“IoT Based Industrial Fault Detection and Monitoring System.”



SONARGAON UNIVERSITY

Supervised by:

MD. RAIS UDDIN MOLLAH

Lecturer and Assistant Co-Ordinator (EEE)

Submitted By:

MIZANUR RAHMAN	ID: EEE1803015003
MD. YUSUF AHMED	ID: EEE1702011048
MD. NUR HOSSAIN	ID: EEE1803015018
MD. TUHIN HOSSAIN	ID: EEE1803015036
MD. MOHIDUL ISLAM	ID: EEE1803015052
MD. ASIBUL ISLAM	ID: EEE1803015040

A project submitted to the Department of Electrical & Electronics Engineering in partial fulfillment for the degree of Bachelor in Science Electrical & Electronics Engineering.

Department of Electrical and Electronics Engineering

Sonargaon University (SU)

147/1 Green Road, Panthapath. Tejgaon, Dhaka-1215, Bangladesh

Section:15A

Date of Submission: 03-06-2022

DECLARATION OF AUTHENTICATION

We certified that the project-based titled, — **"IoT Based Industrial Fault Detection and Monitoring System"** is completely our group work. All sources and knowledge for this paper which were found by other researchers are acknowledged by reference. Materials of work such as images, Figures, tables, and citations in this paper are accepted by our supervisor. We hereby declare that this project has not yet been previously submitted either in whole or in part, for any other degree or publication.

Signature of the Supervisor

……..

MD. RAIS UDDIN MOLLAH
Lecturer and Assistant Co-Ordinator (EEE)
Department of Electrical and Electronics Engineering

Signature of Authors,

……..	……..
MIZANUR RAHMAN	MD.YUSUF AHMED
EEE1803015003	EEE1803015036
……..	……..
MD.NUR HOSSAIN	MD.TUHIN HOSSAIN
EEE1803015018	EEE1803015036
……..	……..
MD.ASIBUL ISLAM	MD.MOHIDUL ISLAM
EEE1803015040	EEE1803015052

SUPERVISORS APPROVAL

This is certified that this project titled "**IoT Based Industrial Fault Detection and Monitoring System**" prepared by the following students under my direct supervision. This project work has been carried out by them in the laboratories of the Department of Electrical and Electronics Engineering under the Faculty of Engineering, Sonargaon University in partial fulfillment for the degree of Bachelor in Science Electrical & Electronics Engineering.

Supervisor

……..

MD. RAIS UDDIN MOLLAH
Lecturer and Assistant Co-Ordinator (EEE)
Department of Electrical and Electronics Engineering
Sonargaon University (SU)

ACKNOWLEDGEMENT

At First, we thank to Almighty of Allah for giving patience to complete the project work on — **IoT Based Industrial Fault Detection and Monitoring System**. We would like to convey our heartiest felicitation, best gratitude, humblest thanks to our honorable supervisor **Md. Rais Uddin Mollah, Asst. Coordinator** of Electrical & Electronics Engineering Sonargaon University (SU) for his sincere guidance, kind Co-operation, Supervision, Constant encouragement and valuable suggestion. We would like to express heartiest gratitude to our Head (EEE), Prof. **Dr. M. Bashir Uddin** and Coordinator, **Ferdous khan** for their valuable support and guidance. Also, thanks to our parents, all friends and lab assistants who helped us and inspired us to do the best.

Finally, we wish to acknowledge those who directly or indirectly helped us in connection with the work.

ABSTRACT

The Internet of Things refers to connecting things and people through internet, it has imposed itself as the new business practices in different sectors. To make quick and efficient response in real time, IoT enhances the way and provides emergency managers with the necessary information and communication to make use of those assets. In this paper it is proposed that a quick response for fire hazards is evaluated and examined by using IoT based model. Fire is one of the major reasons of accidental deaths in the world. Gas leakages may lead to fires leading to huge industrial losses, also instant fire detection is needed in case of furnace blasts or other conditions. Also, low lighting in industries may create improper work conditions increasing the probability of accidents. To implement this proposed system a low-cost Wi-Fi module, gas detection sensor, Flame detection sensor, buzzer to alert and temperature sensors are used. The sensors detects and alerts the local emergency with the data collected by the system, and alerts organizations like fire departments, police stations and hospitals by sending the exact location to both user and operator through module which all are well connected with. Thus, an integrated intelligent system is designed through IoT to manage such hazards where innocent lives and property.

Keywords: Node MCU, Temperature sensor, Flame sensor, Gas sensor, IoT, Industrial Safety Alert System.

TABLE OF CONTENTS

Acknowledgement ii

Abstract iii

INTRODUCTION

CHAPTER-1

1.1 Introduction 1

1.2 Objectives 3

1.3 Methodology 3

1.4 Organization of Book 3

LITERATURE REVIEW

CHAPTER 2

2.1 Literature Review 4

CHAPTER-3 **Circuit Component's**

3.1 Introduction 7

3.2 Node MCU 7

3.3 SMPS 11

3.4 5V Regulator IC 12

3.5 Resistor 13

3.6 Flame Sensor 14

3.7 MQ2 Smoke/Gas Sensor 15

3.8 Buzzer 17
3.9 LCD Display 18
3.10 Temperature Sensor 21
3.11 Arduino Software 23
3.12 Proteus Software 27
3.13 Remote XY 28

CHAPTER-4 Design and Implementation
4.1 System Description 33
4.2 Block Diagram 34
4.3 Schematic Diagram 35
4.4 Working Principle 36

CHAPTER-5 Result & Performances
5.1 Result & Performances 37

CHAPTER-4 Discussion and Conclusion
6.1 Discussion 40
6.2 Advantage 40
6.3 Application 41

6.4 Limitation 41
6.5 Future Scope 41
6.6 Conclusion 42

Reference 43

Appendix 45

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
3.1	Node MCU	8
3.2	Node MCU Schematic Diagram	9
3.3	Node MCU PIN out	10
3.4	SMPS POWER SUPPLY	11
3.5	5V Regulator IC	13
3.6	Flame Sensor	15
3.7	MQ2 Smoke/Gas Sensor	16
3.8	Buzzer	18
3.9	16*2 LCD display	19
3.10	DHT11 Temperature Sensor	22
3.11	Arduino Software Interface IDE	23
3.12	Proteus Software Interface	27
3.13	Remote XY Software.	29
4.1	System Circuit Design in Proteus	33
4.2	Block Diagram	34
4.3	Schematic Diagram	35

CHAPTER 1

INTRODUCTION

1.1 Introduction:

As the society is growing with various developments, the outmoded forms of storing various food products in cold store rooms is failing to satisfy human needs. As the society is growing with various developments, the outmoded forms of storing various food products in cold store rooms fails to satisfy human need. Through the monitoring of the temperature and humidity inside cold storage rooms, the goodness of the products can be ensured for a longer time. Recent research has revealed that operations of wireless sensor systems are largely affected by their on-board temperature [1]. We can implement sensors in wide area over the machines and instruments and control and monitor the circumstances by using concept of IoT [2].

In this paper we have designed a gateway which will be the central part of this whole system. The function of the gateway is to gather data, process them, upload them and process user control information. If the network connection is not established then the data will be stored and upon reestablishment of the network it will be uploaded. The terms of "things" in the IoT vision is very broad and includes a variety of physical elements. The terms of things include portable personal items such as smart phones, tablets and digital cameras. With the rapid increase in the number of users of and digital cameras. Furthermore, IoT includes elements of our environments (be it home, car or office), and things equipped with RFID tags connected to a gateway device. From those mentioned so far, a huge number of devices and things will be connected to the Internet, each providing data and information and some even services.

Gas leakage is a serious problem and nowadays it is observed in many places like residences, industries, and vehicles like Compressed Natural Gas (CNG), buses, cars, etc. It is noticed that due to gas leakage, dangerous accidents occur.[3] The Liquefied petroleum gas (LPG), or propane, is a flammable mixture of hydrocarbon gases used as fuel in many applications like homes, hostels, industries, automobiles, and vehicles

because of its desirable properties which include high calorific value, less smoke, less soot, and meager harm to the environment. Liquid petroleum gas (LPG) is highly inflammable and can burn even at some distance from the source of leakage.[4] This energy source is primarily composed of propane and butane which are highly flammable chemical compounds. These gases can catch fire easily. In homes, LPG is used mainly for cooking purposes.

When a leak occurs, the leaked gases may lead to an explosion. Gas leakage leads to various accidents resulting in both material loss and human injuries. Home fires have been occurring frequently and the threat to human lives and properties has been growing in recent years.[5] The risks of explosion, fire, suffocation are based on their physical properties such toxicity, flammability, etc. The number of deaths due to the explosion of gas cylinders has been increasing in recent years. The Bhopal gas tragedy is an example of accidents due to gas leakage.

The reason for such explosions is due to substandard cylinders, old valves, no regular checking of gas cylinders, worn out regulators and a lack of awareness of handling gas cylinders.[6] Therefore, the gas leakage should be detected and controlled to protect people from danger. An odorant such as ethane thiol is added to LPG, so that leaks can be detected easily by most people. However, some people who have a reduced sense of smell may not be able to rely upon this inherent safety mechanism.[7] A gas leakage detector becomes vital and helps to protect people from the dangers of gas leakage. A number of research papers have been published on gas leakage detection techniques.

1.2 Objective:

In the course of design and installation of a **IoT Based Industrial Fault Detection & Monitoring System**, the following criteria are set to fulfill its objectives:

- To study of the **IoT Based Industrial Fault Detection & Monitoring System**
- To design and construct of a temperature, fire and gas detection system.
- To implement and study of take proper step to reduce these incidents.
- To take necessary notes from the project for future improvements.

1.3 Methodology:

- Creating an idea for Design and construction of **IoT Based Industrial Fault Detection & Monitoring System**. And designing a block diagram & circuit diagram to know which components need to construct it.

- Collecting the all components and programming for the micro controller to controlled the system.

- Setting all components in a PCB board & soldering. Then assembling the all block in a board and finally run the system & checking.

1.4 Organization of Book:

This project book consists of five chapter. The first chapter contains the statement of the introduction, objectives of the study and the project organization. Chapter two contains literature review in details. Chapter three describes the project theory and working principle, , working principle and shows the complete prototype of the project that we have built.. Chapter four deals with the result and discussion. In the final chapter, we have conclusion and future work of the project and also about some aspects we had to overcome while doing the project and lastly, we gave the conclusion of the book.

CHAPTER 2

LIRERATURE REVIEW

2.1 LITRATURE REVIEW:

In this section topics related to A Real Time Fire and Smoke Detection System are included. These provide a sampling of problems appropriate for application of A Real Time Fire and Smoke Detection System. The references are summarized below.

[1] H. S. Raju, Sanath Shenoy Siemens Technologies and Services Private Limited in "Real-Time remote monitoring and operation of commercial Devices using IoT and Cloud" 978-1-5090-5256-1/16/31.00- 2016 IEEE have explained recent times there has been significant advances in managing differing types of sensors and industrial devices by IoT (Internet of Things) protocol. Alongside the availability of massive amount of processing power provided by the Cloud new opportunities have emerged for complete automation of commercial devices. IoT features a vast application in several sectors and domains which are yet to be explored. In industry domain, industrial automation is need of the hour to extend the time to plug with high grade quality and enhanced productivity. During this paper we explain proven ways to utilize the capabilities of Cloud and IoT to regulate the device and analyze the info generated by them.

[2] "IoT-based Integration of IEC 61131 Industrial Automation Systems: The case of UML4IoT" by Foivos Christoulakis, 22 Page 20-28 © MANTECH PUBLICATIONS 2018. All Rights Reserved, Kleanthis Thramboulidis- 978-1-5090-0873-5/16/$31.00 ©2016 IEEE have explained that Internet of Things (IoT) plays a key role in the new generation of industrial automation systems (IASs). Evolving IoT standards if effectively used may address many challenges in the development of IASs. However, the use of the IoT and the REST architectural paradigm that IoT is based on, is not an easy task for the automation engineer. In this paper, a model driven system engineering process is adopted for IASs and it is extended to exploit IoT standardization efforts in IEC 61131

based system. IoT is considered as an enabling technology for the integration of cyber physical and cyber components of the system and humans, bringing into the industrial automation domain the benefits of this technology. A UML profile for IoT is exploited to automate the generation process of the IoT wrapper, i.e., the software layer that is required on top of the IEC 61131 cyber part of the cyber physical component to expose its functionality to the modern IoT environment. A prototype implementation and performance measurements prove the feasibility of the presented approach.

A. Need for Wireless Fire Detection Systems using IOT: The interconnection of physical devices embedded with electronics, software and sensors, which is capable of collecting required data from the surrounding and sent those data over internet is called IOT. The fire detection composed of all of the techniques and processes that ensures the early detection of a fire. There are three main categories Smoke detection, Flame detection and Temperature detection. The automatic fire detection system will do a real-time surveillance. Mainly an automatic fire alarm system is designed for high-rise buildings. To do early extinguishing of a fire disaster, large numbers of detectors are deployed in buildings, which will periodically measure smoke concentration or temperature. There are many techniques we had been already used to detect fire, such as fire detection using sensors, image processing, CCTV technology.

B. Unified Smoke and Flame Detection for Intelligent Surveillance System: This paper explains the way of unification of flame and smoke detection algorithms by merging the common steps into a single processing flow. Smoke detection and flame detection can be done using color probability, boundary roughness, edge density and area variability filtering processes. The obtained preliminary results show that, by processing smoke and flame in parallel, we can reduce the processing time. The results obtained from unified processing part can be used to feed as input for different tasks of intelligent surveillance systems.

C. Design and Implementation of Smoke Early Warning System: In this paper the author proposes a simple hardware design, which consists of different fire sensors that enables

every user to use this wireless fire security system. The challenges, while designing the nodes with different types of fire sensors are discussed and the methods to overcome this problem is analyzed. The system is interfaced with the many types of sensors to sense different fire sources like gas leakage, smoke, and heat. This circuit is fully controlled by the PIC micro-controller. All the sensors and detectors are wired to PIC micro-controller. The PIC micro-controller will periodically monitor all the connected sensors and when it senses any security problem then notification sent to the PC central monitoring station wirelessly. The distance should be in a range of 300m indoor/1500m outdoor using Zigbee. Also, gas, light, smoke, IR, temperature & humidity sensors can be interfaced with this micro-controller to find out abnormal fire conditions in the surroundings in all possible ways. ARM7 based micro-controller can also be used to build such systems.

Chapter -3

Circuit Component's

3.1 Introduction: In this chapter we will discuss about Node MCU, SMPS, Flame Sensor, Gas Sensor, Temperature Sensor, LCD Display, Buzzer, Voltage Regulator IC, Resistor, Capacitor.

1. Node MCU
2. SMPS
3. Flame Sensor
4. Gas Sensor
5. Temperature Sensor
6. LCD Display
7. Buzzer
8. Voltage Regulator IC
9. Resistor
10. Capacitor

3.2 Node MCU:

Node MCU is an open-source firmware for which open-source prototyping board designs are available. The name "Node MCU" combines "node" and "MCU" (micro-controller unit). The term "Node MCU" strictly speaking refers to the firmware rather than the associated development kits. Both the firmware and prototyping board designs are open source.

The firmware uses the Lua scripting language. The firmware is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. It uses many open-source

projects, such as lua-cjson and SPIFFS. Due to resource constraints, users need to select the modules relevant for them

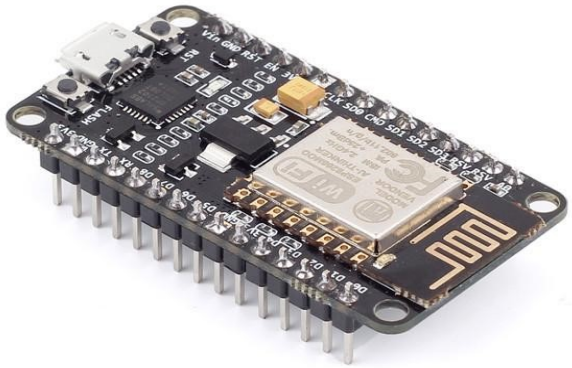


Figure 3.1: Node MCU

project and build a firmware tailored to their needs. Support for the 32-bit ESP32 has also been implemented.

The prototyping hardware typically used is a circuit board functioning as a dual in-line package (DIP) which integrates a USB controller with a smaller surface-mounted board containing the MCU and antenna. The choice of the DIP format allows for easy prototyping on breadboards. The design was initially was based on the ESP-12 module of the ESP8266, which is a Wi-Fi SoC integrated with a Ten silica Extensa LX106 core, widely used in IoT applications.

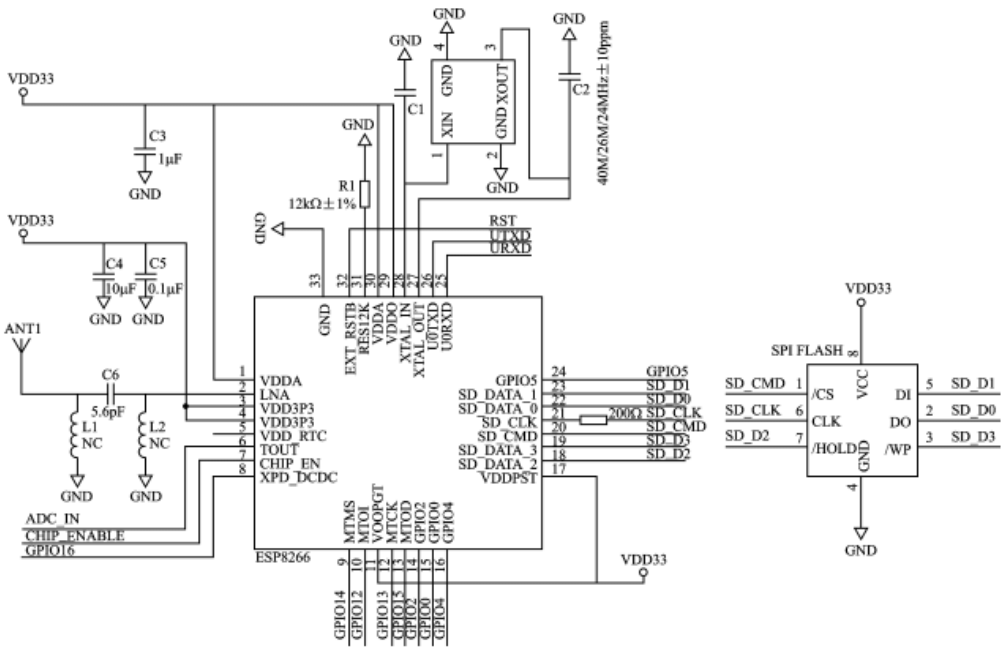


Figure 3.2: Node MCU Schematic Diagram

This an open source IoT platform. It includes firmware which runs on the ESP8266 Wi-Fi SoC from Express if Systems, and hardware which is based on the ESP-12 module. The term "Node MCU" by default refers to the firmware rather than the development kits. The firmware uses the Lua scripting language. It is based on the Lua project, and built on the Espressif Non-OS SDK for ESP8266. Node MCU was created shortly after the ESP8266 came out. On December 30, 2013, Espressif Systems began production of the ESP8266. The ESP8266 is a Wi-Fi SoC integrated with a Tensilica Xtensa LX106 core, widely used in IoT applications (see related projects). Node MCU started on 13 Oct 2014, when Hong committed the first file of node MCU-firmware to GitHub. Two months later, the project expanded to include an open-hardware platform when developer Huang R committed the gerber file of an ESP8266 board, named devkit v0.9. Later that month, Tuan PM ported MQTT client library from Contiki to the ESP8266 SoC platform, and committed to Node MCU project, then Node MCU was able to support the MQTT IoT protocol, using Lua to access the MQTT broker. Another important update was made on 30 Jan 2015, when Devsaurus ported the u8glibto Node MCU project, enabling Node MCU to easily drive LCD, Screen, OLED, even VGA displays. In summer 2015 the creators

abandoned the firmware project and a group of independent contributors took over. By summer 2016 the Node MCU included more than 40 different modules. Due to resource constraints users need to select the modules relevant for their project and build a firmware tailored to their needs.

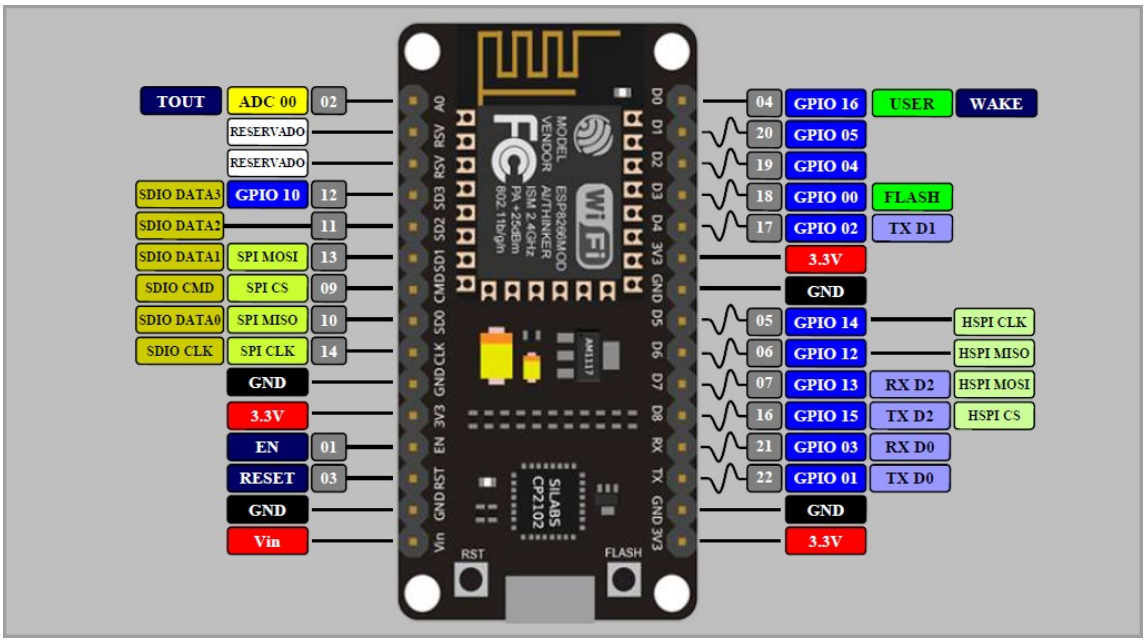


Figure 3.3: Node MCU Pin Out

Node MCU V3 ESP8266 ESP-12E is Wi-Fi development board that helps you to prototype your IoT product with few Lua script lines, or through Arduino IDE. The board is based on ESP8266 ESP-12E variant, unlike other ESP-12E, you won't need to buy a separate breakout board, USB to serial adapter, or even solder it to a PCB to get started, you will only need a USB cable (Micro USB).

Features:

1. Communication interface voltage: 3.3V.
2. Antenna type: Built-in PCB antenna is available.
3. Wireless 802.11 b/g/n standard
4. Wi-Fi at 2.4GHz, support WPA / WPA2 security mode

5. Support STA/AP/STA + AP three operating modes
6. Built-in TCP/IP protocol stack to support multiple TCP Client connections (5 MAX)
7. D0 ~ D8, SD1 ~ SD3: used as GPIO, PWM, IIC, etc., port driver capability 15mA
8. AD0: 1 channel ADC
9. Power input: 4.5V ~ 9V (10VMAX), USB-powered
10. Current: continuous transmission: ≈70mA (200mA MAX), Standby: <200uA
11. Transfer rate: 110-460800bps
12. Support UART / GPIO data communication interface
13. Remote firmware upgrade (OTA)
14. Flash size: 4MByte.

3.3 SMPS:

A switched-mode power supply is an electronic power supply that incorporates a switching regulator to convert electrical power efficiently. Like other power supplies, an SMPS transfers power from a DC or AC source to DC loads, such as a personal computer, while converting voltage and current characteristics.



Figure 3.4: SMPS Power Supply

A switched-mode power supply (switching-mode power supply, switch-mode power supply, switched power supply, SMPS, or switcher) is an electronic power supply that incorporates a switching regulator to convert electrical power efficiently.

Like other power supplies, an SMPS transfers power from a DC or AC source (often mains power) to DC loads, such as a personal computer, while converting voltage and current characteristics. Unlike a linear power supply, the pass transistor of a switching-mode supply continually switches between low-dissipation, full-on and full-off states, and spends very little time in the high dissipation transitions, which minimizes wasted energy. Ideally, a switched-mode power supply dissipates no power. Voltage regulation is achieved by varying the ratio of on-to-off time. In contrast, a linear power supply regulates the output voltage by continually dissipating power in the pass transistor. This higher power conversion efficiency is an important advantage of a switched-mode power supply. Switched-mode power supplies may also be substantially smaller and lighter than a linear supply due to the smaller transformer size and weight.

Switching regulators are used as replacements for linear regulators when higher efficiency, smaller size or lighter weight are required. They are, however, more complicated; their switching currents can cause electrical noise problems if not carefully suppressed, and simple designs may have a poor power factor.

3.4 5V Regulator IC:

Voltage sources in a circuit may have fluctuations resulting in not providing fixed voltage outputs. A voltage regulator IC maintains the output voltage at a constant value. 7805 IC, a member of 78xx series of fixed linear voltage regulators used to maintain such fluctuations, is a popular voltage regulator integrated circuit (IC). The xx in 78xx indicates the output voltage it provides. 7805 IC provides +5 volts regulated power supply with provisions to add a heat sink.

7805 IC Rating

- Input voltage range 7V- 35V
- Current rating IC = 1A
- Output voltage range Vmax. =5.2V, VMin=4.8V

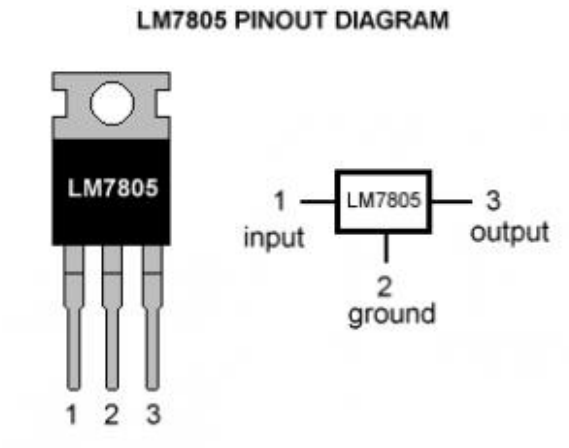
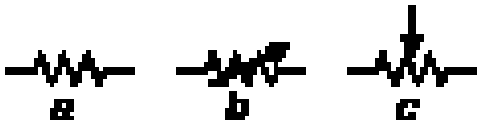


Figure 3.5: 5V Regulator IC

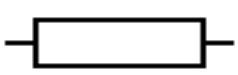
3.5 Resistor:

A **resistor** is a passive two-terminal electrical component that implements electrical resistance as a circuit element. In electronic circuits, resistors are used to reduce current flow, adjust signal levels, to divide voltages, bias active elements, and terminate transmission lines, among other uses. High-power resistors that can dissipate many watts of electrical power as heat, may be used as part of motor controls, in power distribution systems, or as test loads for generators. Fixed resistors have resistances that only change slightly with temperature, time or operating voltage. Variable resistors can be used to adjust circuit elements (such as a volume control or a lamp dimmer), or as sensing devices for heat, light, humidity, force, or chemical activity.

Two typical schematic diagram symbols are as follows:



(a) resistor, (b) rheostat (variable resistor), and (c) potentiometer



IEC resistor symbol

One common scheme is the RKM code following IEC 60062. It avoids using a decimal separator and replaces the decimal separator with a letter loosely associated with SI prefixes corresponding with the part's resistance. For example, *8K2* as part marking code, in a circuit diagram or in a bill of materials (BOM) indicates a resistor value of 8.2 kΩ. Additional zeros imply a tighter tolerance, for example *15M0* for three significant digits. When the value can be expressed without the need for a prefix (that is, multiplicator 1), an "R" is used instead of the decimal separator. For example, *1R2* indicates 1.2 Ω, and *18R* indicates 18 Ω.

Specifications:

- Resistance: 220 Ohms
- Power (Watts): 0.25W, 1/4W
- Temperature Coefficient: 350ppm/Celsius
- Tolerance: +/- 5%
- Case: Axial
- Size: 0.094" Dia x 0.248" L (2.40mm x 6.30mm)

3.6 Flame Sensor:

The flame sensor can detect flame and infrared light sources with wavelengths ranging from 760 nm to 1100 nm. It uses the LM393 comparator chip, which gives a clean, stable digital output signal and driving ability of 15 mA. This flame detector that can be used in fire alarms and other fire detecting devices.

Specifications:

- LM393 comparator chip
- Detection Range: 760 nm to 1100 nm
- Operating Voltage: 3.3 V to 5 V
- Maximum Output Current: 15 mA

- Digital Outputs: 0 and 1
- Detection Angle: about 60 degrees
- Adjustable sensitivity via potentiometer
- LED lights indicators: power (red) and
- digital switching output (green)
- Fixed bolt holes for easy installation



Figure 3.6: Flame Sensor

3.7 MQ 2 Smoke/Gas Sensor:

The utility model can be used for Smoke/gas leakage monitoring devices in families and factories, and is suitable for the detection of liquefied petroleum gas, butane, propane, methane, Hydrogen, smoke, etc. This is a very easy to use low-cost semiconductor Gas sensor Module with analog and digital output.

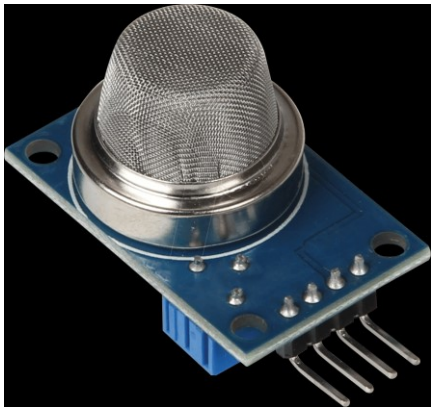


Figure 3.7: MQ 2 Smoke Sensor

Features:

- Adopt high quality double panel design, with power indication and TTL signal output indication.
- It has DO switch signal (TTL) output and AO analog signal output.
- TTL output valid signal is low level. When the output is low, the signal light is on, and the micro controller or relay module can be directly connected.
- The analog output voltage increases with the concentration, the higher the voltage.
- It has better sensitivity to liquefied petroleum gas, natural gas, urban gas and smoke.
- MQ-2 MQ2 Smoke Gas LPG Butane Methane Sensor Detector Module
- With four screw holes, easy to locate.
- Product size: 32 (L), *20 (W), *22 (H)
- With long service life and reliable stability.
- Fast response recovery features

Specifications:

- Input voltage: DC5V
- Power dissipation (current): 150mA
- DO output: TTL, numeric quantities 0 and 1 (0.1 and 5V)

- AO output: 0.1-0.3V (relatively pollution-free), the highest concentration of about 4V voltage
- Special reminder: after the sensor is energized, you need to preheat 20S or so, the data to be stable, sensor heating is a normal phenomenon, because the internal heating wire, if hot, it is not normal.

Connection mode:

- VCC: power supply positive (5V)
- GND: power supply negative pole
- DO: TTL switch signal output
- AO: analog signal output
- Functions: This version supporting test procedures
- Using chips: AT89S52
- Crystal oscillator: 11.0592MHZ
- Since this Gas Sensor module is sensitive to smoke it can be used in for fire detection. MQ2 Gas Sensor is also sensitive to flammable/combustible gasses like LPG, Propane & Hydrogen.
- Baud rate: 9600

3.8 Buzzer:

Piezo elements are actually one of the coolest crystals you'll play with. Flex them, they generate a voltage (pretty high, actually). Pass power through them, and they'll flex. Do it fast enough, and they'll start to make a sound! So besides making annoying little beep noises, you could use it as a sensor to detect a knock or other similar vibration.



Figure 3.8: Buzzer

Specifications:

- Operating Voltage: 1.0 ~ 20.0V
- Rated Voltage: 3.0V
- Diameter: Ø12.5mm
- Total Height: 6.3mm
- Capacitance: 15,000pF
- Current Consumption: ≤2mA
- Resonance Frequency: 4,000Hz
- Sound Pressure Level at 10cm: ≥75db
- Self-Drive: No

3.9 LCD Display:

Some of the most common LCDs connected to the Arduino Uno are 16*2. This means 16 characters per line by 2 lines and in recent years the LCD is finding widespread use replacing LED's.

Fortunately, a very popular standard exists which allows us to communicate with the vast majority of LCDs regardless of their manufacturer. The standard is referred to as HD44780U, which refers to the controller chip which receives data from an external source and communicates directly with the LCD. The 44780 standard requires 3 control lines as well as either 4 or 8 I/O lines for the data bus. The user may select whether the LCD is to operate with a 4-bit data bus or an 8-bit data bus. If a 4-bit data bus is used the LCD will require a total of 7 data lines (3 control lines plus the 4 lines for the data bus). If an 8-bit data bus is used the LCD will require a total of 11 data lines (3 control lines plus the 8 lines for the data bus).



Figure 3.9: 16*2 LCD Display

Features of LCD Display:

- Display data RAM
- Character generator ROM
- 160 different 57 dot matrix character patterns.
- Display data RAM and Character generator RAM may be accessed by the microprocessor.
- Numerous instruction
- Clear Display, Cursor Home, Display ON/OFF, Cursor ON/OFF, Blink Character, Cursor Shift.
- Built in reset circuit is triggered at power ON.

Pin Symbol I/O Description:

PIN NO	Symbol	Fuction
1	VSS	GND
2	VDD	+5V
3	V0	Contrast adjustment
4	RS	H/L Register select signal
5	R/W	H/L Read/Write signal
6	E	H/L Enable signal
7	DB0	H/L Data bus line
8	DB1	H/L Data bus line
9	DB2	H/L Data bus line
10	DB3	H/L Data bus line
11	DB4	H/L Data bus line
12	DB5	H/L Data bus line
13	DB6	H/L Data bus line
14	DB7	H/L Data bus line
15	A	+4.2V for LED
16	K	Power supply for BKL(0V)

Logic Status on control lines

- E- 0 Access to LCD Disabled
- -1 Access to LCD enabled
- R/W – 0 Writing data to LCD
- 1 Reading data from LCD
- RS – 0 Instructions

Writing data to the LCD

- Set R/W bit to low
- Set RS bit to logic 0 or 1
- Set data to data lines
- Set E line to high
- Set E line to low

Read data from data lines

- Set R/W bit to high
- Set RS bit to logic 0 or 1
- Set data to data lines
- Set E line to high
- Set E line to low

3.10 Temperature Sensor:

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). It's fairly simple to use, but requires careful timing to grab data.

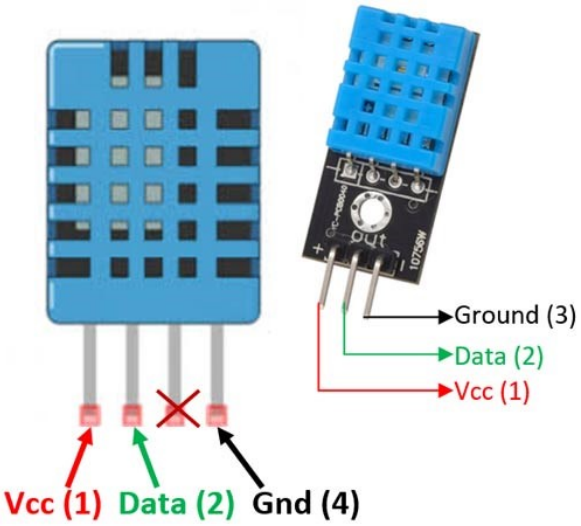


Figure3.10**:** DHT11 Temperature Sensor

DHT11 Specifications:

- Operating Voltage: 3.5V to 5.5V
- Operating current: 0.3mA (measuring) 60uA (standby)
- Output: Serial data
- Temperature Range: 0°C to 50°C
- Humidity Range: 20% to 90%
- Resolution: Temperature and Humidity both are 16-bit
- Accuracy: ±1°C and ±1%

3.11 Arduino software:

The smart microcontroller unit named as Arduino Nano can be programmed with the Arduino software. There is no any requirement for installing other software rather than Arduino. Firstly, Select "Arduino Nano from the Tools, Board menu (according to the microcontroller on your board). The IC used named as ATmega328 on the Arduino Nano comes pre burned with a boot loader that allows you to upload new code to it without the use of an external hardware programmed.

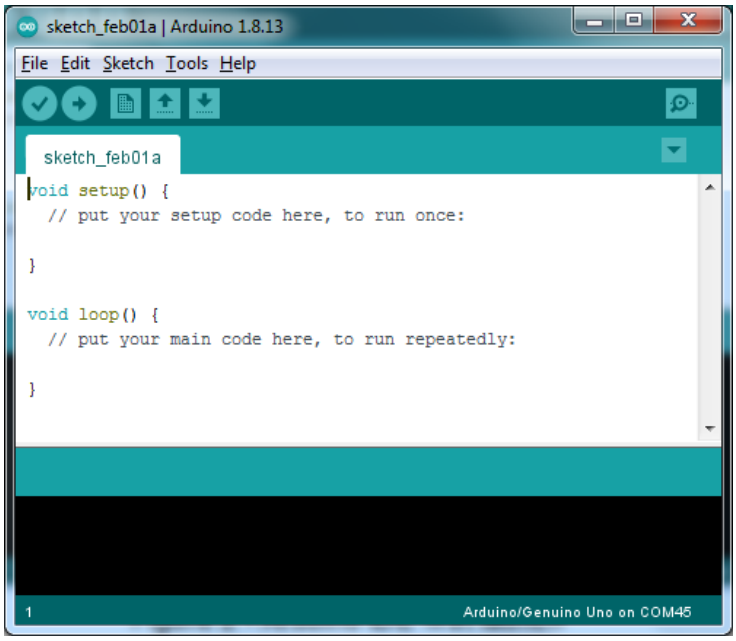


Figure 3.11: Arduino Software Interface IDE.

Communication is using the original STK500 protocol (reference, C header files). We can also bypass the boot loader and programs the microcontroller through the ICSP (In Circuit Serial Programming) header. The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available. The ATmega16U2/8U2 is loaded with a DFU boot loader, which can be activated by: On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then 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.

The Arduino Nano is one of the latest smart microcontroller units and has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL at (5V) with serial communication, which is available on digital pins 0 -(RX) for receive the data and pin no.1 (TX) for transmit the data. An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, an .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.

Writing Sketches

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension. ino. The editor has features for cutting/pasting and for searching/replacing text. 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 righthand 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.

Find in Reference

This is the only interactive function of the Help menu: it directly selects the relevant page in the local copy of the Reference for the function or command under the cursor.

Sketchbook

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

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

Beginning with version 1.0, files are saved with a .ino file extension. Previous versions use the .pde extension. You may still open .pde named files in version 1.0 and later, the software will automatically rename the extension to .ino.

Tabs, Multiple Files, and Compilation

Allows you 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 an 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 you upload a sketch, you're using the Arduino bootloader, a small program that has been loaded on to the microcontroller on your board. It allows you to upload code

without using any additional hardware. The bootloader is active for a few seconds when the board resets; then it starts whichever sketch was most recently uploaded to the microcontroller. The bootloader will blink the on-board (pin 13) LED when it starts (i.e. when the board resets).

Libraries

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from the Sketch > Import Library menu. This will insert one or more #include statements at the top of the sketch and compile the library with your sketch. Because libraries are uploaded to the board with your 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 your code.

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

Third-Party Hardware

Support for third-party hardware can be added to the hardware directory of your sketchbook directory. Platforms installed there may include board definitions (which appear in the board menu), core libraries, bootloaders, and programmer definitions. To install, create the hardware directory, then unzip the third-party platform into its own sub-directory. (Don't use "Arduino" as the sub-directory name or you'll override the built-in Arduino platform.) To uninstall, simply delete its directory. For details on creating packages for third-party hardware, see the Arduino IDE 1.5 3rd party Hardware specification.

3.12 Proteus Software:

The Proteus Design Suite is a proprietary software tool suite used primarily for electronic design automation. The software is used mainly by electronic design engineers and technicians to create schematics and electronic prints for manufacturing printed circuit boards.

The first version of what is now the Proteus Design Suite was called PC-B and was written by the company chairman, John Jameson, for DOS in 1988. Schematic Capture support followed in 1990, with a port to the Windows environment shortly thereafter. Mixed mode SPICE Simulation was first integrated into Proteus in 1996 and microcontroller simulation then arrived in Proteus in 1998. Shape based auto routing was added in 2002 and 2006 saw another major product update with 3D Board Visualization. More recently, a dedicated IDE for simulation was added in 2011 and MCAD import/export was included in 2015. Support for high-speed design was added in 2017.Feature led product releases are typically biannual, while maintenance-based service packs are released as required.

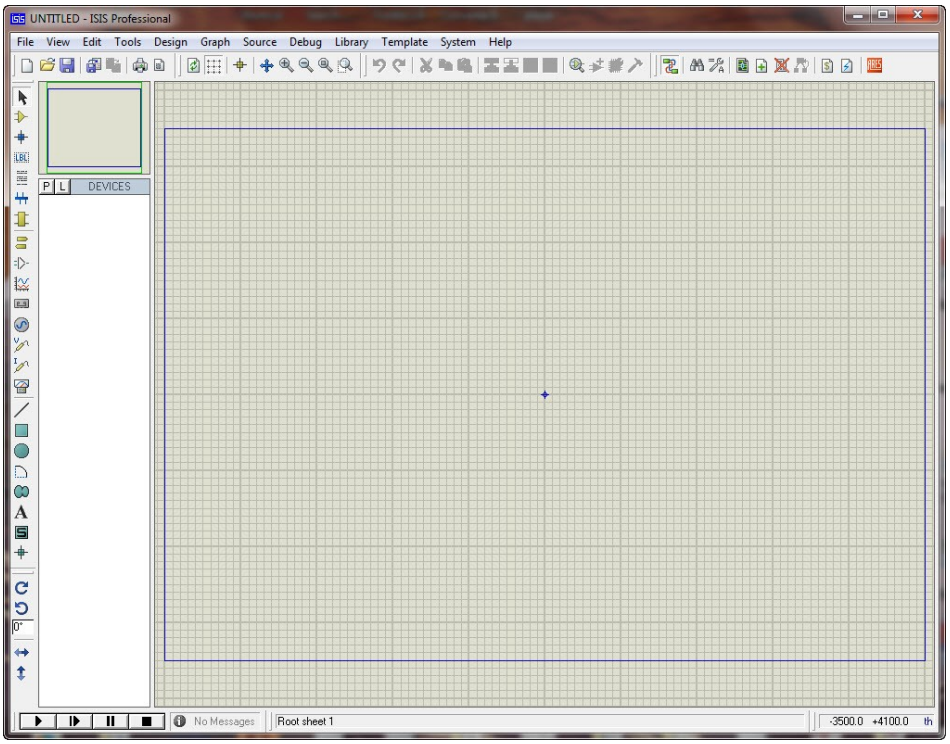


Figure 3.12: Proteus Software Interface.

3.13 Remote XY:

Remote XY is easy way to make and use a mobile graphical user interface for controller boards to control via smartphone or tablet. The system includes:

- Editor of mobile graphical interfaces for controller boards, located on the site remotexy.com
- Mobile app Remote XY that allows to connect to the controller and control it via graphical interface. Download app.

Distinctive features:

- The interface structure is stored in the controller. When connected, there is no interaction with servers to download the interface. The interface structure is downloaded to the mobile application from the controller.
- One mobile application can manage all your devices. The number of devices is not limited.

Supported connection methods:

- Internet over Cloud Server;
- Wi-Fi client and access point;
- Bluetooth;
- Ethernet by IP or URL;
- USB OTG (Android only that support USB OTG);

Supported boards:

- Arduino UNO, MEGA, Leonardo, Pro Mini, Nano, MICRO and compatible AVR boards;
- ESP8266 boards;
- ESP32 boards;
- Chip KIT UNO32, Chip KIT uC32, Chip KIT Max32;

Supported communication modules:

- Bluetooth HC-05, HC-06 or compatible;
- Bluetooth BLE HM-10 or compatible;
- ESP8266 as modem;
- Ethernet W5100;

Supported IDE:

- Arduino IDE;
- FLProg IDE;
- MPIDE;

Supported mobile OS:

- Android;
- iOS;

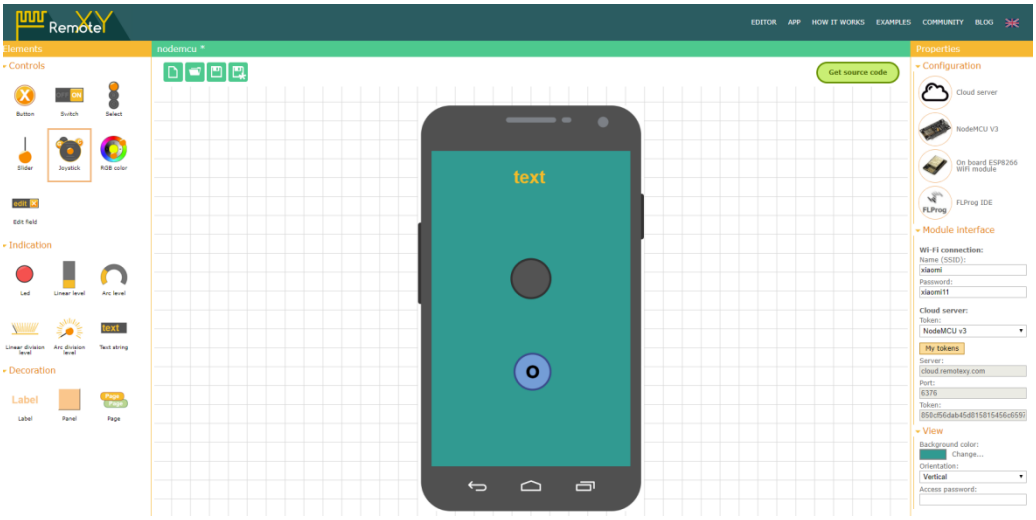


Figure 3.13: Remote XY PC Interface

Remote XY is easy way to make a unique graphical interface to control microcontroller device via mobile application, Arduino for example.

Remote XY allows:

- To develop any graphical management interface, using the control, display and decoration elements any combination thereof. You can develop the graphical interface for any task, placing the elements on the screen using the online editor. Online editor posted on the website remotexy.com.
- After the development of the graphical interface, you get the source code for the microcontroller that implements your interface. The source code provides a structure for interaction between your program with the controls and display. Thus, you can easily integrate the control system into your task for which you are developing the device.
- To manage microcontroller device using your smartphone or tablet with the graphical interface. For manage used mobile application Remote XY.
- Using one mobile application, you can manage a large number of devices with different graphical management interfaces. As the interface description is stored on board the microcontroller device.

The source code for the microcontroller, for example a sketch for the Arduino, you opened it in the IDE. In the simplest case, you can download the sketch in Arduino board and test your interface immediately, connecting via a smartphone. But often you need to integrate the developed graphical interface with your problem you want to solve it using the Arduino. Next, we will show how it is easy to do.

The generated source code has a part that is starts with tag **Remote XY include library** and ends with tag **END Remote XY include**. This part of the code defines the structure of the graphical interface that you have created. In turn, this structure is also divided into parts. Consider the order.

Connection settings

This part of the code in more detail defines the characteristics of the selected connection. For example, if you are using a connection via Software Serial, you can determine which pins microcontroller will be connected to the Bluetooth module. You can also determine the rate of exchange with the Bluetooth module, if it is different from the default 9600 baud. All definitions are set by default, but you can change them as you need. This is the only section of code in the included section of Remote XY that you can change.

ESP8266 Wi-Fi module

ESP8266 module allows to use Remote XY with microcontroller device for Wi-Fi. Implemented support for all Arduino boards. The source code of GUI can be download for the Arduino IDE and PLProg.

Remote XY allows to configure module for operate in one of two modes: access point and client. Client allows to connect the module to an existing Wi-Fi access point. Access point mode of ESP8266 configures the module as an access point and allows to connect Arduino directly to this point. The access point is available to connect to it from smartphone or tablet within a radius of availability of the radio signal. It does not require any other network infrastructure. This connection mode can be used in the far away place where there is no computer networks and the Internet. To configure this mode in the configuration properties of editor need to select the type of connection "Wi-Fi access point."

Client mode of ESP8266 configures the module for automatically connect to an existing Wi-Fi access point, such as a home router or enterprise access point. At the same time connected to the Arduino module ESP8266 must be located in the physical availability of the radio signal of access point. Connecting to the device from smartphone or tablet will be not directly, but through the Ethernet network, an IP address provided by the DHCP server to ESP8266 module. It allows to connect to the Arduino device from anywhere in the local network as well as from the Internet. Connect from the Internet is possible with

the correct configuration of the router, such as the use of the virtual server. To configure this mode in the configuration properties of editor need to select the connection type "Ethernet" and select ESP8266 module.

The module is connected to the microcontroller board via serial interface. You can choose to use software serial or hardware serial. Module connection option is selected in the module interface settings of editor. The module is controlled via AT commands.

Connection via Software Serial

It allows to connect the module to an arbitrary microcontroller pins. The pins used must be select in the module interface settings of editor. Also in the settings must be select the baud rate for serial port. ESP8266 default configured to 115200 baud. To work through Software Serial, must configure the transmission speed ESP8266 to a lower speed. Stable operation of the module is possible at a speed of no more than 19200 baud. To change the speed of module serial interface, must to connect it to the console to the possibility of the AT commands. Module serial interface speed can be changed with the AT command "AT+UART_DEF=19200,8,1,0,0".There are some limitations on the use of the RX pin for Arduino boards. Limitations associated with do not support to pins interrupt of the microcontroller.

- On **Arduino** RX doesn't work on pin 13;
- On **Mega** and **Mega2560** only the following can be used for RX: 10, 11, 12, 13, 14, 15, 50, 51, 52, 53, A8(62), A9(63), A10(64), A11(65), A12(66), A13(67), A14(68), A15(69);
- On **Leonardo** and **Micro** only, the following can be used for RX: 8, 9, 10, 11, 14, 15, 16;

Also, when using Software Serial, you should take the following limitations:

- No ability to work at high baud rates. It is not recommended to use a baud rate of more than 19200.
- Some libraries, which also uses interrupt may not work correctly. For example Servo library, the servos will twitch

Chapter 4

Design and Implementation

4.1 System Description:

In this system here we use a Node MCU micro controller for controlling this whole system. This system we use three sensors (Temperature Sensor, Gas Sensor, Fire Sensor) in a demo industry to detect this kind of condition. Node MCU can get notification from far way of industry through internet by Remote XY mobile Apps.

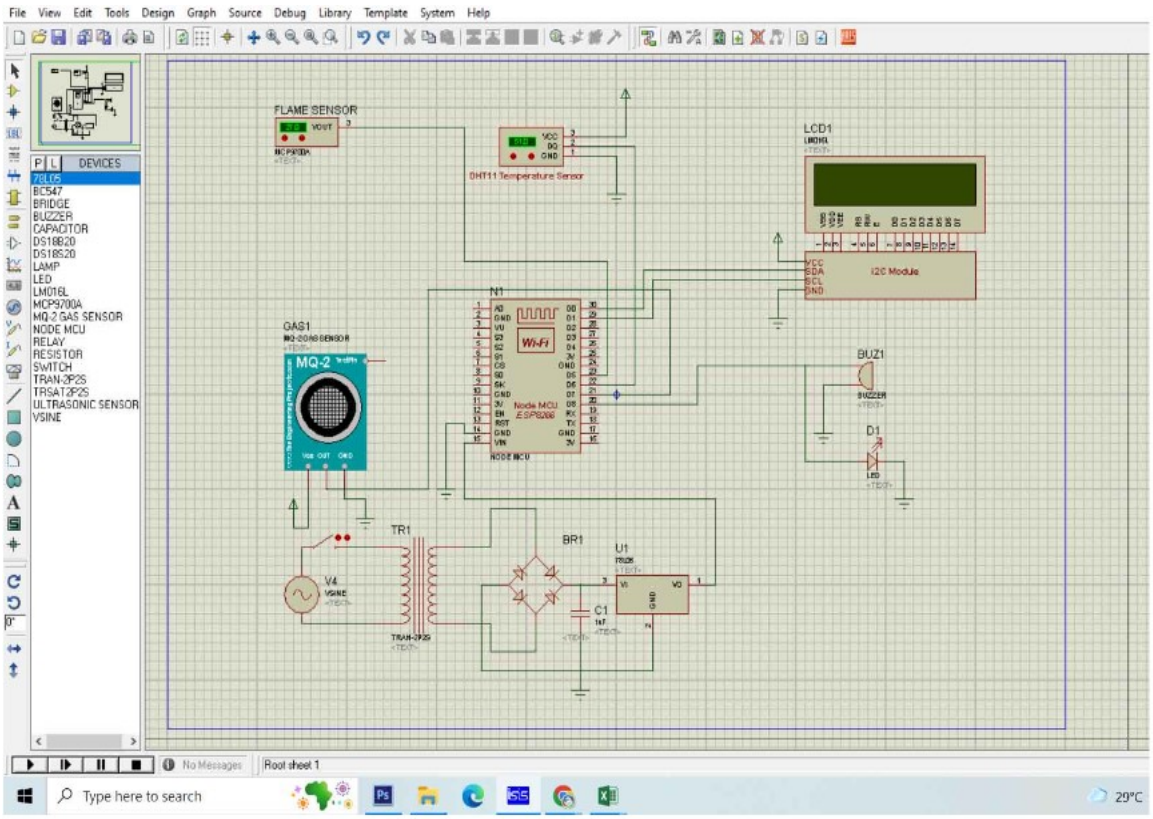


Figure 4.1: System Structure Circuit Design in Proteus.

4.2 Block Diagram:

In our project we have set up a Temperature, Fire and Gas detection system. Here temperature sensor always measures the industrial inside temperature, fire sensor and gas sensor detect the fire and the gas. This reading will be show in a LCD display and mobile app display. Here the main power supply is AC 220V. In this circuit we have used a Node MCU for controlling this system.

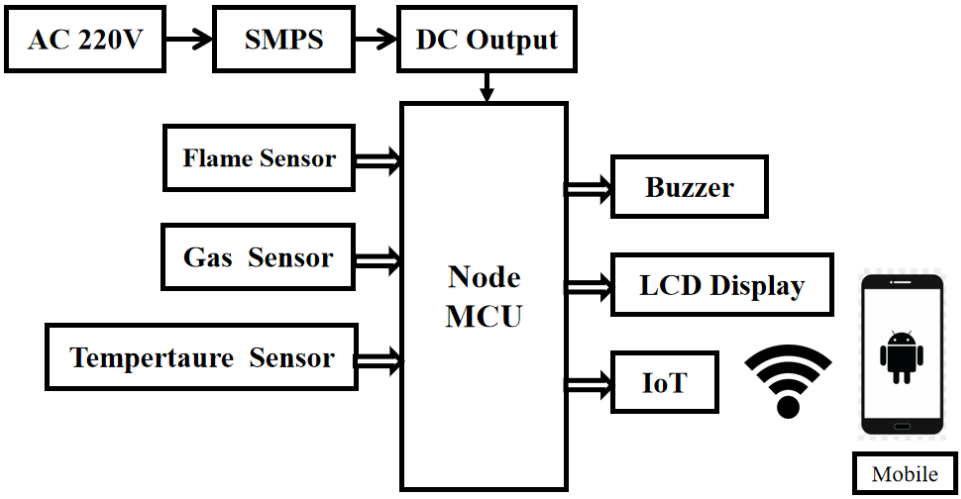


Figure 4.2: Block Diagram.

4.3 Schematic Diagram:

Node MCU has been used as main controller for this project. With the help of Node MCU which the project can be controlled from far away through internet. Here we have connected all of our instrument through wire.

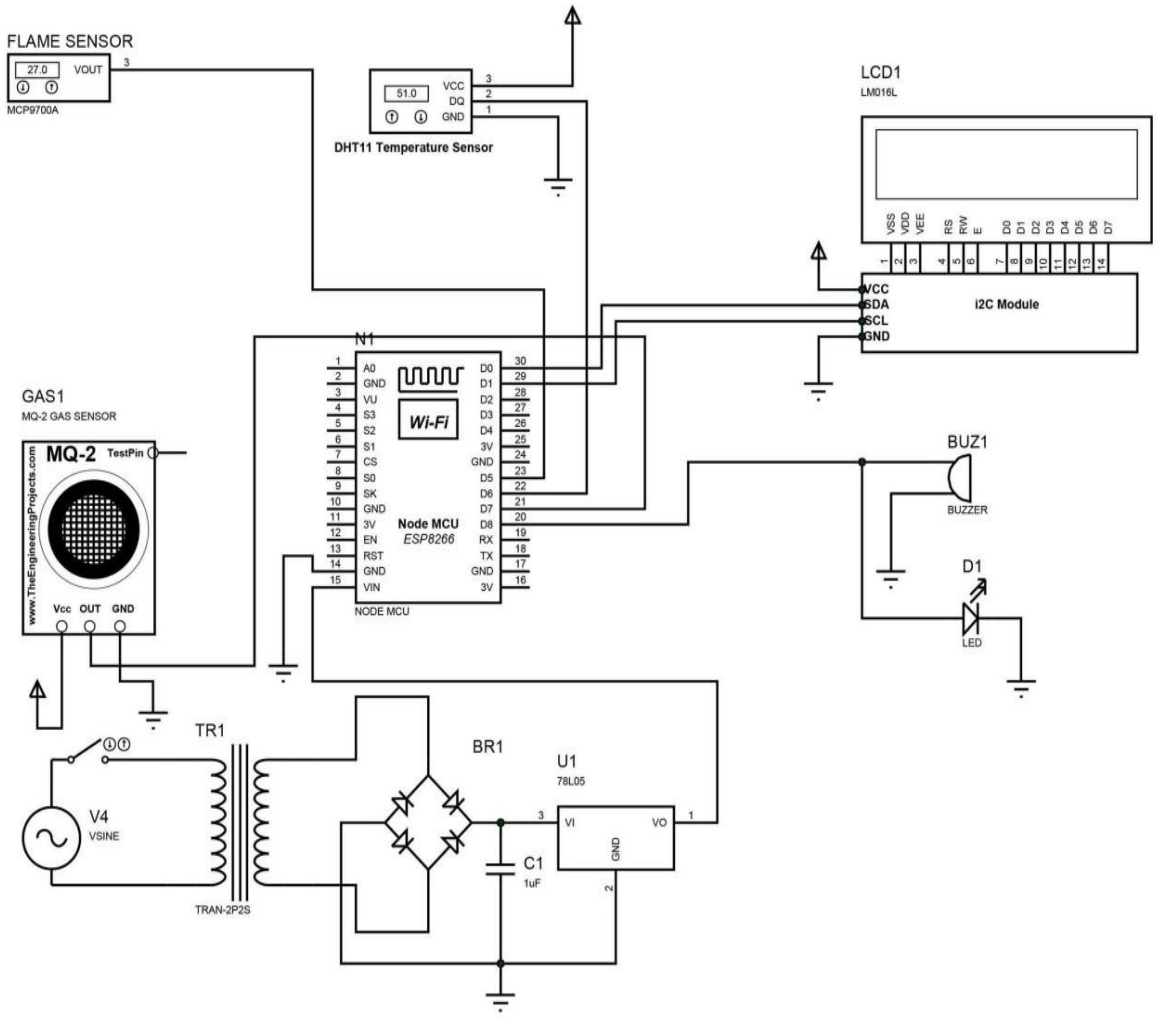


Figure 4.3: Schematic diagram

4.4 Working Principle:

The main brain of our system is the Node MCU. The way of whole project works is that we take 220V (rms) AC power from the supply voltage and then feed it to a Switch Mode Power Supply or in short SMPS module. The SMPS simply converts the 220V AC to a pure DC of 5V. We will use this 5V DC output from the SMPS to run our micro-controller, sensor and other units. A temperature sensor detects the industrial inside temperature,

flame sensor was used for detect the fire, MQ2 sensor is detect the Smoke/gas. When there creates a fire then firstly detect it by flame sensor, after that when there creates some gas then it will detect by the MQ2 sensor. Then it will ring a buzzer for alerting system. All kind of operational output will be display in a LCD Display and Mobile Apps Display. This is the main process of our system.

CHAPTER 5

Result & Performances

5.1 Result:

This chapter contains the results obtained and discussion about the full project. Our project is temperature, smoke and gas detection system. In our project making we used PVC boards for total hardware making. After finally completing this project, we ran it & we observed the output of this project. We can see that it is working well as expected. After making our project we observe it very careful. It works as we desire. Our project give output perfectly and all equipment are work perfectly. We check how much it works and we get perfect output from this project.

- Finally, we have completed our project successfully & check our project its run accurately according to our objective.
- At first, we start our system.
- Then it was show industrial inside temperature in LCD display and mobile display.
- Then if there create any fire and gas incident then it will sense and ring a buzzer for alert.
- Beside of this work this system also send a notification in mobile apps.

Project Activities

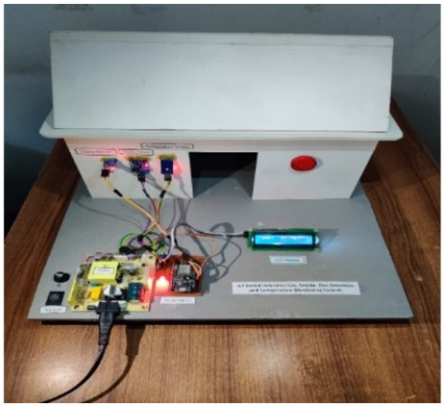
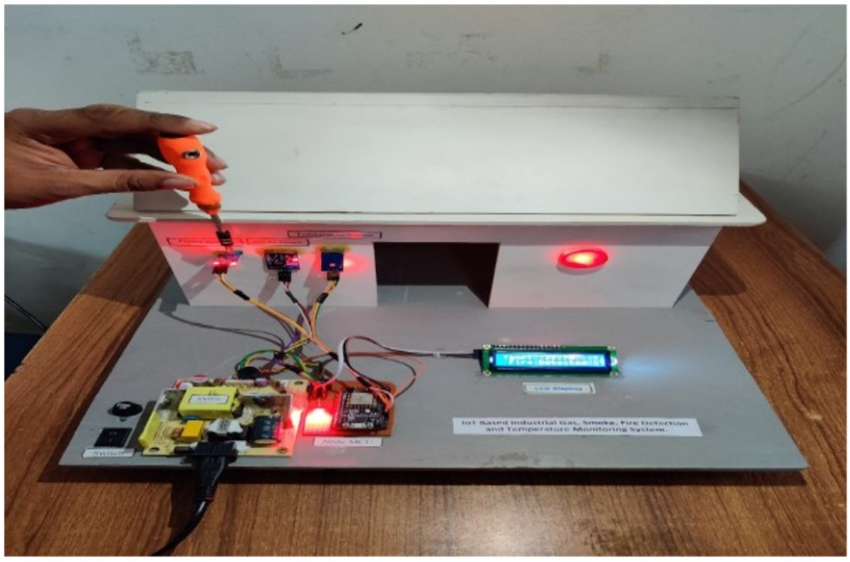
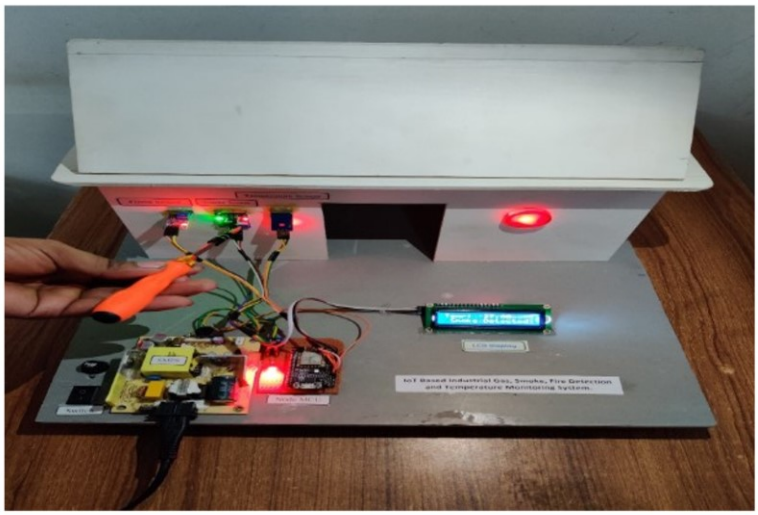


Figure: Project Prototype

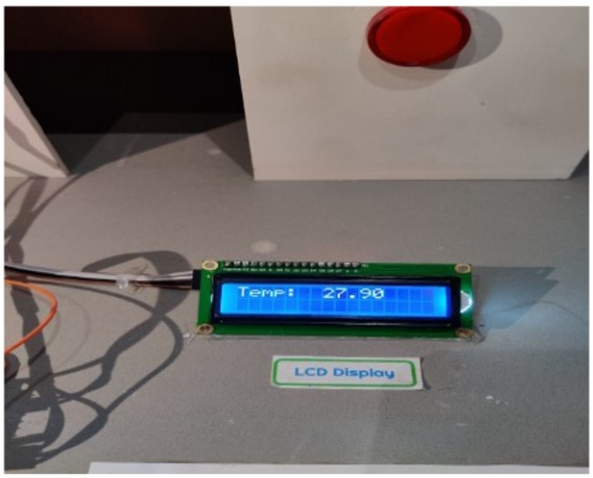
Presentation in flame sensor Detection



Presentation in smoke/Gas sensor Detection



Presentation in Temperature monitoring .



Chapter 6

Discussion & Conclusion

6.1 Discussion:

Temperature, Fire and Gas Detection System is really needed for country right now because it can very well save people from unwanted harms. This project lets the user control the unwanted fire and gas related incident. Our project is simple to use but very smart. It doesn't require much energy to run as well. The components that were used to build the system is cheap and available so maintenance is not issue. This type of machine reduces human efforts and also provides accurate results. While doing our project, we did face some difficulties such as faulty wires, coding problem, structure building problems etc. But at the end, the results we came up with were up to the mark. As we have some limitations in our project, so it means that there are lots of things that we can do with this project in the future and we are already focusing on that.

6.2 Advantage:

There are certainly many advantages of our project and some of the major ones have been given below:

- Can measure the temperature of inside industry.
- Can detect smoke/gas and fire incident automatically.
- Also send notification in IoT mobile apps.
- This project is easy to use
- Cost effective
- User-friendly
- Automatic and requires less human interaction
- Can be operated virtually from anywhere in the world
- The whole system consumes very little energy.

6.3 Application:

This project has applications in many fields due its necessity. We have selected a few of them and they are given below:

- It can be used in any educational institutions
- It can be used in any industry or corporate offices
- It can be used in Hospitals
- It can be used in residential areas

6.4 Limitation:

Though this project has many advantages but it has some limitations as well and they are listed below:

- Used cheap Chinese products for the prototype so there's some processing delay present in the circuit
- This project can now be only used for small scale purposes.
- Must be have internet to get IoT notification.

.

6.5 Future Scope:

The model can be improved by making some changes in the hardware and components. Some suggestions are given below-

- In future development this project can be develop by more sensor & alarm system.
- In future we will add camera monitoring system.

6.6 Conclusion:

On our system, we read three different parameters for detecting fire, these three parameters are flame, smoke, and temperature this will helps to improve the accuracy of detecting fire. It will help to detect fire detection on any condition more accurately and fast. As most of the fire detection systems sense any one parameter for detecting fire and there is a chance to take more time for detection of fire until that sensor particular parameter does not exceed above the threshold value. So, we try to overcome this issue by sensing different parameters for this reason we interface different parameters for sensing detection of fire. Also, this system is based on IoT technologies so Realtime sensor data and alert messages send to the concerned person so our system detects fire faster and accurately.

Reference

[1] D. Gutmacher, U. Hoefer, and J. Wöllenstein, “Gas sensor technologies for fire detection,” *Sensors Actuators, B Chem.*, 2012, doi: 10.1016/j.snb.2011.11.053.

[2] P. S. Murvay and I. Silea, “A survey on gas leak detection and localization techniques,” *Journal of Loss Prevention in the Process Industries*. 2012, doi: 10.1016/j.jlp.2012.05.010.

[3] S. J. Chen, D. C. Hovde, K. A. Peterson, and A. W. Marshall, “Fire detection using smoke and gas sensors,” *Fire Saf. J.*, 2007, doi: 10.1016/j.firesaf.2007.01.006.

[4] E. Scorsone, A. M. Pisanelli, and K. C. Persaud, “Development of an electronic nose for fire detection,” *Sensors Actuators, B Chem.*, 2006, doi: 10.1016/j.snb.2005.12.059.

[5] S. Rappert and R. Müller, “Odor compounds in waste gas emissions from agricultural operations and food industries,” *Waste Manag.*, 2005, doi: 10.1016/j.wasman.2005.07.008.

[6] W. Krüll, R. Tobera, I. Willms, H. Essen, and N. Von Wahl, “Early forest fire detection and verification using optical smoke, gas and microwave sensors,” in *Procedia Engineering*, 2012, doi: 10.1016/j.proeng.2012.08.208.

[7] D. Gutmacher, C. Foelmli, W. Vollenweider, U. Hoefer, and J. Wöllenstein, “Comparison of gas sensor technologies for fire gas detection,” in *Procedia Engineering*, 2011, doi: 10.1016/j.proeng.2011.12.276.

[8] H. Jin, L. Zhang, W. Liang, and Q. Ding, “Integrated leakage detection and localization model for gas pipelines based on the acoustic wave method,” *J. Loss*

[9] Reshma Shinde, Ritika Pardeshi, Archana Vishwakarma and Nayan Barhate, ” Need for Wireless Fire Detection Systems using IOT ” , International Research Journal of Engineering and Technology, Volume 04, Issue 01, pp-1078-1081, January 2017.

[10] Alexander Filonenko, Danilo Caceres Hernandez, Ajmal Shahbaz and Kang-Hyun Jo , ” Unied Smoke and Flame Detection for Intelligent Surveillance System ” , IEEE, Volume 08, Issue 03, pp-953-957, May 2016.

[11] Xiaofei Ji, Xuan Xie, Xinmeng Zuo and Jiangtao Cao, ” Design and Implementation of Smoke Early Warning System ” , IEEE, Volume 03, Issue 17, pp-351-356, July 2017.

Appendix

Micro-controller Code:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27 ,16,2);
#define REMOTEXY_MODE__ESP8266WIFI_LIB_CLOUD
#include <ESP8266WiFi.h>

#include <RemoteXY.h>
#include "DHTesp.h" // Click here to get the library: http://librarymanager/All#DHTesp

#ifdef ESP32
#pragma message(THIS EXAMPLE IS FOR ESP8266 ONLY!)
#error Select ESP8266 board.
#endif

DHTesp dht;
#define REMOTEXY_WIFI_SSID "abcde"
#define REMOTEXY_WIFI_PASSWORD "12345678"
#define REMOTEXY_CLOUD_SERVER "cloud.remotexy.com"
#define REMOTEXY_CLOUD_PORT 6376
#define REMOTEXY_CLOUD_TOKEN "318b70249d13777029280f11bcd1be44"

  char str[] = "Temperature High";
  char str1[] = "Fire Detected";
  char str2[] = "Smoke Detected";
  char nor[] = "Normal Condition";

// RemoteXY configurate
```

```
#pragma pack(push, 1)
uint8_t RemoteXY_CONF[] =   // 96 bytes
  { 255,0,0,59,0,89,0,16,31,1,67,4,1,69,61,9,36,31,31,69,
  0,4,6,10,10,1,69,0,26,6,10,10,1,69,0,47,6,10,10,1,
  67,4,41,29,22,8,135,31,11,67,4,41,45,21,8,163,31,11,129,0,
  1,30,40,6,203,84,101,109,112,101,114,97,116,117,114,101,58,32,0,129,
  0,2,46,28,6,204,72,117,109,105,100,105,116,121,58,0 };

// this structure defines all the variables and events of your control interface
struct {

    // output variables
  char notify[31];  // string UTF8 end zero
  int16_t sound_1; // =0 no sound, else ID of sound, =1001 for example, look sound list in
          app
  int16_t sound_2; // =0 no sound, else ID of sound, =1001 for example, look sound list in
          app
  int16_t sound_3; // =0 no sound, else ID of sound, =1001 for example, look sound list in
          app
  char temp[11];  // string UTF8 end zero
  char humidity[11];  // string UTF8 end zero

    // other variable
  uint8_t connect_flag;  // =1 if wire connected, else =0

} RemoteXY;
#pragma pack(pop)

/////////////////////////////////////////////
//          END RemoteXY include          //
/////////////////////////////////////////////
void setup()
{
```

```
  RemoteXY_Init ();

  pinMode(D7,INPUT);
  pinMode(D5,INPUT);
  pinMode(D8,OUTPUT);

  dht.setup(12, DHTesp::DHT11); // Connect DHT sensor to GPIO 17
  lcd.init();
lcd.backlight();
  lcd.begin(16,2);
   lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Industrial Fault");
  lcd.setCursor(0,1);
  lcd.print("Detection System");
  delay(2000);
}

void loop()
{
  RemoteXY_Handler ();
    delay(dht.getMinimumSamplingPeriod());

    float humidity = dht.getHumidity();
  float temperature = dht.getTemperature();
int fire = digitalRead(D5);
int smoke = digitalRead(D7);

  dtostrf(temperature, 0, 2, RemoteXY.temp); // result: "1234.32"
  dtostrf(humidity, 0, 2, RemoteXY.humidity); // result: "1234.32"
lcd.clear();
lcd.setCursor(0,0);
  lcd.print("Temp:");
```

```
 lcd.setCursor(7,0);
  lcd.print(temperature);

  if(temperature > 36){
    strcpy  (RemoteXY.notify, str);
  RemoteXY.sound_1 = 2001;
digitalWrite(D8,HIGH);
  lcd.setCursor(0,1);
  lcd.print("Temperature High ");
    delay(2000);
    }
if(temperature < 36){
  digitalWrite(D8,LOW);
  RemoteXY.sound_1 = 0;

}
if(fire == LOW ){
    strcpy  (RemoteXY.notify, str1);
  RemoteXY.sound_2 = 2001;
digitalWrite(D8,HIGH);
  lcd.setCursor(0,1);
  lcd.print(" Fire Detected  ");
    delay(2000);
}
if(fire == HIGH){
  RemoteXY.sound_2 = 0;

}
if(smoke == LOW ){
    strcpy  (RemoteXY.notify, str2);
  RemoteXY.sound_3 = 2001;
digitalWrite(D8,HIGH);
  lcd.setCursor(0,1);
```

```
  lcd.print(" Smoke Detected  ");
    delay(2000);
}
if(smoke == HIGH){
  RemoteXY.sound_3 = 0;
}
}
```