HANDY WEATHER STATION

A project report submitted in partial fulfillment of the requirements

for the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

Submitted by

Pranathi Mandava	[19L61A0528]
B.Himja Ramani	[19L61A0529]
U.Lavanya	[19L61A0534]
U.Sai Jagadeesh	[19L61A0539]
K.Prasanth	[19L61A0532]
V.Akhil	[17L61A0507]

Under the esteemed Guidance of

Dr.CH. PAVAN SATISH Ph.D

Professor

Department of CSE

Department of Computer Science and Engineering

CHAITANYA ENGINEERING COLLEGE

Accredited by NBA, Approved by A.I.C.T.E.

(Affiliated to Jawaharlal Nehru Technological University, Kakinada, A.P)

KOMMADI VILLAGE, VISAKHAPATNAM-530048

2019– 2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CHAITANYA ENGINEERING COLLEGE

(Affiliated to Jawaharlal Nehru Technological University, Kakinada, A.P)

CERTIFICATE

This is to certify the project report entitled **"HANDY WEATHER STATION"** is original work done by "Pranathi.Mandava(19L61A0528), B.HimjaRamani(19L61A0529), U.Lavanya(19L61A0534), U.SaiJagadeesh(19L61A0539), K.Prasanth(19L61A0532), V.Akhil(17L61A0507) this team in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science** during the academic year 2022-2023. The result presented in this thesis has been verified and are found to be satisfactory.

PROJECT GUIDE	**HEAD OF THE DEPARTMENT**
Dr.CH.PAVAN SATISH Ph.D	**Dr.CH.PAVAN SATISH Ph.D**
HOD & Professor	**HOD & Professor**
Dept. Of Computer Science Engineering	**Dept. Of Computer Science Engineering**

EXTERNAL EXAMINER

DECLARATION

I hereby declare that this report for the main project entitled as **"HANDY WEATHER STATION"** has been carried out by us and submitted in partial fulfillment of the curriculum for the award of degree of Bachelor of Technology in **COMPUTER SCIENCE AND ENGINEERING,** Chaitanya Engineering College, Visakhapatnam.

The various contents incorporated in this document have not been submitted for the award of any other degree of any other institution or university.

PROJECT ASSOCIATES

Pranathi. Mandava	**[19L61A0528]**
B.Himja Ramani	**[19L61A0529]**
U.Lavanya	**[19L61A0534]**
U.Sai Jagadeesh	**[19L61A0539]**
K.Prasanth	**[19L61A0532]**
V. Akhil	**[17L61A0507]**

ACKNOWLEDGEMENTS

I wish to express our deep sense of gratitude to our project guide, **Dr.CH.PAVAN SATISH,Ph.D,** in the department of Computer Science and Engineering for his valuable advice and technical support from the beginning of our project. He has always been a source of inspiration for us and has given us wholehearted help and deep routed support. I sincerely thank **Dr.CH.PAVAN SATISH,Ph.D ,** Head of Department for providing access to all required facilities in department of Computer Science and Engineering and for constantly motivating us throughout the project. I would like to extend our sincere thanks to all our faculty, on-teaching staff, classmates in department and our family members for helping us to make this project successful. Take the privilege to thank our principal **Dr. K. SURESH, Ph.D.,** who has made the atmosphere so early to work, I remember him for his valuable encouragement.

Pranathi. Mandava	**[19L61A0528]**
B.Himja Ramani	**[19L61A0529]**
U.Lavanya	**[19L61A0534]**
U.Sai Jagadeesh	**[19L61A0539]**
K.Prasanth	**[19L61A0532]**
V. Akhil	**[17L61A0507]**

CONTENTS

CONTENTS

ABSTRACT

ABSTRACT

A Handy Weather Station is a device that measures various meteorological parameters such as temperature, humidity, pressure, wind speed. These measurements can be displayed on a digital screen or sent wirelessly to a computer or smartphone for analysis and forecasting. A solar-based Handy Weather Station is a type of weather station that uses solar energy to power its sensors and electronic components. This WILL BE A BENEFICIARY for the station to be placed in remote locations where access to electricity may be limited. It's important to note that the accuracy and reliability of a solar-based Handy Weather Station will depend on the quality of the sensors and electronic components used as well as the solar panel's ability to provide sufficient power to the station.

Objectives:

- Accurate weather report
- Maintenance of weather records
- Portable device and no need of external power source

INTRODUCTION

1. INTRODUCTION

Weather forecasting is done using predicting the weather and values obtained from sensors or instruments. We human use an approach of algorithms having certain or no input and valid output. Considering there is nothing random in nature and everything around us follows a particular pattern. On the basis of these weather forecasting patterns people can take precautions on even harsh weather conditions. The wireless arduino weather station has a capability of working on low power. Hence it is not much dependent on power source. The device is also made of low cost items and around ±1 unit error accuracy. As an application, a normal person can place this device at various places like in his backyard garden and remote areas for accurate weather report such as temperature, humidity, pressure and wind speed the report is transferred from the device placed place to anywhere through a message from the GSM module at even intervals of time period and even emergency situations there will be a call feature too.

1.1 Project Purpose

The objective of **Handy Weather Station** is to provide accurate weather report at a particular location where the device is placed and the report is transmitted to various persons though a text message using GSM module and if there exit an emergency situation there will be a call to alert. This type of Handy Weather Station is mainly used in factories, powerplants, research areas and even for some agriculture fields also.

The Handy Weather Station provides the temperature, humidity, pressure and wind speed through a message and also displayed on the lcd screen this will reduce man power in industrial areas and it is cost effective too and it is portable and works in robust conditions.

1.2 Project Scope

Without a **Handy Weather Station** there will be no accurate weather condition of a particular place as we even get the data though weather app in mobiles and in news reports but they are not accurate as they are weather stations located far away from the location most of the weather report is an assumption and the weather stations are large in size and occupies lots of space the above problems are solved by the proposed model.

1.3 Project Overview

The overview of this project is to provide accurate results and make the device portable that can be carried any where and to provide with an internal power source without any external power consumptions.

1.4 Features

The features provided by this application are:

Portability

Durability

Source of power

Accessability

1.5 Existing System

- It is a device or station that sense or study the weather/atmospheric conditions and helps in the process of weather forecasting and weather estimation. The device looks like the fig given below.

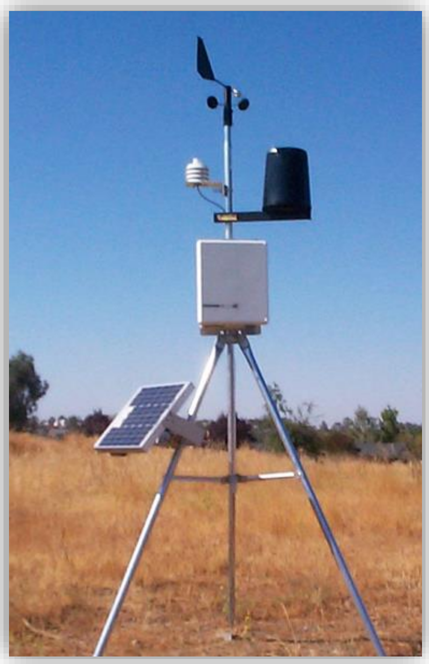


Fig1: The weather station model

- A ‘Handy Weather Station’ is also same as regular weather station but when we compare we face few pros and cons between them such as PORTABILITY, ACCESSIBILITY, etc. an example is shown in the picture given below.

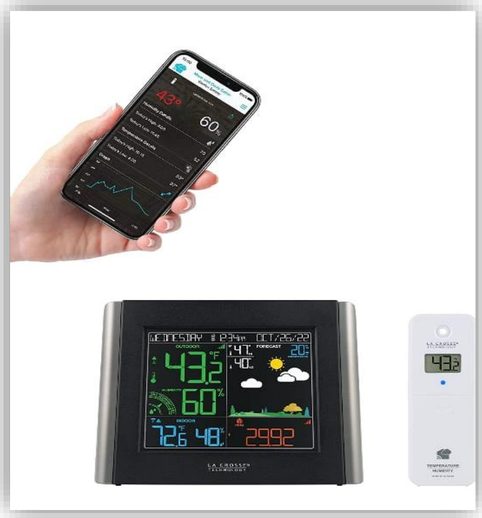


Fig 2:The existing model of handy weather station

Architecture of the Existing system

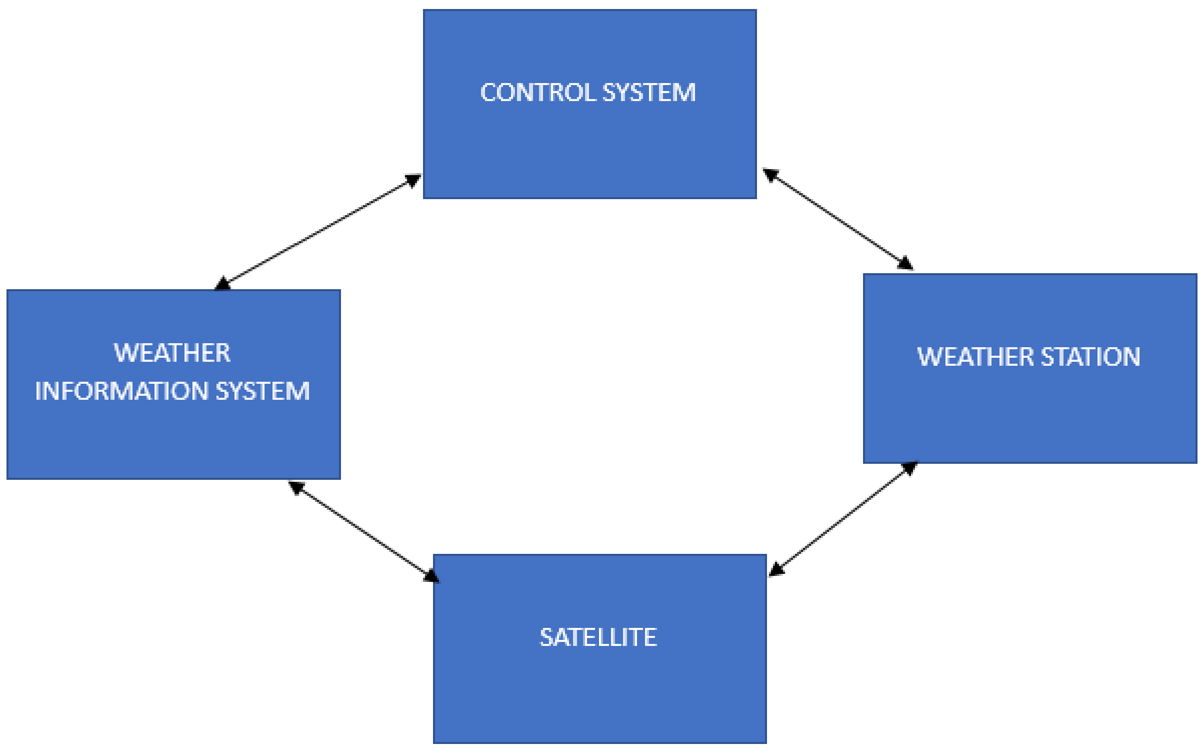


Fig 3:Existing model Architecture

A handy weather station typically consists of several components that work together to provide accurate and up-to-date weather information. Some of the common components include:

1. Sensors: These are the devices that measure various weather parameters such as temperature, humidity, barometric pressure, wind speed and direction, rainfall, and UV index. The sensors can be either analog or digital, and they transmit data wirelessly to the central processing unit.
2. Central Processing Unit (CPU): This is the brain of the weather station that collects, analyzes, and stores data from the sensors. The CPU may also have a display screen that shows the current weather conditions and other information.
3. Power Source: The weather station requires a power source, which is supplied externally.

4. Communication Interface: The weather station may have a wireless communication interface, such as Wi-Fi or Bluetooth, that allows it to connect to the internet and transmit data to a smartphone or computer.
5. Software: The weather station software is responsible for interpreting the data from the sensors, displaying the current weather conditions, and providing alerts and notifications for severe weather events.

Overall, the existing architecture of a handy weather station typically consists of sensors, a central processing unit, power source, communication interface, and software.

Disadvantages

Durability -

In robust environment conditions the system/device fails.

Accessibility -

In the aspect of information transmission many designs are with wi-fi connection oriented so in few areas wi-fi is not possible.

1.6 Proposed System

A Handy Weather Station is a device that measures various weather parameters such as temperature, humidity, pressure, wind speed using respective sensors.We mainly concentrated on the drawbacks of existing system and designed with This Device is having capability of detecting fire, smoke, gas sensing feature for the safety All the captured or measured data is given by device as output. The output is also displayed on LCD display module of device and also through message to respective mobile number using GSM module or through Wi-Fi module if available.This is a Solar based weather station the solar energy is charge the lithium-ion rechargeable batteries so that the device can stand by with out any external power source.

Architecture of the proposed system

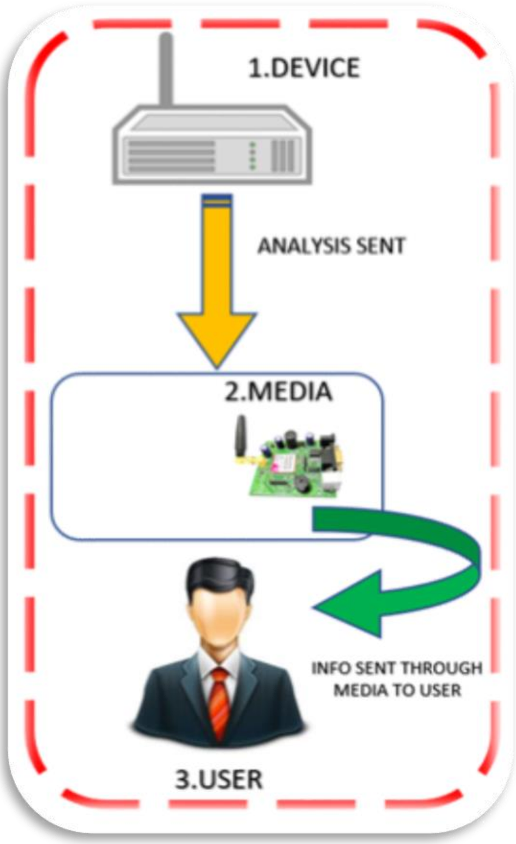


Fig 4:Proposed model architecture

A handy weather station architecture developed using Arduino Uno, GSM, DHT11, anemometer, BMP280, and a solar panel for power supply would consist of the following components:

1. Arduino Uno: This microcontroller board is the heart of the weather station. It is responsible for collecting data from the sensors and sending it to the GSM module for transmission.

2. GSM module: This component is used to transmit the collected data to a remote server or user's mobile phone. It communicates with the Arduino Uno using serial communication.

3. DHT11 sensor: This sensor measures temperature and humidity in the environment. It is connected to the Arduino Uno through a digital pin.

4. Anemometer: This component measures wind speed and direction. It is connected to the Arduino Uno through an analog pin.

5. BMP280 sensor: This sensor measures atmospheric pressure and temperature. It is connected to the Arduino Uno through the I2C interface.

6. Solar panel: This component is used to provide power to the weather station. It converts sunlight into electrical energy, which is stored in a battery for use by the system.

The architecture would include a software program that runs on the Arduino Uno, which collects data from the sensors, formats it into a message, and sends it to the GSM module for transmission. The program would also manage the power supply, using the solar panel to charge the battery and ensure that the system has sufficient power to operate.

Users can receive the data transmitted by the weather station on their mobile phone or on a remote server. The data can be used for a variety of purposes, such as monitoring weather conditions for agriculture, construction, or outdoor activities. The handy weather station architecture provides a low-cost and flexible solution for collecting and transmitting weather data.

1.7 OBJECTIVE

The objective of a handy weather station using GSM and Arduino is to provide an effective solution for remote weather monitoring. The system is designed to collect various weather-related data, such as temperature, humidity, wind speed, and rainfall, using various sensors. The data is then transmitted to a central monitoring station using the GSM network.

With this system, users can monitor weather conditions in remote locations and take necessary actions to protect their property and crops. For example, farmers can use this system to monitor weather conditions in their fields and adjust irrigation schedules based on real-time weather data. Similarly, emergency response teams can use this system to monitor weather conditions in disaster-prone areas and take necessary actions to prevent damage and save lives.

Overall, the objective of a handy weather station using GSM and Arduino is to provide an easy-to-use, affordable, and reliable solution for remote weather monitoring that can help people make informed decisions and take proactive measures to protect themselves and their assets.

2.SYSTEM REQUIREMENTS

2.1 Hardware requirements

- **GSM**

Fig 5

SIM900A is an ultra compact and reliable wireless module. This is a complete GSM/GPRS module in a SMT type and designed with a very powerful single-chip processor integrating AMR926EJ-S core, allowing you to benefit from small dimensions and cost-effective solutions. Specification. Dual-Band 900/ 1800 MHz.

- **ARDUINO UNO**

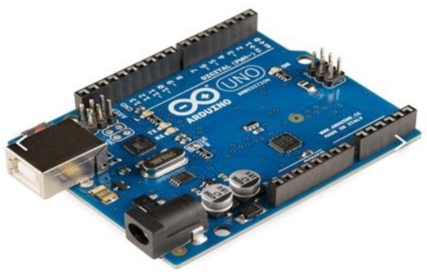


Fig 6

Arduino UNO is a low-cost, flexible, and easy-to-use programmable open-source microcontroller board that can be integrated into a variety of electronic projects. This board can be interfaced with other Arduino boards, Arduino shields, Raspberry Pi boards and can control relays, LEDs, servos, and motors as an output.

- **LCD**

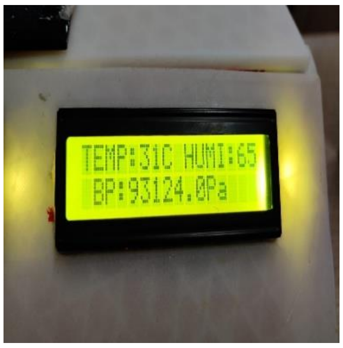


Fig 7

A 16x2 LCD means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. The 16 x 2 intelligent alphanumeric dot matrix display is capable of displaying 224 different characters and symbols. This LCD has two registers, namely, Command and Data.

- **DHT 11**

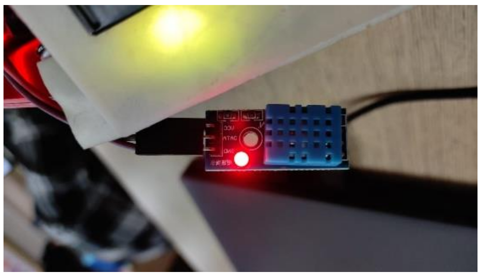


Fig 8

The DHT11 is a basic, ultra low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air and spits out a digital signal on the data pin (no analog input pins needed).

Anemometer

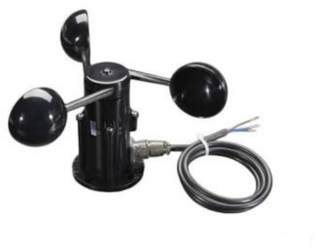


Fig 9

An anemometer is an instrument that measures the air velocity. The main applications of wind sensor are in the field of meteorology and industrial applications such as in wind turbine business or high-rise crane safety monitoring.[1]

- **BMP280**

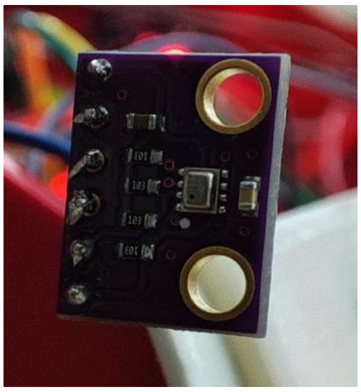


Fig 10

BMP280 is an absolute barometric pressure sensor especially designed for mobile applications. The sensor module is housed in an extremely compact package. Its small dimensions and its low power consumption allow for the implementation in battery driven devices such as mobile phones, GPS modules or watches.[4]

2.2 Software requirements

- ARDUINO IDE
- It is common Arduino web editor for all Arduino boards
- It is a integral development program/programming.
- The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.

3.SYSTEM ANALYSIS

3.1Data Flow Diagram

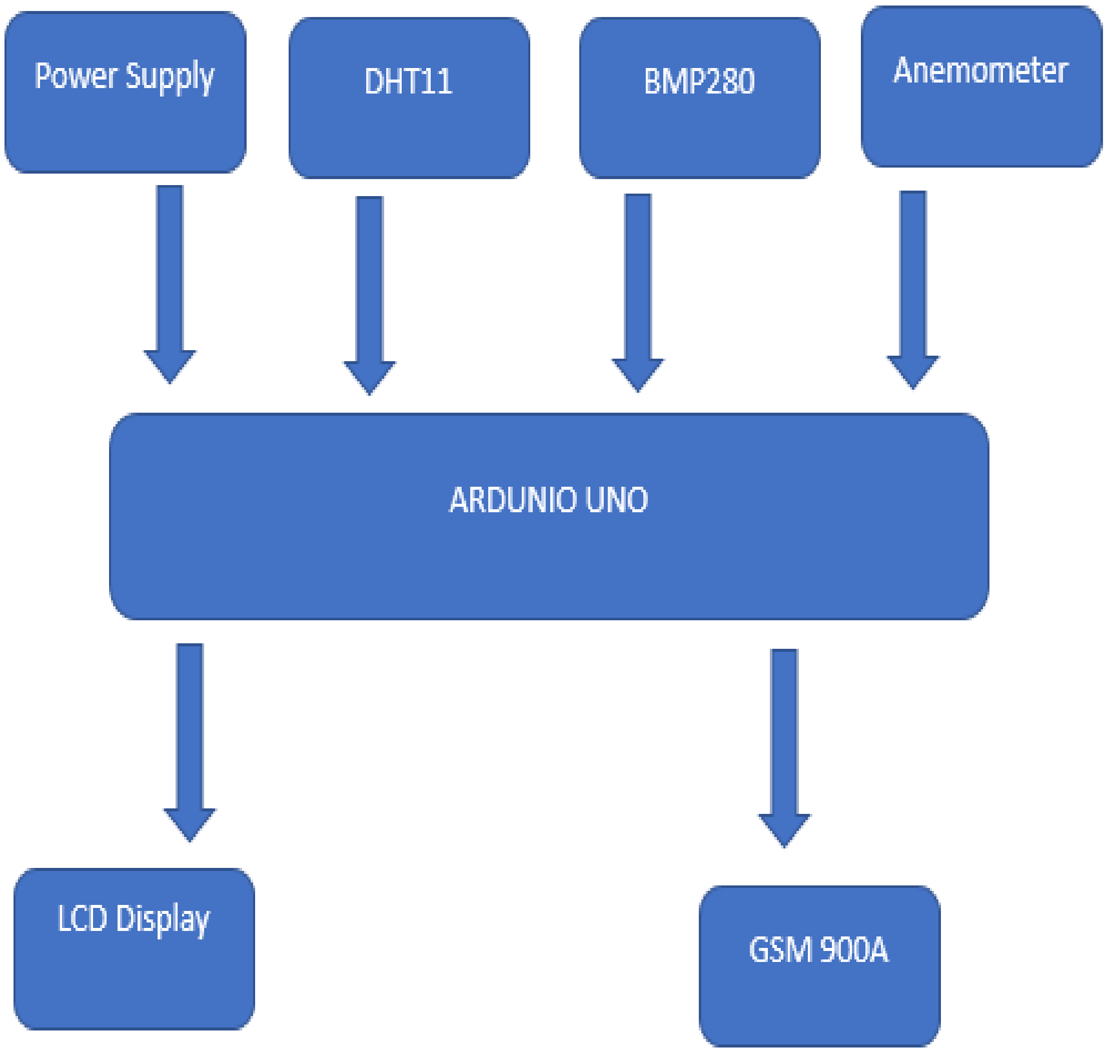


Fig 11:It describes how the model works

A handy weather station architecture developed using Arduino Uno, GSM, DHT11, anemometer, BMP280, and a solar panel for power supply would consist of the following components:[3]

1. Arduino Uno: This microcontroller board is the heart of the weather station. It is responsible for collecting data from the sensors and sending it to the GSM module for transmission.

2. GSM module: This component is used to transmit the collected data to a remote server or user's mobile phone. It communicates with the Arduino Uno using serial communication.

3. DHT11 sensor: This sensor measures temperature and humidity in the environment. It is connected to the Arduino Uno through a digital pin.

4. Anemometer: This component measures wind speed and direction. It is connected to the Arduino Uno through an analog pin.

5. BMP280 sensor: This sensor measures atmospheric pressure and temperature. It is connected to the Arduino Uno through the I2C interface.

6. Solar panel: This component is used to provide power to the weather station. It converts sunlight into electrical energy, which is stored in a battery for use by the system.

The architecture would include a software program that runs on the Arduino Uno, which collects data from the sensors, formats it into a message, and sends it to the GSM module for transmission. The program would also manage the power supply, using the solar panel to charge the battery and ensure that the system has sufficient power to operate.

Users can receive the data transmitted by the weather station on their mobile phone or on a remote server. The data can be used for a variety of purposes, such as monitoring weather conditions for agriculture, construction, or outdoor activities. The handy weather station architecture provides a low-cost and flexible solution for collecting and transmitting weather data.

3.2 System Architechture

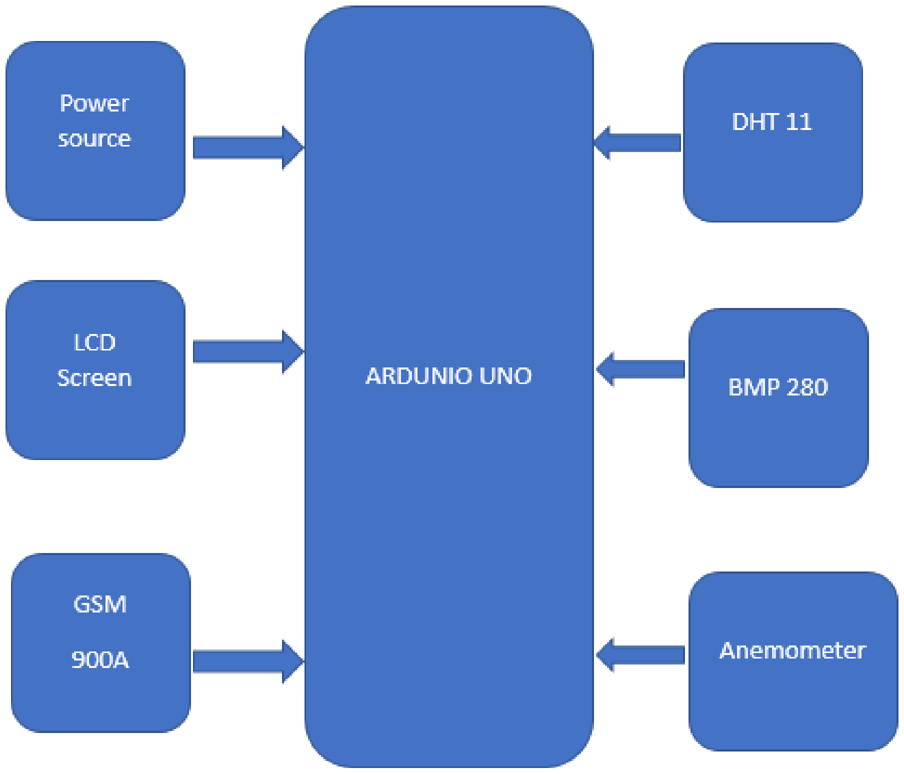


Fig:12:System Architecture

3.3 Module analysis

Sensor module:

This module includes various weather sensors such as temperature, humidity, air pressure, and wind speed sensors. The sensors are connected to the Arduino board and read the weather data which is then stored in the Arduino memory.

Arduino Board:

The Arduino board acts as the main controller for the weather station. It is responsible for interfacing with the weather sensors, reading the data, processing it, and communicating it to the

GSM module for transmission. The board is programmed using the Arduino IDE and the C++ programming language.

GSM Module:

This module is responsible for sending and receiving data to and from the internet. The GSM module is connected to the Arduino board and uses a SIM card to establish a data connection to the internet. The module sends the weather data to a remote server using HTTP or MQTT protocol.

3.4 Feasibility Analysis

The feasibility analysis suggests that the handy weather station using GSM and Arduino is a practical and viable solution for remote weather monitoring, with several advantages such as affordability, flexibility, and ease of use.

Economical Feasibility

- The system is relatively low-cost compared to other weather monitoring solutions, making it affordable for individuals and organizations with limited budgets.
- The maintenance cost of the system is low as it uses off-the-shelf components that are easily replaceable.

Technical Feasibility

- The system utilizes readily available Arduino microcontroller boards and GSM modules, making it easy to source components and build the system.
- The weather sensors used in the system are widely available and can provide accurate and reliable data.
- The data transmission process through GSM technology is reliable and has a wide coverage area, making it possible to transmit data from remote locations.

.

Social Feasibility

- The use of GSM technology may be subject to local regulations, and appropriate permissions may be required to use the system in certain locations.
- The data collected by the system may be subject to data protection regulations and should be handled accordingly.

4. UML DIAGRAMS

UML

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process.

The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.

UML (Unified Modeling Language) DESIGN DIAGRAMS

UML stands for Unified Modeling Language. UML is a language for specifying, visualizing and documenting the system. This is the step while developing any product after analysis. The goal from this is to produce a model of the entities involved in the project which later need to build. The representation of the entities involved in the project which later need to be built.

- The unified modeling language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic, semantic and pragmatic rules
- A UML system is represented using five different views that describe the system from distinctly perspective.

Each view is defined by a set of diagrams, which is as follows.

- Use-case view
- Logical view
- Implementation view
- Process view
- Deployment view

Use-case view

A view showing the functionality of the system as perceived by external actors. An actor interacts with system. The actor can be a user or another system. The use-case view is used by customers, designers, developers and testers.

It is described in use-case diagrams, sometimes with support from activity diagrams. The desired usage of the system is described as a number of use cases in the use-case view, where a use case is a generic description of a function requested.

Logical view

A view showing how the functionality is designed inside the system, in terms of the system's static structure and dynamic behavior. It is mainly for the designers and developers. It describes both static structure (classes, objects and relationships) and the dynamic collaborations that occur when the objects send messages to each other to provide a given function.

Properties such as persistence and concurrency are also defined, as well as the interfaces and the internal structure of classes. The static structure is described in class and object diagrams. The dynamic modeling is described in state machines and interaction and activity diagrams.

Implementation view

The implementation view describes the main modules and their dependencies. It is mainly for developers and consists of the main software artifacts. The artifacts include different types of code modules shown with their structure and dependencies.

Process view-

A view showing main elements in the system related to process performance. This view includes scalability, throughput, and basic time performance and can touch on some very complex calculations for advanced systems.

The view consists of dynamic diagrams(state machines, interaction and activity diagrams) and implementation diagrams(interaction and deployment diagrams).

Deployment view

Finally the deployment view shows the physical deployment of the system such as the computers and devices (nodes) and how they connect to each other. The various execution environments within the processors can be specified as well.

The deployment view is used by the developers, integrators and testers. It is represented by the deployment diagram.

Diagrams

The diagrams contain the graphical elements arranged to illustrate a particular part or aspect of the system. A system model typically has several diagrams of varying types, depending on the goal for the model. Software design is a process that gradually changes as various new, better and more Complex methods with a broader understanding of the whole problem in general come in to existence. There are various kinds of diagrams used in software design.

Mainly These Are As Follows:

- Use case diagrams
- Class diagram
- Object diagram

- Sequence diagrams
- Collaboration diagrams
- Activity diagram
- State chart diagram
- Component diagram
- Deployment diagram.

USE CASE DIAGRAM

Use Case diagrams identify the functionality provided by the system (use cases), the users who interact with the system (actors), and the association between the users and the functionality. Use Cases are used in the Analysis phase of software development to articulate the high-level requirements of the system**.**

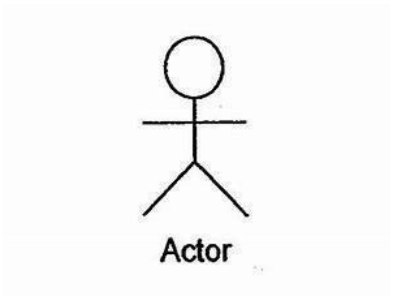
The primary goals of Use Case diagrams include:

- Providing a high-level view of what the system does
- Identifying the users ("actors") of the system
- Determining areas needing human-computer interfaces Use Cases extend beyond pictorial diagrams. In fact, text-based use case descriptions are often used to supplement diagrams and explore use case functionality in more detail.

Graphical Notation:

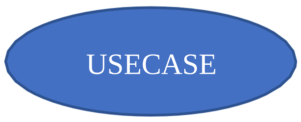
The basic components of Use Case diagrams are the Actor, the Use Case, and the Association.

Actor:



An Actor, as mentioned, is a user of the system, and is depicted using a stick figure. The role of the user is written beneath the icon. Actors are not limited to humans. If a system communicates with another application, and expects input or delivers output, then that application can also be considered an actor.

Use Case

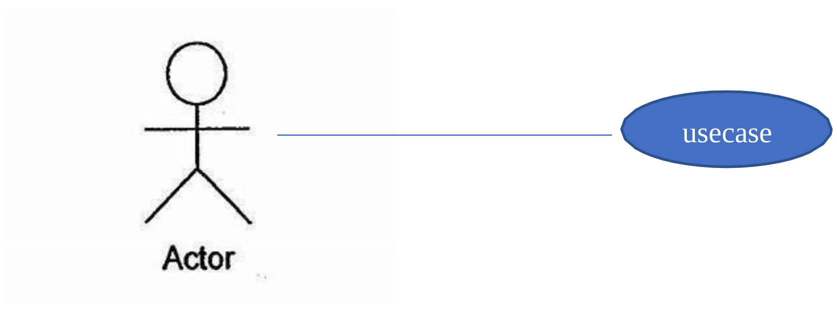


A Use Case is functionality provided by the system, typically described as verb object (ex. Register Car, Delete User). Use Cases are depicted with an ellipse. The name of the use case is written within the ellipse.

Association :

Associations are used to link Actors with Use Cases and indicate that an Actor participates in the Use Case in some form. Associations are depicted by a line connecting the Actor and the Use Case.

The following image shows how these three basic elements work together to form a use case diagram.



Text Notation

Behind each Use Case is a series of actions to achieve the proper functionality, as well as alternate paths for instances where validation fails, or errors occur. These actions can be further defined in a Use Case description. Because this is not addressed in UML 1.4, there no standards for Use Case descriptions. However, there are some common templates you follow, and whole books on the subject writing of Use Case descriptions.

Common methods of writing Use Case descriptions include:

- Write a paragraph describing the sequence of activities in the Use Case
- List two columns, with the activities of the actor and the responses by the system
- Use a template (such as those from the Rational Unified Process or Alistair Cockburn's book, Writing Effective Use Cases) identifying actors, preconditions, post conditions, main success scenarios, and extensions. Remember, the goal of the process is to be able to communicate the requirements of the system, so use whichever method is best for your team and your organization.

SYSTEM USECASE DIAGRAM:

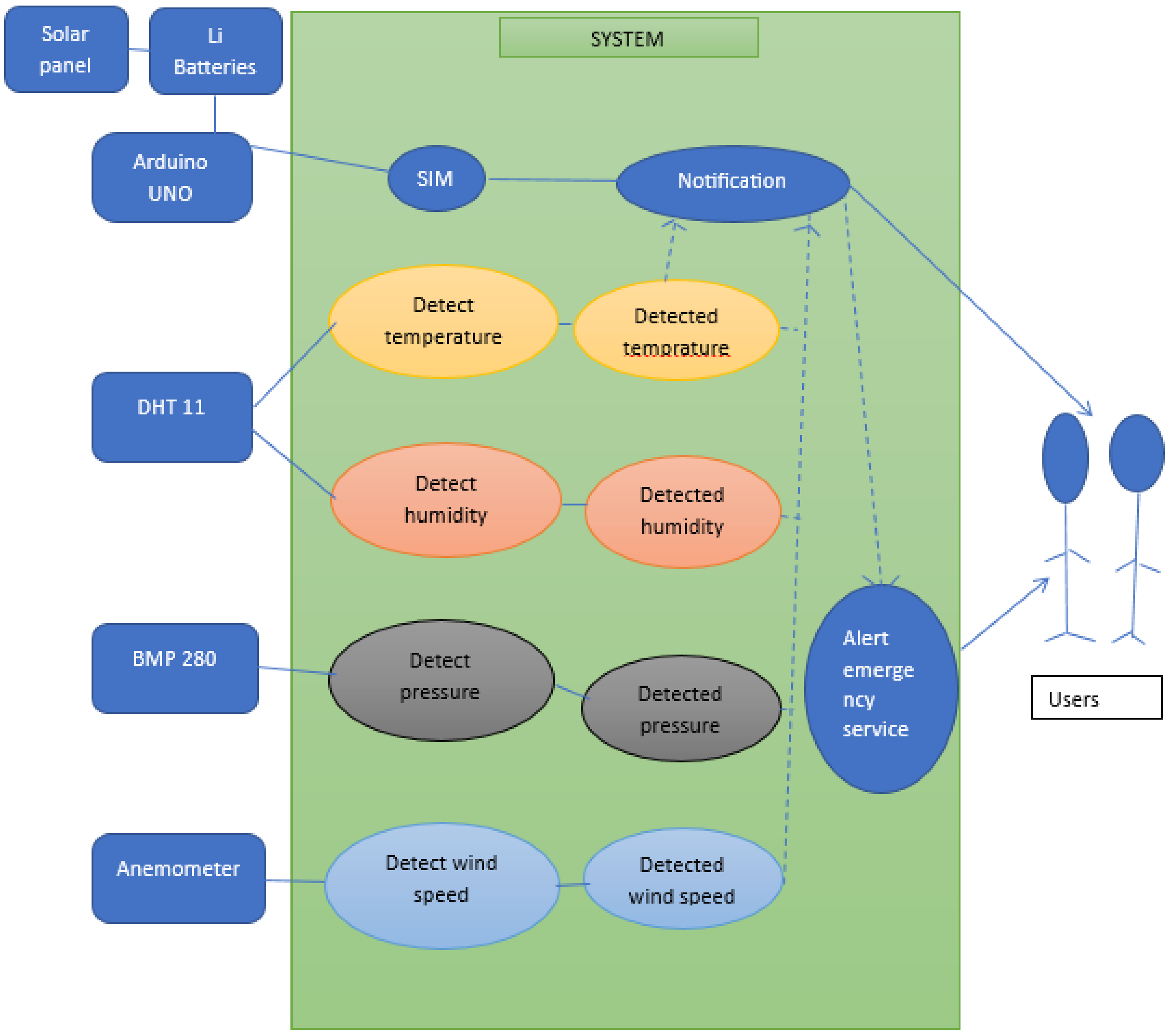


Fig:13:Proposed system use case

Sure, here's a description of the use case diagram for the handy weather station system with all the sensors mentioned above:

1. Collect Weather Data: This use case represents the system's ability to collect weather data from various sensors, including BMP280 for temperature and pressure, DHT11 for temperature and humidity, and Anemometer for wind speed.

2. Store Data: This use case represents the system's ability to store the collected weather data in the data storage system.

3. Generate Weather Reports: This use case represents the system's ability to perform data analysis on the collected weather data and generate weather reports based on the analysis.

4. Send Reports: This use case represents the system's ability to send the generated weather reports to the user through the GSM module.

5. Update Settings: This use case represents the system's ability to receive and update the settings, such as the reporting interval and reporting format, from the user.

6. Check Battery Levels: This use case represents the system's ability to monitor the battery levels of the system's power supply, which includes solar panels and rechargeable batteries.

7. Manage Power: This use case represents the system's ability to manage the power supply by controlling the power usage of the components based on the battery levels and available solar power.

8. Send Low Battery Alert: This use case represents the system's ability to send a low battery alert to the user when the battery levels are critically low.

Overall, the use case diagram represents the core functionalities of the weather station system, which include collecting weather data, generating weather reports, and sending the reports to the user through the GSM module. The diagram also highlights the system's power management capabilities, which are critical for ensuring continuous operation of the system, especially in remote and off-grid locations. Finally, the diagram shows the system's ability to receive and update the settings from the user, making the system more customizable and user-friendly.

5. SYSTEM DESIGN

5.1 Circuit diagrams

Software Design The software is designed in other to support the effectiveness of the hardware device. The complex and intricate operating routine of the software is achieved by writing sections as demonstrated in the program flowchart in figure below. Arduino IDE (integrated developer environment) was used throughout the software building stage. The software was written in C language and was developed in sections for easy debugging and then later integrated. The integrated program was built and verified and the hex file was generated which was then loaded into the microcontroller. Simulations with proteus After a successful completion of stages one and two, the design was virtually implemented using proteus vsm software to detect likely logical errors in the program and improper operations in the hardware design. During and after simulations on proteus, faults were detected and adjustments were made appropriately.[7]

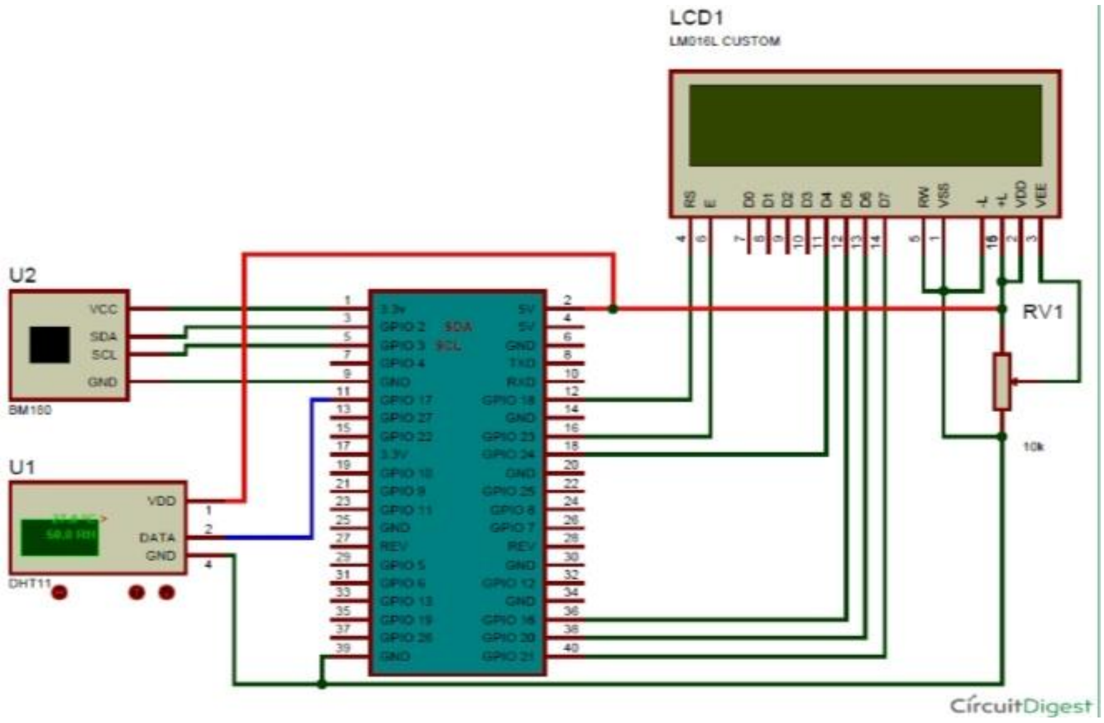


Fig:14:Circuit diagram with connections

Arduino Uno

The Arduino board is a microcontroller-based board that can be programmed to control various electronic devices. It consists of a number of components that work together to provide a functional platform for building projects. Here's a brief overview of the key components on an Arduino board:

1. Microcontroller: This is the heart of the Arduino board. It's responsible for executing the code that's uploaded to the board, and for controlling the various input and output pins. The microcontroller used in most Arduino boards is an Atmel AVR microcontroller.

2. Crystal oscillator: This component provides a clock signal to the microcontroller, which is necessary for it to operate at a consistent speed. Most Arduino boards use a 16 MHz crystal oscillator.

3. Power supply: The Arduino board can be powered by a variety of sources, including USB, a DC power jack, or a battery. The board contains voltage regulators that ensure the correct voltage is supplied to the various components.

4. USB interface: This allows the Arduino board to communicate with a computer and upload new programs.

5. Digital input/output pins: These pins can be used to control or read digital signals, such as turning an LED on or off or reading a switch state.

6. Analog input pins: These pins can be used to read analog signals, such as the output of a sensor or the position of a potentiometer.

7. PWM pins: Pulse-width modulation (PWM) allows for the simulation of an analog signal using a digital signal. These pins can be used to control the brightness of an LED or the speed of a motor, for example.

8. ICSP header: This header allows for in-circuit serial programming of the microcontroller.

Overall, the Arduino board provides a versatile platform for building a wide range of electronic projects, from simple LED blinkers to complex robotic systems.

Connections

The connections for each component are as follows:[2]

- BMP280: Connect the VCC pin to the 3.3V output of the Arduino, GND pin to the GND pin of the Arduino, SDA pin to the A4 pin of the Arduino, and SCL pin to the A5 pin of the Arduino.
- DHT11: Connect the VCC pin to the 5V output of the Arduino, GND pin to the GND pin of the Arduino, and DATA pin to the digital pin 8 of the Arduino.
- Anemometer: Connect the VCC pin to the 5V output of the Arduino, GND pin to the GND pin of the Arduino, and SIGNAL pin to the digital pin 0 of the Arduino.

- Solar Panel: Connect the positive terminal of the solar panel to the VIN pin of the Arduino, and the negative terminal of the solar panel to the GND pin of the Arduino.
- Rechargeable Batteries: Connect the positive terminal of the rechargeable batteries to the VIN pin of the Arduino, and the negative terminal of the rechargeable batteries to the GND pin of the Arduino.
- GSM Module: Connect the VCC pin to the 5V output of the Arduino, GND pin to the GND pin of the Arduino, TXD pin to the digital pin 2 of the Arduino, and RXD pin to the digital pin 3 of the Arduino.
- LCD Display: Connect the VCC pin to the 5V output of the Arduino, GND pin to the GND pin of the Arduino, SDA pin to the A4 pin of the Arduino, SCL pin to the A5 pin of the Arduino, and R/W pin to the GND pin of the Arduino.

In addition to the above connections, you also need to connect a 10K resistor between the VCC and DATA pins of the DHT11 sensor, and a 10K resistor between the SIGNAL pin and GND pin of the Anemometer sensor. Please note that the actual circuit diagram and connections may vary based on the specific model and version of the sensors and modules used in the system. It's important to refer to the datasheets and manuals of the components for accurate and safe connections.

6. SYSTEM IMPLEMENTATION

Source Code

```
//smoke sensor,gsm, lcd, bmp, dht11
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
int smokeA0=A0;
int buzzer =10;
float sensorValue;
#include <DFRobot_DHT11.h>
DFRobot_DHT11 DHT;
#define DHT11_PIN 8
#include <SoftwareSerial.h>
SoftwareSerial mySerial(2,3);
LiquidCrystal_I2C lcd(0x27,16,2);  // set the LCD address to 0x27 for a 16 chars and 2 line display
LiquidCrystal_I2C lcd2(0x26, 16,2);
#include <Wire.h>
#include "i2c.h"
#include "i2c_BMP280.h"
BMP280 bmp280;

void setup()
{
   mySerial.begin(9600);
  Serial.begin(9600);
  lcd.init();                      // initialize the lcd
  lcd.backlight();
    lcd2.init();                      // initialize the lcd
  lcd2.backlight();
    pinMode(buzzer,OUTPUT);
  pinMode(smokeA0,INPUT);
    Serial.println("Gas sensor warming up!\n");
  delay(2000);
  noTone(buzzer);
   // Setting the baud rate of GSM Module
```

```
       Serial.begin(9600);

     Serial.print("Probe BMP280: ");
     if (bmp280.initialize()) Serial.println("Sensor found");
     else
     {
         Serial.println("Sensor missing");
         while (1) {}
     }

     // onetime-measure:
     bmp280.setEnabled(0);
     bmp280.triggerMeasurement();
}

void loop()
{
  lcd.setCursor(0,0);
  lcd.backlight();
   sensorValue=analogRead(smokeA0);
  if(sensorValue > 300)
  {
    Serial.print(" | Smoke detected!\n");
    tone(buzzer,1000,200);
  }
  else
  {
     Serial.print(" | Smoke  not detected!\n");

    noTone(buzzer);

  }
    DHT.read(DHT11_PIN);
  lcd.setCursor(0,0);
  lcd.backlight();
lcd.print("TEMP:");
lcd.setCursor(6,0);
int y = DHT.temperature;
lcd.print(y);
lcd.setCursor(9,0);
lcd.print("HUMI:");
 lcd.setCursor(14,0);
 int x =DHT.humidity;
lcd.println(x);
bmp280.awaitMeasurement();
  float pascal;
  bmp280.getPressure(pascal);

  bmp280.triggerMeasurement();

  Serial.print(" BP: ");
  Serial.print(pascal);
  Serial.print(" Pa");
  lcd.setCursor(0,1);
  lcd.print(" BP: ");
  lcd.setCursor(4,1);
  lcd.print(pascal);
  lcd.setCursor(11,1);
  lcd.print("Pa");


  if (Serial.available()>0)

    mySerial.println("AT+CMGF=1");    //Sets the GSM Module in Text Mode
   delay(1000);  // Delay of 1 second
   mySerial.println("AT+CMGS=\"+919949823605\"\r"); // Replace x with mobile number
   delay(1000);
   mySerial.println("Tempurature: ");// The SMS text you want to send
   mySerial.println(y);
   mySerial.println("Humidity:");
```

```
     mySerial.println(x);
     mySerial.println("Pressure:");
     mySerial.println(pascal);
     delay(100);
     mySerial.println((char)26);// ASCII code of CTRL+Z for saying the end of sms to  the module
      delay(10000);

 if (mySerial.available()>0)
   Serial.write(mySerial.read());

   float sensorValue = analogRead(A1);
  Serial.print("Analog Value =");
  Serial.println(sensorValue);

  float voltage = (sensorValue / 1024) * 5; //Arduino ADC resolution 0-1023
  Serial.print("Voltage =");
  Serial.print(voltage);
  Serial.println(" V");

  float wind_speed = mapfloat(voltage, 0.4, 2, 0, 32.4);
  float speed_mph = ((wind_speed *3600)/1609.344);
  Serial.print("Wind Speed =");
  Serial.print(wind_speed);
  Serial.println("m/s");
  Serial.print(speed_mph);
  Serial.println("mph");

  lcd2. setCursor (0, 0);
  lcd2.print ("Wind Speed");
//Here cursor is placed on second line
lcd2. setCursor (0, 1);
lcd2.print (wind_speed);
lcd2.print ("m/s");

  Serial.println(" ");
  delay(300);
}

float mapfloat(float x, float in_min, float in_max, float out_min, float out_max)
{
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
```

7.TESTING

7.1 TESTING TECHNIQUES

This section deals with the descriptions of tests performed on the various sections of the overall system and their corresponding results. To verify the correct functionality of the system, each component had to be tested individually. In achieving the effective testing of the component, the breadboard and digital multi-meter were used. Testing was performed on each of the components and section that made up the circuit so as to ensure proper and satisfactory operation of the system. The results obtained were sectioned into four to tally with the objectives. The first objective which is to convert analog data obtained from sensors and fed to one of the ADC channel of Microcontroller has been achieved. The second objective which entails wireless monitoring of the weather system in an enclosure has also been achieved through sensing and display of real time environmental emperature, relative humidity and light intensity on LCD screen as shown in figure 12 below.

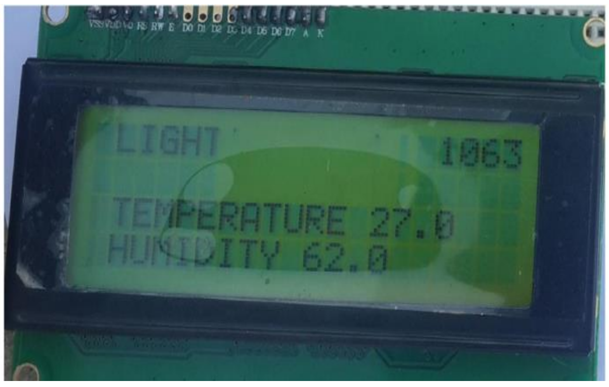
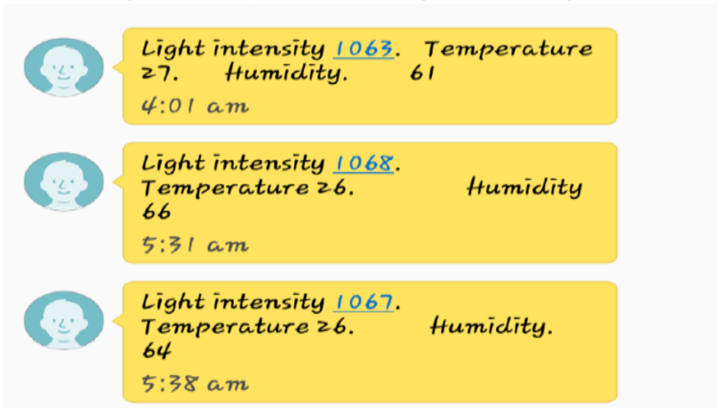


Fig:15:LCD display to check

Display of real time Light Intensity, Temperature and Humidity The obtained results which are the measured temperature, relative humidity and light intensity are then sent to the user with the help of GSM module (SIM 800) via SMS. A sample SMS is shown.


Light intensity 1063. Temperature 27. Humidity. 61
4:01 am

Light intensity 1068. Temperature 26. Humidity 66
5:31 am

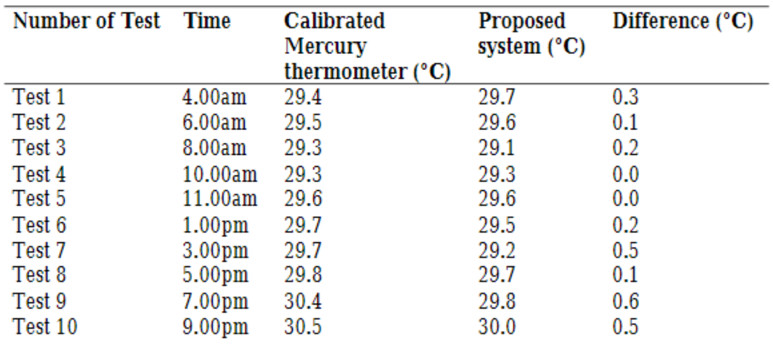
Light intensity 1067. Temperature 26. Humidity. 64
5:38 am

Fig:16:Message example

7.2 Test Cases

After the accuracy of the proposed system has been tested through extensive experiments. The results obtained are summarized in tables

Comparison of temperature measurments:



Number of Test	Time	Calibrated Mercury thermometer (°C)	Proposed system (°C)	Difference (°C)
Test 1	4.00am	29.4	29.7	0.3
Test 2	6.00am	29.5	29.6	0.1
Test 3	8.00am	29.3	29.1	0.2
Test 4	10.00am	29.3	29.3	0.0
Test 5	11.00am	29.6	29.6	0.0
Test 6	1.00pm	29.7	29.5	0.2
Test 7	3.00pm	29.7	29.2	0.5
Test 8	5.00pm	29.8	29.7	0.1
Test 9	7.00pm	30.4	29.8	0.6
Test 10	9.00pm	30.5	30.0	0.5

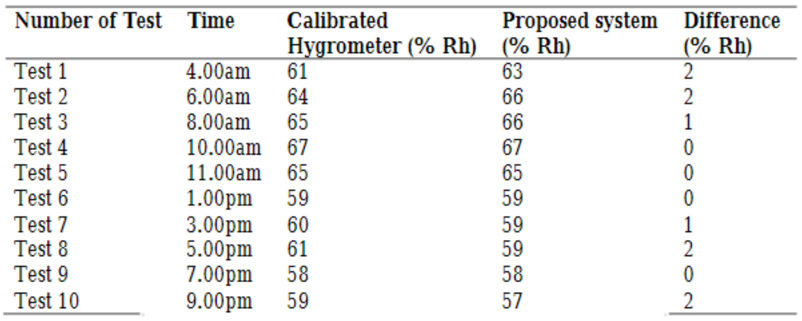
Table:1

Mercury thermometers are traditional thermometers that use a column of mercury to measure temperature. They consist of a glass tube with a small reservoir at the bottom that contains mercury. As the temperature increases, the mercury expands and rises up the tube, indicating the temperature on a scale. Mercury thermometers are typically accurate to within 1 degree Celsius.

On the other hand, the DHT11 sensor is a digital temperature and humidity sensor that uses a thermistor to measure temperature. It has a built-in microcontroller that converts the analog signal from the thermistor into a digital signal, which can be read by a microprocessor. The DHT11 sensor is typically accurate to within 2 degrees Celsius.

A temperature comparison table would show the temperature readings obtained from both thermometers side by side for a given set of conditions. The table may also include other relevant information, such as the time of measurement, the location of measurement, and any relevant notes or observations. The purpose of such a table would be to compare the accuracy and reliability of the two different types of thermometers under the same conditions.

Comparison of relative humidity:



Number of Test	Time	Calibrated Hygrometer (% Rh)	Proposed system (% Rh)	Difference (% Rh)
Test 1	4.00am	61	63	2
Test 2	6.00am	64	66	2
Test 3	8.00am	65	66	1
Test 4	10.00am	67	67	0
Test 5	11.00am	65	65	0
Test 6	1.00pm	59	59	0
Test 7	3.00pm	60	59	1
Test 8	5.00pm	61	59	2
Test 9	7.00pm	58	58	0
Test 10	9.00pm	59	57	2

Table:2

A calibrated hygrometer is a device specifically designed to measure humidity accurately. It typically uses a hair bundle or a capacitance sensor to measure the amount of moisture in the air. The hygrometer is calibrated using a reference standard, which ensures that the readings obtained are accurate and reliable. Calibrated hygrometers are typically accurate to within 1-2% relative humidity.

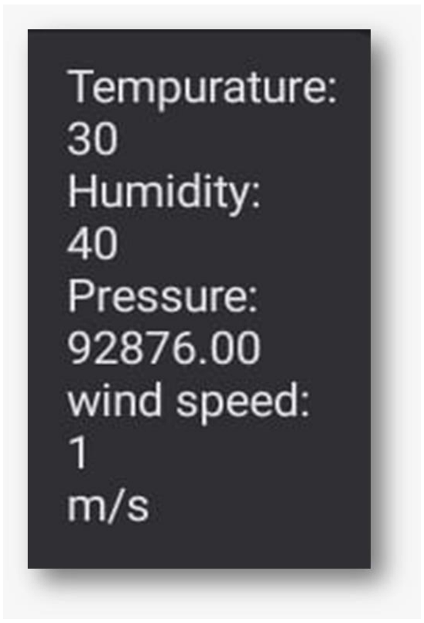
The DHT11 sensor, on the other hand, is a digital temperature and humidity sensor that uses a thermistor and a capacitive humidity sensor to measure both temperature and humidity. It has a built-in microcontroller that converts the analog signals from the sensors into digital signals, which can be read by a microprocessor. The DHT11 sensor is typically accurate to within 5% relative humidity.

A humidity comparison table would show the humidity readings obtained from both sensors side by side for a given set of conditions. The table may also include other relevant information, such as the time of measurement, the location of measurement, and any relevant notes or observations. The purpose of such a table would be to compare the accuracy and reliability of the two different types of sensors under the same conditions. It is important to note that the accuracy of the DHT11 sensor may be affected by factors such as temperature, which can impact the readings of the capacitive humidity sensor.

8. OUTPUT SCREENS

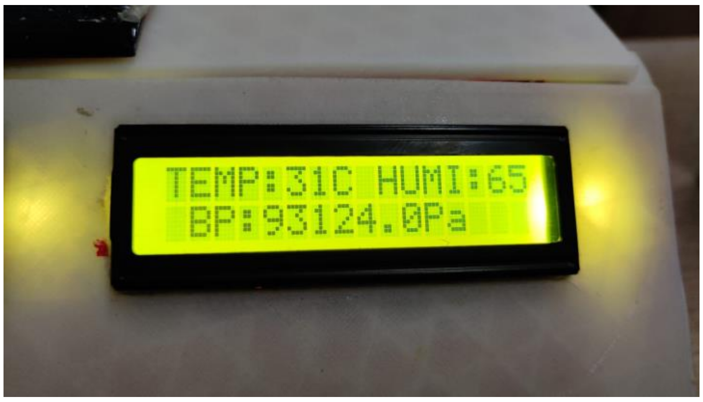
8. OUTPUT SCREENS

Temperature, humidity and pressure measurements:



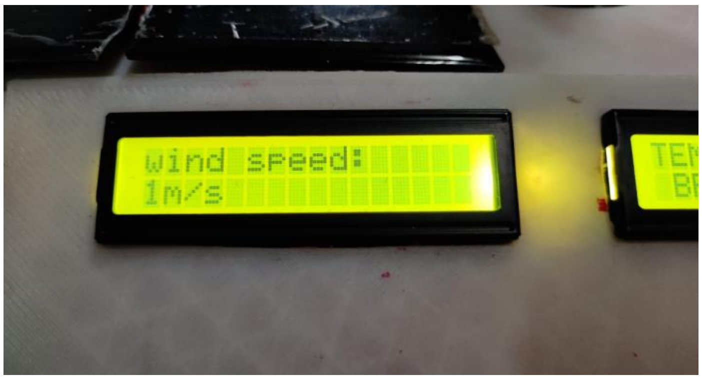
Tempurature:
30
Humidity:
40
Pressure:
92876.00
wind speed:
1
m/s

Fig:17 :GSM output display



```
TEMP:31C HUMI:65
 BP:93124.0Pa
```

Fig:18:LCD output display



```
wind speed:
1m/s
```

Fig:19:LCD output display

9.CONCLUSION

9.CONCLUSION

The developed low-cost Arduino-based weather station was effective in collecting some chosen weather element, transmitted the data, processed the data into information, stored the information, and output the information in digital format. The system is portable due to the use of the Arduino microcontroller, unlike the conventional weather station. It is recommended that the work can further be analyzed by taking readings for two or three months for two or three different seasons in order to determine optimal conditions of an environment which will help farmers and event organizers

10. FUTURE ENHANCEMENT

10. FUTURE ENHANCEMENT

There are several future enhancements that can be made to the handy weather station using GSM and Arduino to improve its functionality and performance. Some of these enhancements include:

- Cloud Connectivity: The addition of cloud connectivity would allow the system to store weather data on the cloud and provide users with easy access to the data through various devices.
- Data Visualization: The inclusion of data visualization tools would allow users to easily interpret weather data through graphs and charts.
- Wireless connectivity: Adding Wi-Fi or Bluetooth connectivity to the Handy Weather Station would allow users to access weather data remotely and receive alerts on their mobile devices when weather conditions change. This could be especially useful for outdoor enthusiasts, farmers, and other people who need to stay informed about weather conditions in real-time.
- Solar power: Adding a solar panel to the device would allow it to operate for longer periods of time without needing to be recharged or plugged in. This would be especially useful for people who use the device in remote locations or for extended periods of time.
- Data logging: Adding a data logging feature would allow the Handy Weather Station to store weather data over time, which could be useful for tracking weather patterns and trends. This could be especially useful for researchers, meteorologists, and farmers.
- Additional sensors: Adding sensors for wind speed and direction, UV radiation, and precipitation would provide even more comprehensive weather information. This could be especially useful for people who need to make decisions based on detailed weather information, such as pilots, sailors, and farmers.
- Mobile app integration: Developing a mobile app that works with the Handy Weather Station would provide users with a more user-friendly interface for accessing and analyzing weather data. The app could include features such as customizable alerts, historical data analysis, and social sharing.

11.BIBLOGRAPHY

- Bowles, G.A.J. “Corsock automatic weather station”.
- Vincent, Kim Sung Tae “Automatic wireless mobile weather station”.
- Strange ways, I.C “ a cold region automatic weather station”.

REFERENCES

[1] https://robu.in/product-tag/anemometer/

[2] https://www.hackster.io/ronfrtek/arduino-weather-station-using-bmp280-dht11-af7d6a

[3] https://iotprojectsideas.com/interface-dht11-sensor-with-arduino-and-lcd/

[4]https://quartzcomponents.com/blogs/electronics-projects/temperature-and-humidity-measurement-using-arduino-uno-and-dht11

[5] https://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-an-arduino/

[6] https://www.geeksforgeeks.org/how-to-interface-i2c-lcd-display-with-arduino/

[7] https://www.geeksforgeeks.org/how-to-interface-temperature-sensor-with-arduino/