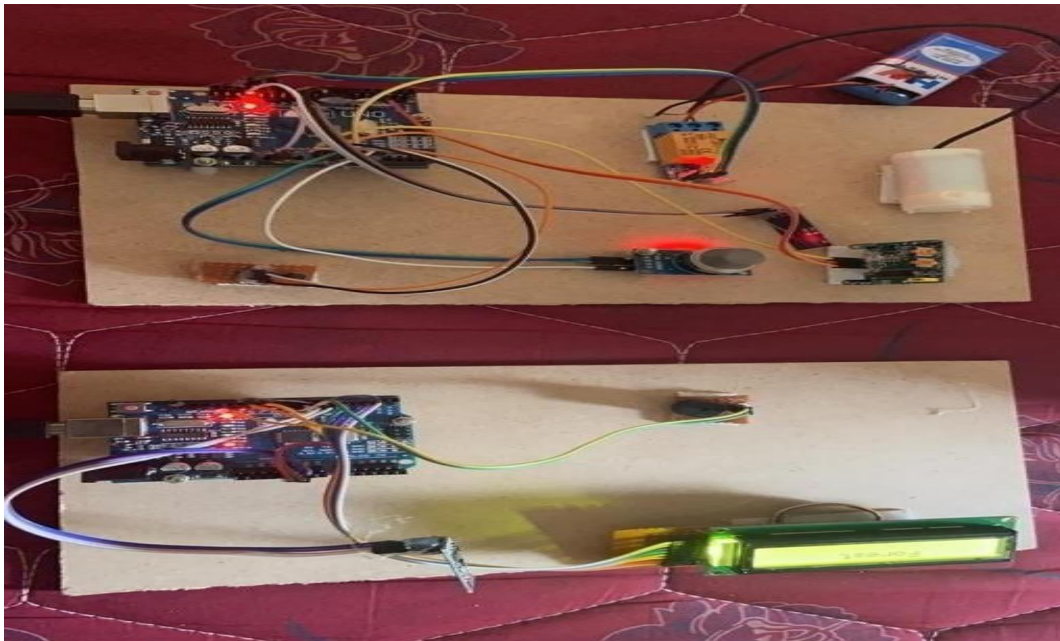


Fig.6.3.Demonstration



Fig.6.4. Forest Normal Condition

The LCD (Liquid Crystal Display) shows us the forest Normal condition Through the Display if the forest in a Fire Free Area Also in a Normal Condition.

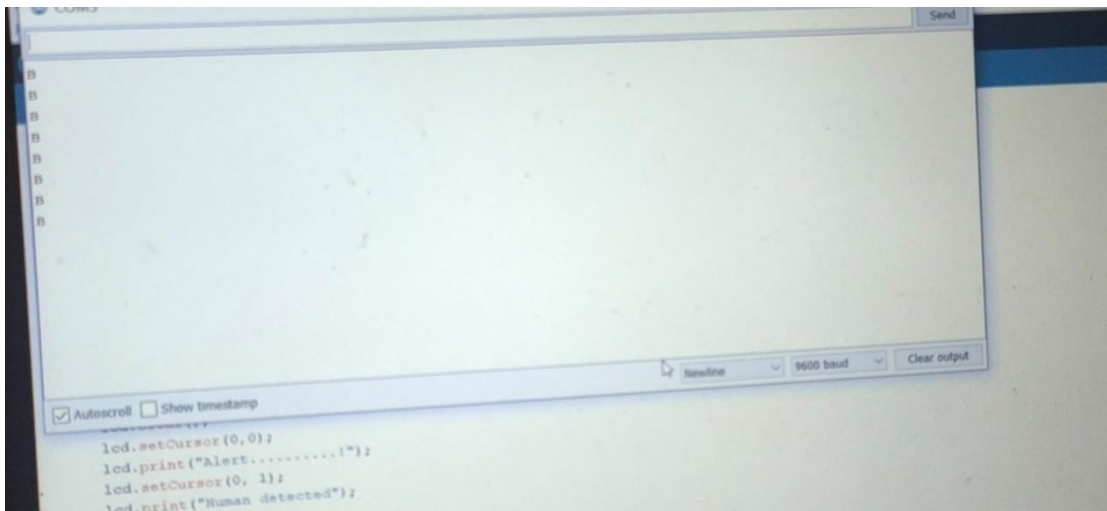


Fig 6.5 Output Display

The above Display Shows us the Output of the entire project which describes about the change in the forest in a alternate way. In the above diagram where the continuous C Represents the forest in a normal condition whether the B represents the forest in a fire Condition.



Fig.6.5.6. Output through LCD Display

The above Figure Represents Whether a Human is Detected under fire Condition if He/She is in Forest under fire Condition this Will Displays a Human is Detected

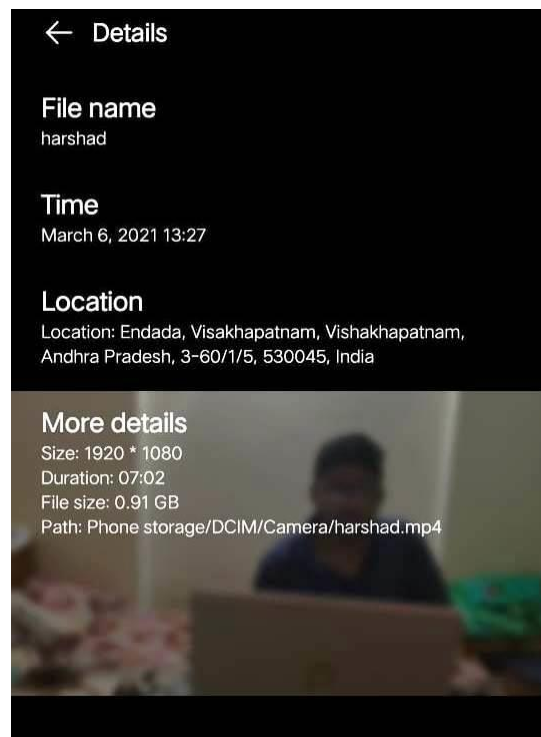


Fig.6.5.7. Geotagged Photo

CHAPTER 7

CONCLUSION

This Project introduced here a fire checking framework that can alarm the client distantly utilizing IOT. This Project has been planned and carried out effectively. The Project expresses that the wood land Fire Detection and Implementation across the world. Fire Accidents are Common Incidents All Around the World Most of the Time This closures of in Human Life Loss. This Accident Can Happen Any Environment Such as Home, Industries, and Cinemas Etc. In the event that Accident happens in an Industrial Environment the Fatality of the Situation is extremely High. This Can Be Solved by Designing an overall System to Monitor any Type Environment Remotely or Locally. This framework as been Tested in Deliberately Created Fire Accident Situation and Response is Very High.

.

CHAPTER 8

REFERENCES

- [1] A. Aksamovic, M. Hebibovic, and D. Boskovic, "Forest fire early detection system design utilising the WSN simulator," in *2017 XXVII International Conference on Information, Communication and Automation Technologies (ICAT)*, 2017, pp. 1-5.
- [2] Y. Liu, Y. Liu, H. Xu, and K. L. Teo, "Forest fire monitoring, detection and decision making systems by wireless sensor network," in *2018 Chinese Control And Decision Conference (CCDC)*, 2018, pp. 5482- 5486.
- [3] D. Pant, S. Verma, and P. Dhuliya, "A study on disaster detection and management using WSN in Himalayan region of Uttarakhand," in *2017 3rd International Conference on Advances in Computing, Communication & Automation (ICACCA) (Fall)*, 2017, pp. 1-6.
- [4] C. Aranzazu-Suescun and M. Cardei, "Distributed algorithms for event reporting in mobile-sink WSN for Internet of Things," *Tsinghua Science and Technology*, vol. 22, no. 4, pp. 413-426, 2017.
- [5] V. P. Illiano and E. C. Lupu, "Detecting Malicious Data Injections in Event Detection Wireless Sensor Networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 3, pp. 496-510, 2015.
- [6] G. Hristov, J. Raychev și D. K. a. P. Zahariev, „Emerging methods for early detection of forestfires using unmanned aerial vehicles and LoRaWAN sensor networks,” 2018.
- [7] L. Yu, N. Wang și X. Meng, „Real-time Forest Fire Detection with Wireless Sensor,” pp1-4, 2005.
- [8] R. Vega-Rodríguez, S. Sendra, J. Lloret și P. R.-D. a. J. L. GarciaNavas, „Low Cost LoRa based Network for Forest Fire Detection,” 2019.
- [9] „Pycom,” [Interactiv]. Available: <https://pycom.io/product/lopy4/>. [Accesat 15 February 2020].
- [10] „Sparkfun,” [Interactiv]. Available: <https://www.sparkfun.com/products/14675>. [Accesat 9 February 2020].
- [11] „Arduino,” 25 February 2020. [Interactiv]. Available: <https://store.arduino.cc/arduino-mega-2560-rev3>.
- [12] „Wikipedia,” 29 February 2020. [Interactiv]. Available: https://en.wikipedia.org/wiki/Flame_detector.
- [13] A. Lavric și A. I. Petrariu, „LoRaWAN Communication Protocol: The New Era of IoT,”

Conference: 2018 International Conference on Development and Application Systems (DAS), 2018, pp 74-77.

[14] A. Lavric, V. Popa, „A LoRaWAN: Long Range Wide Area Networks Study”, 11-th International Conference on Electromechanical and Power Systems, pp. 435-438, 2017.

[15] O. Khutsoane și B. I. a. A. M. Abu-Mahfouz, „IoT devices and applications based onLoRa/LoRaWAN,” IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society, Beijing, pp. 6107-6112, 2017.

[16] B. Sarwar, I. S. Bajwa, N. Jamil și S. R. a. N. Sarwar, „An Intelligent Fire Warning ApplicationUsing IoT and an Adaptive Neuro-Fuzzy Inference System,” Sensors 19(14):3150, doi:10.3390/s19143150, 2019.

REPORT

ORIGINALITY REPORT

8%

SIMILARITY INDEX

22%

INTERNET SOURCES

8%

PUBLICATIONS

7%

STUDENT PAPERS

PRIMARY SOURCES

1

www.cs.unca.edu
Internet Source

8%

Exclude quotes On
Exclude bibliography On

Exclude matches < 7%