CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Air pollution is the biggest problem of every nation, whether it is developed or developing. Health problems have been growing at faster rate especially in urban areas of developing countries where industrialization and growing number of vehicles leads to release of lot of gaseous pollutants. Harmful effects of pollution include mild allergic reactions such as irritation of the throat, eyes and nose as well as some serious problems like bronchitis, heart diseases, pneumonia, lung and aggravated asthma. According to a survey, due to air pollution 50,000 to 1,00,000 premature deaths per year occur in the US alone whereas in EU number reaches to 3,00,000 and over 30,00,000 worldwide. Various kinds of anthropogenic emissions named as primary pollutants are pumped into the atmosphere that undergoes chemical reaction and further leads to the formation of new pollutants normally called as secondary pollutants. For instance, according to the Fifth Assessment Report of the Intergovernmental panel of climate change (IPCC), nearly all climate-altering pollutants either directly or indirectly (by contributing to secondary pollutants in the atmosphere) are responsible for health problems. Almost every citizen spends 90% of their time in indoor air. Outdoor air quality of the cities of developed countries improved considerably in recent decades. In contrast to this, indoor air quality degraded during this same period because of many factors like reduced ventilation, energy conservation and the introduction to new sources and new materials that cause indoor pollution. The design of buildings for lower power consumption resulted in decrease of ventilation which further decreases the quality of air inside the building. This increases the need for indoor air quality (IAQ) monitoring due to this fact and use of new building materials, IAQ often reaches to unacceptable levels.

1.2 STATEMENT ABOUT PROBLEM

During past decades, as result of civilization and urbanization there is a huge growth in Polluting industries, open burning of refuse and leaves, massive quantities of construction waste, substantial loss of forests and vehicles (particularly diesel-driven cars) on roads that give rise to health endangering pollution. Therefore, it is necessary to regularly monitor and report the hazardous impacts from air pollution.

To monitor the quality of air, a new framework is proposed that monitors the parameters of the environment around us such as CO_2, CO, presence of smoke, alcohol, LPG, temperature and humidity with the help of GSM, Bluetooth and WSN.

1.3 LITERATURE SURVEY

Table 1: Literature Survey

Author name	Journal/Paper name	Technique used	Work Focused
Navreetinder Kaur, Rita Mahajan and Deepak Bagai	Air quality monitoring system based on arduino microcontroller	Digital signal acquisition technique	Air quality monitoring
Dr. Xirong Li	Applications of wireless sensors in monitoring Indoor Air Quality in the classroom environment	Stagnant monitoring technique	To investigate indoor air quality monitoring technologies, government regulations and policies, and best practices to improve IAQ while minimizing the adverse effect of poor IAQ in classroom environment
Vikhyat chaudhry	Arduair: Air Quality Monitoring	Conventional Carbon monoxide measurement technique	Air quality monitoring

1.4 OBJECTIVES AND SCOPE

- The basic mission of the Air Quality Planning and Standards is to preserve and improve the quality of our nation's air. The level of pollution in air can be measured by measuring the pollutants present in the air of that area.
- If the quantity of air pollutants increases in the air, it can prove harmful for humans. The substance can be solid particles, liquid droplets, or gases. A pollutant can be of natural origin or man-made.

- The need to monitor and control air pollution is because of hazardous effects on individuals or groups or whole population when exposed to these pollutants.
- The objective of designing a project that monitors and controls air pollution is to prevent the harmful effects of pollutants present in air so that a healthy surroundings can be maintained using telemetry.
- A network of IoT is used to connect it to the internet and access it globally. IoT typically utilize sensors to assist in environmental protection by monitoring air or water quality, atmospheric or soil conditions, and can even include areas like monitoring the movements of wildlife and their habitats. IOT devices typically span a large geographic area and can also be mobile.

1.5 BLOCK DIAGRAM

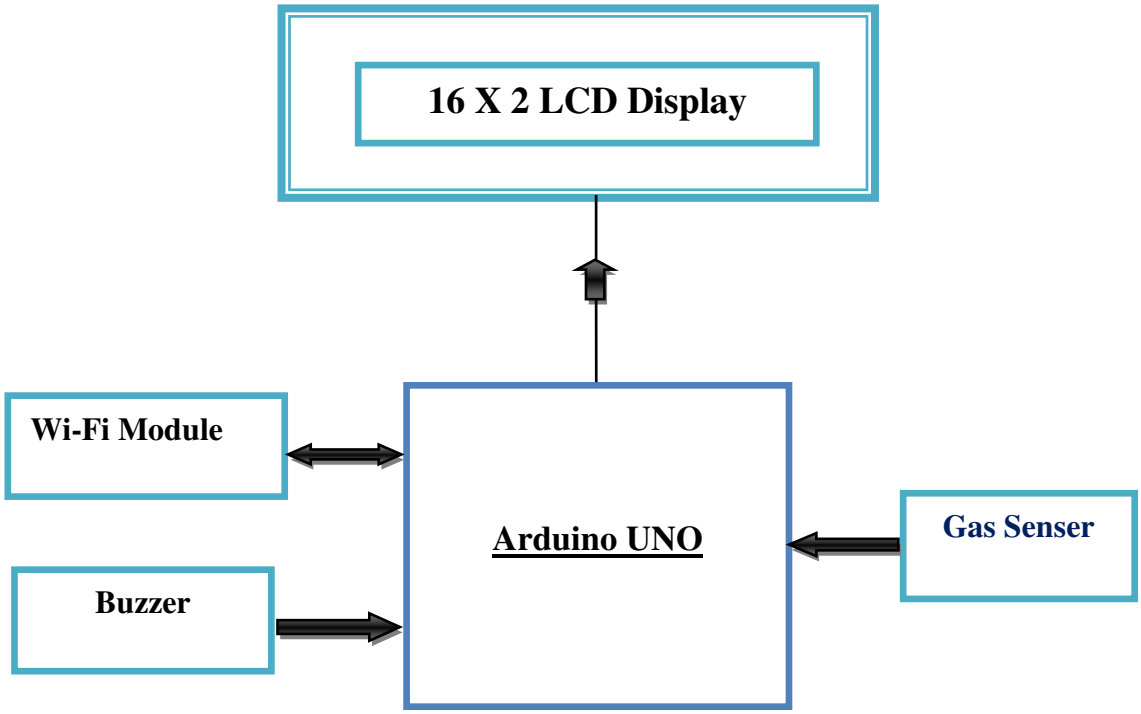


Fig. Block Diagram of Air Monitoring System by using Arduino and Gas Sensor

The Interfacing of various hardware components with arduino uno ATmega328P is shown in the Figure 1. Gas Sensor collects the quantity of pollutants in the atmospheric air, this information will be received by arduino uno. This information is then sent by the arduino to the Wi-Fi module, LCD. The Wi-Fi module then sends this message to IP address which is preprogrammed in the Arduino Uno. The LCD displays the message as per the quantity of the pollutants sensed by the gas sensor. If the quantity of pollutants

sensed by the sensor reaches above the recommended level then the arduino triggers the buzzer.

1.6 METHODOLOGY

- The MQ135 sensor can sense NH3, NO2, alcohol, Benzene, smoke, CO_2 and some other gases, so it is perfect gas sensor for our **Air Quality Monitoring Project**.
- When we will connect it to Arduino then it will sense the gases, and we will get the Pollution level in PPM (parts per million).
- MQ135 gas sensor gives the output in form of voltage levels and we need to convert it into PPM.
- So for converting the output in PPM, here we have used a library for MQ135 sensor.
- Safe level of air quality is 350 PPM and it should not exceed 1000 PPM.
- When it exceeds the limit of 1000 PPM, then it starts cause Headaches, sleepiness and stagnant, stale, stuffy air and if exceeds beyond 2000 PPM then it can cause increased heart rate and many other diseases.
- When the value will be less than 1000 PPM, then the LCD and webpage will display “Fresh Air”.
- Whenever the value will increase 1000 PPM, then the buzzer will start beeping and the LCD and webpage will display “Poor Air, Open Windows”.
- If it will increase 2000 then the buzzer will keep beeping and the LCD and webpage will display “Danger! Move to fresh Air”.

1.7 HARDWARE REQUIREMENT

- MQ135 Gas sensor
- Arduino Uno
- Wi-Fi module ESP8266
- 16X2 LCD
- Breadboard
- 10K potentiometer
- 1K ohm resistors
- 220 ohm resistor
- Buzzer

1.8 SOFTWARE REQUIREMENT

- Arduino IDE software

1.9 ORGANISATION OF THE REPORT

CHAPTER 1:

Chapter 1 begins with a very brief introduction of the project. This set the stage for the literature survey, where the basic knowledge for the project is discussed. The chapter discusses the advantages and applications of the project. It concludes with the discussion of resources used.

CHAPTER 2:

Chapter 2 continues to provide the brief theory about the project. This chapter is divided into 2 parts due to the length and amount of detail. The first section of the chapter introduces the hardware resources used. It also provides the concept and feature of the components used. The second part of the chapter discusses software resources used and provides the brief introduction and feature of the software used.

CHAPTER 3:

Chapter 3 deals with Design, implementation and simulation part of the project. The chapter covers the details of the interfaced components. This chapter provides the information about how the system is implemented.

CHAPTER 2

BRIEF THEORY ABOUT THE PROJECT

2.1 HARDWARE DESCRIPTION

2.1.1 ARDUINO UNO

OVERVIEW

Arduino is an open-source computer hardware and software company, project and user community that designs and manufactures microcontroller-based kits for building digital devices and interactive objects that can control objects in the physical world. Arduino had used the Atmel ATmega AVR series of chips, specifically the ATmega8, ATmega168, ATmega328, ATmega1280, and ATmega2560. The Arduino UNO is a 8 bit microcontroller board on the ATmega328. It has 14 digital pins and 6 analog pins and other power pins such as, GND, VCC. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It has SRAM 2kb and a flash memory 32kb, EEPROM with 1kb. Arduino is open source hardware board with many open source libraries to interface it on board microcontroller with many other external components like LED, motors, IR sensors and many other things one want to interface with Arduino board.

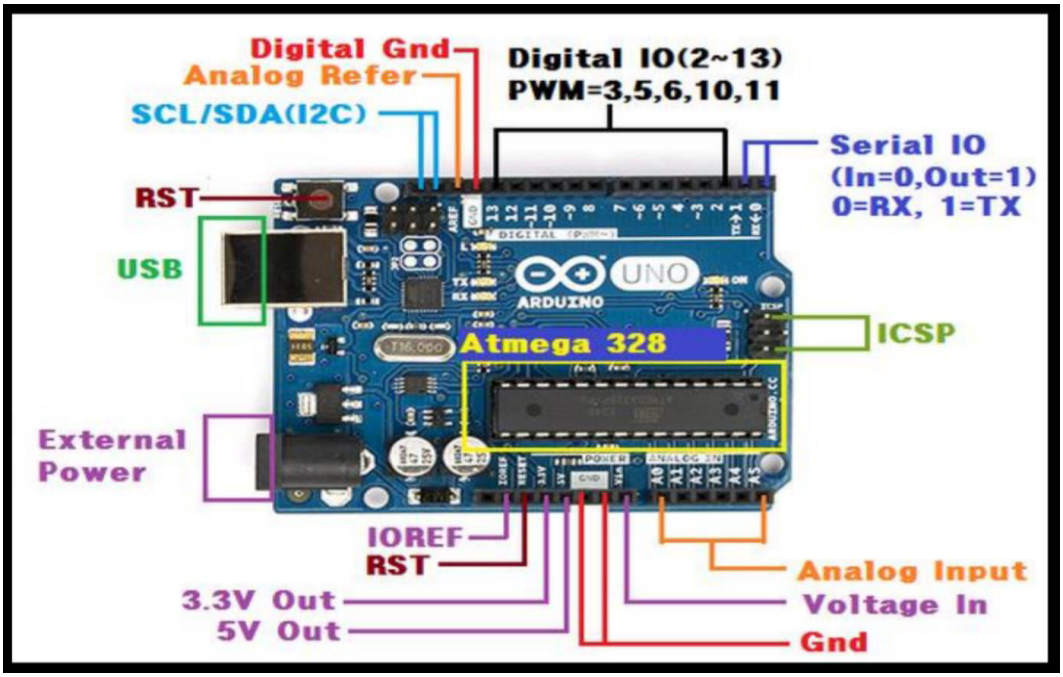


Fig 2: Arduino UNO Microcontroller

Arduino is a complete board which includes all things to connect with external peripheral and to program through computer. Fig 2 shows the Arduino UNO Microcontroller with all it's specifications labeled. It contains everything needed to support the microcontroller. Arduino UNO is needed to connect it to a computer using a USB cable or power it with an AC-to-DC (7-12V) adapter. Arduino also consists of a reset button. As with a normal computer, sometimes things can go wrong with Arduino, and when all else fails, one may need to reset the system and restart the Arduino. The Arduino circuit acts as an interface between the software part and the hardware part of the project.

POWER

The Arduino Uno can be powered via the USB connection or with an externalpower 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 of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- **VIN:** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You 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 5V from 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.

MEMORY

The ATmega328 has 32 KB (with 0.5 KB used for the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

INPUT AND OUTPUT

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. 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 attachInterrupt() function for details.
- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the analogWrite() function.

- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCLK).** These pins support SPI communication using the SPI library.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

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 to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:

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

There are a couple of other pins on the board:

- **AREF:** Reference voltage for the analog inputs. Used with analogReference().
- **RESET:** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

COMMUNICATION

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides 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, a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino 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 and 1).

A SoftwareSerial library allows for serial communication on any of the Uno's digital pins.

The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. For SPI communication, use the SPI library.

AUTOMATIC (SOFTWARE) RESET

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno 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 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader 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 bootloader 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 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.

USB OVERCURRENT PROTECTION

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

PHYSICAL CHARACTERISTICS

The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Four screw holes allow the board to be attached to a surface or case. Note that

the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

ATMEL ATMEGA328P PIN MAPPING

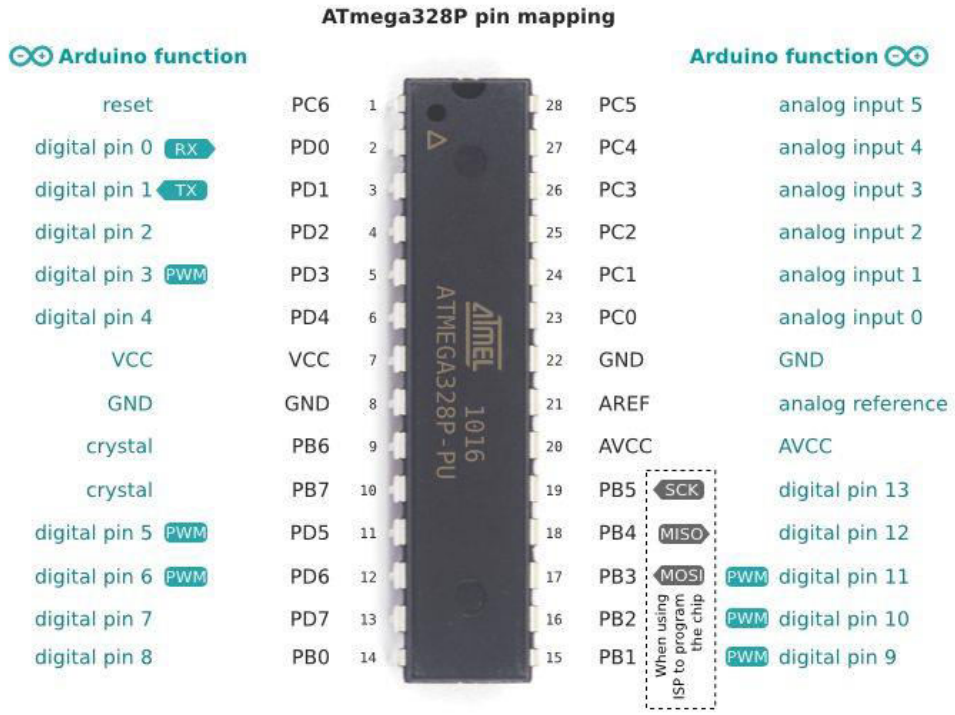


Figure 3: ATmega328P pin mapping

Figure 3 shows the pin mapping of ATmega328P. It is a high-performance 8-bit AVR RISC-based microcontroller combines 32KB ISP flash memory with read-while-write capabilities, 1KB EEPROM, 2KB SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible timer/counters with compare modes, internal and external interrupts, serial programmable USART, a byte-oriented 2-wire serial interface, SPI serial port, a 6-channel 10-bit A/D converter (8-channels in TQFP and QFN/MLF packages), programmable watchdog timer with internal oscillator, and five software selectable power saving modes. The device operates between 1.8-5.5 volts.

By executing powerful instructions in a single clock cycle, the device achieves throughputs approaching 1 MIPS per MHz, balancing power consumption and processing speed.

COMPARISON BETWEEN ATMEGA48PA, ATMEGA88PA, ATMEGA168PA AND ATMEGA328P

The ATmega48PA, ATmega88PA, ATmega168PA and ATmega328P differ only in memory sizes, boot loader support, and interrupt vector sizes. Table 1.1 summarizes the different memory and interrupts vector sizes for the three devices.

Table 2: Memory Size Summary

Device	Flash	EEPROM	RAM	Interrupt Vector Size
ATmega48PA	4K Bytes	256 Bytes	512 Bytes	1 instruction word/vector
ATmega88PA	8K Bytes	512 Bytes	1K Bytes	1 instruction word/vector
ATmega168PA	16K Bytes	512 Bytes	1K Bytes	2 instruction words/vector

ATmega88PA, ATmega168PA and ATmega328P support a real Read-While-Write Self Programming mechanism. There is a separate Boot Loader Section, and the SPM instruction can only execute from there. In ATmega48PA, there is no Read-While-Write support and no separate Boot Loader Section. The SPM instruction can execute from the entire Flash.

2.1.2 Wi-Fi module ESP8266

TECHNICAL OVERVIEW

ESP8266 is a complete and self-contained Wi-Fi network solutions that can carry software applications, or through Another application processor uninstall all Wi-Fi networking capabilities. Built-in cache memory will help improve system performance and reduce memory requirements. Another situation is when wireless Internet access assumes the task of Wi-Fi adapter, you can add it to any microcontroller-based design, and the connection is simple, just by SPI / SDIO interface or central processor AHB bridge interface as shown in Figure 4.

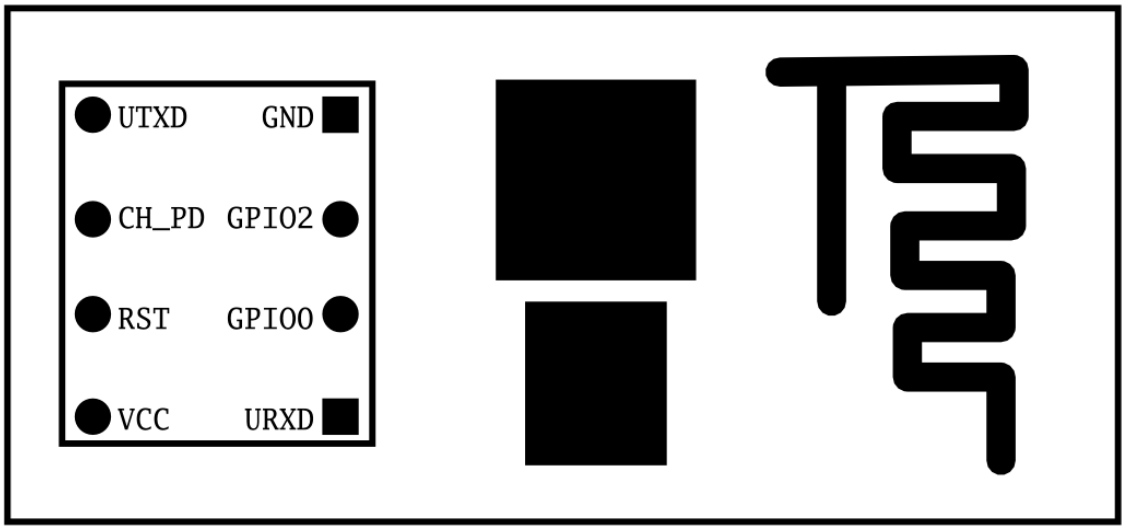


Figure 4: ESP8266 Wi-Fi Module Top View

Processing and storage capacity on ESP8266 powerful piece, it can be integrated via GPIO ports sensors and other applications specific equipment to achieve the lowest early in the development and operation of at least occupy system resources. The ESP8266 highly integrated chip, including antenna switch balun, power management converter, so with minimal external circuitry, and includes front-end module, including the entire solution designed to minimize the space occupied by PCB. The system is equipped with ESP8266 manifested leading features are: energy saving VoIP quickly switch between the sleep / wake patterns, with low-power operation adaptive radio bias, front-end signal processing functions, troubleshooting and radio systems coexist characteristics eliminate cellular / Bluetooth / DDR / LVDS / LCD interference.

CHARACTERISTICS

- 802.11 b / g / n
- Wi-Fi Direct (P2P), soft-AP
- Built-in TCP / IP protocol stack

- Built-in TR switch, balun, LNA, power amplifier and matching network
- Built-in PLL, voltage regulator and power management components
- 802.11b mode + 19.5dBm output power
- Built-in temperature sensor
- Support antenna diversity
- off leakage current is less than 10uA
- Built-in low-power 32-bit CPU: can double as an application processor
- SDIO 2.0, SPI, UART
- STBC, 1x1 MIMO, 2x1 MIMO
- A-MPDU, A-MSDU aggregation and the 0.4 Within wake
- 2ms, connect and transfer data packets
- standby power consumption of less than 1.0mW (DTIM3)

ULTRA-LOW POWER TECHNOLOGY

ESP8266 specifically for mobile devices, wearable electronics and networking applications design and make the machine to achieve the lowest energy consumption, together with several other patented technology. This energy-efficient construction in three modes: active mode, sleep mode and deep sleep mode type. When ESP8266 using high-end power management technology and logic systems to reduce non-essential functions of the power conversion regulate sleep patterns and work modes, in sleep mode, it consumes less than the current 12uA, is connected, it consumes less power to 1.0mW (DTIM = 3) or 0.5mW (DTIM = 10). Sleep mode, only calibrated real-time clock and watchdog in working condition. Real-time clock can be programmed to wake ESP8266 within a specific period of time. Through programming, ESP8266 will automatically wake up when detected certain to happen. ESP8266 automatic wake-up in the shortest time, this feature can be applied to the SOC for mobile devices, so before you turn Wi- Fi SOC are in a low-power standby mode. To meet the power requirements of mobile devices and wearable electronics products, ESP8266 at close range when the PA output power can be reduced through software programming to reduce overall power consumption in order to adapt to different applications.

MAXIMUM INTEGRATION

ESP8266 integrates the most critical components on the board, including power management components, TR switch, RF balun, a peak power of + 25dBm of PA,

therefore, ESP8266 only guarantee the lowest BOM cost, and easy to be embedded in any system. ESP8266 BOM is the only external resistors, capacitors, and crystal.

CPU AND MEMORY

CPU INTERFACE

The chip embedded in an ultra-low-power 32-bit micro-CPU, with 16 compact modes. Can be connected to the CPU via the following interfaces:

Connecting storage controllers can also be used to access external code memory RAM/ROM interface (iBus). Also attached storage controller data RAM interface (dBus), Access Register of AHB interface JTAG debug interface.

Storage Controller

Storage controller contains ROM and SRAM. CPU can iBus, dBus and AHB interface to access the storage controller. Any one of these interfaces can apply for access to ROM or RAM cells, memory arbiter to determine the running order in the order of arrival.

AHB and APB module

AHB module acts as arbiter, through the MAC, and SDIO host CPU control AHB interface. Since sending Address different, AHB data requests may arrive the following two slaves in one: APB module, or flash memory controller (usually in the case of off-line applications) to the received request is a high speed memory controllers often request, APB module receives register access is often Request. APB module acts as a decoder, but only you can access the ESP8266 main module programmable registers. Since the sending address different, APB request may reach the radio receiver, SI / SPI, hosts SDIO, GPIO, UART, real-time clock (RTC), MAC or digital baseband.

INTERFACE

ESP 8266 contains multiple analog and digital interfaces, as follows:

Main SI / SPI control (optional)

Main Serial Interface (SI) can run at two, three, four-wire bus configuration, is used to control the EEPROM or other I2C / SPI devices. Multiple devices share the two-wire I2C bus. Multiple SPI devices to share the clock and data signals, and according to the chip select, each controlled by software alone GPIO pins. SPI can be used to control external devices, such as serial flash, audio CODEC or other slave devices, installation, effectively giving it three different pins, making it the standard master SPI device.

SPI_EN0

SPI_EN1

SPI_EN2

SPI slave is used as the primary interface, giving SPI master and slave SPI support. In the built-in applications, SPI_EN0 is used as an enable signal, the role of external serial flash, download firmware and/or MIB data to baseband. In host-based applications, the firmware and you can choose one MIB data downloaded via the host interface both. This pin is active low when not should be left unconnected. SPI_EN1 often used for user applications, such as controlling the built-in applications or external audio codec sensor ADC. This pin is active low when not should be left unconnected. SPI_EN2 often used to control the EEPROM, storing individual data such as MIB information, MAC address, and calibration data, or for general purposes. This pin is active low when not should be left unconnected.

BASIC AT COMMAND SET

Table 3: Basic AT Commands

COMMAND	DESCRIPTION
AT	Test AT Stratup
AT+RST	Restart module
AT+GMR	View version info
AT+GSLP	Enter deep sleep
ATE	AT commands echo or not
AT+RESTORE	Factory reset
AT+UART	UART configuration
AT+UART_CUR	UART current configuration
AT+UART_DEF	UART default configuration, save to flash
AT+SLEEP	Sleep mode

AT+RFPOWER	Set maximum value of RF TX power
AT+RFVDD	Set RF TX power according to VDD33

Wi Fi Functions Overview

Table 4: Wi-Fi connectivity functions invoked by AT commands

COMMAND	DESCRIPTION
AT+CWMODE	WIFI mode (sta/AP/sta+AP)
AT+CWMODE_CUR	WIFI mode (sta/AP/sta+AP)
AT+CWMODE_DEF	WIFI default mode (sta/AP/sta+AP), Save to flash
AT+CWJAP	Connect to AP
AT+CWJAP_CUR	Connect to AP, won't save to flash.
AT+CWJAP_DEF	Connect to AP, save to flash.
AT+CWLAP	List available AP's
AT+CWQAP	Disconnect from AP
AT+CWSAP	Set configuration of ESP8266 softAP.
AT+CWSAP_CUR	Set configuration of ESP8266 soft AP Won't save to flash
AT+CWSAP_DEF	Set configuration of ESP8266 softAP Save to flash
AT+CWLIF	Get station's IP which is connected to ESP8266 softAP
AT+CWDHCP	Enable/Disable DHCP
AT+CWDHCP_CUR	Enable/Disable DHCP, won't save to flash
AT+CWDHCP_DEF	Enable/Disable DHCP, save to flash
AT+CWAUTOCONN	Connect to AP automatically when power on
AT+CIPSTAMAC	Set MAC address of ESP8266 station
AT+CIPSTAMAC_CUR	Set MAC address of ESP8266 station Won't save to flash
AT+CIPSTAMAC_DEF	Set MAC address of ESP8266 station Save to flash
AT+CIPAPMAC	Set MAC address of ESP8266 softAP
AT+CIPAPMAC_CUR	Set MAC address of ESP8266 softAP Won't save to flash
AT+CIPAPMAC_DEF	Set MAC address of ESP8266 softAP

	Save to flash
AT+CIPSTA	Set IP address of ESP8266 station
AT+CIPSTA_CUR	Set IP address of ESP8266 station, Won't save to flash
AT+CIPSTA_DEF	Set IP address of ESP8266 station Save to flash
AT+CIPAP	Set IP address of ESP8266 softAP
AT+CIPAP_CUR	Set IP address of ESP8266 softAP, Won't save to flash
AT+CIPAP_DEF	Set IP address of ESP8266 softAP Save to flash

2.1.3 MQ135 GAS SENSOR

MQ135 SENSOR CIRCUIT AND WORKING

The MQ series of gas sensors utilizes a small heater inside with an electro chemical sensor these sensors are sensitive to a range of gasses are used at room temperature. MQ135 alcohol sensor is a Sno2 with a lower conductivity of clean air. When the target explosive gas exists, then the sensor's conductivity increases along with the gas concentration rising levels. By using simple electronic circuits, it converts the charge of conductivity to correspond output signal of gas concentration.

The MQ135 gas sensor has high sensitivity in ammonia, sulfide, benzene steam, smoke and in other harm full gas. It is low cost and suitable for different applications. There are different types of alcohol sensors like MQ-2, MQ-3, MQ-4, MQ-5, MQ-6, etc.

MQ-135 GAS SENSOR

The MQ-135 gas sensor senses the gases like ammonia nitrogen, oxygen, alcohols, aromatic compounds, sulfide and smoke. The boost converter of the chip MQ-3 gas sensor is PT1301. The operating voltage of this gas sensor is from 2.5V to 5.0V. The MQ-3 gas sensor has a lower conductivity to clean the air as a gas sensing material. In the atmosphere we can find polluting gases, but the conductivity of gas sensor increases as the concentration of polluting gas increases. MQ-135 gas sensor can be implementation to detect the smoke, benzene, steam and other harmful gases. It has potential to detect different harmful gases. The MQ-135 gas sensor is low cost to purchase. The basic image of the MQ-135 sensor is shown in figure 5.

Figure 5: MQ-135 Gas Sensor

WORKING PRINCIPLE AND CIRCUIT DIAGRAM

The MQ-135 gas sensor consists of a tin dioxide (SnO2), a perspective layer inside aluminum oxide micro tubes (measuring electrodes) and a heating element inside a tubular casing. The end face of the sensor is enclosed by a stainless steel net and the back side holds the connection terminals. Ethyl alcohol present in the breath is oxidized into acetic acid passing through the heat element. With the ethyl alcohol cascade on the tin dioxide sensing layer, the resistance decreases. By using the external load resistance the resistance variation is converted into a suitable voltage variation. The circuit diagram and the connection arrangement of an MQ 135 is as shown in the Figure 6.

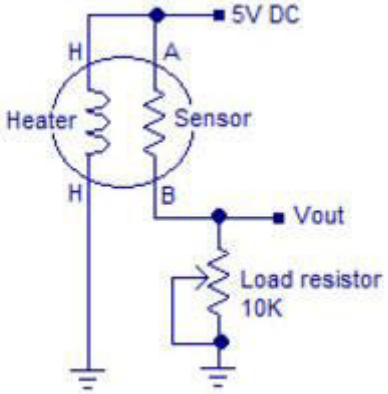


Figure 6: MQ-135 Circuit Diagram

MQ – 135 AIR QUALITY SENSOR

The air quality sensor is also a MQ-135 sensor for detecting venomous gases that are present in the air in homes and offices. The gas sensor layer of the sensor unit is made

up of tin dioxide (SnO2); it has lower conductivity compare to clean air and due to air pollution the conductivity is increases. The air quality sensor detects ammonia, nitrogen oxide, smoke, CO2 and other harmful gases.

The air quality sensor has a small potentiometer that permits the adjustment of the load resistance of the sensor circuit. The 5V power supply is used for air quality quality sensor as shown in Figure 7.

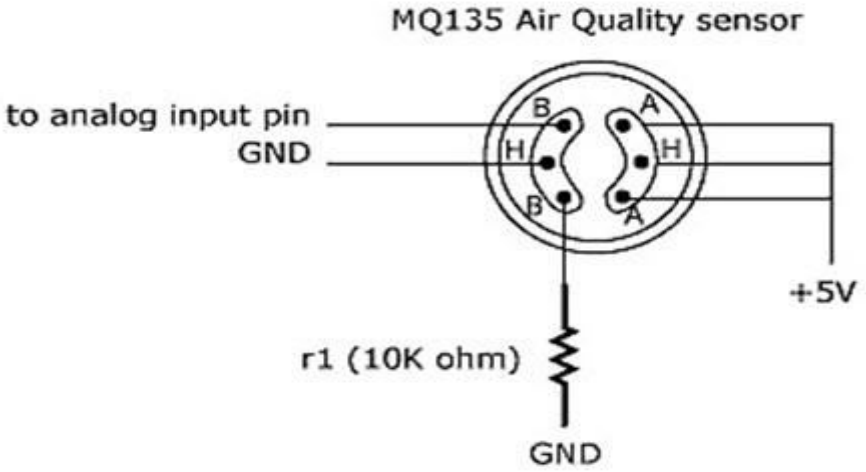


Figure 7: MQ – 135 Air Quality Sensor

The air quality sensor is a signal output indicator instruction. It has two outputs: analog output and TTL output. The TTL output is low signal light which can be accessed through the IO ports on the Microcontroller. The analog output is a concentration, i.e., increasing voltage is directly proportional to increasing concentration. This sensor has a long life and reliable stability as well.

FEATURES

Table 5: Features of MQ-135

Sensitive gas	**Ammonia, nitrogen oxide, alcohols, Aromatic compounds, sulfide and Smoke**
Boost converter chip	PT1301
Operating voltage	2.5V-5.0V
Dimensions	40.0mm*21.0mm
Fixing hole size	2.0mm

INTERFACES

Table 6: Pin Description

Pin No.	Symbol	Description
1	DOUT	Digital output
2	AOUT	Analog output
3	GND	Power ground
4	VCC	Positive power supply (2.5V-5.0V)

TECHNIQUE USED

In this section, we illustrate the usage of the module with an example of sensitive gas detection by connecting a development board

1. Download the relative codes to the development board.
2. Connect the development board to PC via a serial wire and the module to the development board then, power up the board and start the serial debugging software.

Here is the configuration of the connection between the module and the development board.

Table 7: Configuration of the connection

PORT	STM32 MUC pin
DOUT	GPIOA.4
AOUT	GPIOA.6
GND	GND
VCC	3.3V

PORT	Arduino pin
DOUT	D2
AOUT	A0
GND	GND
VCC	5V

3. Warn-up the sensor for a minute.
4. The detected result can be checked by the LED indicator on the module. Put the sensor into a container filled with sensitive gas, you will find the indicator turns

on. While take the sensor out of the container, you can see the indicator turns off .

APPLICATIONS OF MQ 135 GAS SENSOR

The following are the applications of the MQ 135 gas sensor:

- Air quality monitor
- Detection of harmful gases
- Domestic air pollution detection
- Industrial pollution detection
- Portable air pollution detection

CHARACTERISTICS OF MQ 135

- Good sensitivity to harmful gases in wide range.
- It has long life and low cost.
- Possesses high sensitivity to ammonia, benzene, sulfide gases.
- It is a simple drive circuit.

2.1.4 LCD

A liquid crystal display (LCD) shown in Figure 8 is a flat panel display, electronic visual display, based on Liquid Crystal Technology. A liquid crystal display consists of an array of tiny segments called pixels that can be manipulated to present information. Liquid crystals do not emit light directly instead they use light modulating techniques

LCD (Liquid Crystal Display) screen is an electronic display module and find a wide range of applications, including computer monitors, television, instrument panels, aircraft cockpit displays, signage etc. They are common in consumer devices such as video players, gaming devices, clocks, watches, calculators and telephones. The term liquid crystal is used to describe a substance in a state between liquid and solid but which exhibits the properties of both. Molecules in liquid crystal tend to arrange themselves until the all point in the same specific direction. This arrangement of molecules enables the medium to flow as a liquid. A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits. These modules are preferred over seven segments and other multi segment LEDs. The reasons being: LCDs are economical; easily programmable; have no limitation of displaying special & even custom characters (unlike in seven segments), animations and so on.

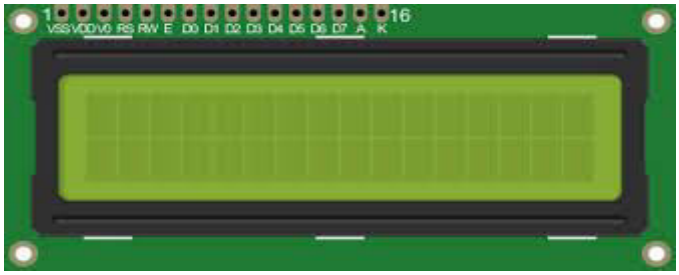


Figure 8: 16x2 LCD display

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. This LCD has two registers, namely, Command and Data. The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD.

Depending on the temperature and particular nature of a substance, liquid crystal can exist in one of several distinct phases. Liquid crystal in a nematic phase, in which there is no spatial ordering of the molecules. LCDs use liquid crystals because they react predictably to electric current in such a way as to control the passage of light.

PIN CONFIGURATION IN LCD

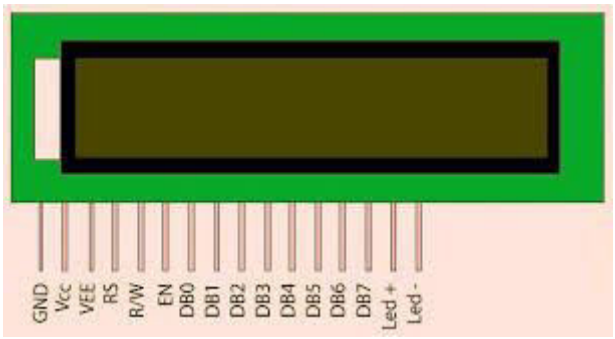


Figure 9: Pin configuration in LCD

Table 8: Pin Description

Pin No	Symbol	Level	Description
1	VSS	0V	Ground
2	VDD	5.0V	Supply voltage for logic

3	V0	----	Input voltage for LCD
4	RS	H/L	H:Data signal, L:Instruction signal
5	R/W	H/L	H:Read mode, L:Write mode`
6	E	H, H→L	Chip enable signal
7	DB0	H/L	Data bit 0
8	DB1	H/L	Data bit 1
9	DB2	H/L	Data bit 2
10	DB3	H/L	Data bit 3
11	DB4	H/L	Data bit 4
12	DB5	H/L	Data bit 5
13	DB6	H/L	Data bit 6
14	DB7	H/L	Data bit 7
15	LED A(+)	4.2V	Back light anode
16	LED K(-)	0V	Back light cathode

V0: To set LCD display sharpness, this pin is used. Best way is to use variable resistor such as potentiometer a variable current makes the character contrast sharp. Connect the output of the potentiometer to this pin. Rotate the potentiometer knob forward and backward to adjust the LCD contrast.

RS (Register Select): A 16X2 LCD has two registers, namely, command and data. The register select is used to switch from one register to other. RS=0 for command register, whereas RS=1 for data register.

The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. Processing of commands happens in the command register.

The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD. When we send data to LCD it goes to the data register and is processed there. When RS=1, data register is selected.

R/W (Read/Write): A 16x2 LCD can perform 2 functions namely read and write. When the LCD writes to a variable or register, then R/W=0. When the LCD reads from a variable or register, then R/W=1.

E (Enable): This is the controlling factor of a LCD. For the LCD to perform read and write operations, the enable must always be high i.e. enable pin is always kept high. If enable pin is low, then the LCD can perform neither read or write operation.

D0-D7: These are the 8 data bits present in a 16x2 LCD from pin number 7 to 14.

A and K: These are the 15^{th} and 16^{th} pins of a 16x2 LCD. These are the pins which control the backlight of the display. Power supply is given to A (LED+) and K (LED-) is grounded. These pins are not mandatory. They can be used if highlighting display is needed.

2.2 SOFTWARE DESCRIPTION

The Arduino Uno can be programmed with the (Arduino Software (IDE)). Select "Arduino/Genuino Uno" from the Tools > Board menu (according to the microcontroller on your board).

The ATmega328 o*n the Arduino Uno comes preprogrammed with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol.

Bootloader can be bypassed and program the microcontroller through the ICSP (In-Circuit Serial Programming) header using Arduino ISP or similar; see these instructions for details.

The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available in the Arduino repository. The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by:

- On Rev1 boards: connecting the solder jumper on the back of the board and then resetting the 8U2.
- 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.

Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) can be used to load a new firmware. Or ISP header can be used with an external programmer (overwriting the DFU bootloader).

2.2.1 ARDUINO IDE

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. The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board.

WRITING SKETCHES

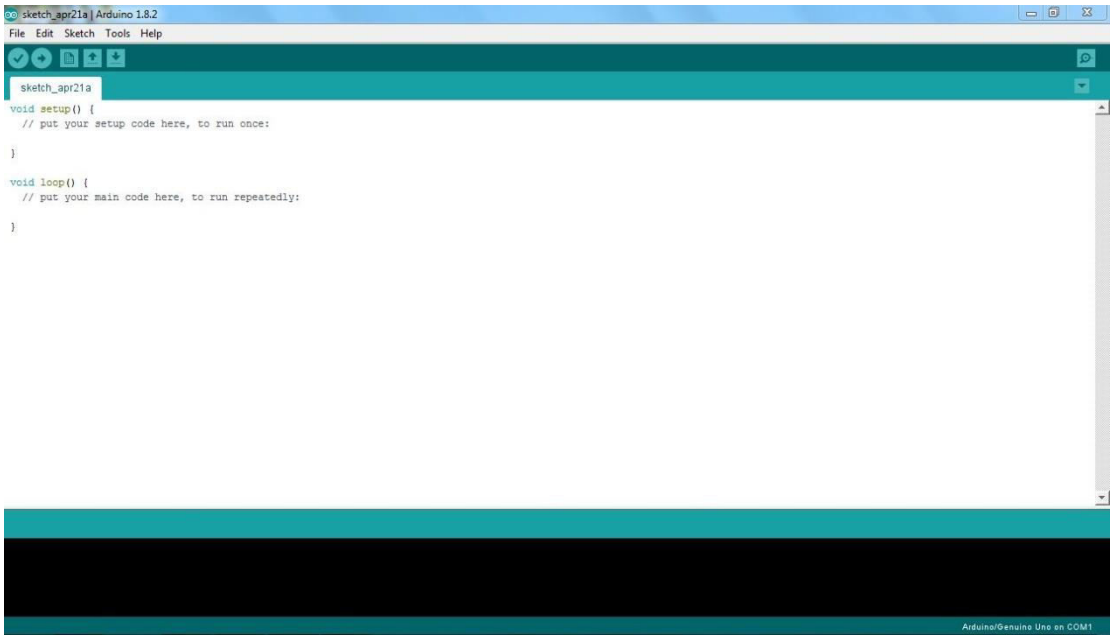


Figure 10: Screenshot of Arduino 1.8.2

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom right-hand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

Verify: Checks the code for errors compiling it.

Upload: Compiles your code and uploads it to the configured board.

New: Creates a new sketch.

Open: Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window overwriting its content.

Save: Saves the sketch.

 **Serial Monitor**: Opens the serial monitor.

The options in the main menu include File**,** Edit**,** Sketch**,** Tools**,** Help.

SKETCHBOOK

The Arduino Software (IDE) uses the concept of a sketchbook: a standard place to store your programs (or sketches). The sketches in your sketchbook can be opened from the **File > Sketchbook** menu or from the **Open** button on the toolbar. The first time you run the Arduino software, it will automatically create a directory for your sketchbook. You can view or change the location of the sketchbook location from with the **Preferences** dialog. Beginning with version 1.0, files are saved with a .ino file extension.

TABS, MULTIPLE FILES, AND COMPILATION

Allows to manage sketches with more than one file (each of which appears in its own tab). These can be normal Arduino code files (no visible extension), C files (.c extension), C++ files (.cpp), or header files (.h).

UPLOADING

Before uploading your sketch, you need to select the correct items from the **Tools > Board** and **Tools > Port** menus. The boards are described below. On the Mac, the serial port is probably something like **/dev/tty.usbmodem241** (for a Uno or Mega2560 or Leonardo) or **/dev/tty.usbserial-1B1** (for a Duemilanove or earlier USB board), or **/dev/tty.USA19QW1b1P1.1** (for a serial board connected with a Keyspan USB-to-Serial adapter). On Windows, it's probably **COM1** or **COM2** (for a serial board) or **COM4**, **COM5**, **COM7**, or higher (for a USB board) - to find out, you look for USB serial device in the ports section of the Windows Device Manager. On Linux, it should be **/dev/ttyACMx** , **/dev/ttyUSBx** or similar. Once you've selected the correct serial port and board, press the upload button in the toolbar or select the **Upload** item from the **Sketch** menu. Current Arduino boards will reset automatically and begin the upload. With older boards (pre-Diecimila) that lack auto-reset, you'll need to press the reset button on the board just before starting the upload. On most boards, you'll see the RX and TX LEDs blink as the sketch is uploaded. The Arduino Software (IDE) will display a message when the upload is complete, or show an error.

When a sketch is uploaded, a small program called Arduino **bootloader is used** that has been loaded on to the microcontroller on the board. It allows uploading code

without using any additional hardware. The bootloader is active for a few seconds when the board resets; then it starts whichever sketch was most recently uploaded to the microcontroller.

LIBRARIES

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from the **Sketch > Import Library** menu. This will insert one or more **#include** statements at the top of the sketch and compile the library with the sketch. Because libraries are uploaded to the board with the sketch, they increase the amount of space it takes up. If a sketch no longer needs a library, simply delete its **#include** statements from the top of the code.

Some libraries are included with the Arduino software. Others can be downloaded from a variety of sources or through the Library Manager. Starting with version 1.0.5 of the Arduino IDE, a library can be imported from a zip file and use it in an open sketch.

SERIAL MONITOR

It displays serial data being sent from the Arduino board. To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down that matches the rate passed to **Serial.begin()** in the sketch. Note that on Windows, Mac or Linux, the Arduino board will reset (rerun your sketch execution to the beginning) when connected with the serial monitor

BOARDS

The board selection has two effects: it sets the parameters (e.g. CPU speed and baud rate) used when compiling and uploading sketches; and sets the file and fuse settings used by the burn bootloader command. Some of the board definitions differ only in the latter, so even if uploading is successful with a particular selection, check it before burning the bootloader.

Arduino Software (IDE) includes the built in support for the boards in the following list, all based on the AVR Core.

- Arduino Yùn
- Arduino/Genuino Uno
- Arduino Diecimila or Duemilanove
- Arduino Nano

- Arduino/Genuino Mega 2560
- Arduino MegaArduino Mega ADK
- Arduino Leonardo
- Arduino/Genuino Micro
- Arduino Esplora
- Arduino Mini
- Arduino Ethernet
- Arduino Fio
- Arduino BT
- LilyPad Arduino USB
- LilyPad Arduino
- Arduino Pro or Pro Mini (5V, 16 MHz)
- Arduino NG
- Arduino Robot Control
- Arduino Robot Motor
- Arduino Gemma
- Arduino Uno WiFi

CHAPTER 3

DESIGN, IMPLEMENTATION AND SIMULATION

The implementation of individual components with Arduino is as follows:

3.1 ESP8266

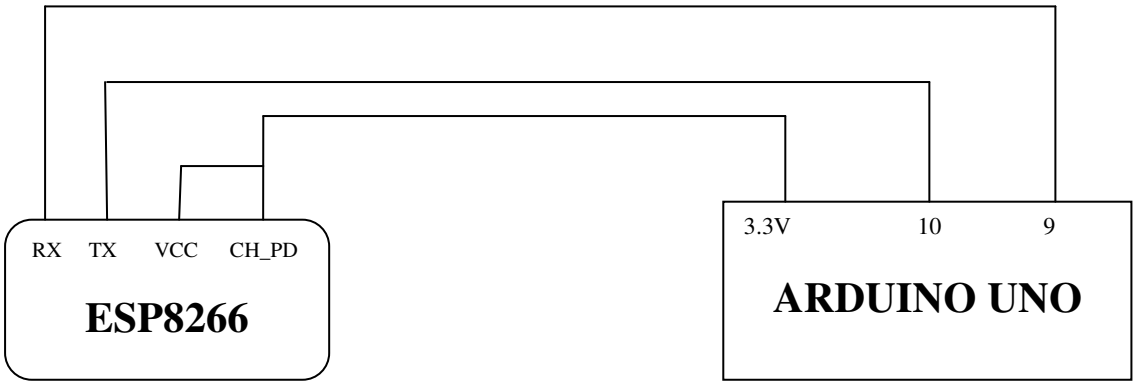


Figure 11: Implementation of ESP8266 with Arduino Uno

- Connect the VCC and the CH_PD to the 3.3V pin of Arduino.
- Connect the TX pin of the ESP8266 to the pin 10 of the Arduino.
- Connect the RX pin of the esp8266 to the pin 9 of Arduino through the resistors.

3.2 MQ135

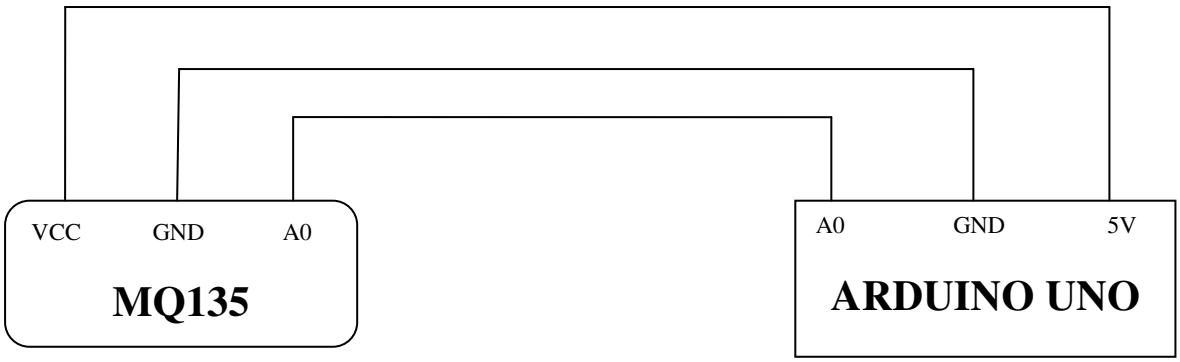


Figure 12: Implementation of MQ135 with Arduino Uno

- Connect the VCC and the ground pin of the sensor to the 5V and ground of the Arduino.
- Analog pin of sensor to the A0 of the Arduino.

3.3 BUZZER

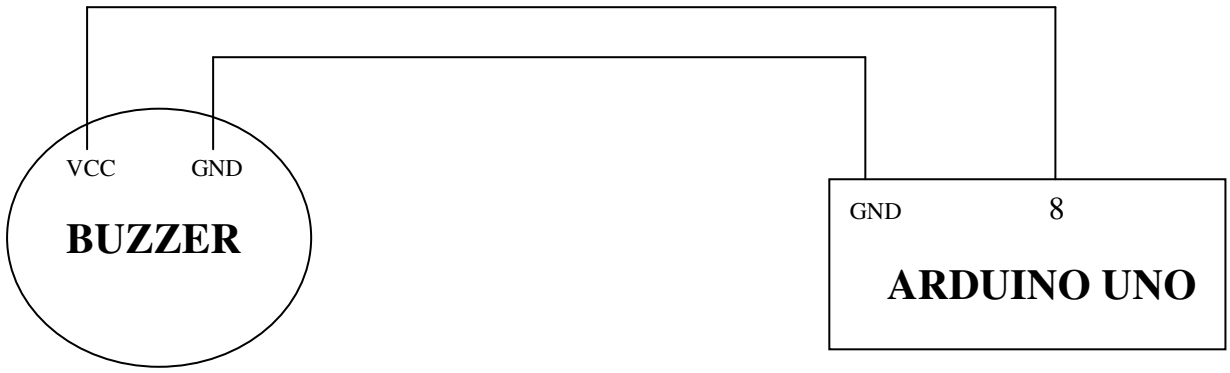


Figure 13: Implementation of Buzzer with Arduino Uno

- Connect a buzzer to the pin 8 of the Arduino which will start to beep when the condition becomes true.

3.4 LCD

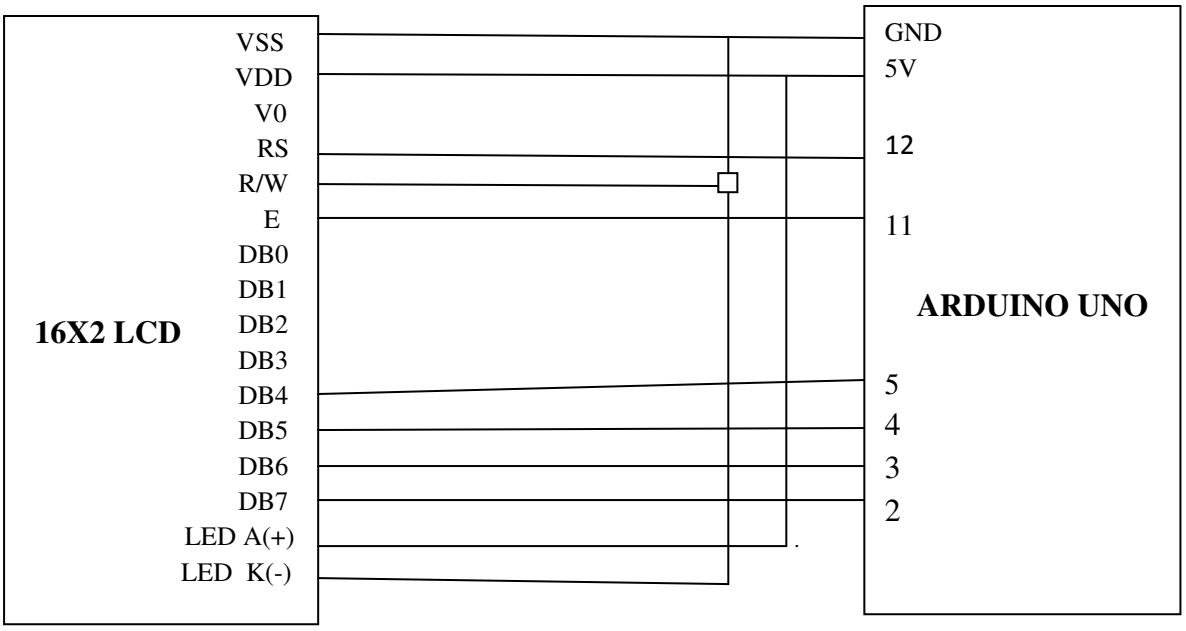


Figure 14: Implementation of LCD with Arduino Uno

The connections of the LCD are as follows

- Connect pin 1 (VEE) to the ground.
- Connect pin 2 (VDD or VCC) to the 5V.

- Connect pin 4 (RS) to the pin 12 of the Arduino.
- Connect pin 5 (Read/Write) to the ground of Arduino.
- Connect pin 6 (E) to the pin 11 of the Arduino.
- The following four pins are data pins which are used to communicate with the Arduino.

  Connect pin 11 (D4) to pin 5 of Arduino.

  Connect pin 12 (D5) to pin 4 of Arduino.

  Connect pin 13 (D6) to pin 3 of Arduino.

  Connect pin 14 (D7) to pin 2 of Arduino.
- Connect pin 15 to the VCC through the 220 ohm resistor.
- Connect pin 16 to the Ground.

3.5 HARDWARE IMPLEMENTATION

First connect the **ESP8266 with the Arduino**. ESP8266 runs on 3.3V and if it is connected to 5V of the Arduino then it won't work properly and it may get damage. Connect the VCC and the CH_PD to the 3.3V pin of Arduino. The RX pin of ESP8266 works on 3.3V and it will not communicate with the Arduino when it is directly connected to the Arduino. So a voltage divider circuit has to be made for it which will convert the 5V into 3.3V. This can be done by connecting three resistors in series as shown in the Figure. Connect the TX pin of the ESP8266 to the pin 10 of the Arduino and the RX pin of the esp8266 to the pin 9 of Arduino through the resistors.

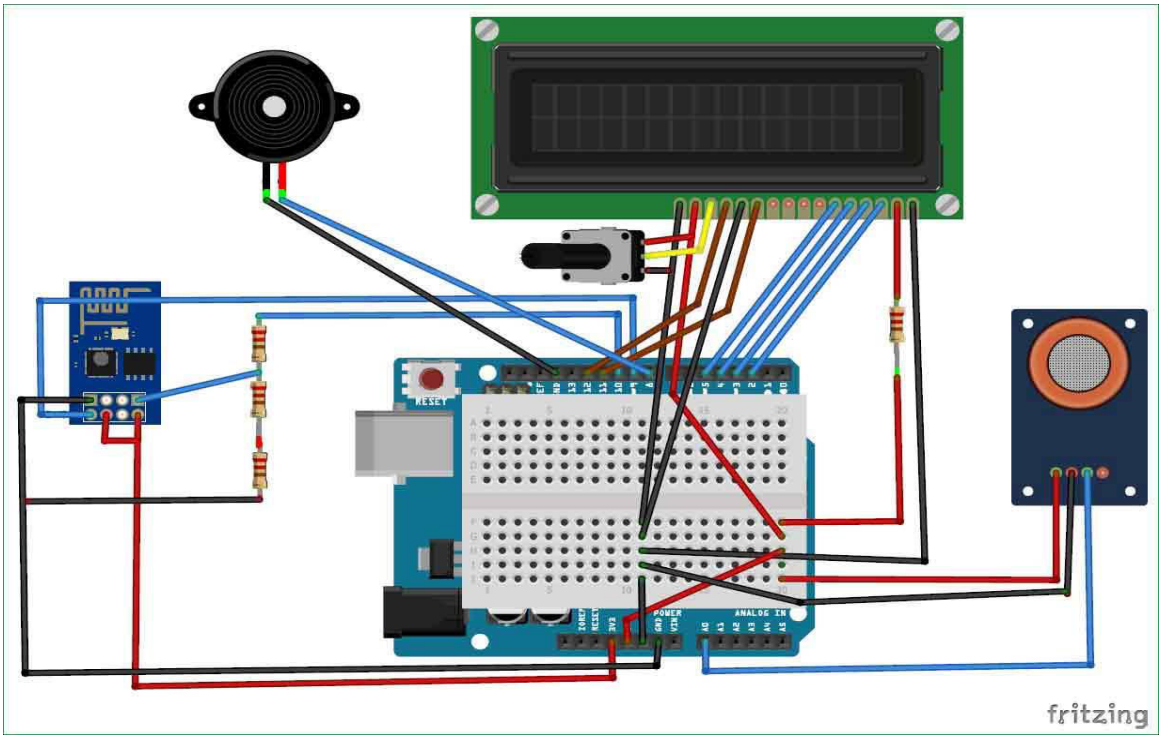


Figure 15: Interfacing of Components with Arduino Uno

ESP8266 Wi-Fi module gives the project **access to Wi-Fi or internet**. It is a very cheap device and makes the project very powerful. It can communicate with any microcontroller and it is the most leading devices in the IOT platform.

Connect the **MQ135 sensor with the Arduino**. Connect the VCC and the ground pin of the sensor to the 5V and ground of the Arduino and the Analog pin of sensor to the A0 of the Arduino.

Connect a buzzer to the pin 8 of the Arduino which will start to beep when the condition becomes true.

The connections of the LCD are as follows

- Connect pin 1 (VEE) to the ground.
- Connect pin 2 (VDD or VCC) to the 5V.
- Connect pin 3 (V0) to the middle pin of the 10K potentiometer and connect the other two ends of the potentiometer to the VCC and the GND. The potentiometer is used to control the screen contrast of the LCD. Potentiometer of values other than 10K will work too.
- Connect pin 4 (RS) to the pin 12 of the Arduino.

- Connect pin 5 (Read/Write) to the ground of Arduino. This pin is not often used so it can be connected to the ground.
- Connect pin 6 (E) to the pin 11 of the Arduino. The RS and E pin are the control pins which are used to send data and characters.
- The following four pins are data pins which are used to communicate with the Arduino.

  Connect pin 11 (D4) to pin 5 of Arduino.

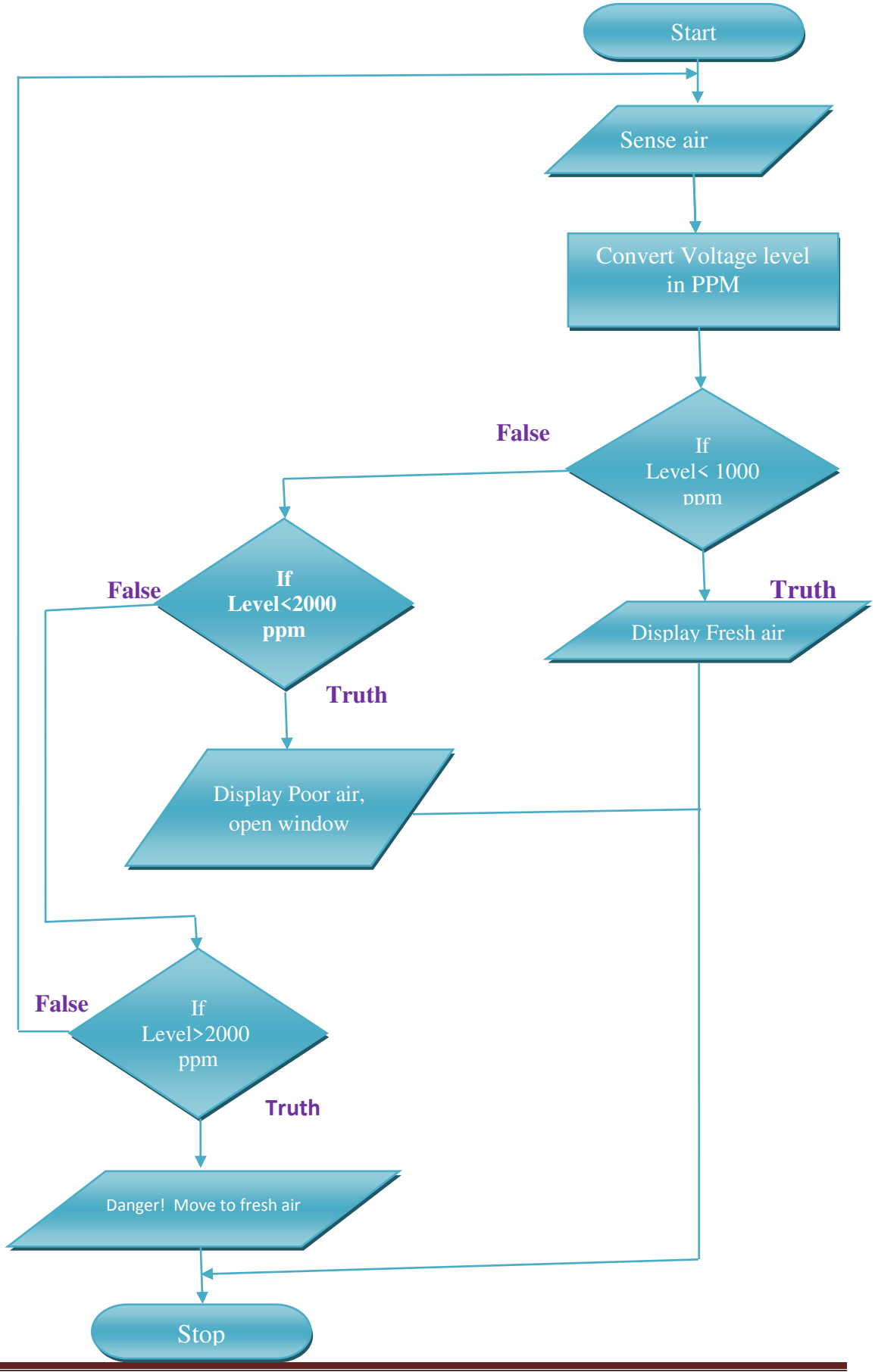
  Connect pin 12 (D5) to pin 4 of Arduino.

  Connect pin 13 (D6) to pin 3 of Arduino.

  Connect pin 14 (D7) to pin 2 of Arduino.
- Connect pin 15 to the VCC through the 220 ohm resistor. The resistor will be used to set the back light brightness. Larger values will make the back light much darker. Connect pin 16 to the Ground.

3.6 Design Flow

3.6.1 FLOW CHART



6.2 CODE EXPLANATION

Before beginning the coding for this project, we need to first Calibrate the MQ135 Gas sensor. There are lots of calculations involved in converting the output of sensor into PPM value, here we are using the Library for MQ135.

Using this library you can directly get the PPM values, by just using the below two lines:

```
MQ135 gasSensor = MQ135(A0);
float air_quality = gasSensor.getPPM();
```

But before that we need to **calibrate the MQ135 sensor**, for calibrating the sensor upload the below given code and let it run for 12 to 24 hours and then get the *RZERO* value.

```
#include "MQ135.h"
void setup (){
Serial.begin (9600);
}
void loop() {
MQ135 gasSensor = MQ135(A0); // Attach sensor to pin A0
float rzero = gasSensor.getRZero();
Serial.println (rzero);
delay(1000);
}
```

After getting the *RZERO* value, **Put the RZERO value in the library file** "MQ135.h": *#define RZERO 494.63*

Now we can begin the actual code for our Air quality monitoring project.

In the code, first of all we have defined the libraries and the variables for the Gas sensor and the LCD. By using the Software Serial Library, we can make any digital pin as TX and RX pin. In this code, we have made Pin 9 as the RX pin and the pin 10 as the TX pin for the ESP8266. Then we have included the library for the LCD and have defined the pins for the same. We have also defined two more variables: one for the sensor analog pin and other for storing air quality value.

```
#include <SoftwareSerial.h>
#define DEBUG true
SoftwareSerial esp8266(9,10);
#include <LiquidCrystal.h>
LiquidCrystal lcd(12,11, 5, 4, 3, 2);
const int sensorPin= 0;
int air_quality;
```

Then we will declare the pin 8 as the output pin where we have connected the buzzer. *l*cd.begin(16,2) command will start the LCD to receive data and then we will set the cursor to first line and will print the 'Welcome'. Then we will set the cursor on the second line and will print 'Sensor Warming'.

```
pinMode(8, OUTPUT);
lcd.begin(16,2);
lcd.setCursor (0,0);
lcd.print ("Welcome ");
lcd.setCursor (0,1);
lcd.print ("Sensor Warming ");
delay(1000);
```

Then we will set the baud rate for the serial communication. Different ESP's have different baud rates so write it according to your ESP's baud rate. Then we will send the commands to set the ESP to communicate with the Arduino and show the IP address on the serial monitor.

```
Serial.begin(115200);
esp8266.begin(115200);
  sendData("AT+RST\r\n",2000,DEBUG);
  sendData("AT+CWMODE=2\r\n",1000,DEBUG);
  sendData("AT+CIFSR\r\n",1000,DEBUG);
  sendData("AT+CIPMUair_quality=1\r\n",1000,DEBUG);
  sendData("AT+CIPSERVER=1,80\r\n",1000,DEBUG);
```

```
pinMode(sensorPin, INPUT);
lcd.clear();
```

For printing the output on the webpage in web browser, we will have to use **HTML programming**. So, we have created a string named webpage and stored the output in it. We are subtracting 48 from the output because the read() function returns the ASCII decimal value and the first decimal number which is 0 starts at 48.

```
if(esp8266.available())
  {
   if(esp8266.find("+IPD,"))
   {
    delay(1000);
    int connectionId = esp8266.read()-48;
    String webpage = "<h1>IOT Air Pollution Monitoring System</h1>";
      webpage += "<p><h2>";
      webpage+= " Air Quality is ";
      webpage+= air_quality;
      webpage+=" PPM";
      webpage += "<p>";
```

The following code will call a function named sendData and will send the data & message strings to the webpage to show.

```
    sendData(cipSend,1000,DEBUG);
    sendData(webpage,1000,DEBUG);


    cipSend = "AT+CIPSEND=";
    cipSend += connectionId;
    cipSend += ",";
    cipSend +=webpage.length();
    cipSend +="\r\n";
```

The following code will print the data on the LCD. We have applied various conditions for checking air quality, and LCD will print the messages according to conditions and buzzer will also beep if the pollution goes beyond 1000 PPM.

```
lcd.setCursor (0, 0);
lcd.print ("Air Quality is ");
lcd.print (air_quality);
lcd.print (" PPM ");
lcd.setCursor (0,1);
if (air_quality<=1000)
{
lcd.print("Fresh Air");
digitalWrite(8, LOW);
```

Finally the below function will send and show the data on the webpage. The data we stored in string named *'webpage'* will be saved in string named *'command'*. The ESP will then read the character one by one from the *'command'* and will print it on the webpage.

```
String sendData(String command, const int timeout, boolean debug)
{
   String response = "";
   esp8266.print(command); // send the read character to the esp8266
   long int time = millis();
   while( (time+timeout) > millis())
   {
     while(esp8266.available())
     {
       // The esp has data so display its output to the serial window
       char c = esp8266.read(); // read the next character.
       response+=c;
     }
   }
```

```
   if(debug)
   {
    Serial.print(response);
   }
   return response;
}
```

3.6.3 CODE

```
#include "MQ135.h"
#include <SoftwareSerial.h>
#define DEBUG true
SoftwareSerial esp8266(9,10); // This makes pin 9 of Arduino as RX pin and pin 10 of
Arduino as the TX pin
const int sensorPin= 0;
int air_quality;
#include <LiquidCrystal.h>
LiquidCrystal lcd(12,11, 5, 4, 3, 2);
void setup() {
pinMode(8, OUTPUT);
lcd.begin(16,2);
lcd.setCursor (0,0);
lcd.print ("Welcome ");
lcd.setCursor (0,1);
lcd.print ("Sensor Warming ");
delay(1000);
Serial.begin(115200);
esp8266.begin(115200); // your esp's baud rate might be different
  sendData("AT+RST\r\n",2000,DEBUG); // reset module
  sendData("AT+CWMODE=2\r\n",1000,DEBUG); // configure as access point
  sendData("AT+CIFSR\r\n",1000,DEBUG); // get ip address
  sendData("AT+CIPMUair_quality=1\r\n",1000,DEBUG); // configure for multiple
connections
  sendData("AT+CIPSERVER=1,80\r\n",1000,DEBUG); // turn on server on port 80
pinMode(sensorPin, INPUT);        //Gas sensor will be an input to the arduino
lcd.clear();
}
void loop() {
MQ135 gasSensor = MQ135(A0);
float air_quality = gasSensor.getPPM();
if(esp8266.available()) // check if the esp is sending a message
  {
   if(esp8266.find("+IPD,"))
   {
    delay(1000);
```

```
    int connectionId = esp8266.read()-48; /* We are subtracting 48 from the output
because the read() function returns the ASCII decimal value and the first decimal
number which is 0 starts at 48*/
    String webpage = "<h1>IOT Air Pollution Monitoring System</h1>";
      webpage += "<p><h2>";
      webpage+= " Air Quality is ";
      webpage+= air_quality;
      webpage+=" PPM";
      webpage += "<p>";
    if (air_quality<=1000)
{
  webpage+= "Fresh Air";
}
else if(air_quality<=2000 && air_quality>=1000)
{
  webpage+= "Poor Air";
}
else if (air_quality>=2000 )
{
webpage+= "Danger! Move to Fresh Air";
}
webpage += "</h2></p></body>";
    String cipSend = "AT+CIPSEND=";
    cipSend += connectionId;
    cipSend += ",";
    cipSend +=webpage.length();
    cipSend +="\r\n";

    sendData(cipSend,1000,DEBUG);
    sendData(webpage,1000,DEBUG);

    cipSend = "AT+CIPSEND=";
    cipSend += connectionId;
    cipSend += ",";
    cipSend +=webpage.length();
    cipSend +="\r\n";

    String closeCommand = "AT+CIPCLOSE=";
    closeCommand+=connectionId; // append connection id
    closeCommand+="\r\n";

    sendData(closeCommand,3000,DEBUG);
   }
  }
lcd.setCursor (0, 0);
lcd.print ("Air Quality is ");
lcd.print (air_quality);
lcd.print (" PPM ");
lcd.setCursor (0,1);
if (air_quality<=1000)
```

```
{
lcd.print("Fresh Air");
digitalWrite(8, LOW);
}
else if( air_quality>=1000 && air_quality<=2000 )
{
lcd.print("Poor Air, Open Windows");
digitalWrite(8, HIGH );
}
else if (air_quality>=2000 )
{
lcd.print("Danger! Move to Fresh Air");
digitalWrite(8, HIGH);   // turn the LED on
}
lcd.scrollDisplayLeft();
delay(1000);
}
String sendData(String command, const int timeout, boolean debug)
{
   String response = "";
   esp8266.print(command); // send the read character to the esp8266
   long int time = millis();
   while( (time+timeout) > millis())
   {
     while(esp8266.available())
     {
       // The esp has data so display its output to the serial window
       char c = esp8266.read(); // read the next character.
       response+=c;
     }
   }
   if(debug)
   {
     Serial.print(response);
   }
   return response;
}
```

3.7 SIMULATION

- The MQ135 sensor can sense NH3, NO2, alcohol, Benzene, smoke, CO_2 and some other gases, so it is perfect gas sensor for our **Air Quality Monitoring Project**.
- When we will connect it to Arduino then it will sense the gases, and we will get the Pollution level in PPM (parts per million).
- MQ135 gas sensor gives the output in form of voltage levels and we need to convert it into PPM.

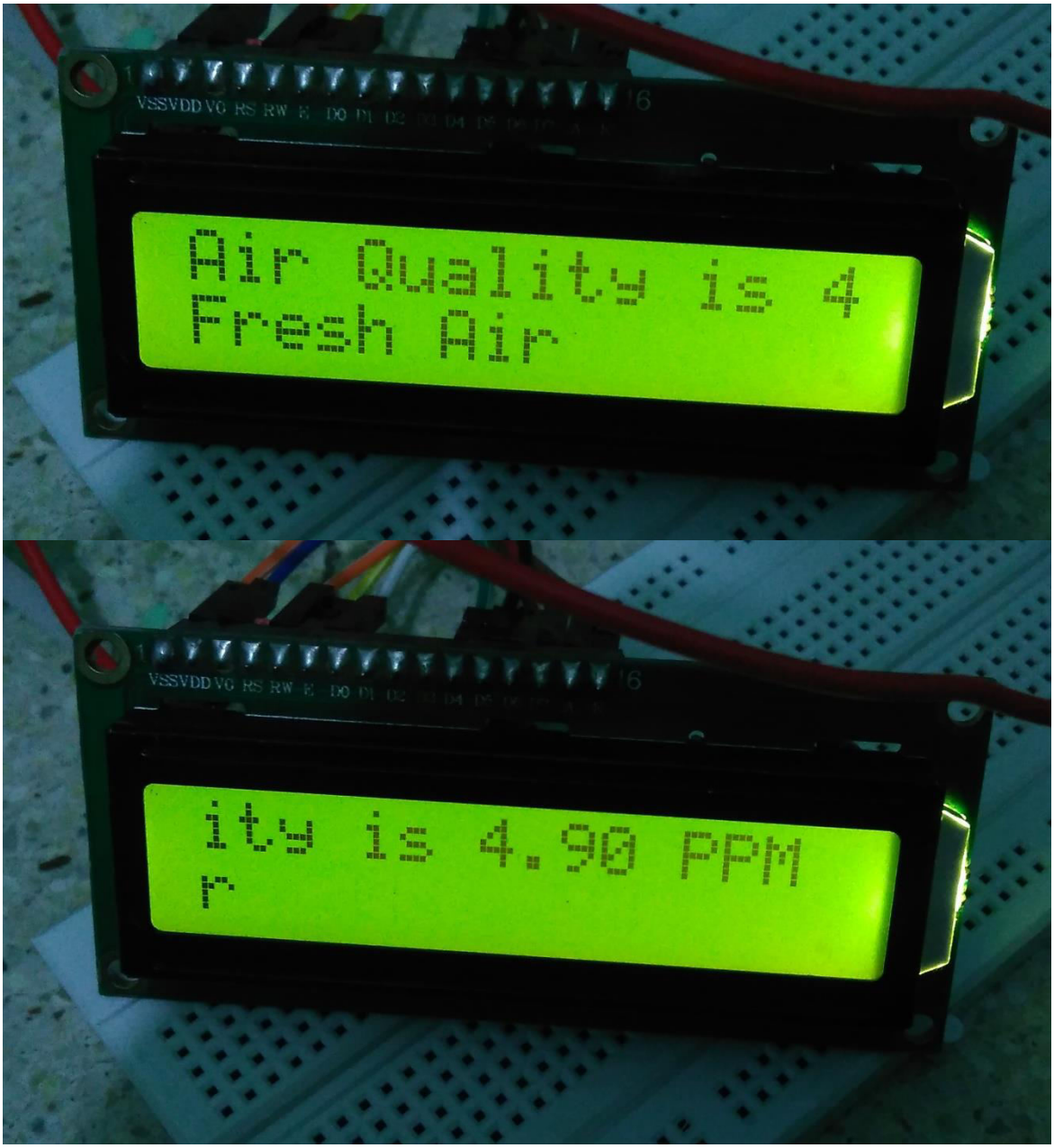


Figure 16: Simulation Result

- So for converting the output in PPM, here we have used a library for MQ135 sensor.
- Safe level of air quality is 350 PPM and it should not exceed 1000 PPM.
- When it exceeds the limit of 1000 PPM, then it starts cause Headaches, sleepiness and stagnant, stale, stuffy air and if exceeds beyond 2000 PPM then it can cause increased heart rate and many other diseases.
- When the value will be less than 1000 PPM, then the LCD and webpage will display “Fresh Air”.

- Whenever the value will increase 1000 PPM, then the buzzer will start beeping and the LCD and webpage will display “Poor Air, Open Windows”.
- If it will increase 2000 then the buzzer will keep beeping and the LCD and webpage will display “Danger! Move to fresh Air”.
- The Software used for uploading the program to arduino uno is arduino IDE 1.8.2.
- The program is then written in the arduino IDE window, then the MQ135.h library file is added to the program, the program is checked for the errors using the **verify** button.
- If no errors then the program is uploaded to arduino board.
- The simulation result is shown in the Figure 16.

CHAPTER 4

4.1 ADVANTAGES

- Easy to implement.
- Low cost.
- Assessing public health impacts caused by poor air quality.
- Determining whether an area is meeting the standards.
- Evaluating changes in air quality as a result of state implementation plans.

4.2 LIMITATIONS

- There are limited number of monitors that can be sited and maintained.
- The priority is for monitors to be located in areas to determine whether an air basin is in compliance with state and federal regulation.
- So the resulting measurements are not always a good indication of the burden of air pollution in a specific area.
- For example air monitors are not located near source of high air pollution since the goal is to obtain a picture of the ambient background levels.

4.3 APPLICATIONS

- **Urban Air Monitoring:-** Organizations responsible for urban and national air monitoring use our sensor-based instruments in place of or in support of traditional analyzer-based monitoring stations.
- **Industrial Perimeter:-** Industrial operators use these products to cost effectively monitor and manage emissions on their perimeter, which helps them improve relationships with regulators and communities.
- **Roadside Monitoring:**- For those interested in monitoring and managing emissions from transportation , these instruments create the opportunity for new monitoring sites, and minute by minute reporting.
- **Research & Consultancy:**- For researchers and consultants these instruments are new tools that gather robust air quality data at lower cost and higher temporal and spatial resolution than traditional analyzer instruments.

CONCLUSION

The system to monitor various parameters of environment using Arduino microcontroller, WSN and GSM Technology is proposed to improve quality of air. With the use of technologies like WSN and GSM enhances the process of monitoring various aspects of environment such as air quality monitoring issue proposed in this paper. The detection and monitoring of dangerous gases is taken into account in a serious manner and related precautions have been considered here in the form of an alert message and a buzzer so that the necessary action may be taken. It is estimated that this system will have a great acceptance in the market as it is a centralized system for a complete monitoring function.

This monitoring system can be enhanced by adding wireless network card for storage of values from sensors attached to microcontroller as well as more gas sensors could be used like Nitrogen dioxide (NO2), Ammonia (NH3), Sulfurated Hydrogen (H2S), alcohol etc. Another aspect of measuring particulate matter can be introduced to make it more advanced.

REFERENCES

[1] Anjaiah Guthi, "Implementation of an Efficient Noise and Air Pollution monitoring system using Internet of Things (IOT)", July 2016. www.ijarcce.com/upload/2016/july-16/IJARCCE%2047.pdf

[2] Navreetinder Kaur, Rita Mahajan and Deepak Bagai "Air quality monitoring system based on arduino microcontroller" June2016. https://www.ijirset.com/upload/2016/june/18_**Air**.pdf

[3] Al-Haija, Q. A Al-Qadeeb, H and Al-Lwaimi, "Case study: Monitoring of AIR quality in King Faisal University using a microcontroller and WSN", Procedia computer science, volume 21, pp. 517-521, 31 Dec 2013. www.sciencedirect.com/science/article/pii/S187705091300865X

[4] Vikhyat chaudhry, "Arduair: Air Quality Monitoring" 2013, www.ripublication.com/ijeem_spl/ijeemv4n6_19.pdf

[5] Dr. Xirong Li, "Applications of wireless sensors in monitoring Indoor Air Quality in the classroom environment" 2012. www.teo.unt.edu/ret/pdf/IAQ-report.pdf.