VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi-590018

A

PROJECT WORK PHASE II - REPORT

on

“UNDERPASS WATERLOGGING USING IOT AND ML”

Submitted in partial fulfilment of the Bachelor Degree

in

INFORMATION SCIENCE AND ENGINEERING

VIII SEMESTER PROJECT WORK PHASE 2 (18CSP87)

Submitted By

AFREED AHMED	**1HK20IS003**
MOHAMMED ADNAN	**1HK20IS046**
MOHAMMED SHAYAAN A	**1HK20IS051**
MUSKAAN PANDROWALA	**1HK20IS128**

Under the guidance of

Dr. S. Tamilarasan

Professor

Department of Information Science and Engineering

2023-2024

Accredited by NBA

HKBK COLLEGE OF ENGINEERING

22/1, Nagawara, Bengaluru – 5600 045.

E-mail: info@hkbk.edu.in, URL: www.hkbk.edu.in

HKBK COLLEGE OF ENGINEERING

Accredited by NBA

BENGALURU – 560 045

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

VISVESVARAYA TECHNLOGICAL UNIVERSITY

A

PROJECT PHASE 2 - REPORT

on

“UNDERPASS WATERLOGGING USING IOT AND ML”

Submitted in partial fulfilment of the project in

VIII Semester Project Phase 2 (18CSP87)

2023-24

SUBMITTED BY:

AFREED AHMED	**1HK20IS003**
MOHAMMED ADNAN	**1HK20IS046**
MOHAMMED SHAYAAN A	**1HK20IS051**
MUSKAAN PANDROWALA	**1HK20IS128**

ACKNOWLEDGEMENT

We would like to place our regards and acknowledgement to all who helped in making this project possible. There are many people who worked behind the screen to help make it possible the below listed are a few of them.

First of all, we would take this opportunity to express our heartfelt gratitude to **Mr. C.M. Ibrahim,** Chairman, **Mr. C.M. Faiz Mohammed,** Director and **Dr. Mohammed Riyaz Ahmed,** Principal for all the infrastructure provided to complete the project in time.

We are deeply indebted to **Dr. A. Syed Mustafa,** HOD, Information Science and Engineering for the ineffable encouragement he provided in successful completion of the project.

We sincerely thank our guide, **Dr. S. Tamilarasan** for the constant assistance, support, patience, endurance and constructive suggestions for the betterment of the project.

We are extremely thankful to the teaching and non-teaching staff of the **Department of Information Science and Engineering** for their valuable guidance and cooperation throughout our dissertation.

AFREED AHMED	**(1HK20IS003)**
MOHAMMED ADNAN	**(1HK20IS046)**
MOHAMMED SHAYAAN A	**(1HK20IS051)**
MUSKAAN PANDROWALA	**(1HK20IS128)**

ABSTRACT

Recent climate changes have caused significant challenges in disposing enormous amount of water hitting the ground in the form of rain. An underpass getting flooded during intense rain is a catastrophic problem in any growing city. Due to vulnerable infrastructure, more amount of water accumulation can be seen within an underpass. This leads to a dangerous situation for pedestrians as well as vehicle drivers. What makes the condition worse is that Google maps do not render any information pertaining to the level of water in the flooded underpass. This project makes us to develop a frame work that monitors the level of the water in the underpass. This developed frame work shall provide a low cost and environment friendly solution to manage the traffic congestion when an underpass gets clogged due to intense rain. Object detection is a well-known computer technology connected with computer vision and image processing that focuses on detecting objects or its instances of a certain class (such as humans, flowers, animals) in digital images and videos. There are various applications of object detection that have been well researched including face detection, character recognition, and vehicle calculator. Object detection can be used for various purposes including retrieval and surveillance.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER -1
INTRODUCTION

1.1 OVERVIEW

In recent times all the humans and creatures on the earth facing troubles because of growing population, aging infrastructure etc. It’s too important to find the solution for water monitoring & control system due to inadequate drainage mechanisms and also poor design, placing and maintenance is also one of the main reasons for the vulnerable situation of the underpass in the cities and also leads to flooding during heavy rainfall.

In country like India where escaping from the traffic is one of the major issue to the people. The Motorists rack their brains every time the sky opens up in order to figure out is there any underpass that enroute to their destination to escape from the traffic that takes place in the city. If they find any underpass there occurs another tension of they getting stuck in rainwater accumulation in the underpass. And if they choose to wait then they end up being late waiting for hours together till the water drains or till the water is pumped up.

During monsoon season, many underpasses are waterlogged. Most of the cities are facing the problem of underpasses getting flooded. These inoperable underpasses not only cause heavy traffic but also pose grave danger to the pedestrians with dangerous diseases and infections. These factors affect vehicle drivers negatively as they contribute to stress, waste of time and other health issues. Waterlogging in the nearby areas, blockage of storm water, and backflow of the water also leads to spreading waterborne diseases like typhoid, cholera, and various infections such as amoebiasis and toxoplasmosis, glardiasis, etc.

Internet Of Things has become powerful technology that could replace humans based strains. IoT is also referred to as the Internet of everything. It consists of all web-enabled devices that collect the information using embedded sensors and send and process the data using processors and communication hardware for further visualization.

The process of object detection analysis is to determine the number, location, size, position of the objects in the input image. Object detection is the basic concept for tracking and recognition of objects, which affects the efficiency and accuracy of object recognition.

Underpass waterlogging presents a significant challenge in urban environments, often leading to traffic congestion, property damage, and safety risks for commuters. Traditional approaches to managing waterlogging rely on reactive measures, which can be insufficient in preventing or mitigating the impacts of flooding. By integrating IoT and ML technologies, a more proactive and effective solution can be developed. IoT sensors placed in underpasses can continuously monitor water levels, rainfall intensity, and other relevant environmental factors in real-time. This data is then transmitted to a central system for processing. ML algorithms can analyze this data to detect patterns and trends, enabling the prediction of potential waterlogging events before they occur.

Early detection allows authorities to take timely actions, such as deploying pumps, diverting traffic, or issuing warnings, to mitigate the impact of waterlogging. Furthermore, the system can continuously learn and improve through feedback loops, enhancing its predictive capabilities over time.

Additionally, integrating this system with existing urban infrastructure and emergency response systems can improve coordination and efficiency in managing waterlogging events. Overall, the integration of IoT and ML technologies offers a proactive approach to underpass waterlogging, improving urban resilience and reducing the risks associated with flooding in urban areas.

1.2 OBJECTIVES

1. Real-time Monitoring: Implement IoT sensors in underpasses to provide real-time data on water levels, rainfall intensity, and environmental conditions.

2. Predictive Analysis: Utilize ML algorithms to analyze data from IoT sensors to predict waterlogging events before they occur.

3. Early Warning System: Develop an early warning system to alert authorities and residents about potential waterlogging, enabling timely preventive actions.

4. Integration with Urban Infrastructure: Ensure seamless integration with existing urban

infrastructure and emergency response systems to enhance coordination and effectiveness in managing waterlogging events.

5. User-friendly Interface: Provide a user-friendly interface, such as a web dashboard or mobile app, for authorities and residents to monitor real-time data, receive alerts, and access historical data and analysis reports.

Creating a project to address underpass waterlogging using IoT (Internet of Things) and ML (Machine Learning) involves a multi-faceted approach that combines hardware, software, and data analytics. Below is a suggested scope for such a project:

Problem Definition:

- Clearly define the problem of waterlogging in underpasses.
- Identify specific locations or underpasses where waterlogging is a recurring issue.
- Analyze the causes of waterlogging, such as heavy rainfall, drainage issues, or urban development.

Data Collection:

- Install IoT devices (sensors) at strategic points within the underpass to collect real-time data.
- Gather data on rainfall, water levels, drainage flow rates, and other relevant environmental parameters.
- Utilize existing weather APIs or set up weather stations for accurate meteorological data.

Data Integration and Storage:

- Develop a data storage and management system to handle the vast amount of data collected by sensors.
- Integrate data from various sources, including IoT devices, weather stations, and municipal database

Predictive Modelling with ML:

- Train machine learning models to predict the likelihood of waterlogging based on historical data.
- Consider using algorithms for time-series analysis, regression, and classification.
- Incorporate weather forecasts to enhance prediction accuracy.

Early Warning System:

- Implement an early warning system that alerts authorities and the public when there is a high probability of waterlogging.
- Design user-friendly interfaces for mobile apps or web platforms to disseminate warnings.

Control Mechanisms:

- Develop automated control mechanisms, such as smart pumps or drainage systems, to mitigate waterlogging.

1.3 MOTIVATION ABSTRACT

Urbanization has led to the proliferation of underpasses as vital components of modern transportation infrastructure. However, the susceptibility of these underpasses to waterlogging poses a significant challenge, disrupting daily commutes, jeopardizing public safety, and causing substantial economic losses. This research presents a novel approach to address underpass waterlogging by integrating cutting-edge technologies—Internet of Things (IoT) and Machine Learning (ML).

The motivation behind this project lies in the dire need to develop a proactive and intelligent system capable of not only monitoring environmental conditions in real-time but also predicting and mitigating waterlogging incidents. The deployment of IoT sensors strategically placed within underpasses enables the collection of comprehensive data on rainfall, water levels, and drainage patterns. This rich dataset serves as the foundation for the implementation of sophisticated ML algorithms.

Our proposed system leverages ML models to analyze historical data, creating predictive models that anticipate waterlogging occurrences with a high degree of accuracy. These models are dynamic, adapting to changing weather patterns and evolving urban landscapes. Through this predictive capability, our system establishes an early warning mechanism, alerting relevant authorities and the public well in advance, allowing for proactive measures to be taken.

In conclusion, this research aims to revolutionize underpass waterlogging mitigation by fusing IoT and ML technologies. By creating a dynamic and responsive system, we aspire to enhance urban resilience, improve public safety, and contribute to sustainable urban development.

1.4 PROBLEM STATEMENT:

During monsoon season, many underpasses are waterlogged. Most of the cities are facing the problem of underpasses getting flooded. These inoperable underpasses cause heavy traffic and pose a grave danger to pedestrians with dangerous diseases and infections. These factors negatively affect vehicle drivers as they contribute to stress, waste of time, and other health issues.

Urban areas often face challenges with underpass waterlogging during heavy rainfall, leading to traffic disruptions and infrastructure damage. The system incorporates IoT sensors strategically placed in underpasses to continuously monitor water levels. These sensors collect real-time data, creating a dynamic dataset that reflects the changing conditions during rain events. The gathered information is then fed into a sophisticated ML model. The ML model employs predictive analytics, analyzing both current sensor data and historical patterns of waterlogging incidents.

By leveraging this analytical capability, the system can forecast potential waterlogging scenarios with a high degree of accuracy. This proactive approach enables timely intervention to prevent or minimize the impact of waterlogging.

1.5 EXISTING SYSTEM

One example of an existing system for underpass waterlogging using IoT is the Smart Water Management System developed by Tata Consultancy Services (TCS). This system utilizes IoT sensors to monitor water levels in underpasses and other areas prone to flooding. The sensors collect real-time data on water levels and transmit it to a central server using wireless communication. The data is then analyzed using machine learning algorithms to predict potential waterlogging events. Alerts are generated when abnormal water levels are detected, allowing authorities to take timely action. Additionally, the system can be integrated with pumps or other infrastructure to mitigate the effects of waterlogging. Overall, the system helps in improving the efficiency of water management and reducing the impact of floods in urban areas.

Another example of an existing system for underpass waterlogging using IoT is the Flood Mapp system developed by researchers at the University of Technology Sydney (UTS). Flood Mapp uses a network of IoT sensors installed in underpasses and flood-prone areas to monitor water levels and other relevant environmental parameters. The sensors are connected to a cloud-based platform that collects and analyzes the data in real-time. Machine learning algorithms are used to predict the likelihood of waterlogging based on the sensor data and weather forecasts. The system provides early warnings to authorities and residents, allowing them to take preventive measures and minimize the impact of flooding. Flood Mapp also incorporates a mobile app interface for users to receive alerts and updates on the flooding situation in their area.

1.6 PROPOSED SYSTEM

A proposed system for underpass waterlogging using IoT and ML would integrate various components to effectively monitor, predict, and mitigate waterlogging events. The system would begin with the deployment of IoT sensors strategically placed in underpasses to measure water levels, rainfall intensity, and other relevant environmental data. These sensors would transmit real-time data to a central server or cloud platform for storage and analysis.

Data collected from the sensors would be processed using ML algorithms to identify patterns and trends indicative of potential waterlogging. These algorithms would continuously learn and improve through feedback loops, enhancing their predictive capabilities over time. When a potential waterlogging event is detected, the system would generate alerts to notify relevant authorities and stakeholders.

To mitigate the impact of waterlogging, the system would facilitate timely responses, such as deploying pumps, redirecting traffic, or issuing warnings. Integration with existing urban infrastructure and emergency response systems would ensure coordinated efforts in managing waterlogging events.

Additionally, the system would include a user-friendly interface, such as a web dashboard or mobile app, allowing users to monitor real-time data, receive alerts, and access historical data and analysis reports. Overall, the proposed system aims to enhance the efficiency of water management in underpasses, reducing the risk of flooding and improving urban resilience to extreme weather events.

CHAPTER-2

LITERATURE SURVEY

2.1 SURVEY OF THE DOMAIN: IOT AND ML

2.1.1 An efficient way to vehicle movement monitor in flooded underpass using IoT

Humans are now suffering issues because of the rapid growth of the human population. Apart from that, there is a deficiency of appropriate infrastructure in development projects, such as roads and underpasses, which results in a range of challenges, such as water logging and floods. In the event of heavy rain, underpasses and roads get immediately swamped with water, resulting in traffic gridlock and car immobilization. The data collected by our water level sensor, which we are using to monitor water level, is controlled, and uploaded by an Arduino board equipped with an ESP8266 module. A regular stream of data will be delivered to the cloud, and the vehicle movement will be rerouted in the most suitable way. Another feature is the installation of an LCD display near the underpass, which continuously updates the water level, allowing anyone in the neighborhood to determine if it is safe for their car to pass through or not. A DC motor pump is then utilized to pump out the water when the level of the water exceeds a predetermined threshold value.

Published in: 2022 International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES)

Date of Conference: 20-21 May 2022

Date Added to IEEE *Xplore*: 11 August 2022

ISBN Information:

INSPEC Accession Number: 21990755

DOI: 10.1109/CISES54857.2022.9844335

Publisher: IEEE

Conference Location: Greater Noida, India

Authors

Suresh N

Department of Electronics and Communication Engineering, KL. University, India

Bala Vamsi T

Department of Electronics and Communication Engineering, KL. University, India

Aravind Y

Department of Electronics and Communication Engineering, KL. University, India

B. Veena Vani

Department of Electrical Engineering, National Institute of Technology, Surathkal, India

B. Naresh Kumar Reddy

School of Electronic Systems & Automation, Digital University Kerala (IIITM Kerala), India

2.1.2 IoT and ML based Flood Alert and Human Detection System

Floods are one of the world's most frequently occurring and disastrous natural calamity. A severe flood might wipe out the community, putting many lives in jeopardy. The government invests a significant amount of money and time in redeveloping the devastated areas. To mitigate flood threats, a flood monitoring system must be developed. It is critical to provide immediate input on the arrival of a flood in order to inform residents to take immediate action, such as evacuating to a safer location. As a remedy, this study suggests a method that can not only detect but also quantify the rate at which the water level rises, alerting residents and informing the concerned authorities of any human life present. This project's methodology is based on this paradigm. The Arduino collects data from the water sensor and sends it to the GSM module, which sends an alert via SMS and analyses any human presence using a machine learning model. The study will demonstrate how the Arduino will be combined with the smartphone to provide an alarm and how the machine learning model will function. In order to confirm that the system can give accurate and dependable data, it is tested in two separate contexts. The first context is to test that the model is able to detect the early signs of a flood and warns accordingly. The second context is to test that the model is able to identify humans in a flood affected area through machine learning.

Published in: 2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT)

Date of Conference: 20-22 January 2022

Date Added to IEEE *Xplore*: 25 February 2022

ISBN Information:

INSPEC Accession Number: 21707371

DOI: 10.1109/ICSSIT53264.2022.9716441

Publisher: IEEE

Conference Location: Tirunelveli, India

Authors

Subhashree Rath

Department of Computer Science, New Horizon College of Engineering, Bengaluru, India

Vaishali M Deshmukh

Department of Computer Science, New Horizon College of Engineering, Bengaluru, India

Rafeeq Manzoor

Department of Computer Science, New Horizon College of Engineering, Bengaluru, India

Sourav Singh

Department of Computer Science, New Horizon College of Engineering, Bengaluru, India

S Joydeep Singh

Department of Computer Science, New Horizon College of Engineering, Bengaluru, India

2.1.3 A framework for mobile application of flood alert monitoring system for vehicle users using Arduino device

This paper proposes the development of an Android application for all the vehicle owners that uses their smartphones while travelling and getting flood reports on the route that they are going to pass through. Working with an Arduino prototype for detecting the flood, the application alerts the user if their vehicle can either pass through the flood safely, proceed with precaution or shouldn't pass the route at all because of the flood. The application will be using the smartphone's GPS to determine the user's location. Whenever the vehicle owner enters the range of the prototype based on the driver's location, it will notify the vehicle owner through a voice about the flood condition on the area near the user whether it is passable or not. This also helps them not to get stuck in a flooded area or worst their vehicle engines might get damaged because of the flood that got inside of their engines since they didn't have a knowledge of how high the flood and tried to pass through it. The main goal of the study is to implement a mobile application that will help vehicle drivers to monitor the flood in the streets and identify if they can pass through a flooded motorway.

Published in: 2017IEEE 9th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)

Date of Conference: 01-03 December 2017

Date Added to IEEE *Xplore*: 25 January 2018

ISBN Information:

INSPEC Accession Number: 17524913

DOI: 10.1109/HNICEM.2017.8269431

Publisher: IEEE

Conference Location: Manila, Philippines

Authors

Gervy Andrew Amagsila

Philippines College of Computer Studies, National University

Mark Emmanuel Cabuhat

Philippines College of Computer Studies, National University

Jhon Emil Tigbayan

Philippines College of Computer Studies, National University

Edmhar Uy

Philippines College of Computer Studies, National University

Eliseo Ramirez

Philippines College of Computer Studies, National University

2.1 SURVEY OF THE PROJECT: UNDERPASS WATER LOGGING USING IOT AND ML.

2.2.1 Advanced IOT Solutions to Monitor Vehicular Movement in Flooded Underpass

In recent years, humans and other living things face problems due to rapid population growth. Also, there is no proper infrastructure, which leads to various other problems like underpass water logging. In case of intense rain, underpasses are quickly flooded with water, which can cause serious situations for passing vehicles. Also, a substantial vehicle traffic problem is experienced in underpasses. This inadequate drainage mechanism and poor design to filter the rainwater to the ground. This paper introduces an IOT based solution to monitor and control heavy traffic in the underpass. The main objective is to develop a framework that monitors the water level in the underpass by activating the significant signs.

International Journal of Electronics and Communication Engineering

© 2021 by SSRG - IJECE Journal

Volume 8 Issue 1

Year of Publication : 2021

2.2.2 Intelligent Water Level Monitoring System Using IoT

Ever since the evolution of earth, water management has become one of the crucial factors for human survival. In evolving years, significant efforts have been put to come up with solutions based on IoT technology for areas such as water level measurement. The main issue that is being addressed in this work is about developing an efficient level sensor based water monitoring system that monitors the water level in the domestic areas i.e. inside homes. The proposed system will detect the water level through depth sensors and verifies the threshold value that is set i.e. (>20 cm). If the value is less than threshold value, no action needs to be taken and if the value is beyond threshold value, the Arduino UNO alerts the user through call by using GSM module. Simultaneously, with the increase in the water level, the proposed system evacuates the water to a storage tank through submersible water pump. This extracted water can be used for some other purposes like watering plants, domestic usage etc without wasting the water.

Published in: 2020 IEEE International Symposium on Sustainable Energy, Signal Processing and Cyber Security (iSSSC)

Date of Conference: 16-17 December 2020

Date Added to IEEE *Xplore*: 25 February 2021 **ISBN Information:**

INSPEC Accession Number: 20506327

DOI: 10.1109/iSSSC50941.2020.9358827

Publisher: IEEE

Conference Location: Gunupur Odisha, India

Authors

Sandhya.A. Kulkarni

Department of Computer Science and Engineering, K S School of Engineering and Management, Bengaluru, India

Vishal D Raikar

Department of Computer Science and Engineering, K S School of Engineering and Management, Bengaluru, India

B K Rahul

Department of Computer Science and Engineering, K S School of Engineering and Management, Bengaluru, India

L V Rakshitha

Department of Computer Science and Engineering, K S School of Engineering and Management, Bengaluru, India

2.2.3 Early Detection of Flood Monitoring and Alerting System to Save Human Lives

Climatic changes have an adverse effect on certain factors of nature like temperature, humidity, rainfall etc. Also, because of convective activity, there is more rainfall in certain areas during monsoon seasons which might flood the areas near rivers or dam. Another reason for flood is cyclone. While cyclones appear to be natural calamities, they are in essence required by nature to maintain the balance of temperature. It is when the oceans become warm that cyclones form to cool them down. Our activities which are distorting the balance of nature, we can expect more inclement weather, stronger storms to form. Climate change is real and must seriously paid attention to. In the end, it is all about balance and our relationship with nature. This paper deals with proactive measures taken to not be affected by flood and hence give precaution to common man. Updates about the current weather and different parameters that are related to water is continuously monitored. This paper also incorporates communication of the information collected effectively by using ESP32 microcontroller, GSM and IoT (Blynk).

Published in: 2020 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)

Date of Conference: 12-13 November 2020

Date Added to IEEE *Xplore*: 13 January 2021

ISBN Information:

INSPEC Accession Number: 20287525

DOI: 10.1109/RTEICT49044.2020.9315556

Publisher: IEEE

Conference Location: Bangalore, India

Authors

Shiva shankar

Department of Electronics & Communication Engineering, Sri Venkateshwara College of Engineering, Vidyanagar, Bengaluru, India

J J Jijesh

Department of Electronics & Communication Engineering, Sri Venkateshwara College of Engineering, Vidyanagar, Bengaluru, India

Dileep Reddy Bolla

Department of Electronics & Communication Engineering, Sri Venkateshwara College of Engineering, Vidyanagar, Bengaluru, India

Mahaveer Penna

Department of Electronics & Communication Engineering, Sri Venkateshwara College of Engineering, Vidyanagar, Bengaluru, India
P V Sruthi
Department of Electronics & Communication Engineering, Sri Venkateshwara College of Engineering, Vidyanagar, Bengaluru, India.

2.2.4 A Sensor-Based Smart Urban Flood Warning and Management System

In recent years, the rapid global climate change has resulted in frequent urban flooding disasters. The phenomenon of blocked drain covers in urban areas directly affects the passage of vehicles and pedestrians, causing inconvenience to citizens' travel. In severe cases, it can even submerge vehicles, leading to casualties and posing a serious threat to public safety and property. To address this issue, we propose a smart urban flood warning and management system. The system installs terminal devices on rainwater grates along the roadsides. When the water level on the road reaches a certain depth, pressure sensors are used to acquire the water level information, which is then uploaded to the server. The system provides realtime and accurate feedback on urban flooding conditions, demonstrating broad prospects for the prediction and control of urban flooding disasters.

Published in: 2023 4th International Conference on Information Science, Parallel and Distributed Systems (ISPDS)
Date of Conference: 14-16 July 2023
Date Added to IEEE *Xplore*: 05 September 2023
ISBN Information:
INSPEC Accession Number: 23691673
DOI: 10.1109/ISPDS58840.2023.10235462
Publisher: IEEE
Conference Location: Guangzhou, China

Authors

Sunyang Song
College of Engineering, China Agricultural University, Beijing, China
Xue Deng
College of Engineering, China Agricultural University, Beijing, China
Chao Gao
College of Engineering, China Agricultural University, Beijing, China

CHAPTER-3
REQUIREMENTS

3.1 HARDWARE REQUIREMENTS

- Arduino
- Lcd Display
- Rain Sensor.
- Water Level Sensor.
- DC Motor/Servo motor
- Relay
- Water pump
- Node Micro control unit.
- Power Supply.

3.2 SOFTWARE REQUIREMENTS

- Arduino Ide
- Embedded C.
- Python
- Open cv.

3.1.1 Arduino Uno:

Arduino/ GenuinoUno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started. You can tinker with your UNO without worrying too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again. "Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino , now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform; for an extensive list of current, past or outdated boards see the Arduino index of boards.[1]

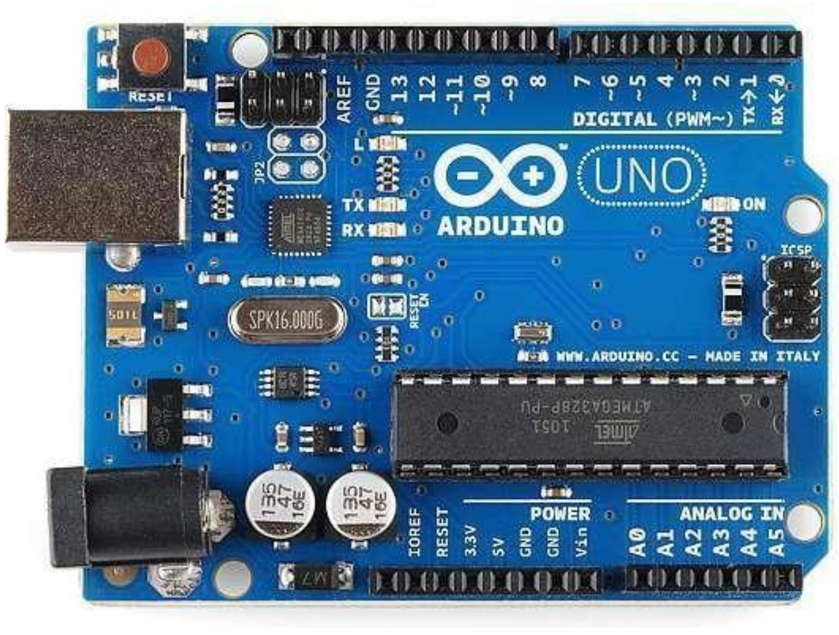


Fig 3.1.1: Arduino

Arduino programming:

Microcontroller	ATmega328P
Operating Voltage	5v
Input voltage	7-12v
Input voltage limit	6-20v
Digital I/O Pins	6
Analogue input Pins	6
DC current perI/O pins	20 Ma
DC current for 3.3v Pin	50 Ma
Flash Memory	Of which o.5KB is used
SRAM	2 KB
EEPROM	1KB
Clock Speed	16MHz
Length	68.6mm
Width	53.4nm
Weight	25g

Table 3.1.1: Arduino programming

The Arduino /GenuinoUno can be programmed with the (Arduino Software (IDE)).Select "Arduino/GenuinoUno from the Tools>Board menu (according to the microcontroller on your board). The ATmega328 on the Arduino/GenuinoUno comes preprogrammed with a boot loader that allows us to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference ,C header files). We can also bypass the boot loader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header using Arduino ISP or similar. The ATmega16U2/8U2 is loaded with a DFU boot loader, which can be activated by:

- On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then reusing the8U2.
- On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode.[1]

3.1.1.1 Warnings:

The Arduino/GenuinoUno has a resettable poly fuse that protects your computer's USB ports from shorts and over current. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.[1]

Differences with other boards:

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.[1]

3.1.1.2 Power:

The Arduino/GenuinoUno board can be powered via the USB connection or with an external power supply. The power source is selected automatically.
External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the GND and VIN pin headers of the POWER connector.
The board can operate on an external supply from 6 to 20 volts. If supplied with less than7V, however, the 5V pin may supply less than five volts and the board may become unstable. If using more than 12V, the voltage regulator may over heat and damage the board. The recommended range is 7 to 12volts.
The power pins are as follows:

- VIN. The input voltage to the Arduino/ Genuino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). One can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- 5V.This pin outputs a regulated 5Vfrom the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.

- 3V3. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- GND. Ground pins.
- IOREF. This pin on the Arduino/Genuinoboard provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs to work with the 5V or 3.3V.[1]

3.1.1.3 Memory:

The ATmega328 has 32 KB (with 0.5 KB occupied by the boot loader). It also has2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

3.1.2 Input &Output:

Each of the 14 digital pins on the Uno can be used as an input or output, using pin Mode(), digital write (), and digital read () functions. They operate at 5 volts. Each pin can provide or receive 20 mA as recommended operating condition and has an internal pull- up resistor (disconnected by default) of 20-50k ohm.A maximum of 40mAis the value that must not be exceeded on any I/O pin to avoid permanent damage to the microcontroller.

In addition, some pins have specialized functions:

- Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attach interrupt () function for details.
- PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analog write () function.
- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.
- LED: 13. There is a built-in LED driven by digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

- TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to5 volts, though is it possible to change the upper end of their range using the AREF pin and the analog reference ()function.

There are a couple of other pins on the board:

- AREF. Reference voltage for the analog inputs. Used with analog Reference().
- Reset. Bring this line LOW to reset the microcontroller. Typically usedto add a reset button to shields which block the one on the board.[1]

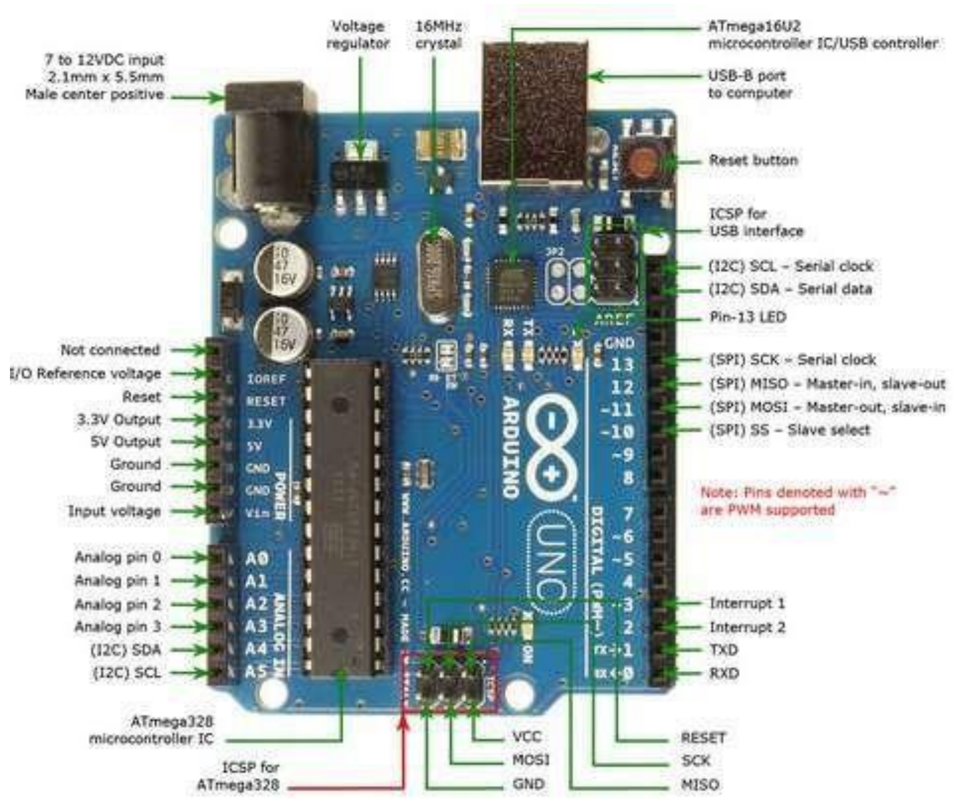


Fig 3.1.2: Pin Specification

3.1.2.1 Communication:

Arduino/GenuinoUno has a number of facilities for communicating with a computer, another Arduino/Genuinoboard,orothermicrocontrollers.TheATmega328provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The 16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, an inffile is required. The Arduino Software (IDE) includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and1).
A Software serial library allows serial communication on any of the Uno's digital pins. The ATmega328alsosupports I2C(TWI) and SPI communication. The Arduino Software (IDE) includes a Wire library to simplify use of the I2C bus; seethe documentation for details. For SPI communication, use the SPI library.[1]

3.1.2.2 Automatic (Software) Reset:

Rather than requiring a physical press of the reset button before an upload, the Arduino/GenuinoUno board is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nano farad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino Software (IDE) uses this capability to allow you to upload code by simply pressing the upload button in the interface toolbar. This means that the boot loader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload. This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the boot loader is running on the Uno.While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data. The Uno board contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line.[1]

3.1.3 Relay

A relay is an electrically operated switch. Current flowing through the coil of the relay creates a magnetic field which attracts a lever and changes the switch contacts. The coil current can be on or off so relays have two switch positions and they are double throw (changeover) switches. The relay’s switch connections are usually labeled COM(POLE), NC and NO. In order to trigger the laser we use driver relay.

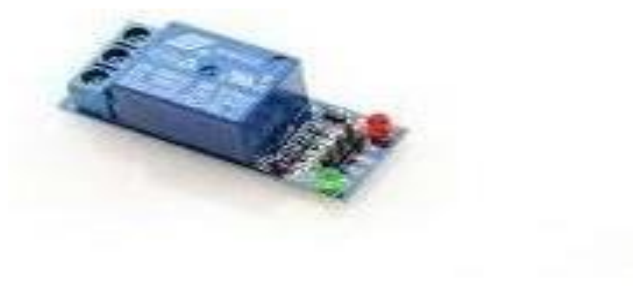


Fig 3.1.3: Relay

Many relays use an electromagnet to mechanically operate a switch, but other operating principles are also used, such as solid-state relays. Relays are used where it is necessary to control a circuit by a separate low-power signal, or where several circuits must be controlled by one signal. The first relays were used in long distance telegraph circuits as amplifiers: they repeated the signal coming in from one circuit and re-transmitted it on another circuit. Relays were used extensively in telephone exchanges and early computers to perform logical operations.

3.1.2 Water Level Sensor:

The HC-SR04 ultrasonic sensor uses SONAR to determine the distance of an object just like the bats do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package from 2 cm to 400 cm or 1" to 13 feet.

The operation is not affected by sunlight or black material, although acoustically, soft materials like cloth can be difficult to detect. It comes complete with ultrasonic transmitter and receiver module.

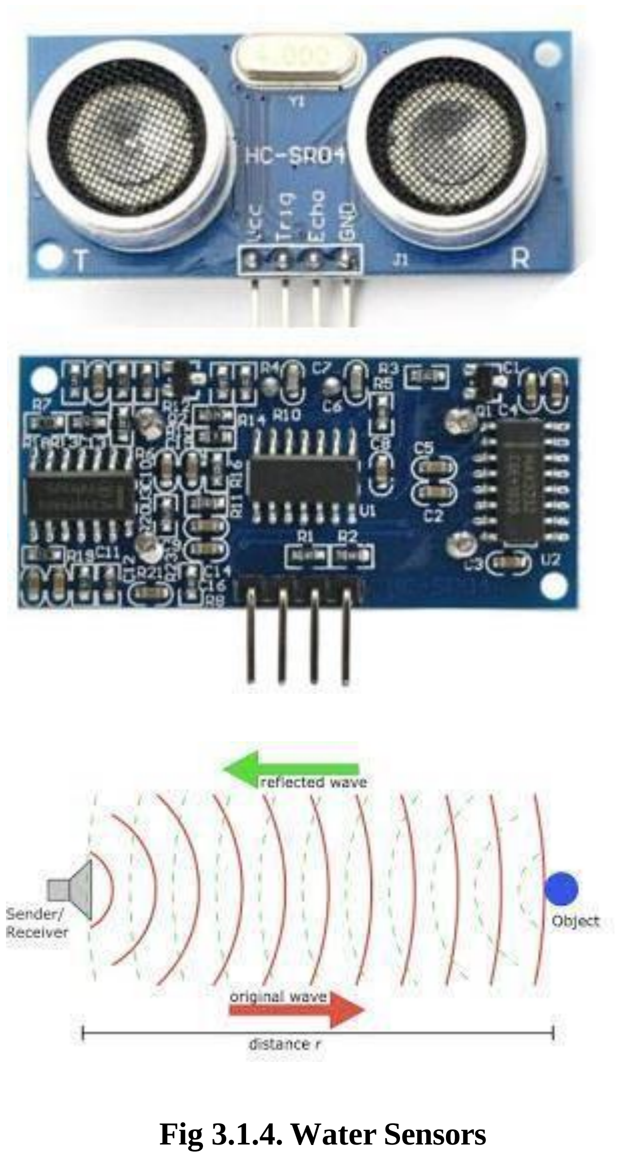


Fig 3.1.4. Water Sensors

Technical Specifications

- Power Supply ±5V DC
- Quiescent Current ≤ 2mA
- Working Current 15mA
- Effectual Angle ≤15°
- Ranging Distance − 2cm – 400 cm/1″ – 13ft
- Resolution − 0.3 cm
- Measuring Angle − 30 degree

3.1.3 DC Motor

DC motors convert electrical into mechanical energy and they consist of permanent magnets and loops of wire inside, when current is applied, the wire loops generate a magnetic field, which reacts against the outside field of the static magnets. The interaction of the fields produces the movement of the shaft/armature. Thus, electromagnetic energy becomes motion.

Fig 3.1.5 DC Motor

Here we use two DC motors for the movement of rover. A DC motor is any of a class of rotary electrical machines that converts direct current electrical energy into mechanical energy. The most common type serially on the forces produced by magnetic fields. Nearly all Types of DC motor shave some internal mechanism, either electromechanical or electronic, to periodically change the direction of current flow in part of the motor. DC motors were the first form of motor widely used, as they could be powered from existing direct-current lighting power distribution systems

Pin Number	**Pin Name**	**Description**
1	Enable 1,2	This pin enables the input pin Input 1(2) and Input 2(7)
2	Input 1	Directly controls the Output 1 pin. Controlled by digital circuits
3	Output 1	Connected to one end of Motor 1
4	Ground	Ground pins are connected to ground of circuit (0V)
5	Ground	Ground pins are connected to ground of circuit (0V)
6	Output 2	Connected to another end of Motor 1
7	Input 2	Directly controls the Output 2 pin. Controlled by digital circuits
8	Vcc2 (Vs)	Connected to Voltage pin for running motors (4.5V to 36V)
9	Enable 3,4	This pin enables the input pin Input 3(10) and Input 4(15)
10	Input 3	Directly controls the Output 3 pin. Controlled by digital circuits
11	Output 3	Connected to one end of Motor 2
12	Ground	Ground pins are connected to ground of circuit (0V)
13	Ground	Ground pins are connected to ground of circuit (0V)

14	Output 4	Connected to another end of Motor 2
15	Input 4	Directly controls the Output 4 pin. Controlled by digital circuits
16	Vcc2 (Vss)	Connected to +5V to enable IC function

Table 3.1.5: DC Motor Pin Specification

3.1.4 H-Bridge L293D:

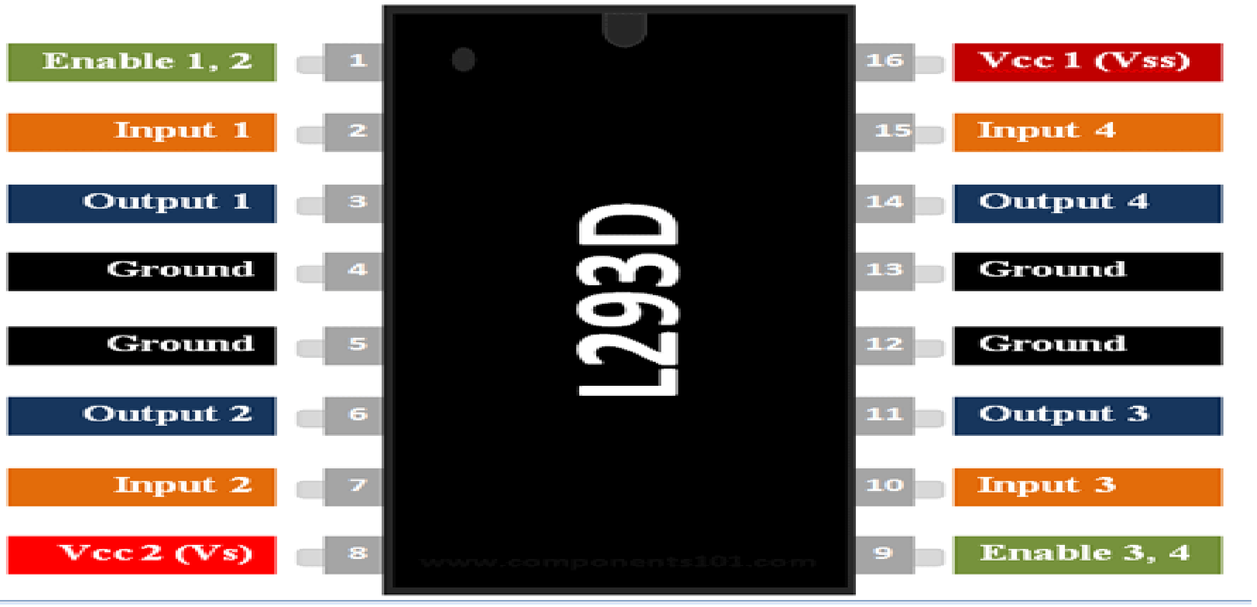


Fig 3.1.6: **L293D Pin Configuration**

Features:

- Can be used to run Two DC motors with the same IC.
- Speed and Direction control is possible
- Motor voltage Vcc2 (Vs): 4.5V to 36V
- Maximum Peak motor current: 1.2A
- Maximum Continuous Motor Current: 600mA
- Supply Voltage to Vcc1(vss): 4.5V to 7V
- Transition time: 300ns (at 5Vand 24V)
- Automatic Thermal shutdown is available

- Available in 16-pin DIP, TSSOP, SOIC packages

Specification of L293D:

Pin Category	**Name**	**Description**
Power	Micro-USB, 3.3V, GND, Vin	**Micro-USB:** NodeMCU can be powered through the USB port **3.3V:** Regulated 3.3V can be supplied to this pin to power the board **GND:** Ground pins **Vin:** External Power Supply
Control Pins	**EN, RST**	The pin and the button resets the microcontroller
Analog Pin	A0	Used to measure analog voltage in the range of 0-3.3V
GPIO Pins	GPIO1 to GPIO16	NodeMCU has 16 general purpose input-output pins on its board
SPI Pins	SD1, CMD, SD0, CLK	NodeMCU has four pins available for SPI communication.
UART Pins	TXD0, RXD0, TXD2, RXD2	NodeMCU has two UART interfaces, UART0 (RXD0 & TXD0) and UART1 (RXD1 & TXD1). UART1 is used to upload the firmware/program.
I2C Pins		NodeMCU has I2C functionality support but due to the internal functionality of these pins, you have to find which pin is I2C.

Table 3.1.6: Specification of L293D

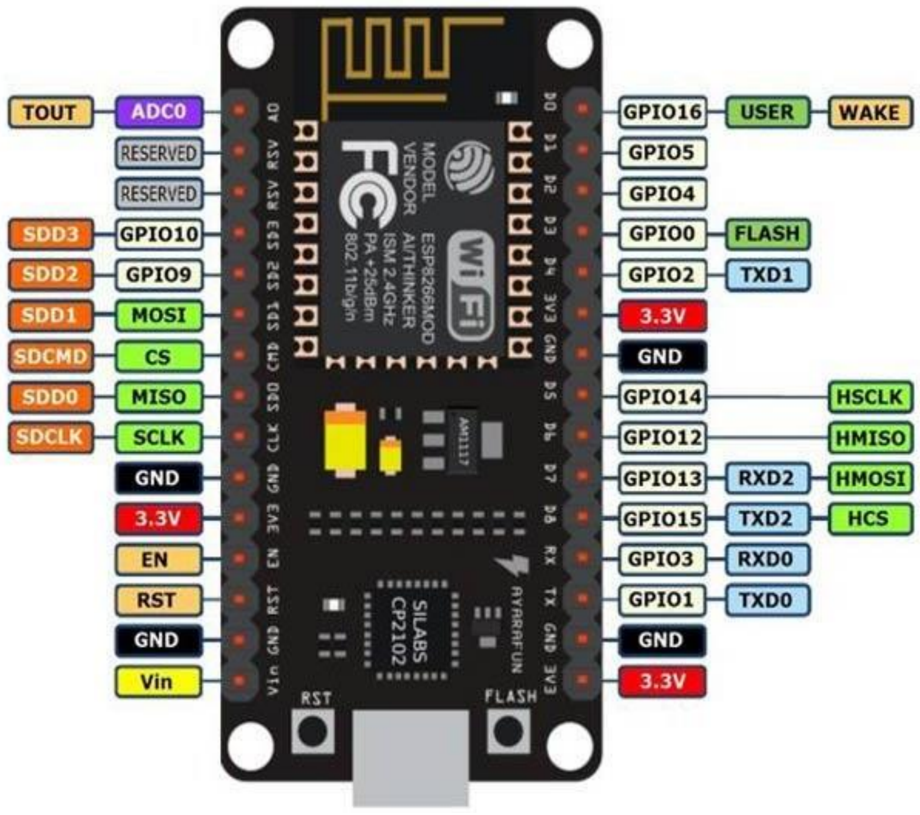


Fig 3.1.7: Pin out Configuration

3.1.5 Node MCU ESP8266 Features

- Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
- Operating Voltage: 3.3V
- Input Voltage: 7-12V
- Digital I/O Pins (DIO): 16
- Analog Input Pins (ADC): 1
- UARTs: 1
- SPIs: 1
- I2Cs: 1
- Flash Memory: 4 MB
- SRAM: 64 KB
- Clock Speed: 80 MHz
- USB-TTL based on CP2102 is included onboard, Enabling Plug n Play

- PCB Antenna
- Small Sized module to fit smartly inside your IoT projects

3.1.6 Buzzer:

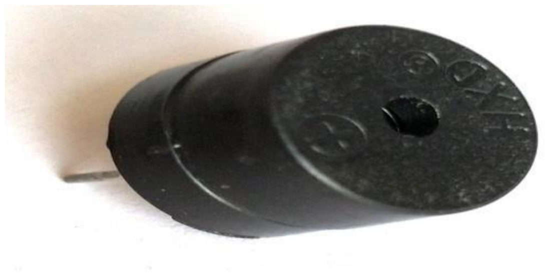


Fig 3.1.8: Buzzer

Buzzer Pin Configuration

Pin Number	**Pin Name**	**Description**
1	Positive	Identified by (+) symbol or longer terminal lead. Can be powered by 6V DC
2	Negative	Identified by short terminal lead. Typically connected to the ground of the circuit

Table 3.1.8: Buzzer pin configuration

Buzzer Features and Specifications

- Rated Voltage: 6V DC
- Operating Voltage: 4-8V DC
- Rated current: <30mA
- Sound Type: Continuous Beep
- Resonant Frequency: ~2300 Hz

- Small and neat sealed package
- Breadboard and Perf board friendly

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 are buzzers that are commonly available. The one shown here is a simple buzzer which when powered will make a Continuous Beeeeeppp. 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.

Applications of Buzzer

- Alarming Circuits, where the user has to be alarmed about something
- Communication equipments
- Automobile electronics
- Portable equipments, due to its compact size

3.1.7 Regulated power supply:

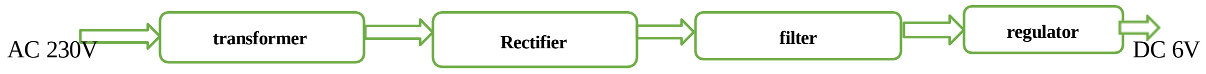


Fig 3.1.9: regulated power supply

3.1.7.1 Transformer:

A transformer is a device that transfers electrical energy from one circuit to another through inductively coupled conductors without changing its frequency. A varying current in the first or primary winding creates a varying magnetic flux in the transformer's core, and thus a varying magnetic field through the secondary winding. This varying magnetic field induces a varying electromotive force (EMF) or "voltage" in the secondary winding. This effect is called mutual induction. If a load is connected to the secondary, an electric current will flow in the secondary winding and electrical energy will be transferred from the primary circuit through the transformer to the load. This field is made up from lines of force and has the same shape as a bar magnet. If the current is increased, the lines of force move outwards from the coil. If the current is reduced, the lines of force move inwards. If another coil is placed adjacent to the first coil then, as the field moves out or in, the moving lines of force will "cut" the turns of the second coil. As it does this, a voltage is induced in the second coil. With the 50 Hz AC mains supply, this will happen 50 times a second. This is called MUTUAL INDUCTION and forms the basis of the transformer.

3.1.7.2 Rectifier:

A rectifier is an electrical device that converts alternating current (AC) to direct current (DC), a process known as rectification. Rectifiers have many uses including as components of power supplies and as detectors of radio signals. Rectifiers may be made of solid-state diodes, vacuum tube diodes, mercury arc valves, and other components. A device that it can perform the opposite function (converting DC to AC) is known as an inverter. When only one diode is used to rectify AC (by blocking the negative or positive portion of the waveform), the difference between the term diode and the term rectifier is merely one of usage, i.e., the term rectifier describes a diode that is being used to convert AC to DC. Almost all rectifiers comprise a number of diodes in a specific arrangement for more efficiently converting AC to DC than is possible with only one diode. Before the development of silicon semiconductor rectifiers, vacuum tube diodes and copper (I) oxide or selenium rectifier stacks were used.

3.1.7.3 Filter:

The process of converting a pulsating direct current to a pure direct current using filters is called as filtration. Electronic filters are electronic circuits, which perform signal-processing functions, specifically to remove unwanted frequency components from the signal, to enhance wanted ones.

3.1.7.4 Regulator:

A voltage regulator (also called a regulator) with only three terminals appears to be a simple device, but it is in fact a very complex integrated circuit. It converts a varying input voltage into a constant _regulated ‘output voltage. Voltage Regulators are available in a variety of outputs like 5V, 6V, 9V, 12V and 15V. The LM78XX series of voltage regulators are designed for positive input. For applications requiring negative input, the LM79XX series is used. Using a pair of _voltage-divider resistors can increase the output voltage of a regulator circuit. It is not possible to obtain a voltage lower than the stated rating. You cannot use a 12V regulator to make a 5V power supply. Voltage regulators are very robust. These can withstand over-current draw due to short circuits and also over-heating. In both cases, the regulator will cut off before any damage occurs. The only way to destroy a regulator is to apply reverse voltage to its input. Reverse polarity destroys the regulator almost instantly. Fig: 3 shows voltage regulator.

3.1.8 LCD display

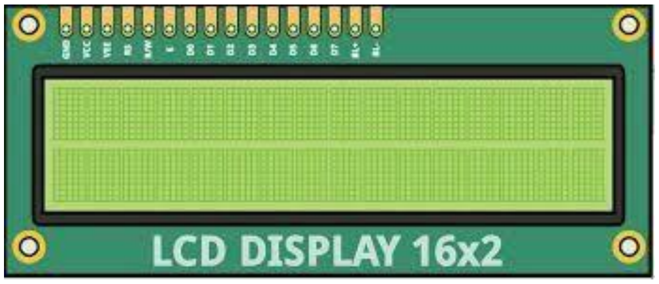


Fig 3.1.10. 16*2 LCD display

A **liquid-crystal display** (**LCD**) is a flat-panel display or other electronically modulated optical device that uses the light-modulating properties of liquid crystals. Liquid crystals do not emit light directly, instead using a backlight or reflector to produce images in color or monochrome. LCDs are available to display arbitrary images (as in a general-purpose computer display) or fixed images with low information content, which can be displayed or hidden, such as preset words, digits, and 7-segment displays, as in a digital clock. They use the same basic technology, except that arbitrary images are made up of a large number of small pixels, while other displays have larger elements.

LCD is used in wide range application including computer monitors, televisions, instrument panels, aircraft cockpit displays, and indoor and outdoor signage. Small LCD screens are common in portable consumer devices such as digital cameras, watches, calculators, and mobile telephones, including smartphones. LCD screens are also used on consumer electronics products such as DVD players, video game devices and clocks. LCD screens have replaced heavy, bulky cathode ray tube (CRT) displays in nearly all applications. LCD screens are available in a wider range of screen sizes than CRT and plasma displays, with LCD screens available in sizes ranging from tiny digital watches to huge, big- screen television sets.

Since LCD screens do not use phosphors, they do not suffer image burn-in when a static image is displayed on a screen for a long time (e.g., the table frame for an aircraft schedule on an indoor sign). LCDs are, however, susceptible to image persistence.

Interfacing an LCD with an Arduino

The 16x2 LCD has a total of 16 pins. As shown in the table below, eight of the pins are data lines (pins 7-14), two are for power and ground (pins 1 and 16), three are used to control the operation of LCD (pins 4-6), and one is used to adjust the LCD screen brightness (pin 3). The remaining two pins (15 and 16) power the backlight. The details of the LCD terminals are as follows

LCD Terminals

Terminal 1	GND
Terminal 2	+5V
Terminal 3	Mid terminal of potentiometer (for brightness control)
Terminal 4	Register Select (RS)
Terminal 5	Read/Write (RW)
Terminal 6	Enable (EN)
Terminal 7	DB0
Terminal 8	DB1
Terminal 9	DB2
Terminal 10	DB3
Terminal 11	DB4
Terminal 12	DB5
Terminal 13	DB6
Terminal 14	DB7
Terminal 15	+4.2-5V
Terminal 16	GND

Table 3.1.10: LCD terminals

3.1.9 LCD MODULE

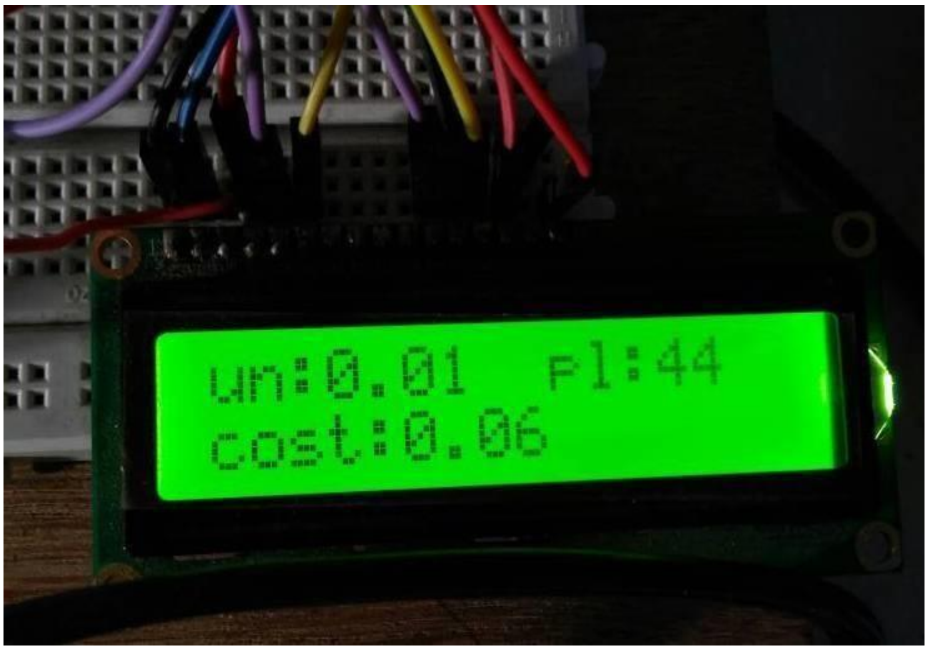


Fig 3.1.11: Output of LCD

The name and functions of each pin of the 16×2 LCD module is given below

Pin1 (Vss): Ground pin of the LCD module.
Pin2 (Vcc): Power to LCD module (+5V supply is given to this pin)
Pin3 (VEE): Contrast adjustment pin. This is done by connecting the ends of a 10K potentimeter to +5V and ground and then connecting the slider pin to the VEE pin.
Pin4(RS): Register select pin. Logic HIGH at RS pin selects data register and logic LOW at RS pin selects command register. If we make the RS pin HIGH and feed an input to the data lines (DB0 to DB7), this input will be treated as data to display on LCD screen. If we make the RS pin LOW and feed an input to the data lines, then this will be treated as a command (a command to be written to LCD controller – like positioning cursor or clear screen or scroll).
Pin5 (R/W): Read/Write modes. This pin is used for selecting between read and write modes. Logic HIGH at this pin activates read mode and logic LOW at this pin activates write mode.
Pin6 (E): This pin is meant for enabling the LCD module. A HIGH to LOW signal at this pin will enable the module.

Pin7 (DB0) to Pin14(DB7): These are data pins. The commands and data are fed to the LCD module though these pins.
Pin15 (LED+): Anode of the back light LED. When operated on 5V, a 560 ohm resistor should be connected in series to this pin. In Arduino based projects the back light LED can be powered from the 3.3V source on the Arduino board.
Pin16 (LED-): Cathode of the back light LED.
RS pin of the LCD module is connected to digital pin 12 of the Arduino. R/W pin of the LCD is grounded. Enable pin of the LCD module is connected to digital pin 11 of the Arduino. This method is very simple, requires less connections and we can almost utilize the full potential of the LCD module. Digital lines DB4, DB5, DB6 and DB7 are interfaced to digital pins 5, 4, 3 and 2 of the Arduino. The 10K potentiometer is used for adjusting the contrast of the display. The Arduino can be powered through the external power jack provided on the board. +5V required in some other parts of the circuit can be tapped from the 5V source on the Arduino board. The Arduino can be also powered from the PC through the USB port.

3.1.10 Rain Sensor:

Nowadays, conserving water as well as its proper usage is essential in everyone’s life. Here is a **sensor** namely rain sensor which is used to detect the rain and generate an alarm. So, we can conserve water to use it later for different purposes. There are several methods available for conserving water like harvesting, etc Using this method we can increase the level of underground water. These sensors are mainly used in the field like automation, irrigation, automobiles, **communication**, etc. This article discusses a simple as well as reliable sensor module which can be available at low cost in the market.

What is a Rain Sensor?

A rain sensor is one kind of switching device which is used to detect the rainfall. It works like **a switch** and the working principle of this sensor is, whenever there is rain, the switch will be normally closed.

Rain Sensor Module

The rain sensor module/board is shown below. Basically, this board includes nickel coated lines and it works on the resistance principle. This **sensor module** permits to gauge moisture through analog output pins & it gives a digital output while moisture threshold surpasses.

This module is similar to **the LM393 IC** because it includes the electronic module as well as a **PCB.** Here PCB is used to collect the raindrops. When the rain falls on the board, then it creates a parallel resistance path to calculate through the **operational amplifier.**
This sensor is a resistive dipole, and based on the moisture only it shows the resistance. For example, it shows more resistance when it is dry and shows less resistance when it is wet.

Pin Configuration

The pin configuration of this sensor is shown below. This sensor includes four pins which include the following.

- Pin1 (VCC): It is a 5V DC pin
- Pin2 (GND): it is a GND (ground) pin
- Pin3 (DO): It is a low/ high output pin
- Pin4 (AO): It is an analog output pin

Specifications

The specifications of the rain sensor include the following.

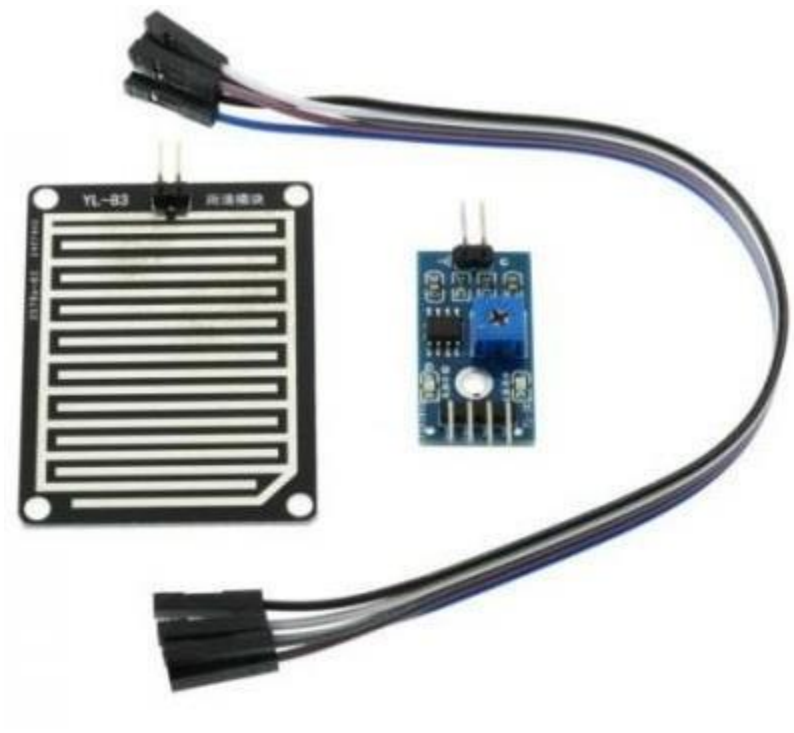


Fig 3.1.12. Rain Sensor

- This sensor module uses good quality of double-sided material.
- Anti-conductivity & oxidation with long time use
- The area of this sensor includes 5cm x 4cm and can be built with a nickel plate on the side
- The sensitivity can be adjusted by a potentiometer
- The required voltage is 5V
- The size of the small PCB is 3.2cm x 1.4cm
- For easy installation, it uses bolt holes
- It uses an LM393 comparator with wide voltage
- The output of the comparator is a clean waveform and driving capacity is above 15mA

Applications

The applications of rain sensor include the following.

- This sensor is used as a water preservation device and this is connected to the irrigation system to shut down the system in the event of rainfall.
- This sensor is used to guard the internal parts of an automobile against the rainfall as well as to support the regular windscreen wiper's mode.
- This sensor is used in specialized satellite communications aerials for activating a rain blower over the opening of the aerial feed, to get rid of water droplets from the mylar wrap to keep pressurized as well as dry air within the waveguides.

3.1.11 Jumper Wires

Fig 3.1.13. Jumper wires

A **jump wire** (also known as jumper, jumper wire, jumper cable, DuPont wire, or DuPont cable) is an electrical wire or group of them in a cable with a connector or pin at each end (or sometimes without them – simply "tinned"), which is normally used to interconnect the components of a breadboard or other prototype or test circuit, internally or with other equipment or components, without soldering.

Individual jump wires are fitted by inserting their "end connectors" into the slots provided in a breadboard, the header connector of a circuit board, or a piece of test equipment.

3.1.12 LDR Sensor:

What is a Light Dependent Resistor or a Photo Resistor?

A *Light Dependent Resistor* (LDR) or a photo resistor is a device whose **resistivity** is a function of the incident electromagnetic radiation. Hence, they are light sensitive devices. They are also called as photo conductors, photo conductive cells or simply photocells. They are made up of **semiconductor** materials having high resistance. There are many different symbols used to indicate a **LDR**, one of the most commonly used symbol is shown in the figure below. The arrow indicates light falling on it.

3.2 SOFTWARE REQUIREMENTS

3.2.1 Arduino IDE

A program for Arduino may be written in any programming language for a compiler that produces binary machine code for the target processor. Atmel provides a development environment for their microcontrollers, AVR Studio and the newer Atmel Studio.

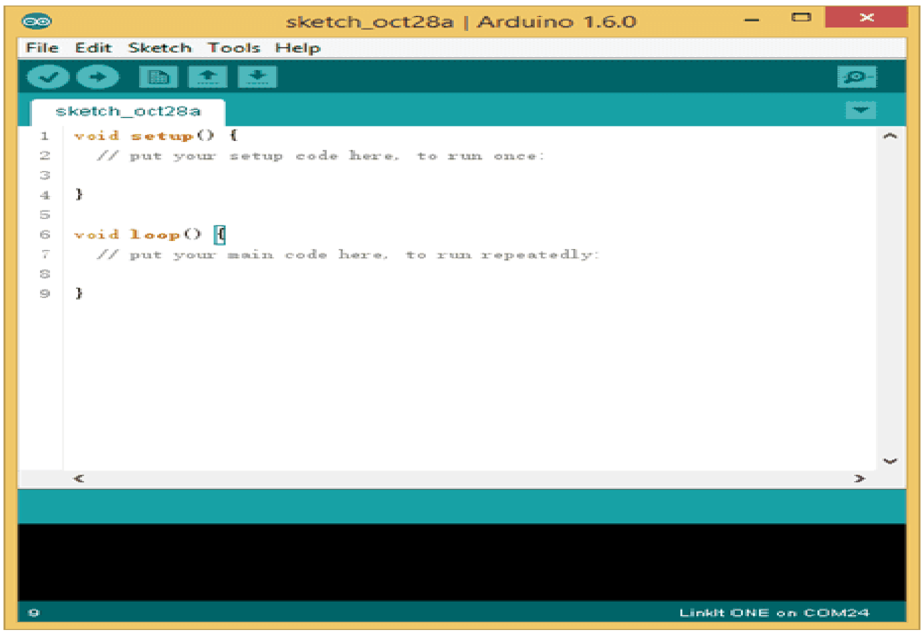
The Arduino project provides the Arduino integrated development environment (IDE), which is a cross-platform application written in the programming language Java. It originated from the IDE for the languages Processing and Wiring. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple one-click mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus.

A program written with the IDE for Arduino is called a sketch. Sketches are saved on the development computer as text files with the file extension .ino. Arduino Software (IDE) pre-1.0 saved sketches with the extension. pde.

The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution.

A minimal Arduino C/C++ sketch, as seen by the Arduino IDE programmer, consist of only two functions:

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



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

}

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

}
```

Fig 3.2.1. An Arduino Sketch

3.2.2 Embedded C:

When designing software for a smaller embedded system with the 8051, it is very common place to develop the entire product using assembly code. With many projects, this is a feasible approach since the amount of code that must be generated is typically less than 8 kilobytes and is relatively simple in nature. If a hardware engineer is tasked with designing both the hardware and the software, he or she will frequently be tempted to write the software in assembly language. The trouble with projects done with assembly code can is that they can be difficult to read and maintain, especially if they are not well commented. Additionally, the amount of code reusable from a typical assembly language project is usually very low. Use of a higher-level language like C can directly address these issues. A program written in C is easier to read than an assembly program. Since a C program possesses greater structure, it is easier to understand and maintain. Because of its modularity, a C program can better lend itself to reuse of code from project to project. The division of code into functions will force better structure of the software and lead to functions that can be taken from one project and used in another, thus reducing overall development time. A high order language such as C allows a developer to write code, which resembles a human's thought process more closely than does the equivalent assembly code. The developer can focus more time on designing the algorithms of the system rather than having to concentrate on their individual implementation. This will greatly reduce development time and lower debugging time since the code is more understandable. By using a language like C, the programmer does not have to be intimately familiar with the architecture of the processor. This means that someone new to a given processor can get a project up and running quicker, since the internals and organization of the target processor do not have to be learned. Additionally, code developed in C will be more portable to other systems than code developed in assembly. Many target processors have C compilers available, which support ANSI C. All of this is not to say that assembly language does not have its place. In fact, many embedded systems (particularly real time systems) have a combination of C and assembly code. For time critical operations, assembly code is frequently the only way to go. One of the great things about the C language is that it allows you to perform low-level manipulations of the hardware if need be, yet provides you the functionality and abstraction of a higher order language.

3.2.3 PYTHON

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is interpreted:** Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

- **Python is Interactive:** You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented:** Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language:** Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Python Features

- **Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read:** Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain:** Python's source code is fairly easy-to-maintain.
- **A broad standard library:** Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode:** Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable:** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable:** You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases:** Python provides interfaces to all major commercial databases.
- **GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable:** Python provides a better structure and support for large programs than shell scripting.

Python’s standard library

- Pandas
- Numpy
- Sklearn
- Seaborn
- Matplotlib

- Importing Datasets
- OpenCV

IMAGE PROCESSING USING OPENCV IN PYTHON

Image processing is the process of manipulating pixel data in order to make it suitable for computer vision applications or to make it suitable to present it to humans. For example, changing brightness or contrast is an image processing task which make the image visually pleasing for humans or suitable for further processing for a certain computer vision application.

PYTHON

It is an object-oriented programming language. The processing happens during the runtime, and this is performed by the interpreter. Python's simple to learn and easy to use is an advantage and thus makes it developer friendly. It is easier to read and understand as the syntax is conventional. The code can be executed line by line using the interpreter. Python can support multiple platforms like Linux, UNIX, windows, Macintosh, and so on. The paradigms of Object-oriented programming are supported by python. The functions such as polymorphism, operator overloading and multiple inheritance is supported python.

3.2.4 Open CV

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library is cross platform and free for use under the open-source BSD license. OpenCV supports deep learning frameworks Tensor Flow, Torch/PyTorch and Cafe.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

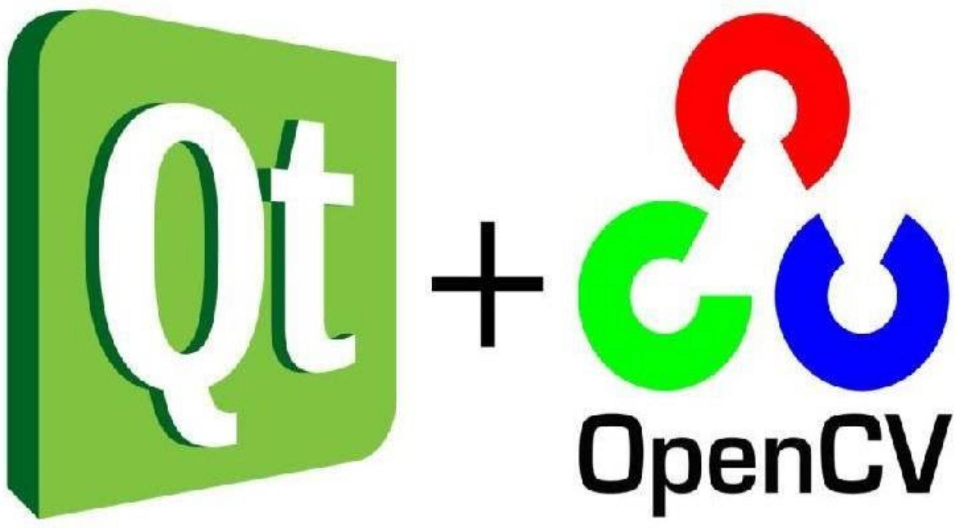


Fig 3.2.4. Structure of Open CV

Once OpenCV is installed, the OPENCV_BUILD\install directory will be populated with three types of files:

- **Header files**: These are located in the OPENCV_BUILD\install\includesubdirectory and are used to develop new projects with OpenCV.
- **Library binaries**: These are static or dynamic libraries (depending on the option selected with CMake) with the functionality of each of the OpenCV modules. They are located in the bin subdirectory (for example, x64\mingw\bin when the GNU compiler is used).
- **Sample binaries**: These are executables with examples that use the libraries. The sources for these samples can be found in the source package.

General description

- Open source computer vision library in C/C++.
- Optimized and intended fr real-time applications.
- OS/hardware/window-manager independent.
- Generic image/video loading, saving, and acquisition.

Features

- Image data manipulation (allocation, release, copying, setting, conversion).
- Image and video I/O (file and camera based input, image/video file output).
- Matrix and vector manipulation and linear algebra routines (products, solvers,, SVD).
- Various dynamic data structures (lists, queues, sets, trees, graphs).
- Basic image processing (filtering, edge detection, corner detection, sampling and interpolation, color conversion, morphological operations, histograms, image pyramids).
- Structural analysis (connected components, contour processing, distance transform, various moments, template matching, Hough transform, polygonal approximation, line fitting, ellipse fitting, Delaunay triangulation).
- Camera calibration (finding and tracking calibration patterns, calibration, fundamental matrix estimation, homography estimation, stereo correspondence).
- Motion analysis (optical flow, motion segmentation, tracking).
- Object recognition (eigen-methods, HMM).
- Basic GUI (display image/video, keyboard and mouse handling, scroll-bars).
- Image labeling (line, conic, polygon, text drawing)

3.3 Functional Requirements:

- **Rain Detection:** The system should be able to detect rain using a rain sensor. Upon detecting rain, the system should trigger appropriate actions.
- **Water Level Monitoring:** The system should monitor the water level using a water level sensor. It should provide accurate information about the water level in the underpass.
- **Gate Control:** The system should control the opening and closing of the underpass gate using a DC motor. The gate should close when water levels rise to prevent flooding.
- **Power Management:** The system should be powered by both solar and battery sources for continuous operation. It should intelligently switch between solar and battery power based on availability.

- **Human Detection:** The camera connected to a laptop should detect humans in the underpass area. The system should trigger alerts if humans are detected during water logging events.
- **Notification System:** NodeMCU should send timely notifications to relevant authorities or users. Notifications should include information about water levels, gate status, and human presence.

3.4 Non-Functional Requirements:

- **Reliability:** The system should be reliable and capable of continuous operation in various weather conditions.
- **Scalability:** The system should be scalable to accommodate additional sensors or devices if required.
- **Security:** Data transmission, especially notifications, should be secured to prevent unauthorized access.
- **Usability:** The system should have a user-friendly interface, such as an LCD display for local monitoring.
- **Power Efficiency:** The system should optimize power consumption to ensure prolonged operation during low sunlight or battery conditions.
- **Accuracy:** Sensors should provide accurate data to ensure precise monitoring and control.
- **Response Time:** The system's response time, especially in triggering gate closure or sending notifications, should be minimized.
- **Maintainability:** The system should be designed for easy maintenance, allowing quick replacement or upgrades of components.
- **Adaptability:** The system should adapt to changes in environmental onditions and adjust its operations accordingly.
- **Integration:** The components (Arduino, sensors, camera, NodeMCU) should seamlessly integrate to form a cohesive system.

CHAPTER 4
PROPOSED METHODOLOGY

Problem Definition: Describe the specifics of the project, including the underpasses that require monitoring and the volume of water logging that is considered troublesome. Determine the parameters that require monitoring, such as the water level, precipitation totals, traffic volume, and pump condition. Data collection: Install Internet of Things (IoT) sensors to monitor water levels, traffic volume, and weather at strategic underpass locations. Gather historical data on water logging problems, rainfall patterns, and road congestion.

Preprocessing of the Data: Remove anomalies and noise from the collected data Combine data. Make use of feature engineering to find relevant characteristics in raw data. Model Creation: Build a machine learning model to predict water logging occurrences using input features such as traffic flow, rainfall intensity, and historical data. Choose the appropriate machine learning (ML) algorithm based on the type of problem and the available data, such as regression, decision trees, or neural networks.

To train and evaluate the model, divide the dataset into training and testing sets. Train the machine learning model with the training dataset, then evaluate its performance with Adjust the model's parameters to maximize performance metrics such as accuracy, precision, Install a real-time monitoring system powered by the Internet of Things (IoT) to continuously Integrate the trained machine learning model into the monitoring system to predict the Install alert systems to notify relevant authorities or stakeholders when the risk of water logging exceeds a preset threshold.

Utilize the predictions of the machine learning model to guide the development of a response This may mean sending out alerts to motorists, altering traffic routes to reduce congestion, or Optimization and the Feedback Loop: Consult with relevant parties and monitor the system's Make use of this feedback to fine-tune the parameters, increase overall system performance, and strengthen the machine learning model.

BLOCK DIAGRAM:

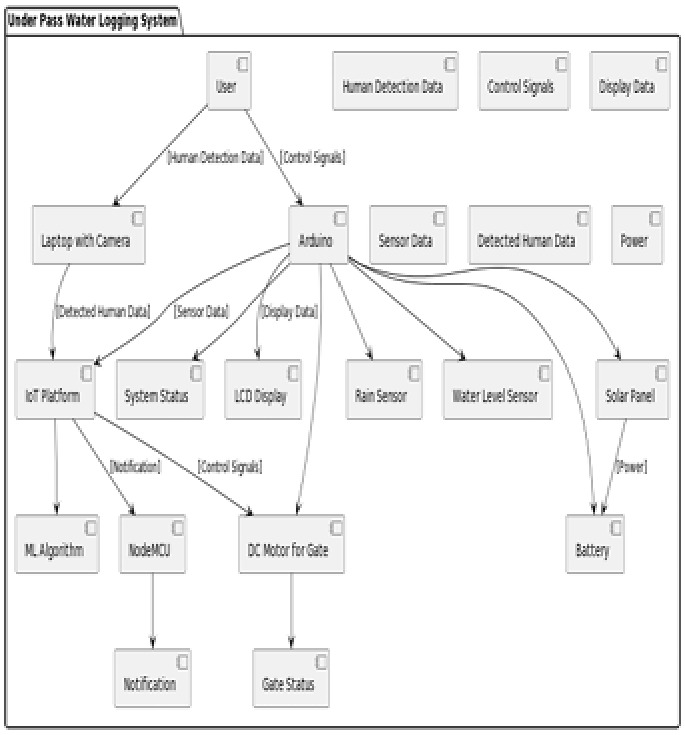


Figure 4.1 Block Diagram

Block Diagram Description:

Understanding the role of the Internet of Things in the water logging management system underpass is the aim of this study. When the underpass floods, this system sends out timely alerts. To help ease traffic congestion, an LCD panel displaying the type of vehicle that can pass through the underpass can be placed in front of it. The block diagram of the complete system is shown in the picture. The complete system consists of front-end devices, cloud architecture, hardware nodes, and a WiFi module. The underpass's entrance needs to have a camera or webcam installed first. When people approach the underpass at the wrong time, the camera will identify the person's face using the Haar cascade technique and send information to

The water level sensor provides an analog input to the Atmega 328p microcontroller. It runs at a clock frequency of 20 MHz and a speed of 20 MIPS in the voltage range of 2.7–5.5V.

Following that, the LCD receives this data from the microcontroller and is positioned to allow cars to veer and avoid the flooded underpass. The data is used by the screen to choose which messages to show. To evacuate the accumulated water in the underpass, we utilize a motor driver to turn on and off the DC pump, which is controlled by a microcontroller or WIFI module. The water level sensor's input is posted to the Think Speak website by creating a channel.

4.2 UML DIAGRAM

4.2.1.CLASS DIAGRAM

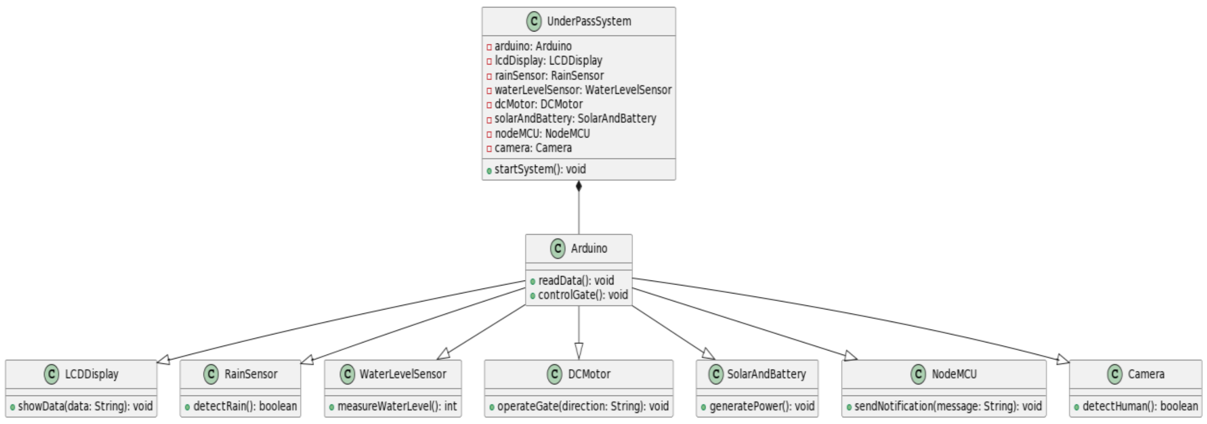


Fig 4.2.1 Class Diagram

In software engineering, a class diagram is a type of static structural diagram that displays an object's classes, properties, operations (or methods), and relationships with other objects in the system.

4.2.2 USE CASE DIAGRAM

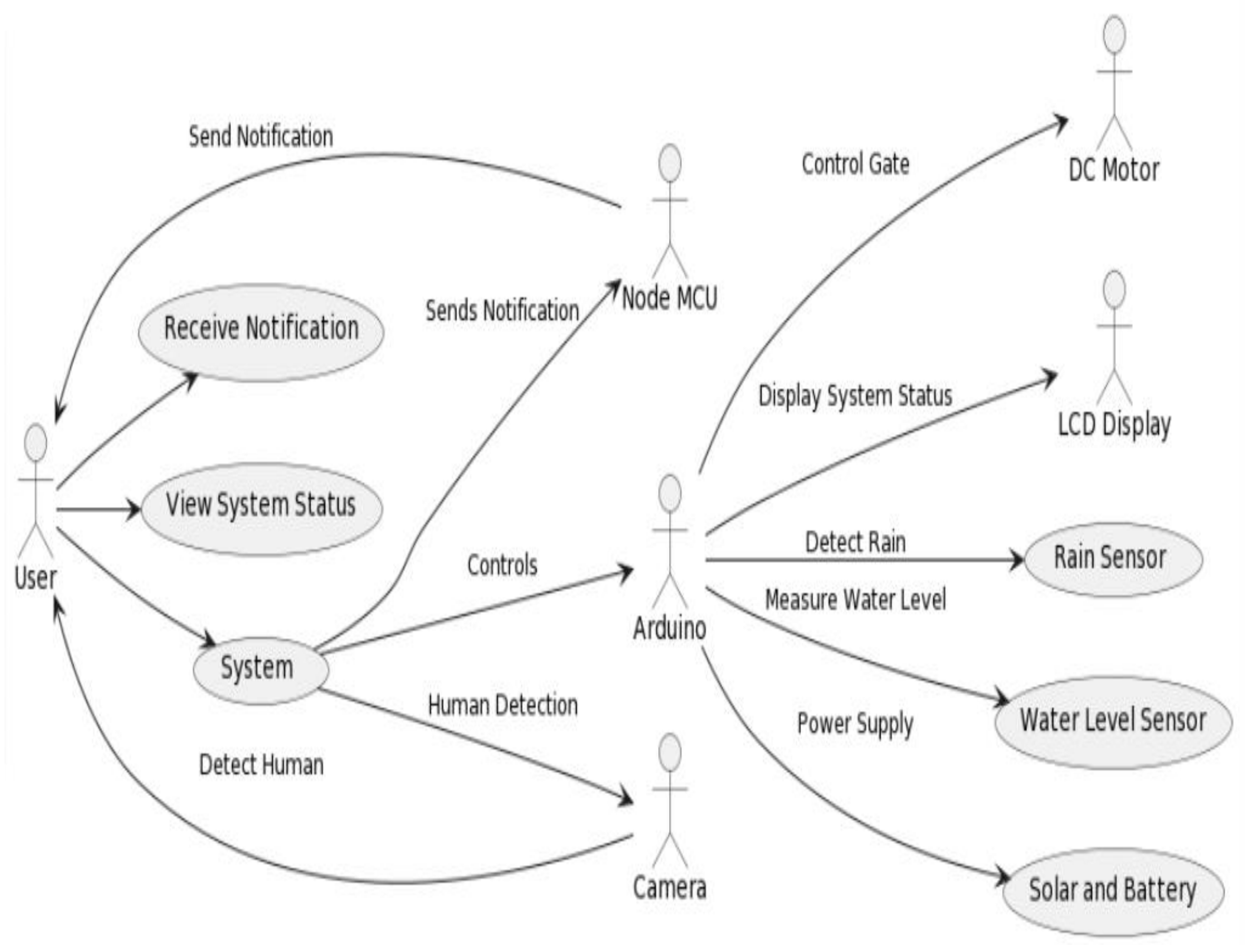


Fig 4.2.2 Use Case Diagram

A use case diagram, in its most basic form, is a representation of a user's interaction with the system that shows how the user and the many use cases they are involved in are connected. A use case diagram can be used to identify the many use cases and user types of a system; it is often used in conjunction with other types of diagrams. While a use case may go into great detail about each situation, a use case diagram can help provide a higher-level view of the system. We've heard the expression "use case diagrams are the blueprints for your system" before. They provide a simplified and graphical representation of what the system actually needs to do.

4.2.3 SEQUENCE DIAGRAM

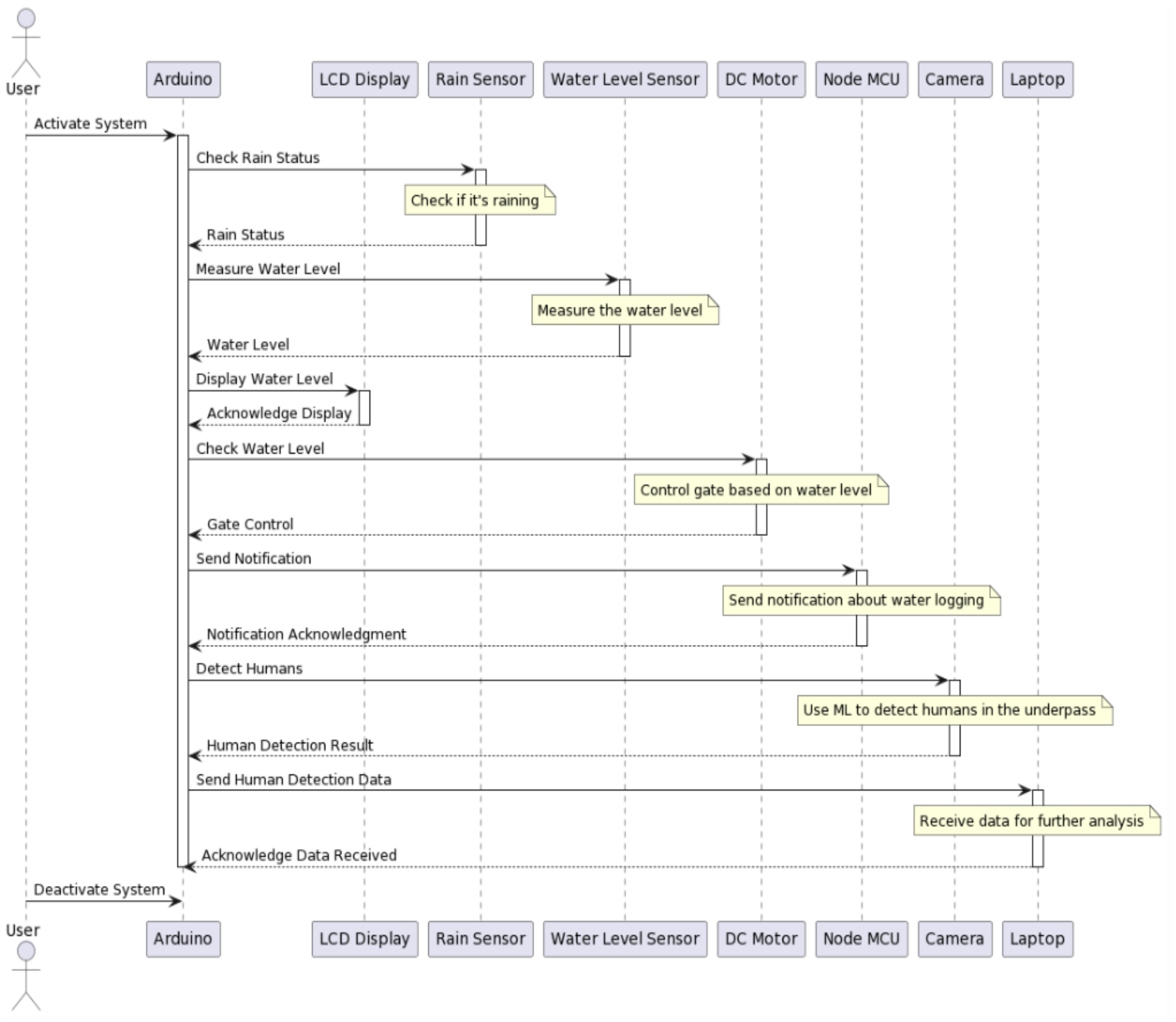


Fig 4.2.3 Sequence Diagram

A sequence diagram shows the interactions of an object in chronological order. In order for the scenario to work, it displays the classes and objects that are a part of it along with the messages that are passed between the objects. Sequence diagrams are frequently associated with the ways Sequence diagrams can also be called event diagrams or event scenarios. In a sequence diagram, several concurrently occurring processes or objects are shown as parallel vertical lines (sometimes called lifelines), and the messages that are sent between them are shown as horizontal arrows, arranged chronologically. This makes it possible to specify fundamental runtime scenarios graphically.

4.2.4 ACTIVITY DIAGRAM

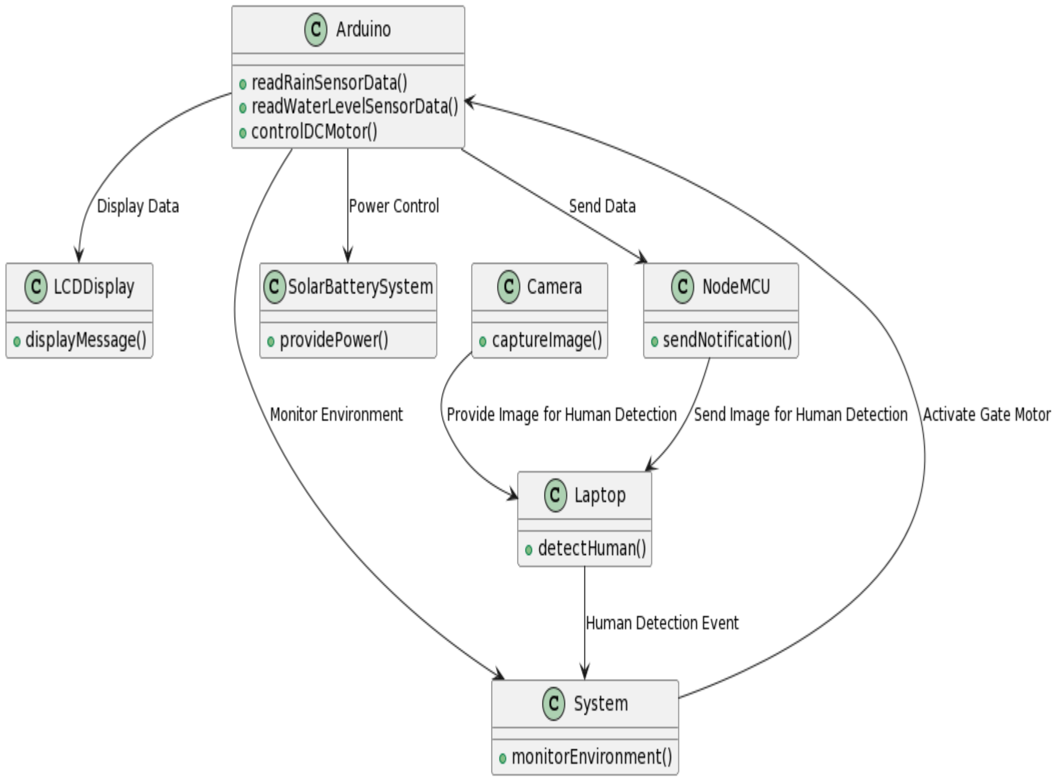


Fig 4.2.4 Activity Diagram

4.2.5 PROPOSED METHOD

Sensor Deployment: To monitor temperature, humidity, water levels, and other relevant variables, strategically place Internet of Things (IoT) sensors throughout the underpass. A central server should be able to receive data from these sensors instantly. Data Collection & Preprocessing: Gather data from the deployed sensors continuously. Preprocessing the collected data is important to remove noise, outliers, and superfluous information. This stage is critical to ensuring the accuracy of later machine learning models.

Using the pre processed data, feature engineering extracts valuable features. Features could include patterns in water levels, traffic flow, rainfall intensity, and past instances of water logging. Feature engineering has a big impact on how well machine learning models function. Model Selection: Choose the most effective machine learning algorithms to predict water logging incidents based on the features that were gathered. Depending on the complexity of the problem and the available data, algorithms like Decision Trees, Random Forests, Gradient Boosting Machines, or even deep learning models like Convolutional Neural Networks (CNNs)

Training and Validation: Train the selected machine learning models using historical data. Employ methods like cross-validation to assess model performance and modify hyper parameters for optimal results. Validation is required to ensure the model applies Actual Forecast: Make instantaneous forecasts regarding water logging incidents using the created machine learning model. Once new data from IoT sensors is received, the model should be able to adapt to changing situations and keep updating its predictions.

Alerting System: Combine the machine learning model with an alerting system to notify relevant authorities or stakeholders when the risk of water logging surpasses a predefined threshold. This enables the implementation of preventive measures, such traffic rerouting and Constant observation and development: Continually monitor the system's performance and get feedback from users and stakeholders. Regularly retrain the machine learning model with new

Evaluation: Find out how successfully the IoT-ML system lowers the frequency of water logging incidents. Consider factors like response speed, false alarm rate, and prediction accuracy when assessing its impact on improving underpass safety and functionality. Iterative Refinement: Based on assessment findings and user input, make iterative system enhancements to address any shortcomings or constraints. This can mean adjusting machine learning algorithms, rearranging sensor locations, or enhancing data preparation techniques.

4.3 Testing

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. Testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

4.3.1 Testing Principle

Before applying methods to design effective test cases, a software engineer must understand the basic principle that guides software testing. All the tests should be traceable to customer requirements.

4.3.2 Testing Methods

There are different methods that can be used for software testing. They are,

1. Black-Box Testing

The technique of testing without having any knowledge of the interior workings of the application is called black-box testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, while performing a black-box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

2. White-Box Testing

White-box testing is the detailed investigation of internal logic and structure of the code. White-box testing is also called glass testing or open-box testing. In order to perform white-box testing on an application, a tester needs to know the internal workings of the code. The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

4.3.3 Levels of Testing

There are different levels during the process of testing. Levels of testing include different methodologies that can be used while conducting software testing. The main levels of software testing are:

- **Functional Testing:**

This is a type of black-box testing that is based on the specifications of the software that is to be tested. The application is tested by providing input and then the results are examined that need to conform to the functionality it was intended for. Functional testing of software is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. There are five steps that are involved while testing an application for functionality.

- The determination of the functionality that the intended application is meant to perform.
- The creation of test data based on the specifications of the application.
- The output based on the test data and the specifications of the application.
- The writing of test scenarios and the execution of test cases.
- The comparison of actual and expected results based on the executed test cases.

➢ **Non-functional Testing**

This section is based upon testing an application from its non-functional attributes. Non-functional testing involves testing software from the requirements which are non-functional in nature but important such as performance, security, user interface, etc. Testing can be done in different levels of SDLC. Few of them are

4.3.3.1 Unit Testing

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. Unit testing is often automated but it can also be done manually. The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality. Test cases and results are shown in the Tables.

Unit Testing Benefits

- Unit testing increases confidence in changing/ maintaining code.
- Codes are more reusable.
- Development is faster.
- The cost of fixing a defect detected during unit testing is lesser in comparison to that of defects detected at higher levels.
- Debugging is easy.

- Codes are more reliable.

Unit Testing:

Sl # Test Case : -	UTC-1
Name of Test: -	Arduino Test
Items being tested: -	Power up the Arduino and boot
Sample Input: -	Turn on the Supply
Expected output: -	Arduino Should boot
Actual output: -	Power on test and upload successful
Remarks: -	Pass.

Table 4.3.1 Unit Testing (1)

Sl # Test Case : -	UTC-2
Name of Test: -	Sensors Test
Items being tested: -	Sensors Values Should change depending on Physical Changes
Sample Input: -	Tested for different inputs
Expected output: -	Should Show Proper Variation
Actual output: -	Variation detected
Remarks: -	Pass

Table 4.3.2 Unit Testing (2)

Sl # Test Case : -	UTC-3
Name of Test: -	Camera Test
Items being tested: -	Capture image
Sample Input: -	Tested for different inputs
Expected output: -	Should capture image
Actual output: -	Image Captured
Remarks: -	Pass

Table 4.3.3 Unit Testing (3)

4.3.3.2 Integration Testing:

Integration testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing. Integration testing is defined as the testing of combined parts of an application to determine if they function correctly. It occurs after unit testing and before validation testing. Integration testing can be done in two ways: Bottom-up integration testing and Top-down integration testing.

1. **Bottom-up Integration**

This testing begins with unit testing, followed by tests of progressively higher-level combinations of units called modules or builds.

2. **Top-down Integration**

In this testing, the highest-level modules are tested first and progressively, lower-level modules are tested thereafter.

In a comprehensive software development environment, bottom-up testing is usually done first, followed by top-down testing. The process concludes with multiple tests of the complete application, preferably in scenarios designed to mimic actual situations. The table shows the test cases for integration testing and their results.

Integration Testing:

Sl # Test Case : -	ITC-1
Name of Test: -	Flood Detection
Item being tested: -	Water Sensor Value , Rain value , Flow Sensor
Sample Input: -	Different input values
Expected output: -	Should Detect Flood and Earthquake
Actual output: -	When Sensor values crosses Threshold Values LCD Display Earth quake and Flood Values
Remarks: -	Pass.

Table 4.3.4 Integration Testing (1)

Sl # Test Case : -	ITC-2
Name of Test: -	Human Detection
Item being tested: -	Human Posture Detection
Sample Input: -	Image or video
Expected output: -	Should detect the human posture in image
Actual output: -	Human Posture Detected
Remarks: -	Pass.

Table 4.3.5 Integration Testing (2)

4.3.3.3 System Testing:

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black-box testing, and as such, should require no knowledge of the inner design of the code or logic. System testing is important because of the following reasons:

- System testing is the first step in the Software Development Life Cycle, where the application is tested as a whole.
- The application is tested thoroughly to verify that it meets the functional and technical specifications.
- The application is tested in an environment that is very close to the production environment where the application will be deployed.
- System testing enables us to test, verify, and validate both the business requirements as well as the application architecture.

System Testing is shown in below tables

System testing:

Sl # Test Case : -	STC-1
Name of Test: -	Intimation and Detection
Item being tested: -	Should Intimate For underpass condition and Human detection
Sample Input: -	Sensor Inputs and Image Input
Expected output: -	Synchronization and Intimation
Actual output: -	Received the Synchronization output
Remarks: -	Pass

Table 4.3.6 System Testing (1)

4.3.3.4 Validation Testing

At the culmination of integration testing, software is completely assembled as a packages; interfacing errors have been covered and corrected, and final series of software tests-validating testing may begin. Validation can be defined in many ways, but a simple definition is that validation succeeds when software functions in a manner that can be reasonably expected by customers. Reasonable expectation is defined in the software requirement specification- a document that describes all users' visible attributes of the software. The specification contains a section title "validation criteria". Information contained in that section forms the basis for validation testing approach.

CHAPTER 5
RESULT

5.1 Discussion about result

The project UNDERPASS WATERLOGGING USING IOT AND ML includes various Hardware Components such as Arduino board, Water level sensor (to detect water level in the underpass), IoT module (e.g., ESP8266), relay module (to control pumps or valves), Power supply (to power the Arduino and other components) Connecting wires and breadboard (for circuit connections).it also requires Software and Libraries such as

Arduino IDE for coding and uploading the program to the Arduino board, Libraries for the water level sensor, IoT module.

The Project Implementation is that we Connect the water level sensor to the Arduino board and calibrate it to detect different water levels. Use the IoT module to transmit the water level data to a cloud platform (e.g., AWS IoT, Azure IoT, or Google Cloud IoT Core). Implement a machine learning model on the cloud platform to analyze the water level data and predict potential water logging events. Based on the ML model's predictions, trigger actions such as activating pumps or sending alerts to prevent water logging.

Use the cloud platform's data logging features to store the water level data for analysis and future reference. Implement data visualization tools (e.g., dashboards) to monitor the water level in real-time and track historical data.

The Safety Considerations is that we Ensure that the system is designed to be safe and reliable, especially when dealing with water management systems.

The Testing and Iteration is that We Test the system thoroughly to ensure its functionality and reliability under various conditions. Iterate on the design and implementation based on testing results and feedback to improve its performance.

Remember to consider factors such as power consumption, connectivity, and scalability while designing your system.

5.2 Efficiency of the project

The efficiency of the "Underpass Water Logging Using IoT and ML" project can be evaluated based on the accuracy and timeliness of water level detection and prediction of water logging events. The project's effectiveness in preventing water logging incidents and reducing the impact of flooding in underpasses is crucial. Additionally, the reliability of IoT connectivity, energy efficiency, scalability, and adaptability of the system are key factors to consider. Overall, the project's efficiency can be measured by its ability to detect, predict, and mitigate water logging events effectively, while being reliable, energy-efficient, and adaptable to different underpass configurations and environmental conditions.

CHAPTER 6
CONCLUSION AND FUTUREWORKS

6.1 Conclusion

The project proposes a simple water level monitoring system with different levels indicated. It also signifies when the water level is below and above then the requirement. Future Work can involve the analysis of water level in a particular area so that the wastage of water is prevented. We can also include the SMS based system where the message will be sent to the actual authorized person when the water level is below the specified level. The advancement of innovation made ready to increment in originations like WSN, IoT, and 5G and numerous others. In this paper, we've made programmed water utilization checking framework dependent on cloud in a lake which lets in plunging the consumption of the water.

6.2 Future Work

For future work, the "Underpass Water Logging Using IoT and ML" project could focus on enhancing sensor technologies for more accurate water level detection, integrating advanced machine learning models for improved prediction of water logging events, and implementing real-time decision-making capabilities. Additionally, integrating weather forecasting data, developing remote monitoring and control features, and optimizing the system through data analytics could further improve its efficiency and effectiveness. Engaging with local communities for monitoring and reporting, as well as raising awareness, could create a more collaborative approach to managing underpass water logging, ensuring a more responsive and proactive system in the future.

In envisioning the future of underpass waterlogging systems, several key areas emerge for exploration and advancement.

Firstly, there's a significant opportunity for innovation in sensor technology. Implementing more advanced sensors capable of detecting water levels with greater precision and speed would enhance the effectiveness of these systems. Furthermore, integrating these sensors with smart algorithms and AI could enable predictive modeling, allowing authorities to anticipate potential waterlogging events and take proactive measures to mitigate them.

Secondly, enhancing the infrastructure itself is crucial. This involves designing underpasses with improved drainage systems, perhaps incorporating advanced materials that resist corrosion and deterioration. Additionally, exploring novel architectural designs could optimize water flow and minimize the risk of blockages during heavy rainfall.

Moreover, leveraging interconnectedness and the Internet of Things (IoT) could revolutionize underpass waterlogging systems. By connecting these systems to a broader urban network, data sharing and real-time monitoring could be streamlined, facilitating rapid response to changing conditions. This interconnectedness could extend beyond underpasses to encompass entire urban drainage systems, providing a comprehensive approach to managing water flow in cities. Furthermore, considering sustainability in the development of these systems is essential. Exploring eco-friendly materials and renewable energy sources for powering pumps and sensors could reduce the environmental impact of underpass waterlogging systems. Additionally, integrating green infrastructure elements, such as permeable pavements or rain gardens, into underpass designs could help absorb excess water and alleviate pressure on drainage systems.

Lastly, public awareness and community engagement play a vital role in the success of underpass waterlogging systems. Educating residents about the importance of proper waste disposal and the consequences of clogged drains can help prevent waterlogging incidents. Moreover, involving local communities in the planning and maintenance of these systems fosters a sense of ownership and responsibility, leading to more sustainable and effective solutions.

In conclusion, the future of underpass waterlogging systems lies in technological innovation, infrastructure improvement, interconnectedness, sustainability, and community engagement. By addressing these aspects, cities can better manage water flow and mitigate the impacts of flooding, creating safer and more resilient urban environments.

REFERENCES

[1] Deccan Herald home page [online] Available:https://www.deccanherald.com/content/434220/underpassparwater html.,(2018)

[2] Abhishek Rameshbhai Patel, Hemal S. Parekh “Study of Stormwater Problem at Akhbarnagar Underpass (Ahmedabad)” in International Journal of Advance Engineering and Research Development (IJAERD), (2015)

[3] Samiksha Sharma,A Review Paper on the Internet of Things., in International Journal of Engineering Research & Technology (IJERT), 5(23)(2017) 1-7.

[4] Pravallika, Devireddy Prathyusha, Srivasa Kumar, IOT based water level monitoring system with an android application, in International journal of recent and Innovation trends in computing and communication, 6(6)(2018) 94-97.

[5] Gowthamy J, Chinta Rohith Reddy, Pijush Meher, Saransh Shrivastava, Guddu Kumar., Smart Water Monitoring System using IoT, in International Research Journal of Engineering and Technology (IRJET), 5(10)(2018) 1-4.

[6] Ujwala Kini H, Tejaswini, Internet of Things (IoT) A Gateway for Smarter Life, in International Journal of Engineering Research & Technology (IJERT), 6(13)(2018) 1-7.

[7] Kamal Patel, Khushboo Patel, Rajvansh Patel, Kalgi Trivedi, Storm Water Drainage for Underpass, in International Journal for Innovative Research Science & Technology, 3(09)(2017).

[8] N. Aishwarya, R. Akila, R. Gayathri, Mr. S. Dhamodharan., IoT Based Soldier monitoring System., in SSRG International Journal of Electronics and Communication Engineering, Special Issue, (2019) 23-27. Project Phase I Underpass Waterlogging using IOT and ML Dept of ISE, HKBKCE 49 2023-24

[9] Ali Al Dahoud, Mohammed Fezari, NodeMCU V3 For Fast IoT Application Development, (2018).

[10] Melchizedek I. Alipio∗, Jess Ross R. Bayanay, “Vehicle Traffic and Flood Monitoring with Reroute System Using Bayesian Networks Analysis., in 2017 IEEE 6th Global Conference on Consumer Electronics (GCCE 2017).

[11] Times of India homepage [Online] Available https://timesofindia.indiatimes.com/city/bengaluru/bbmp-banksoninjection-wells-to-prevent-underpassflooding/ articleshow/73179057.cms., (2020).

[12] Astha Gautam, Anjana Kumari, Pankaj Singh: "The Concept of Object Recognition", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 5, Issue 3, March 2015

[13] Joseph Redmon, Santosh Divvala, Ross Girshick, “You Only Look Once: Unified, Real-Time Object Detection”, The IEEE Conference on Computer Vision and Pattern Recognition

(CVPR),2016,pp. 779- 788

[14] V. Gajjar, A. Gurnani and Y. Khandhediya, "Human Detection and Tracking for Video Surveillance: A Cognitive Science Approach," in 2017 IEEE International Conference on Computer Vision Workshops, 2017.

[15] Gowthamy J, Chinta Rohith Reddy, Pijush Meher, Saransh Shrivastava, Guddu Kumar., Smart Water Monitoring System using IoT, in International Research Journal of Engineering and Technology (IRJET), 5(10)(2018) 1-4.

[16] Ujwala Kini H, Tejaswini, Internet of Things (IoT) A Gateway for Smarter Life, in International Journal of Engineering Research & Technology (IJERT), 6(13)(2018) 1-7.

[17] Kamal Patel, Khushboo Patel, Rajvansh Patel, Kalgi Trivedi, Storm Water Drainage for Underpass, in International Journal for Innovative Research Science & Technology, 3(09)(2017).

[18] N. Aishwarya, R. Akila, R. Gayathri, Mr. S. Dhamodharan., IoT Based Soldier monitoring System., in SSRG International Journal of Electronics and Communication Engineering, Special Issue, (2019) 23-27.

[19] Ali Al Dahoud, Mohammed Fezari, NodeMCU V3 For Fast IoT Application Development, (2018).

[20] Melchizedek I. Alipio∗, Jess Ross R. Bayanay, “Vehicle Traffic and Flood Monitoring with Reroute System Using Bayesian Networks Analysis., in 2017 IEEE 6th Global Conference on Consumer Electronics (GCCE 2017).

APPENDIX-I

SNAPSHOTS:

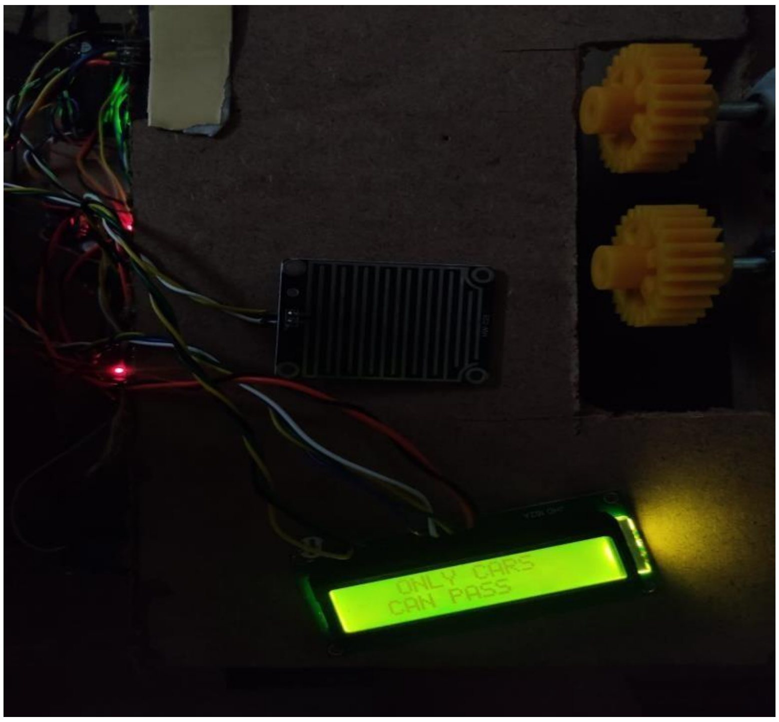


Figure (i): Only cars can pass when the level of water is low

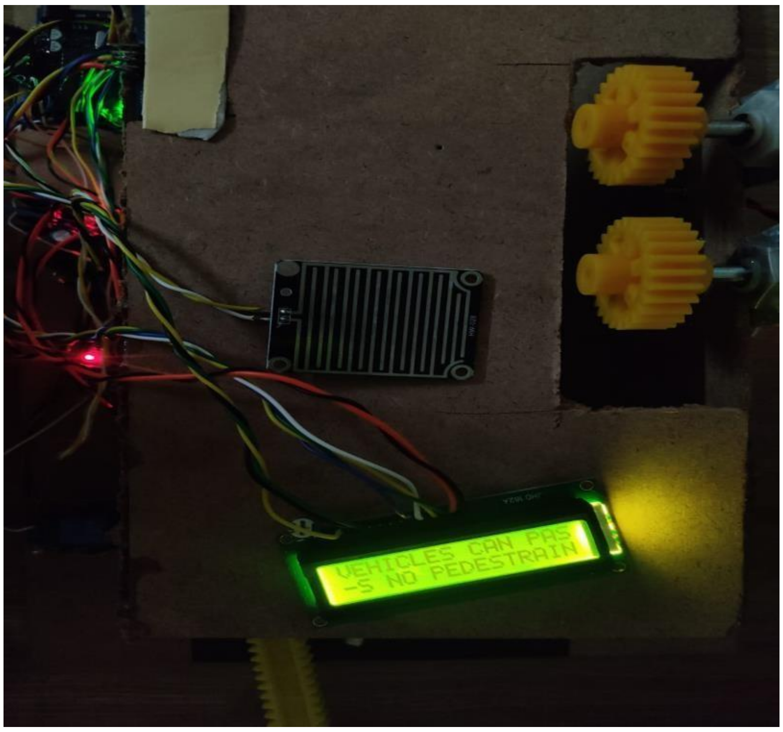


Figure (ii): Vehicles can pass when the level of water is moderate

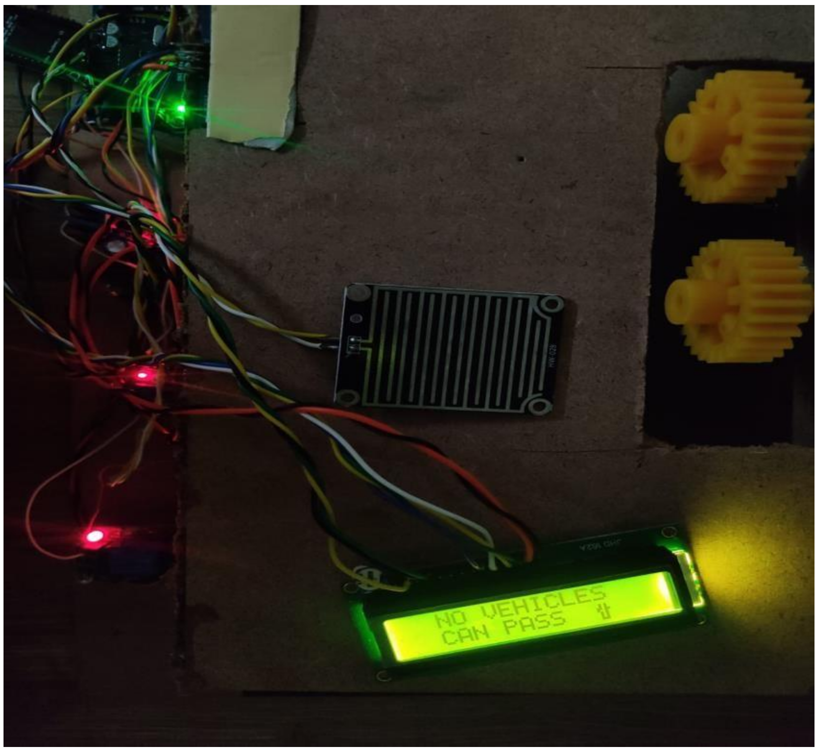


Figure (iii): No vehicles can pass when the level of water is high

APPENDIX-II

SOURCE CODE:

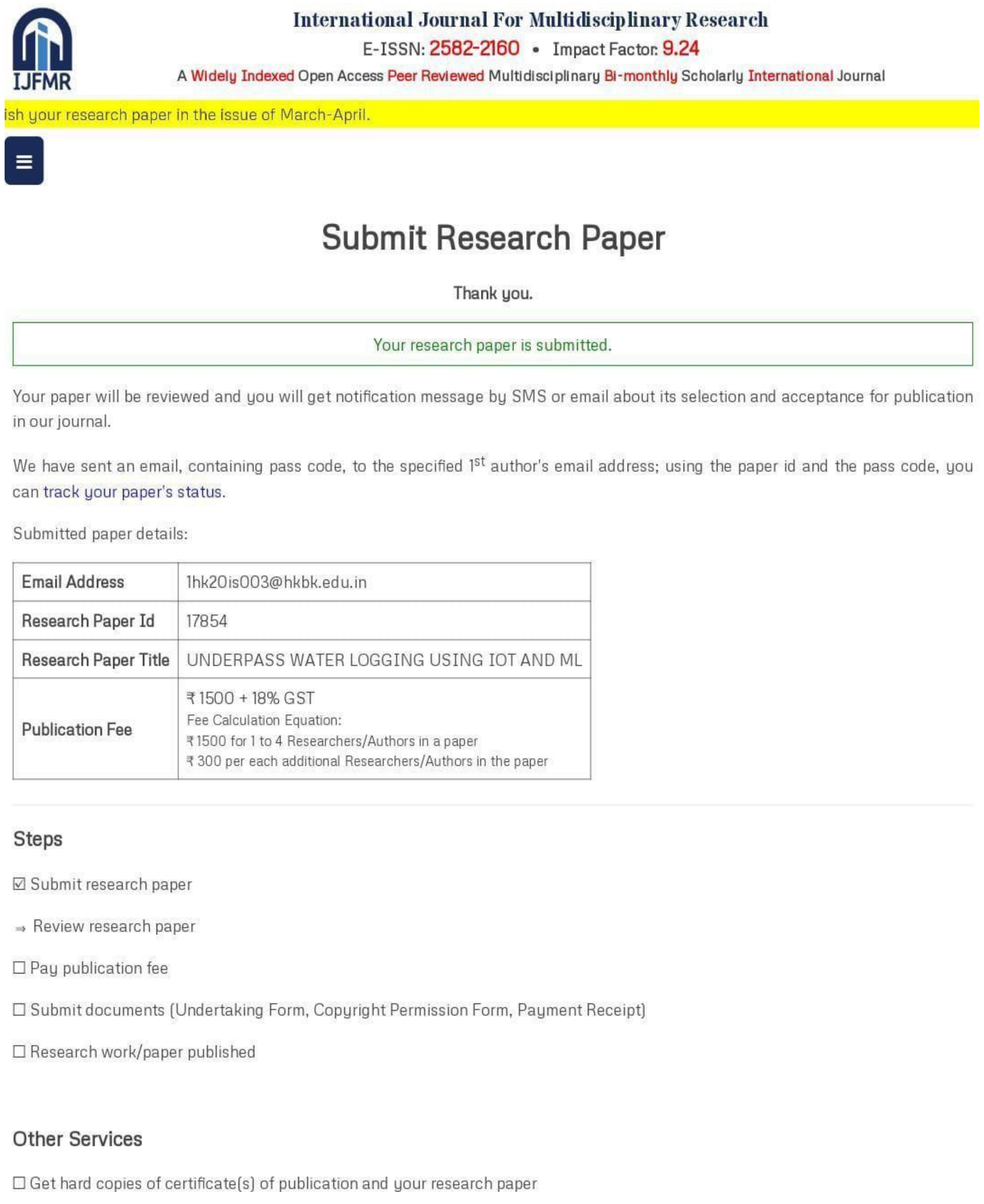
```
import argparse
import csv
import os
import platform
import sys
from pathlib import Path

import torch

FILE = Path(__file__).resolve()
ROOT = FILE.parents[0] # YOLOv5 root directory
if str(ROOT) not in sys.path:
    sys.path.append(str(ROOT))  # add ROOT to PATH
ROOT = Path(os.path.relpath(ROOT, Path.cwd()))  # relative

from ultralytics.utils.plotting import Annotator, colors, save_one_box

from models.common import DetectMultiBackend
from utils.dataloaders import IMG_FORMATS, VID_FORMATS, LoadImages,
LoadScreenshots, LoadStreams
from utils.general import (
    LOGGER,
    Profile,
    check_file,
    check_img_size,
    check_imshow,
    check_requirements,
    colorstr,
    cv2,
    increment_path,
    non_max_suppression,
    print_args,
    scale_boxes,
    strip_optimizer,
    xyxy2xywh,
)
from utils.torch_utils import select_device, smart_inference_mode



@smart_inference_mode()
def run(
    weights=ROOT / "yolov5s.pt", # model path or triton URL
    source=ROOT / "data/images", # file/dir/URL/glob/screen/0(webcam)
    data=ROOT / "data/coco128.yaml",  # dataset.yaml path
    imgsz=(640, 640), # inference size (height, width)
    conf_thres=0.25, # confidence threshold
    iou_thres=0.45, # NMS IOU threshold
    max_det=1000, # maximum detections per image
    device="", # cuda device, i.e. 0 or 0,1,2,3 or cpu
    view_img=False, # show results
    save_txt=False, # save results to *.txt
    save_csv=False,  # save results in CSV format
    save_conf=False, # save confidences in --save-txt labels
    save_crop=False, # save cropped prediction boxes
    nosave=False, # do not save images/videos
    classes=None, # filter by class: --class 0, or --class 0 2 3
    agnostic_nms=False, # class-agnostic NMS
    augment=False,  # augmented inference
    visualize=False, # visualize features
    update=False,  # update all models
    project=ROOT / "runs/detect", # save results to project/name
    name="exp",  # save results to project/name
    exist_ok=False, # existing project/name ok, do not increment
    line_thickness=3, # bounding box thickness (pixels)
    hide_labels=False,  # hide labels
    hide_conf=False,  # hide confidences
    half=False,  # use FP16 half-precision inference
    dnn=False, # use OpenCV DNN for ONNX inference
    vid_stride=1,  # video frame-rate stride
):
    source = str(source)
    save_img = not nosave and not source.endswith(".txt")  # save inference images
    is_file = Path(source).suffix[1:] in (IMG_FORMATS + VID_FORMATS)
    is_url = source.lower().startswith(("rtsp://", "rtmp://", "http://", "https://"))
    webcam = source.isnumeric() or source.endswith(".streams") or (is_url and not is_file)
    screenshot = source.lower().startswith("screen")
    if is_url and is_file:
        source = check_file(source)  # download

    # Directories
    save_dir = increment_path(Path(project) / name, exist_ok=exist_ok) # increment run
    (save_dir / "labels" if save_txt else save_dir).mkdir(parents=True, exist_ok=True)  #
make dir

    # Load model
    device = select_device(device)
    model = DetectMultiBackend(weights, device=device, dnn=dnn, data=data, fp16=half)
    stride, names, pt = model.stride, model.names, model.pt
    imgsz = check_img_size(imgsz, s=stride)  # check image size

    # Dataloader
    bs = 1 # batch_size
    if webcam:
        view_img = check_imshow(warn=True)
        dataset = LoadStreams(source, img_size=imgsz, stride=stride, auto=pt, vid_stride=vid_stride)
        bs = len(dataset)
    elif screenshot:
        dataset = LoadScreenshots(source, img_size=imgsz, stride=stride, auto=pt)
    else:
        dataset = LoadImages(source, img_size=imgsz, stride=stride, auto=pt, vid_stride=vid_stride)
    vid_path, vid_writer = [None] * bs, [None] * bs

    # Run inference
    model.warmup(imgsz=(1 if pt or model.triton else bs, 3, *imgsz))  # warmup
    seen, windows, dt = 0, [], (Profile(device=device), Profile(device=device), Profile(device=device))
    for path, im, im0s, vid_cap, s in dataset:
        with dt[0]:
            im = torch.from_numpy(im).to(model.device)
            im = im.half() if model.fp16 else im.float() # uint8 to fp16/32
            im /= 255 # 0 - 255 to 0.0 - 1.0
            if len(im.shape) == 3:
                im = im[None] # expand for batch dim
            if model.xml and im.shape[0] > 1:
                ims = torch.chunk(im, im.shape[0], 0)

        # Inference
        with dt[1]:
            visualize = increment_path(save_dir / Path(path).stem, mkdir=True) if visualize else False
            if model.xml and im.shape[0] > 1:
                pred = None
                for image in ims:
                    if pred is None:
                        pred = model(image, augment=augment, visualize=visualize).unsqueeze(0)
                    else:
                        pred = torch.cat((pred, model(image, augment=augment,
    visualize=visualize).unsqueeze(0)), dim=0)
                pred = [pred, None]
            else:
                pred = model(im, augment=augment, visualize=visualize)
        # NMS
        with dt[2]:
            pred = non_max_suppression(pred, conf_thres, iou_thres, classes, agnostic_nms,
    max_det=max_det)

        # Second-stage classifier (optional)
        # pred = utils.general.apply_classifier(pred, classifier_model, im, im0s)

        # Define the path for the CSV file
        csv_path = save_dir / "predictions.csv"

        # Create or append to the CSV file
        def write_to_csv(image_name, prediction, confidence):
            """Writes prediction data for an image to a CSV file, appending if the file exists."""
            data = {"Image Name": image_name, "Prediction": prediction, "Confidence":
    confidence}
            with open(csv_path, mode="a", newline="") as f:
                writer = csv.DictWriter(f, fieldnames=data.keys())
                if not csv_path.is_file():
                    writer.writeheader()
                writer.writerow(data)

        # Process predictions
        for i, det in enumerate(pred): # per image
            seen += 1
            if webcam:  # batch_size >= 1
                p, im0, frame = path[i], im0s[i].copy(), dataset.count
                s += f"{i}: "
            else:
                p, im0, frame = path, im0s.copy(), getattr(dataset, "frame", 0)

            p = Path(p)  # to Path
            save_path = str(save_dir / p.name)  # im.jpg
            txt_path = str(save_dir / "labels" / p.stem) + ("" if dataset.mode == "image" else
    f"_{frame}")  # im.txt
            s += "%gx%g " % im.shape[2:]  # print string
            gn = torch.tensor(im0.shape)[[1, 0, 1, 0]] # normalization gain whwh
            imc = im0.copy() if save_crop else im0  # for save_crop
            annotator = Annotator(im0, line_width=line_thickness, example=str(names))
            if len(det):
                # Rescale boxes from img_size to im0 size
                det[:, :4] = scale_boxes(im.shape[2:], det[:, :4], im0.shape).round()

                # Print results
                for c in det[:, 5].unique():
                    n = (det[:, 5] == c).sum()  # detections per class
                    s += f"{n} {names[int(c)]}{'s' * (n > 1)}, "  # add to string

                # Write results
                for *xyxy, conf, cls in reversed(det):
                    c = int(cls) # integer class
                    label = names[c] if hide_conf else f"{names[c]}"
                    confidence = float(conf)
                    confidence_str = f"{confidence:.2f}"

                    if save_csv:
                        write_to_csv(p.name, label, confidence_str)

                    if save_txt:  # Write to file
```

```
                    xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-1).tolist()
# normalized xywh
                    line = (cls, *xywh, conf) if save_conf else (cls, *xywh) # label format
                    with open(f"{txt_path}.txt", "a") as f:
                        f.write(("%g " * len(line)).rstrip() % line + "\n")

                if save_img or save_crop or view_img:  # Add bbox to image
                    c = int(cls)  # integer class
                    if names[c]=="person":
                        label = None if hide_labels else (names[c] if hide_conf else f"{names[c]}
{conf:.2f}")
                        annotator.box_label(xyxy, label, color=colors(c, True))
                if save_crop:
                    save_one_box(xyxy, imc, file=save_dir / "crops" / names[c] /
f"{p.stem}.jpg", BGR=True)

            # Stream results
            im0 = annotator.result()
            if view_img:
                if platform.system() == "Linux" and p not in windows:
                    windows.append(p)
                    cv2.namedWindow(str(p), cv2.WINDOW_NORMAL |
cv2.WINDOW_KEEPRATIO)  # allow window resize (Linux)
                    cv2.resizeWindow(str(p), im0.shape[1], im0.shape[0])
                cv2.imshow(str(p), im0)
                cv2.waitKey(1)  # 1 millisecond

            # Save results (image with detections)
            if save_img:
                if dataset.mode == "image":
                    cv2.imwrite(save_path, im0)
                else:  # 'video' or 'stream'
                    if vid_path[i] != save_path:  # new video
                    vid_path[i] = save_path
                    if isinstance(vid_writer[i], cv2.VideoWriter):
                        vid_writer[i].release()  # release previous video writer
                    if vid_cap:  # video
                        fps = vid_cap.get(cv2.CAP_PROP_FPS)
                        w = int(vid_cap.get(cv2.CAP_PROP_FRAME_WIDTH))
                        h = int(vid_cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
                    else:  # stream
                        fps, w, h = 30, im0.shape[1], im0.shape[0]
                    save_path = str(Path(save_path).with_suffix(".mp4")) # force *.mp4 suffix
on results videos
                    vid_writer[i] = cv2.VideoWriter(save_path,
cv2.VideoWriter_fourcc(*"mp4v"), fps, (w, h))
                vid_writer[i].write(im0)

        # Print time (inference-only)
        # LOGGER.info(f"{s}{'' if len(det) else '(no detections), '}{dt[1].dt * 1E3:.1f}ms")

    # Print results
    t = tuple(x.t / seen * 1e3 for x in dt)  # speeds per image
    LOGGER.info(f"Speed: %.1fms pre-process, %.1fms inference, %.1fms NMS per
image at shape {(1, 3, *imgsz)}" % t)
    if save_txt or save_img:
        s = f"\n{len(list(save_dir.glob('labels/*.txt')))} labels saved to {save_dir / 'labels'}" if
save_txt else ""
        LOGGER.info(f"Results saved to {colorstr('bold', save_dir)}{s}")
    if update:
        strip_optimizer(weights[0])  # update model (to fix SourceChangeWarning)


def parse_opt():
    """Parses command-line arguments for YOLOv5 detection, setting inference options
and model configurations."""
```

```
    parser = argparse.ArgumentParser()
    parser.add_argument("--weights", nargs="+", type=str, default=ROOT / "yolov5s.pt",
help="model path or triton URL")
    parser.add_argument("--source", type=str, default=ROOT / "0",
help="file/dir/URL/glob/screen/0(webcam)")
    parser.add_argument("--data", type=str, default=ROOT / "data/coco128.yaml",
help="(optional) dataset.yaml path")
    parser.add_argument("--imgsz", "--img", "--img-size", nargs="+", type=int,
default=[640], help="inference size h,w")
    parser.add_argument("--conf-thres", type=float, default=0.25, help="confidence
threshold")
    parser.add_argument("--iou-thres", type=float, default=0.45, help="NMS IoU
threshold")
    parser.add_argument("--max-det", type=int, default=1000, help="maximum detections
per image")
    parser.add_argument("--device", default="", help="cuda device, i.e. 0 or 0,1,2,3 or cpu")
    parser.add_argument("--view-img", action="store_true", help="show results")
    parser.add_argument("--save-txt", action="store_true", help="save results to *.txt")
    parser.add_argument("--save-csv", action="store_true", help="save results in CSV
format")
    parser.add_argument("--save-conf", action="store_true", help="save confidences in --
save-txt labels")
    parser.add_argument("--save-crop", action="store_true", help="save cropped prediction
boxes")
    parser.add_argument("--nosave", action="store_true", help="do not save
images/videos")
    parser.add_argument("--classes", nargs="+", type=int, help="filter by class: --classes 0,
or --classes 0 2 3")
    parser.add_argument("--agnostic-nms", action="store_true", help="class-agnostic
NMS")
    parser.add_argument("--augment", action="store_true", help="augmented inference")
    parser.add_argument("--visualize", action="store_true", help="visualize features")
    parser.add_argument("--update", action="store_true", help="update all models")
    parser.add_argument("--project", default=ROOT / "runs/detect", help="save results to project/name")
    parser.add_argument("--name", default="exp", help="save results to project/name")
    parser.add_argument("--exist-ok", action="store_true", help="existing project/name ok, do not increment")
    parser.add_argument("--line-thickness", default=2, type=int, help="bounding box thickness (pixels)")
    parser.add_argument("--hide-labels", default=False, action="store_true", help="hide labels")
    parser.add_argument("--hide-conf", default=False, action="store_true", help="hide confidences")
    parser.add_argument("--half", action="store_true", help="use FP16 half-precision inference")
    parser.add_argument("--dnn", action="store_true", help="use OpenCV DNN for ONNX inference")
    parser.add_argument("--vid-stride", type=int, default=1, help="video frame-rate stride")
    opt = parser.parse_args()
    opt.imgsz *= 2 if len(opt.imgsz) == 1 else 1 # expand
    print_args(vars(opt))
    return opt


def main(opt):
    """Executes YOLOv5 model inference with given options, checking requirements before running the model."""
    check_requirements(ROOT / "requirements.txt", exclude=("tensorboard", "thop"))
    run(**vars(opt))
if__name__== "_main_":
    opt = parse_opt()
    main(opt)
```

APPENDIX-III

PUBLICATION:



International Journal For Multidisciplinary Research

E-ISSN: 2582-2160 • Impact Factor: 9.24

A Widely Indexed Open Access Peer Reviewed Multidisciplinary Bi-monthly Scholarly International Journal

ish your research paper in the issue of March-April.

Submit Research Paper

Thank you.

Your research paper is submitted.

Your paper will be reviewed and you will get notification message by SMS or email about its selection and acceptance for publication in our journal.

We have sent an email, containing pass code, to the specified 1st author's email address; using the paper id and the pass code, you can track your paper's status.

Submitted paper details:

Email Address	1hk20is003@hkbk.edu.in
Research Paper Id	17854
Research Paper Title	UNDERPASS WATER LOGGING USING IOT AND ML
Publication Fee	₹ 1500 + 18% GST Fee Calculation Equation: ₹ 1500 for 1 to 4 Researchers/Authors in a paper ₹ 300 per each additional Researchers/Authors in the paper

Steps

☑ Submit research paper

⇒ Review research paper

☐ Pay publication fee

☐ Submit documents (Undertaking Form, Copyright Permission Form, Payment Receipt)

☐ Research work/paper published

Other Services

☐ Get hard copies of certificate(s) of publication and your research paper

APPENDIX-IV

PLAGIARISM

Finale Report (1.1)

by Turnitin Official

Finale Report (1.1)

ORIGINALITY REPORT

17%	11%	15%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Suresh N, Bala Vamsi T, Aravind Y, B. Veena Vani, B. Naresh Kumar Reddy. "An efficient way to vehicle movement monitor in flooded underpass using IoT", 2022 International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES), 2022 Publication	3%
2	Sunyang Song, Xue Deng, Chao Gao. "A Sensor-Based Smart Urban Flood Warning and Management System", 2023 4th International Conference on Information Science, Parallel and Distributed Systems (ISPDS), 2023 Publication	2%
3	www.semanticscholar.org Internet Source	2%
4	Subhashree Rath, Vaishali M Deshmukh, Rafeeq Manzoor, Sourav Singh, S Joydeep Singh. "IoT and ML based Flood Alert and Human Detection System", 2022 4th	2%

International Conference on Smart Systems and Inventive Technology (ICSSIT), 2022
Publication

5	www.mdpi.com Internet Source	1 %
6	www.researchgate.net Internet Source	1 %
7	Gervy Andrew Amagsila, Mark Emmanuel Cabuhat, Jhon Emil Tigbayan, Edmhar Uy, Eliseo Ramirez. "A framework for mobile application of flood alert monitoring system for vehicle users using Arduino device", 2017IEEE 9th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM), 2017 Publication	1 %
8	kssem.edu.in Internet Source	1 %
9	www.internationaljournalssrg.org Internet Source	1 %
10	Devi Devapal, N. Hashna, V.P. Aparna, C. Bhavyasree, Jeena Mathai, K. Sangeetha Soman. "Object Detection from SAR Images based on Curvelet Despeckling", Materials Today: Proceedings, 2019	1 %

Publication

11	Shiva shankar, J J Jijesh, Dileep Reddy Bolla, Mahaveer Penna, P V Sruthi, Alla Gowthami. "Early Detection of Flood Monitoring and Alerting System to Save Human Lives", 2020 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT), 2020 Publication	1%
12	www.cse.griet.ac.in Internet Source	1%
13	www.sciencegate.app Internet Source	1%
14	www.ijert.org Internet Source	<1%
15	Submitted to Sunway Education Group Student Paper	<1%
16	Submitted to Kingston University Student Paper	<1%
17	www.scilit.net Internet Source	<1%
18	Submitted to American InterContinental University Student Paper	<1%
19	Submitted to Indiana Tech Student Paper	<1%

20	beei.org Internet Source	<1%
21	www.ijircst.org Internet Source	<1%
22	Submitted to North Shore International Academy Student Paper	<1%
23	easychair.org Internet Source	<1%
24	fastercapital.com Internet Source	<1%
25	www.termedia.pl Internet Source	<1%
26	catalog.libraries.psu.edu Internet Source	<1%
27	Submitted to SolBridge International School of Business Student Paper	<1%
28	letsmitigatemontana.com Internet Source	<1%
29	www.coursehero.com Internet Source	<1%
30	www.igi-global.com Internet Source	<1%

31	K Kalaivani, S Subramanian, G K S Swedha, N Vinoth, V Vishnu Priya. "Air Monitoring with Cloud and IoT", 2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS), 2023 Publication	<1%
32	Submitted to National Institute of Technology Karnataka Surathkal Student Paper	<1%
33	docplayer.net Internet Source	<1%
34	technical-outsourcing.com Internet Source	<1%
35	123dok.com Internet Source	<1%
36	assets.researchsquare.com Internet Source	<1%
37	eweb.ouc.edu.cn Internet Source	<1%
38	kaizen.com Internet Source	<1%
39	pdffox.com Internet Source	<1%
40	www.acadlore.com Internet Source	<1%

41	www.tandfonline.com Internet Source	<1%
42	"Table of Contents", 2023 4th International Conference on Information Science, Parallel and Distributed Systems (ISPDS), 2023 Publication	<1%
43	B. Naresh Kumar Reddy, B. Seetharamulu, G. Siva Krishna, B. Veena Vani. "An FPGA and ASIC Implementation of Cubing Architecture", Wireless Personal Communications, 2022 Publication	<1%
44	Sandhya.A. Kulkarni, Vishal D Raikar, B K Rahul, L V Rakshitha, K Sharanya, Vandana Jha. "Intelligent Water Level Monitoring System Using IoT", 2020 IEEE International Symposium on Sustainable Energy, Signal Processing and Cyber Security (iSSSC), 2020 Publication	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		

Finale Report (1.1)