MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION, MUMBAI

GOVERNMENT POLYTECHNIC, JALGAON

DEPARTMENT OF ELECTRONICS & TELECOMMUNICATION ENGINEERING

A CAPSTONE PROJECT

On

"Electroplating bath parameters monitoring system in Spectrum Pvt.Ltd. Electroplating plant"

Submitted by

Saraf Prathamesh G.
Bhangale Mohit R.
Patil Mandar B.
Khadse Mohish N.

Under the guidance of
Prof. K. P. Akole
2020-21

MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION, MUMBAI

GOVERNMENT POLYTECHNIC, JALGAON

DEPARTMENT OF ELECTRONICS & TELECOMMUNICATION ENGINEERING

ACADEMIC YEAR 2020-21

Certificate

This is to certify that Mr Prathamesh Saraf , Mohit Bhangale , Mandar Patil , Mohish Khadse from Government Polytechnic , Jalgoan having enrolment No 1800180265 , 1800180288 , 1800180290 , 1800180291 have completed capstone project on

"Electroplating bath parameters monitoring system in Spectrum Pvt.Ltd. Electroplating plant"

This report is submitted as partial fulfilment of requirement of Outcome Based Education pattern in Electronic & Tele-communication engineering prescribed by M.S.B.T.E Mumbai

Principal **Prof K.P.Akole**

(H.O.D. E&TC Dept. and Project Guide)

Acknowledgement

First we would like to express our appreciation to our capstone project guide ,our department HOD Mr K.P Akole sir. His constant guidance advice palyed a vital role in making the execution of the project . He always gave his suggestions that were crucial in making better. Guidance of the our sir has consistently keep us motivated for project . He has taken every possible step to improve us not only in a project but also in social activities which were going to improve our project and working attitude towards our project. We again thank you sir for your support. And we also like to expressed our gratitude towards our industry project guide and R&D department head of Spectrum electrical Pvt.Ltd. Mr. Sudharshan Rane . also we would like to thanks the principal of government polytechnic for the permission grating us undertake industry based project .
After that we would also like to thank our all teaching and non-teaching staff , our all friends , classmate and at last our parents to giving us their support and encouragemet for the our project .

Prathamesh G. Saraf

Mohit R. Bhangale

Mandar B. Patil

Mohish N. Khadse

INDEX

Diagram content

Chapter 1 -: Introduction

1.1 Rationale [1][2][3]

Electroplating is a common metal finishing process with several industrial applications, ranging from the purely cosmetic to the application of protective coatings. The process, which uses electric current to drive ions towards metal surfaces, is sensitive to multiple key parameters and careful control is crucial to achieving desired plating effects. Several components, including the concentration of ions in the plating bath, the bath temperature, and the electric current density, can be adjusted to affect the final outcome. The process of electroplating has produced many rules of thumb for proper operation but by replacing conventional rules with pH measurement, quantitative feedback can be used to guide electroplating process control for improved product consistency and efficient waste treatment.

Process like deposition, doping, electroplating creates the toxic gases in the processing plant. This gases are used as a catalyst in the process. This gases are surely enabler in the processing but are highly toxic and can cause concussion when inhaled. Additional acids like HCL are also used in these companies for similar purpose fumes can cause irritation and affect the respiration of the inhaler Whether these gases are used for manufacturing or by products of any process in a facility it becomes very important to monitor and control them

Therefore detection system can be easily integrated into existing system and equipment of a company, allowing easy detection of leakages that can result in several catastrophe. Quick action can hence be taken to prevent the spread of gas over a wide region. These systems are an essential commodities in such industries. Since the allow them to detect the leakage of noxious and explosive gases maintain proper oxygen level for workers

1.2 Problem definition[a][3]

The problem given by the industry is that in the electroplating plant of the industry there is a need of a system which require to monitor a parameters in the electroplating bath such as electrolyte solution PH value, toxic gases such as methane , smoke gas from electroplating bath container are monitored and transmitted to control room as well as managers office wireless for monitoring purpose .

1.3 Abstract

An appropriate conditions are required within the electroplating bath for its smooth operation. In order to achieve the reliable product finishing on the product it is necessary to keep the stable parameters in the electroplating bath[1][3]. Those parameters are PH value and temperature of the electrolyte solution ,Gases such as methane, carbon monoxide and various oxides coming out due to the process of deposition in the electroplating bath. So it is important to keep continuously checking this parameters in order to maintain them within the specified required values. As a slight change in the value causes lowering the product finishing, increase in the gases and fumes in the plant due to deposition reaction and most important loss of the raw material used and final product

So by automating the data acquisition process of this parameters which are govern by the electroplating bath, allows the information to be collected at higher frequencies with less labour required. The motive of the project is to simple easy to install (handy) microcontroller based circuit to monitor and record the value of temperature PH and gases such as methane , HCL fumes carbon dioxide, etc that are continuously modified and controlled in order to optimize them to achieve maximum product finishing . The project includes microcontroller which communicate with the various sensor module in real time in order to monitor the parameters and send it wirelessly through transmitter in managers office. An integrated Liquid Crystal Display

(LCD) is used at the receiver end to display the real time status of parameters in the bath. Also the use of easy available components reduces the manufacturing and maintenance cost. The design is quite flexible as a software can be changed at any time. It can be tailor-made to the specific requirements of the users. This makes the proposed system to be an economical, portable and low maintenance solutions for the electroplating plant application especially for the small scale industry .

1.4 Objective

Flexible system -: System should be so flexible which will easily adapt the electroplating plant environment and can be easily transferred to another container. Also if require it can be modified as per further requirement(which includes switching action of the inlet and outlet valve of the electroplating bath,diluting the electrolyte solution etc)

Wireless system-: This monitoring system should be wireless because at the actual project implementing site there is no such place to carry the wires towards office

Alarm provision -: System should have the alarm provision which will inform that bath parameters has cross the threshold level.

Compact Size -: System should be compact so it can be attached at the site and can easily be carried to another bath if required .

1.5 Scope

The next chapters are as .

Chapter 2 (System Model)-: This chapter consist of the system model which is design on the basis of propose feature to achieve the project objectives .

Chapter 3 (System Hardware Design)-: This chapter will show the hardware designing of the system on the basis of the analysis the done for designing each part of the system for eg transducer, controller and transmitter interfacing analysis.

Chapter 4 (System Software Design)-: The software description regarding the programme sketch -editor and microcontroller environment IDE will be consist in the next chapter.

Chapter 5 (Hardware Components)-: This part will give the descriptive information and specification of the of the components used in the system .

Chapter 6 (System Operation)-: Chapter will give the operating procedure and information of the system. It will also include the precaution while using the system and terms related to Calibration of the transducers .

Chapter 7 (Simulation Of Hardware)-: This chapter includes the simulation procedure for project simulation.

Chapter 8 (Future Scope)-: This chapter consist of all about the propose available in future and the amendment and advancement in the project

Chapter 2-: System Model

It has a design which implies to monitor the the parameters govern by the electroplating bath. By considering the problem definition given by the industry model has been designed. It has design features to achieve the objective.[4]

2.1 Proposed feature -:

i. The very first aim is to detect the change in the parameters govern by the bath this parameters are
   - PH and Temperature value of the electrolyte solution.
   - Smoke gas and carbon monoxide within the bath environment .

ii. To compare those detected sensed values with the predefined standard value required for the bath

iii. To obtain the wireless features in the project which will allow to send the collected data to the electroplating plant office(managers office).

iv. To obtain the easy installation, flexible with hardware, compact system (system should be easily installed at another bath site).

v. To achieve the indication of change detected beyond the limited values of parameters .

vi. To achieve a user friendly hardware and software flexible system.

Therefore to achieve these specific objectives features the block as shown in fig 2.1 has been design

2.2 System model -:

The system model includes two parts

2.2.1 Parameters measuring terminal-: First part is at the actual electroplating bath site which work is to measure the values and transmit it to the receiver site which is in the manager's office. It consist of the following stages

i. Transducers -: In this block there are the different types of sensors which senses the change in the values of electrolyte solution and bath environment and also convert them to the digital format compatible to the microcontroller.

- **a.** PH sensor (SKU.SEN0161).
- **b.** Smoke gas sensor (Mq2)
- **c.** Carbon monoxide sensor (Mq7)
- **d.** Temperature sensor (DS18B20)

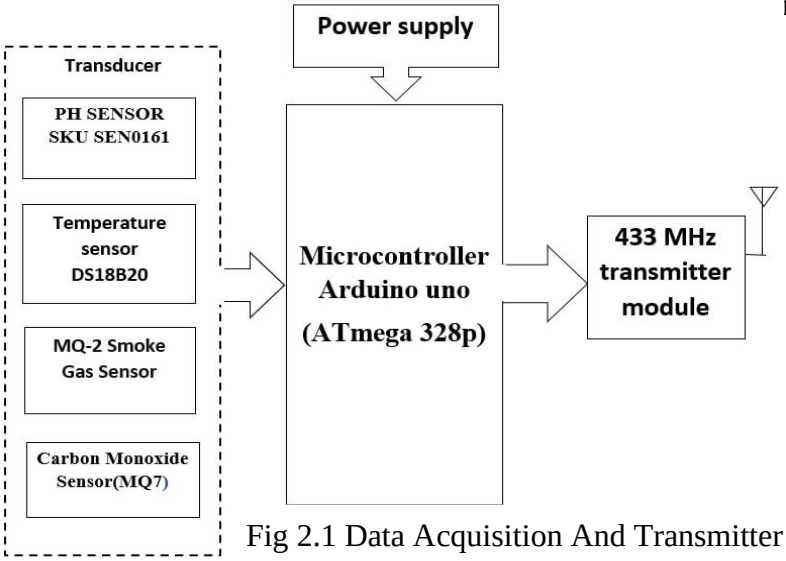


Fig 2.1 Data Acquisition And Transmitter Terminal

ii. Power supply -: This block consist of the supply to the system as per the requirement of the microcontroller. It supply the Arduino uno with **12V 1A** In the electroplating plant the bath itself has a attached power supply so for supply requirement will be fulfilled directly by it. All the sensors are directly compatible with the Arduino so supply for the sensors will be directly taken from the microcontroller (Arduino uno) .

iii. **Microcontroller (arduino uno)-**: It uses the arduino uno which has AVR based controller Atmega 328p the micro controller is the heart of the proposed system. It continuously monitors the digitized parameters of the various sensors and verifies them with predefined threshold level and indicate through the buzzer if values change beyond the limits. It also assign the collected data to the transmitter module which in turns transmit it to the receiver section

iv. **Transmitter module -**: Transmitter section gets the data from the controller and transmit it by arranging it in a frame with a ASK modulation technique.

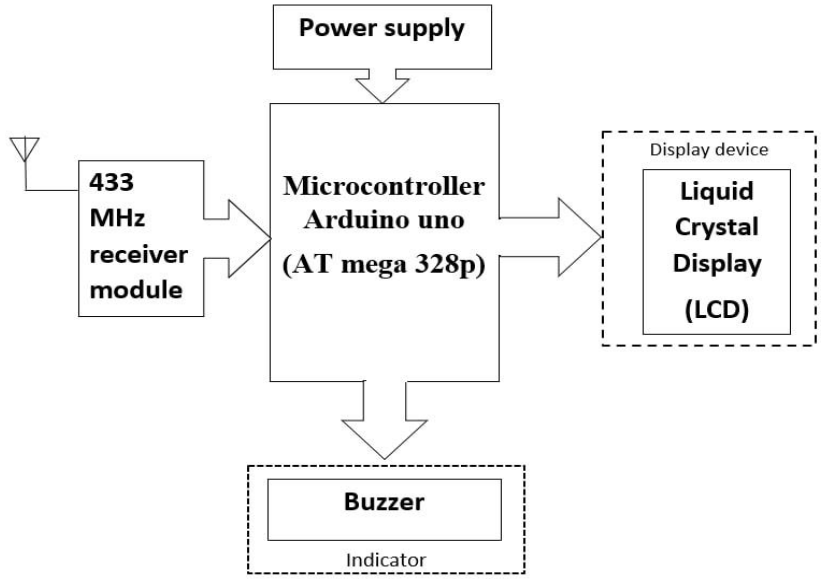


Fig 2.2 Data Receiver and display Terminal

2.2.2 Receiver and Display terminal -: Second part which is located at the control room receives the data and display it and in case if there is change in the values then indicating it. It consist of the following stages.

i. Receiver Module-: It receives the data transmitted by the transmitter section demodulate it and pass it to the controller

ii. Microcontroller (arduino uno) -: It is the heart of the receiver section. It collect the received data and show it on the display and also activate the buzzer if the parameter value reached beyond the threshold.

iii. Power supply -: It functions same as above given section of the power supply but here use of the 12V 1A adapter is used to power the arduino

iv. Display Unit -: A Liquid Crystal Display(JHD204A) is used to indicate the present value of parameters. The data of the each sensor will be display one by one.

v. Indicator device-: Buzzer (KY-012) it is a small buzzer module to indicate the change in the parameters beyond the predefined limit of the parameters.

Chapter 3-: System Hardware Design

The previous chapter consist the proposed block diagram for the system which shows the basic architectural design of the system. So by considering the all aspects of block diagram and to fulfill the objectives of the system the elements of the system has been designed as follows[4][5]

3.1 Functional Circuit

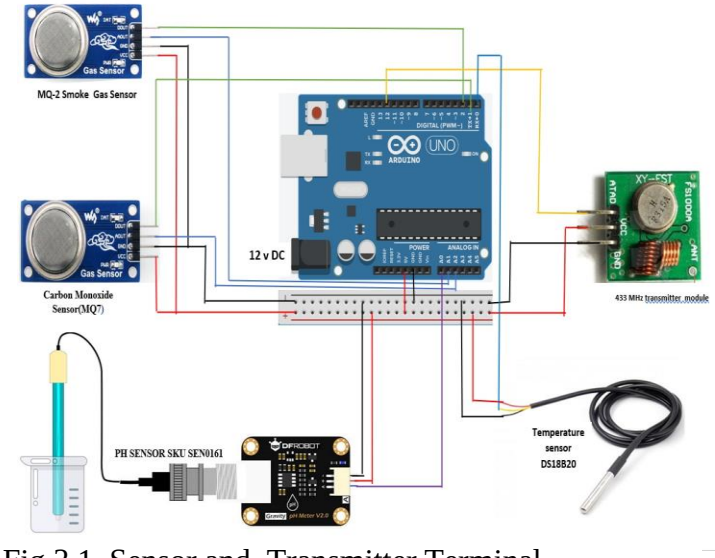


Fig 3.1 Sensor and Transmitter Terminal

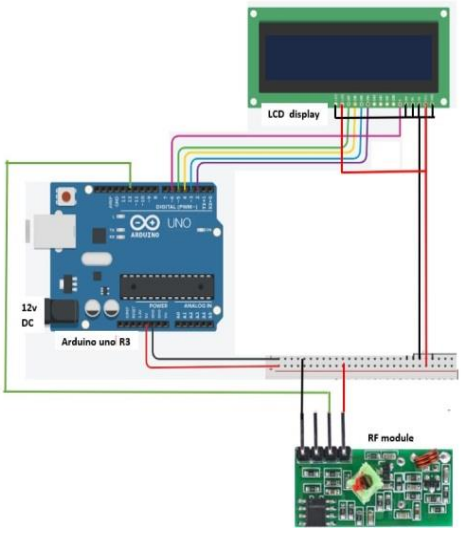


Fig 3.2 Receiver and Display terminal

3.2 Design the system elements

3.2.1 Interfacing Transducer -: It consist all the sensors used in the design

- **PH sensor module (**SKU.SEN0161)**[6]**

  This sensor is directly compatible with arduino and can be interfaced without any extra circuitry added but while interfacing required voltage offset setting

  - **Offset setting** This board have the ability to supply a voltage output to the analogue board that will represent a PH value just like any other sensor that will connect to an analog pin. Ideally PH 0 represent 0v and a PH of 14 to represent 5V.but there is a catch……, this board by default have PH 7 set to 0V (or near it, it differs from one PH probe to another, that is why we have to calibrate the probe as you will see later on), This means that the voltage will go into the minuses when reading acidic PH values and that cannot be read by the analog Arduino port. The offset pot is used to change this so that a PH 7 will read the expected 2.5V to the Arduino analog pin, the analog pin can read voltages between 0V and 5V hence the 2.5V that is halfway between 0V and 5V as a PH 7 is halfway between PH 0 and PH 14, it will need to turn the offset potentiometer to get the right offset, The offset pot is the blue pot nearest to the BNC connector.To set the offset is easy. First, you need to disconnect the probe from the circuit and short-circuit the inside of the BNC connector with the outside to simulate a neutral PH (PH7). It takes a piece of wire, strip both sides, wrap the one side around the outside of the BNC connector and push the other side into the BNC hole. This short-circuit represents about a neutral PH reading of 7 .

There are two ways for the adjustment.

1) By using multimeter measure the value of the PO pin and adjust the offset potentiometer until PO measures 2.5V.
2) Write the sketch given in the appendix a then., open serial monitor and view the reading there. This sketch print the volts it receives from the analog pin and print it to the serial monitor. It of course first changes the digital value to volts to make it easier. Now simply turn the offset pot until it is exactly 2.5Vsmoke gas sensor (Mq2) Connecting the sensor with the arduino . Connection are simple connect the VCC and GND pins to the +5V and GND pins of arduino the connect the P0 pin to A0 pin of arduino and D0 pin to digital pin no 1 of Arduino PH sensor sensitivity in Vtg[6]

VOLTAGE (mV)	pH value	VOLTAGE (mV)	pH value
414.12	0.00	-414.12	14.00
354.96	1.00	-354.96	13.00
295.80	2.00	-295.80	12.00
236.64	3.00	-236.64	11.00
177.48	4.00	-177.48	10.00
118.32	5.00	-118.32	9.00
59.16	6.00	-59.16	8.00
0.00	7.00	0.00	7.00

- **Gas sensors (**MQ2 and MQ7 **)[7]-:**

Both the sensors has same interfacing procedure in order to get correct and accurate data, need to take the following actions first: MQ sensor needs 24-48 hours of preheating time. Connect the power supply and leave for the required time until it gets ready.

This need to calibrate the sensor (we have explained this in the following section)This module has 4 pins. Connect Vcc to 5V and GND to GND. The AO pin returns an analog value based on the concentration of the gas. The DO pin returns HIGH if the concentration of gas is higher than a certain value. This value can be set by the potentiometer on the board. Before using the module you have to calibrate it. This sensor measures the gas concentration based on resistance ratio. This ratio includes R0 (sensor resistance in 1000ppm concentration of Gas) and Rs (Internal resistance of the sensor which changes by gas concentration).

- **Temperature sensor (DS18B20)[8]**

The DS18B20 Digital Thermometer provides 9 to 12-bit (configurable) temperature readings which indicate the temperature of the device. It communicates over a 1-Wire bus that by definition requires only one data line (and ground) for communication with a central microprocessor. In addition it can derive power directly from the data line ("parasite power"), eliminating the need for an external power supply. The core functionality of the DS18B20 is its direct-to-digital temperature sensor. The resolution of the temperature sensor is user-configurable to 9, 10, 11, or 12 bits, corresponding to increments of 0.5°C, 0.25°C, 0.125°C, and 0.0625°C, respectively. The default resolution at power-up is 12-bit.

It has a 3 pins out of which two are VCC and GND which is connected to arduino's +5V and GND , and DQ (data pin) is connected to the digital pin 0 of arduino .

Note while connecting the digital pin it require the pull up register of 4.7k ohm. to insure that the board can read the sensor or say to keep the data transfer stable.[8]

3.2.2 433MHz RF Transmitter module[9][10]

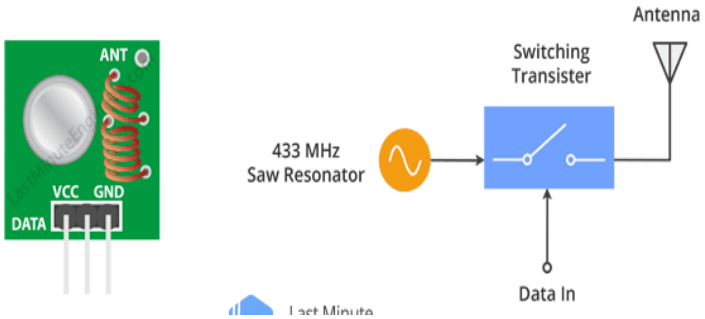


Fig 3.3 RF Transmitter module

This little module is a transmitter among two. It is really simple as it looks. The heart of the module is the SAW resonator which is tuned for 433.xx MHz operation. There is a switching transistor and a few passive components, that's it.

When a logic HIGH is applied to the DATA input, the oscillator runs producing a constant RF output carrier wave at 433.xx MHz and when the DATA input is taken to logic LOW, the oscillator stops.This technique is known as Amplitude Shift Keying,

This one is a receiver module. Though it looks complex, it is as simple as the transmitter module. It consists of a RF tuned circuit and a couple of OP Amps to amplify the received carrier wave from the transmitter. The amplified signal is further fed to a PLL (Phase Lock Loop) which enables the decoder to "lock" onto a stream of digital bits which gives better decoded output and noise immunity.

Connection with Arduino As we will be sending data between two Arduino boards, we will of course need two Arduino boards, two breadboards and a couple of jumper wires.The wiring for the transmitter is fairly simple. It has only three connections. Connect the VCC pin to 5V pin and GND to ground on the Arduino. The Data-In pin should be connected to Arduino's digital pin #12. You should try and use pin 12 as by default the library we'll be using in our sketch uses this pin for data input.

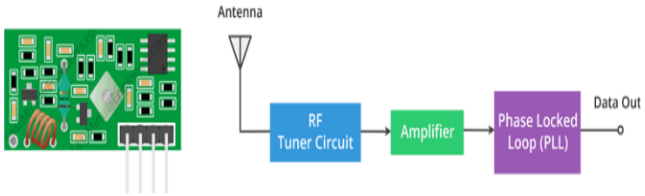


Fig 3.4 RF Receiver module

Once you have the transmitter wired you can move on to the receiver. The wiring for the receiver is just as easy as the transmitter was.Once again there are only three connections to make. Connect the VCC pin to 5V pin and GND to ground on the Arduino. Any of the middle two Data-Out pins should be connected to digital pin #11 on the Arduino.

3.2.3 Liquid Crystal Display [12]

The LCD has a 16 pins .There are 8 Data lines that carry raw data to the display. But, HD44780 LCDs are designed in a way talk to the LCD using only 4 data pins(4-bit mode) instead of 8(8-bit mode). This saves us 4 pins!

So to recap, it will required for LCD to interface using 4-bit mode and hence we need only 6 pins: RS, EN, D7, D6, D5, and D4 to talk to the LCD.

Now connect the LCD Display to the Arduino. Four data pins (D4-D7) from the LCD will be connected to Arduino's digital pins from #2-5. .r/w pin will be connected to the ground as we want only to write data on the LCD. The Enable pin on LCD will be connected to Arduino #6 and the RS pin on LCD will be connected to Arduino #7.

Difference between 4-bit and 8-bit mode

It's faster to use 8-bit mode as it takes half as long to use 4-bit mode . Because in 8-bit mode write the data in just one go. However, in 4-bit mode . It have to split a byte in 2 nibbles, shift one of them 4 bits to the right, and perform 2 write operations. So , The 4-bit mode is often used to save I/O pins. But, 8-bit mode is best used when speed is required in an application and at least 10 I/O pins are available.

3.2.4 Arduino Uno R3 (microcontroller development board).

In the system arduino uno R3 Board which contain ATmega328p micro-controller integrated on it has been used. The large set of the features available of arduino board to fulfill our project objective is the reason for selection of it as a microcontroller.

Selection criteria for the microcontroller-: The selection of the board is based, by considering all the points given below.[11]

- It is an **open-source project**, software/hardware is extremely **accessible** and very flexible to be customized and extended
- It is **easy to use**, connects to computer via USB and communicates using standard serial protocol, runs in standalone mode and as interface connected to PC/Macintosh computers
- It powers directly on 12 V DC supply or 9 V DC battery which saves the power supply cost in the project. However project use the 12V DC supply provided at the bath site.
- It contains Flash 32K bytes (of which 5K is used for the bootloader), SRAM 2K bytes, EEPROM 1K byte which is very much sufficient for our sketch.
- It is inexpensive as compare to other microcontrollers (including all the feature available in it)
- It has a inbuild ADC with the 10 bit resolution 6 analog pins support, which makes the analog sensors compatible with it
- It has inbuild SPI interface (through which controller will be communicating with the RF module)
- It has many features beyond our project requirement (given in components description chapter) which make it compatible for further advancement (given in the future scope chapter).

So based on the above given features point it was a best suited for our project.

Chapter 4-: System Software Design

After designing the hardware now it is required to design the software part of the project .This part include the software used for the developing the program for the system and description of the code developed .

4.1 Software Platform (Arduino) [16]

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. Instruct the board what to do by sending a set of instructions to the microcontroller on the board. To do use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

4.1.1 Program (code) editor (Arduino IDE 1.8.13)[16]

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.

4.1.2 Uploading the code

Before uploading need to select the correct items from the **Tools > Board** and **Tools > Port** menus.[16] 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. After selecting 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. While uploading a sketch, it use the Arduino **bootloader**, a small program that has been loaded on to the microcontroller on board. It allows 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).

4.2 Libraries

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manupulating 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 sketch. Because libraries are uploaded to the board with 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.

4.3 Coding Design

4.3.1 Libraries used in program

Radio Head library(#include <RH_ASK.h>)[13]
This is the RadioHead Packet Radio library for embedded microprocessors. It provides a complete object-oriented library for sending and receiving packetized messages via a variety of common data radios and other transports on a range of embedded microprocessors.

SPI library(#include <SPI.h>)

Serial Peripheral Interface (SPI)[16] is a synchronous serial data protocol used by microcontrollers for communicating with one or more peripheral devices quickly over short distances. It can also be used for communication between two microcontrollers.

With an SPI connection there is always one master device (usually a microcontroller) which controls the peripheral devices. Typically there are three lines common to all the devices:

OneWire library (<OneWire.h>)[14]

OneWire access 1-wire devices made by Maxim/Dallas, such as temperature sensors and ibutton secure memory. For temperature sensors, the DallasTemperature library can be used with this library.

DallasTemperature library (<DallasTemperature.h>)

This library support the temperature sensor DS18B20 as dallastemperature is manufacturer of it

4.3.2 Algorithm

1. **Algorithm for Arduino code at measuring terminal**

Step1: Start

Step2: Include a radiohead library for Rf module interface

Step3: Include dependant SPI Library for communicating with transmitter

Step4: Include a oneWire library for temperature sensor

Step5: Data wire is plugged into digital pin 0 on the Arduino

Step6: Setup a oneWire instance to communicate with any OneWire device

Step7: Pass oneWire reference to DallasTemperature

Step8: Create Amplitude Shift Keying Object

Step9: Initialize function to read carbon monoxide value

Step10: Initialize function to read smoke gas value

Step11: Initialize function to read PH value of solution

Step12: Initialize function to read temperature value of solution

Step13: Initialize ASK Object

Step14: Initialize the variables to store the input value of the sensors

Step15: Call the function to measure the get the PH data

Step16: Converte the data into string for transmitting through the rf module

Step17:Concinate the msg string and converted data to transmit

Step18: Compose output character

Step19: Transmit the data using the inbuild function of RH library

Step20: Wait for the data to be send

Step22: Call the function to measure the get the temperature data

Step23:Converte the data into string for transmitting through the rf module

Step24: Concinate the msg string and converted data to transmit

Step25:Compose output character

Step26: Transmit the data using the function of RH library

Step27:Wait for the data to be send

Step29: Call the function to measure the get the MQ7 data

Step30: Converte the data into string for transmitting through the rf module

Step31:Concinate the msg string and converted data to transmit

Step32: Compose output character

Step33: Transmit the data using the inbuild function of RH library

Step34: Wait for the data to be send

Step36: Call the function to measure the get the MQ2 data

Step37:Converte the data into string for transmitting through the rf module

Step38:Concinate the msg string and converted data to transmit

Step39: Compose output character

Step40:Transmit the data using the function of RH library

Step41:Wait for the data to be send

Step42: Defining the function to read the PH value

Step43: Defining the function to read the temperature value

Step44: Defining the function to read the carbon monoxide value

Step45: Defining the function to read the smoke gas value

Step46:END

Algorithm for Arduino code at display terminal

Step1: Include dependant SPI Library

Step2: Create Amplitude Shift Keying Object

Step3: Include the liquid crystal diaply library

Step4: Creates an LCD object. Parameters: (rs, enable, d4, d5, d6, d7)

Step5: String to store the recived data

Step6: Initialize ASK Object

Step7: Set up the LCD's number of columns and rows:

Step8: Clears the LCD screen

Step9: Set buffer to size of expected message

Step10: Check if received packet is correct size

Step11: Set the cursor to column 0, line 1

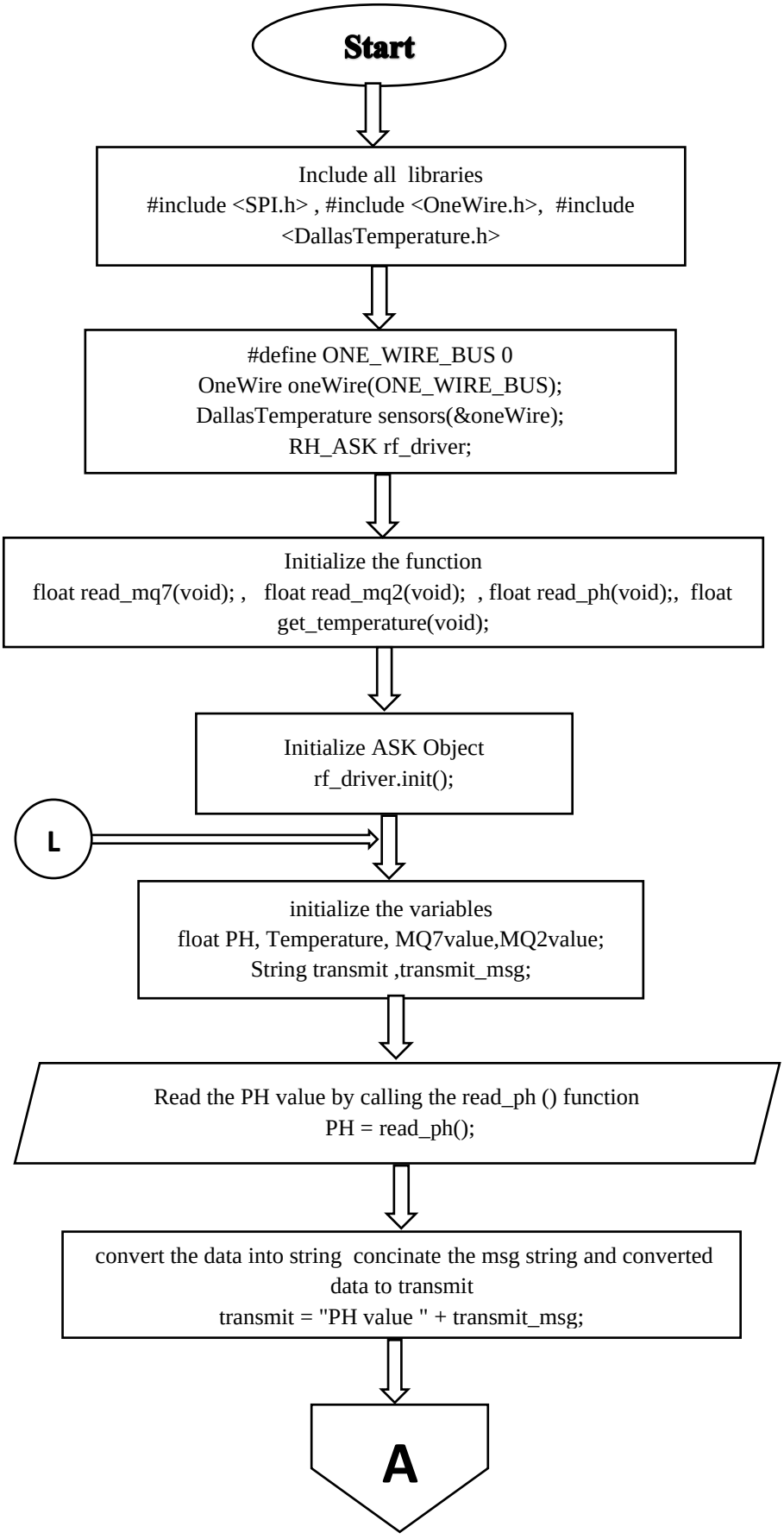
Step12:If is true then print str_out value

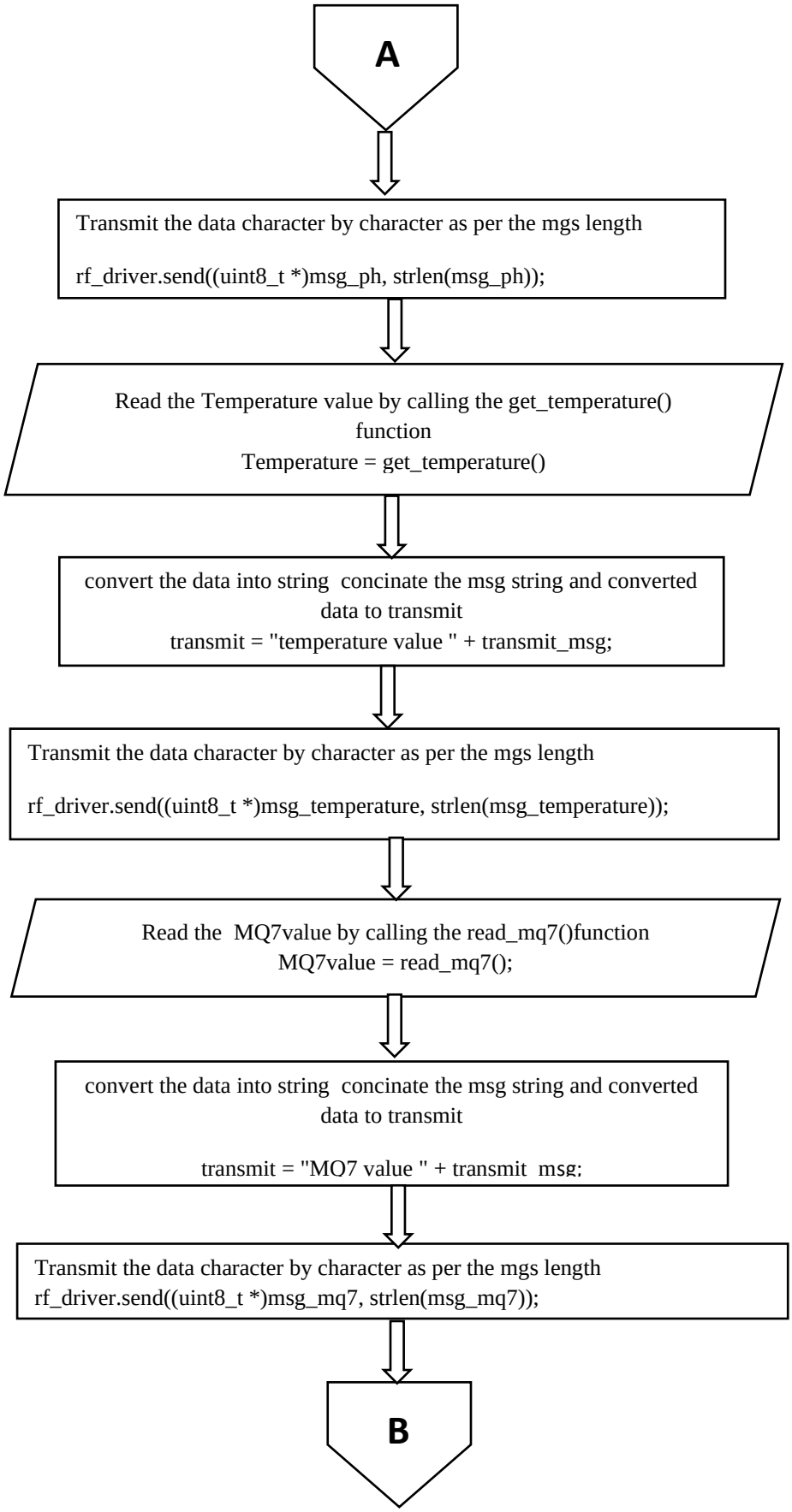
Step13:Else print "data not received"

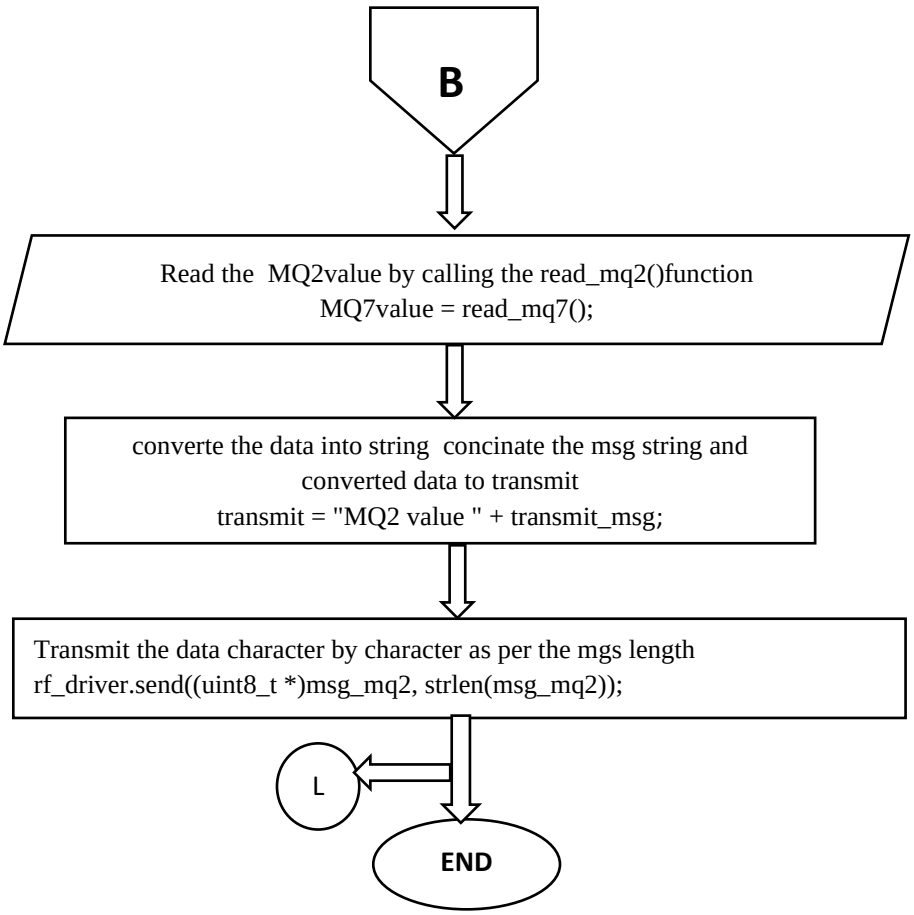
Step14: End

4.3.3 Flowchart

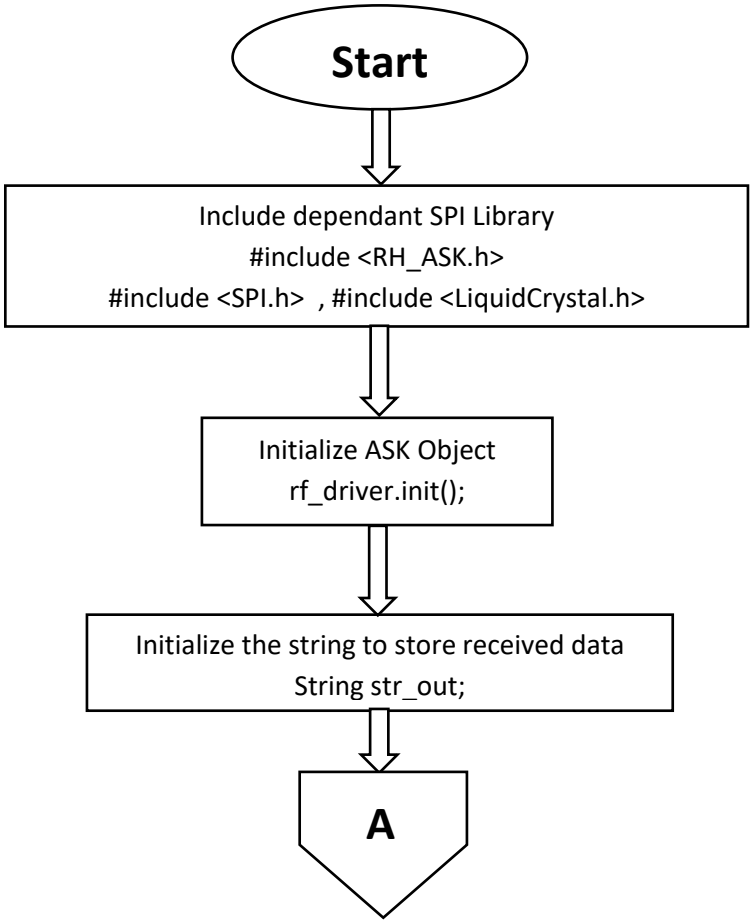
Flowchart for Arduino code at measuring terminal

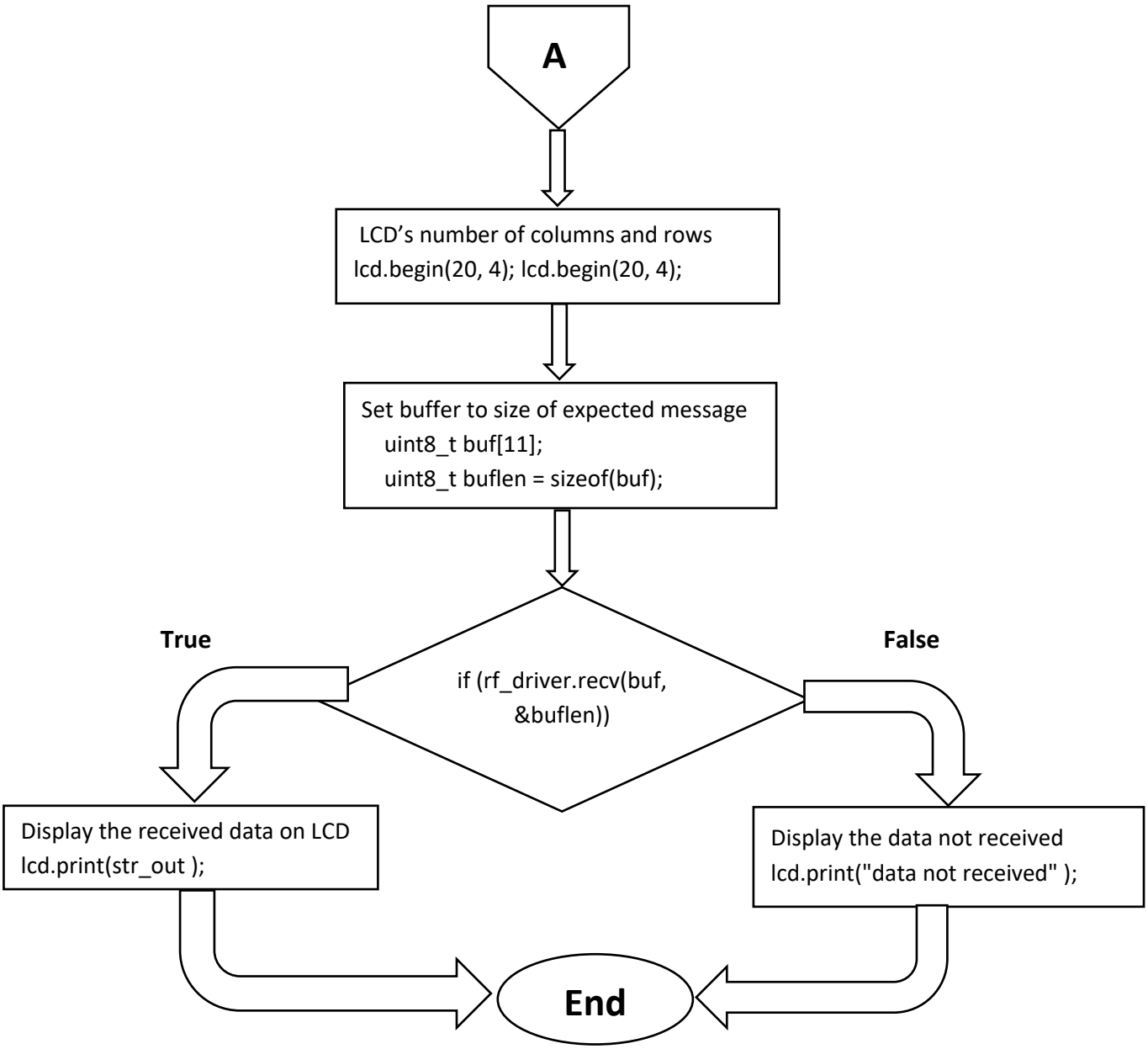



A
Transmit the data character by character as per the mgs length
rf_driver.send((uint8_t *)msg_ph, strlen(msg_ph));
Read the Temperature value by calling the get_temperature() function
Temperature = get_temperature()
convert the data into string concinate the msg string and converted data to transmit
transmit = "temperature value " + transmit_msg;
Transmit the data character by character as per the mgs length
rf_driver.send((uint8_t *)msg_temperature, strlen(msg_temperature));
Read the MQ7value by calling the read_mq7()function
MQ7value = read_mq7();
convert the data into string concinate the msg string and converted data to transmit
transmit = "MQ7 value " + transmit msg;
Transmit the data character by character as per the mgs length
rf_driver.send((uint8_t *)msg_mq7, strlen(msg_mq7));
B



Flowchart for Arduino code at display terminal




A
LCD's number of columns and rows
lcd.begin(20, 4); lcd.begin(20, 4);
Set buffer to size of expected message
uint8_t buf[11];
uint8_t buflen = sizeof(buf);
if (rf_driver.recv(buf,
&buflen))
True
False
Display the received data on LCD
lcd.print(str_out);
Display the data not received
lcd.print("data not received");
End

4.3.4 Program code -:

Program at transmitting terminal

```
#include <RH_ASK.h>  //include a radio head library for Rf module interface
#include <SPI.h>          //Include dependent SPI Library for communicating with transmitter
#include <OneWire.h>   //include a OneWire library for temperature sensor
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 0                // Data wire is plugged into digital pin 0 on the Arduino

OneWire oneWire(ONE_WIRE_BUS);    // Setup a oneWire instance to communicate with any OneWire device
DallasTemperature sensors(&oneWire);  // Pass oneWire reference to DallasTemperature
RH_ASK rf_driver;                          // Create Amplitude Shift Keying Object

//*******FUNCTION INITIALIZATION********

float read_mq7(void);              //initialize function to read carbon monoxide value
float read_mq2(void);              //initialize function to read smoke gas value
float read_ph(void);               //initialize function to read PH value of solution
float get_temperature(void);       //initialize function to read temperature value of solution
void setup()
{
Serial.begin(9600);
rf_driver.init();                  // Initialize ASK Object
}
void loop() {
  float PH, Temperature, MQ7value,MQ2value;  // initialize the variables to store the input value of the sensors
  String transmit ,transmit_msg;                    // Define output strings
 PH = read_ph();                                    //call the function to measure the get the PH data
 transmit_msg=String(PH);                           //convert the data into string for transmitting through the rf module
 transmit = "PH value " + transmit_msg;             //concatenate the msg string and converted data to transmit
 const char *msg_ph = transmit.c_str();              // Compose output character
 rf_driver.send((uint8_t *)msg_ph, strlen(msg_ph));        //transmit the data using the in build function of
                                                    RH library
 rf_driver.waitPacketSent();//wait for the data to be send
 delay(200);

//same procedure for the temperature value
 Temperature = get_temperature();
 transmit_msg=String(Temperature);
 transmit = "Temperature " + transmit_msg;
const char  *msg_temperature = transmit.c_str();
 rf_driver.send((uint8_t *)msg_temperature, strlen(msg_temperature));
 rf_driver.waitPacketSent();
 delay(200);

//same procedure for the carbon monoxide  value
 MQ7value = read_mq7();
 transmit_msg=String(MQ7value);
 transmit = "carbon monoxide " + transmit_msg;
const char  *msg_mq7 = transmit.c_str();
rf_driver.send((uint8_t *)msg_mq7, strlen(msg_mq7));
rf_driver.waitPacketSent();
delay(200);

//same procedure for the smoke gas value
 MQ2value = read_mq2();
  transmit_msg = String(MQ2value);
 transmit = "smoke gas " + transmit_msg;
const char  *msg_mq2 = transmit.c_str();
```

```
 rf_driver.send((uint8_t *)msg_mq2, strlen(msg_mq2));
 rf_driver.waitPacketSent();
 delay(200);

}

//*******DEFINING THE FUNCTION********
//defining the function to read the ph value
float read_ph(void)
{
 const int analogInPin = A0;       //variable to store the data read by analog pin 0
unsigned long int avgValue;       // variable to store the average of sensor value taken

int buf[10],temp;                  // variable to store the 10 samples of ph measured and a "temp" to store
                                   temporary value
for(int i=0;i<10;i++)               //loop to take a 10 samples of the The  PH value
 {
  buf[i]=analogRead(analogInPin);  //read the ph value
  delay(10);
 }for(int i=0;i<9;i++)             //sort the analog values  from small to large
 {
  for(int j=i+1;j<10;j++)
  {
   if(buf[i]>buf[j])
   {
    temp=buf[i];
    buf[i]=buf[j];
    buf[j]=temp;
   }
  }
 }
 avgValue=0;
 for(int i=2;i<8;i++)                              //take the average value of 6 center sample
 avgValue+=buf[i];
 float pHVol=(float)avgValue*5.0/1024/6;           //convert the values in  into volt
 float phValue = -5.70 * pHVol + 21.34;            //convert the volt into pH value
 return phValue;                                   // return the PH value to function call.
 delay(20);
}

//defining the function to read the temperature value
float get_temperature(void)
{
  float Celsius = 0;                               //define the variable to store the temperature value
  sensors.begin();                                 // begin the sensors library
   sensors.requestTemperatures();                  //read the temperature
    Celsius = sensors.getTempCByIndex(0);          //store the  temperature value in the variable
    return Celsius;                                //return the temperature value to function call
}

//defining the function to read the carbon monoxide value
float read_mq7 (void)
{
int pinbuzzer = 11;                //assign the buzzer  pin to digital 11 pin of Arduino
int threshold_level=2;             //assign the threshold pin of sensor to digital pin 2 of arduino
float pinSensor = A2;              //variable to store the analog value of sensor from analog pin2 of arduino
int THRESHOLD = 300;                    //set the threshold level according to the requirement
pinMode(pinbuzzer, OUTPUT);        //assign pin as output
```

```
pinMode(pinSensor, INPUT);                  //assign pin as input
pinMode(threshold_level,INPUT);             //assign pin as input

int analogValue = analogRead(pinSensor);    //read the analog value of carbon monoxide
return analogValue ;                        //return the measured value
digitalWrite(pinbuzzer, LOW);               //low the buzzer
if (analogValue >= THRESHOLD )              //if the read value is above the threshold
  digitalWrite(pinbuzzer,HIGH );            //then sound the buzzer
else
digitalWrite(pinbuzzer,LOW );               //if not then don't sound

}


//defining the function to read the smoke gas  value
float read_mq2 (void)
{
int pinbuzzer = 11;       //assign the buzzer  pin to digital 11 pin of arduino
int threshold_level=1;          //assign the threshold pin of sensor to digital pin 1 of arduino
float pinSensor = A1;           //variable to store the analog value of sensor from analog pin2 of arduino
int THRESHOLD = 450;            //set the threshold level according to the requirement
pinMode(pinbuzzer, OUTPUT);          //assign pin as output
pinMode(pinSensor, INPUT);           //assign pin as input
pinMode(threshold_level,INPUT);      //assign pin as input

int analogValue = analogRead(pinSensor);    //read the analog value of smoke gas
return analogValue ;                        //return the measured value to function call
digitalWrite(pinbuzzer, LOW);               //low the buzzer
if (analogValue >= THRESHOLD )              //if the read value is above the threshold
digitalWrite(pinbuzzer,HIGH );              // then sound the buzzer
else
digitalWrite(pinbuzzer,LOW );               //if not then don't sound

}
```

Program at receiving and display terminal

```
#include <RH_ASK.h>              // Include dependent SPI Library
#include <SPI.h>                  // Create Amplitude Shift Keying Object
RH_ASK rf_driver;
#include <LiquidCrystal.h>              //include the liquid crystal display library
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);          // Creates an LCD object. Parameters: (rs, enable, d4, d5, d6, d7)
String str_out;                           //string to store the received data

void setup()
{
   // Initialize ASK Object
   rf_driver.init();
   // set up the LCD's number of columns and rows:
   lcd.begin(20, 4);
   // Clears the LCD screen
   lcd.clear();
}

void loop()
{
   // Set buffer to size of expected message
   uint8_t buf[11];
   uint8_t buflen = sizeof(buf);
   // Check if received packet is correct size
   if (rf_driver.recv(buf, &buflen))
   {
     str_out = String((char*)buf);            //store the received msg in variable

  // set the cursor to column 0, line 1

 // (note: line 1 is the second row, since counting begins with 0):
      lcd.setCursor(0, 1);
       lcd.print(str_out );
       delay(3000);
       }
       else {
         lcd.print("data not received" );
        }
}
```

Chapter 5-: Hardware Description

5.1 Arduino uno R3

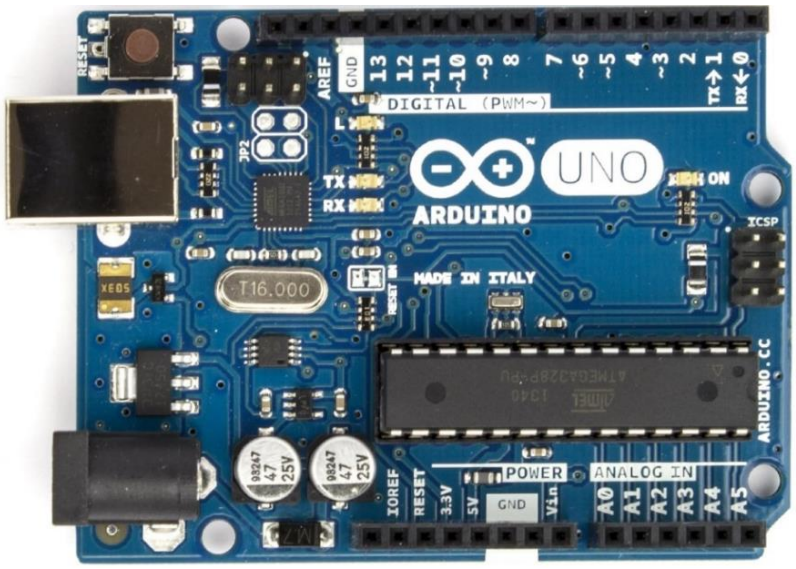


Fig 5.1 Arduino

Overview

The Arduino Uno[16] is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

Revision 2 of the Uno board has a resistor pulling the 8U2 HWB line to ground, making it easier to put into DFU mode.

Revision 3 of the board has the following new features:

- pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible both with the board that use the AVR, which operate with 5V and with the Arduino Due that operate with 3.3V. The second one is a not connected pin, that is reserved for future purposes.
- Stronger RESET circuit.
- Atmega 16U2 replace the 8U2.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

Features of Arduino uno

• Microcontroller	ATmega328
• Operating Voltage	5V
• Input Voltage (recommended)	7-12V
• Input Voltage (limits)	6-20V
• Digital I/O Pins	14 (of which 6 provide PWM output)
• Analog Input Pins	6
• DC Current per I/O Pin	40 mA
• DC Current for 3.3V Pin	50 mA
• Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
• SRAM	2 KB (ATmega328)
• EEPROM	1 KB (ATmega328)
• Clock Speed	16 MHz

Arduino Uno R3 Pin Diagram

The Arduino Uno R3 pin diagram is shown below. It comprises 14-digit I/O pins. From these pins, 6-pins can be utilized like PWM outputs. This board includes 14 digital input/output pins, Analog inputs-6, a USB connection, quartz crystal-16 MHz, a power jack, a USB connection, resonator-16Mhz, a power jack, an ICSP header an RST button.

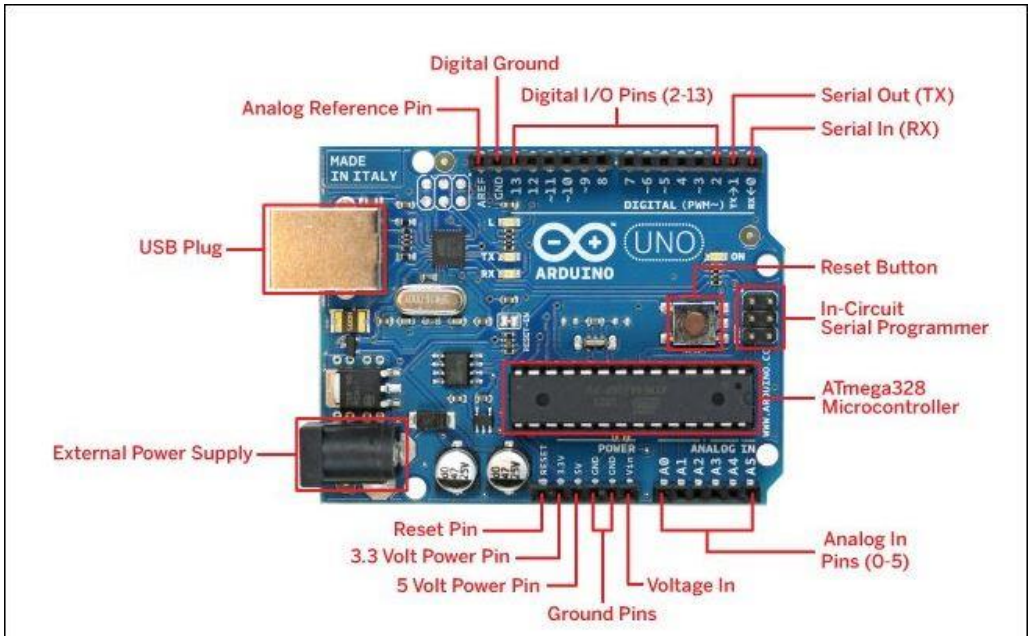


Fig 5.2 Arduino Uno R3 Pin Diagram

Power Supply

The power supply of the Arduino can be done with the help of an exterior power supply otherwise USB connection. The exterior power supply (6 to 20 volts) mainly includes a battery or an AC to DC adapter. The connection of an adapter can be done by plugging a center-positive plug (2.1mm) into the power jack on the board. The battery terminals can be placed in the pins of Vin as well as GND. The power pins of an **Arduino board** include the following.

Vin: The input voltage or Vin to the Arduino while it is using an exterior power supply opposite to volts from the connection of USB or else **RPS (regulated power supply)**. By using this pin, one can supply the voltage.
5Volts: The RPS can be used to give the power supply to the microcontroller as well as components which are used on the Arduino board. This can approach from the input voltage through a regulator.
3V3: A 3.3 supply voltage can be generated with the onboard regulator, and the highest draw current will be 50 mA.
GND: GND (ground) pins

Memory

The memory of an ATmega328 microcontroller includes 32 KB and 0.5 KB memory is utilized for the Boot loader), and also it includes SRAM-2 KB as well as EEPROM-1KB.

Input and Output

We know that an arguing Uno R3 includes 14-digital pins which can be used as an input otherwise output by using the functions like pin Mode (), digital Read(), and digital Write(). These pins can operate with 5V, and every digital pin can give or receive 20mA, & includes a 20k to 50k ohm pull up resistor. The maximum current on any pin is 40mA which cannot surpass for avoiding the microcontroller from the damage. Additionally, some of the pins of an Arduino include specific functions.

Serial Pins

The serial pins of an Arduino board are TX (1) and RX (0) pins and these pins can be used to transfer the TTL serial data. The connection of these pins can be done with the equivalent pins of the ATmega8 U2 USB to TTL chip.

External Interrupt Pins

The external interrupt pins of the board are 2 & 3, and these pins can be arranged to activate an interrupt on a rising otherwise falling edge, a low-value otherwise a modify in value

PWM Pins

The PWM pins of an Arduino are 3, 5, 6, 9, 10, & 11, and gives an output of an 8-bit PWM with the function analog Write ().

SPI (Serial Peripheral Interface) Pins

The SPI pins are 10, 11, 12, 13 namely SS, MOSI, MISO, SCK, and these will maintain the **SPI communication** with the help of the SPI library.

LED Pin

An arguing board is inbuilt with a LED using digital pin-13. Whenever the digital pin is high, the LED will glow otherwise it will not glow.

TWI (2-Wire Interface) Pins

The TWI pins are SDA or A4, & SCL or A5, which can support the communication of TWI with the help of Wire library.

AREF (Analog Reference) Pin

An analog reference pin is the reference voltage to the inputs of an analog i/ps using the function like analog Reference().

Reset (RST) Pin

This pin brings a low line for resetting the microcontroller, and it is very useful for using an RST button toward shields which can block the one over the Arduino R3 board.

Communication

The communication protocols of an Arduino Uno include SPI, I2C, and **UART serial communication**.

UART

An Arduino Uno uses the two functions like the transmitter digital pin1 and the receiver digital pin0. These pins are mainly used in UART TTL serial communication.

I2C

An Arduino UNO board employs SDA pin otherwise A4 pin & A5 pin otherwise SCL pin is used for I2C communication with wire library. In this, both the SCL and SDA are CLK signal and data signal.

SPI Pins

The SPI communication includes MOSI, MISO, and SCK.

MOSI (Pin11)
This is the master out slave in the pin, used to transmit the data to the devices
MISO (Pin12)
This pin is a serial CLK, and the CLK pulse will synchronize the transmission of which is produced by the master.
SCK (Pin13)
The CLK pulse synchronizes data transmission that is generated by the master. Equivalent pins with the SPI library is employed for the communication of SPI. ICSP (in-circuit serial programming) headers can be utilized for programming **ATmega microcontroller** directly with the boot loader.

Arduino Uno R3 Programming

- The programming of an Arduino Uno R3 can be done using IDE software. The microcontroller on the board will come with pre-burned by a boot loader that permits to upload fresh code without using an exterior hardware programmer.
- The communication of this can be done using a protocol like STK500.
- We can also upload the program in the microcontroller by avoiding the boot loader using the header like the In-Circuit Serial Programming.

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 halfsecond 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; see this forum thread for details.

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.

5.2 PH SENSOR SKU SEN0161[6]

Introduction

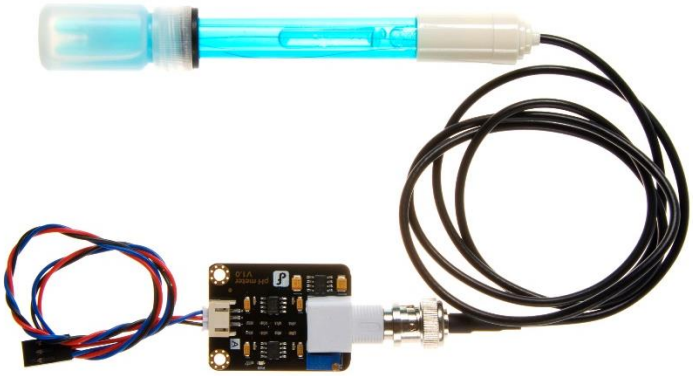


Fig 5.3 PH Sensor

Need to measure water quality and other parameters but haven't got any low cost pH meter? Find it difficult to use with Arduino? Here comes an analog pH meter, specially designed for **Arduino controllers** and has built-in simple, convenient and practical connection and features. It has an LED which works as the Power Indicator, a BNC connector and PH2.0 sensor interface. To use it, just connect the pH sensor with BNC connector, and plug the PH2.0 interface into the analog input port of any Arduino controller. If pre-programmed, you will get the pH value easily. Comes in compact plastic box with foams for better mobile storage. Attention:In order to ensure the accuracy of the pH probe , it will need to use the standard solution to calibrate it regularly . Generally, the period is about half a year. If meaure the dirty aqueous solution, it require need to increase the frequency of calibration.

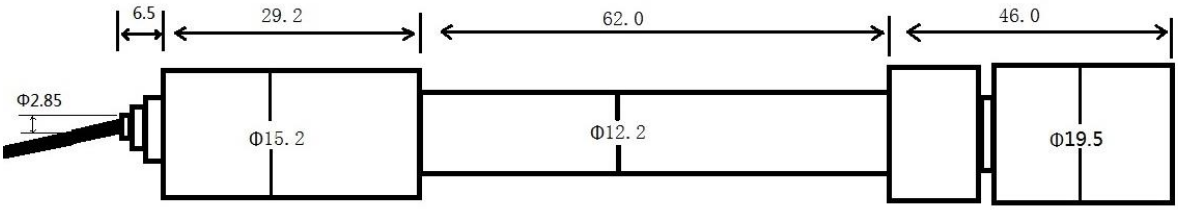


Fig. 5.4 PH Electrode Size

Specification

- Module Power : 5.00V
- Module Size : 43mm×32mm
- Measuring Range:0-14PH
- Measuring Temperature :0-60 ℃
- Accuracy : ± 0.1 PH (25 ℃)
- Response Time : ≤ 1min
- PH Sensor with BNC Connector
- PH2.0 Interface (3 foot patch)
- Gain Adjustment Potentiometer
- Power Indicator LED
- Cable Length from sensor to BNC connector:660mm

5.3 Temperature sensor DS18B20[8][14]

Fig .5.5 DS18B20 Waterproof Temperature Sensor Cable

Description

This Maxim-made item is a digital thermo probe or sensor that employs DALLAS DS18B20. Its unique 1-wire interface makes it easy to communicate with devices. It can converts temperature to a 12-bit digital word in 750ms (max). Besides, it can measures temperatures from -55°C to +125°C (-67F to +257F). In addition, this thermo probe doesn't require any external power supply since it draws power from data line. Last but not least, like other common thermo probe, its stainless steel probe head makes it suitable for any wet or harsh environment.

Fig 5.6 Temperature Sensor Cable

Feature:

- Usable temperature range: -55 to 125°C
- 9 to 12 bit selectable resolution
- Uses 1Wire interface requires only one digital pin for communication
- Unique 64 bit ID burned into chip
- Multiple sensors can share one pin
- ±0.5°C Accuracy from -10°C to +85°C
- Temperature limit alarm system
- Query time is less than 750ms
- Usable with 3.0V to 5.5V power/data
- Stainless steel tube 6mm diameter by 30mm long
- Cable is 36" long and 4mm diameter
- Contains DS18B20 temperature sensor
- Sensor has three wires - Red connects to 3-5V, Blue/Black connects to ground and Yellow/White is data .

DS18B20 Sensor Pinout

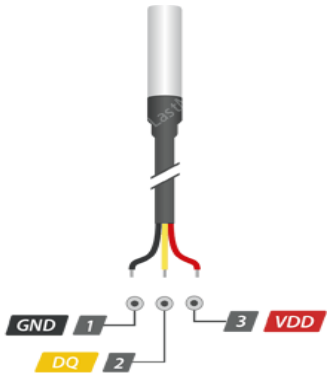


Fig 5.7 pin diagram of temperature sensor

GND is a ground pin.

DQ is 1-Wire Data Bus should be connected to a digital pin on microcontroller.

VDD pin supplies power for the sensor which can be between 3.3 to 5V.

5.4 MQ-2 Smoke Gas Sensor

MQ2 is one of the commonly used gas sensors in MQ[7] sensor series. It is a Metal Oxide Semiconductor (MOS) type Gas Sensor also known as Chemiresistors as the detection is based upon change of resistance of the sensing material when the Gas comes in contact with the material. Using a simple voltage divider network, concentrations of gas can be detected.

Mq2 gas sensor works on 5v dc and draws around 800mw. It can detect lpg, smoke, alcohol, propane, hydrogen, methane and carbon monoxide concentrations anywhere from 200 to 10000ppm.

Specifications-:

Operating voltage	5V
Load resistance	20 KΩ
Heater resistance	33Ω ± 5%
Heating consumption	<800mw
Sensing Resistance	10 KΩ – 60 KΩ
Concentration Scope	200 – 10000ppm
Preheat Time	Over 24 hour

Internal structure of MQ2 Gas Sensor

The sensor is actually enclosed in two layers of fine stainless steel mesh called **Anti-explosion network**. It ensures that heater element inside the sensor will not cause an explosion, as we are sensing flammable gases.

Fig 5.8 MQ2 Sensor

It also provides protection for the sensor and filters out suspended particles so that only gaseous elements are able to pass inside the chamber. The mesh is bound to rest of the body via a copper plated clamping ring.

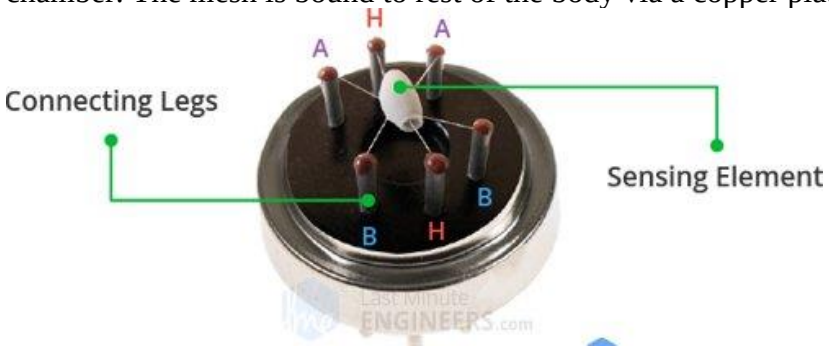
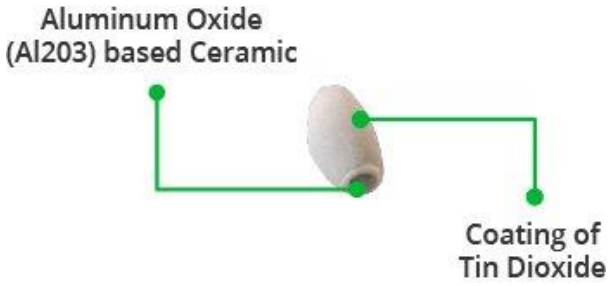


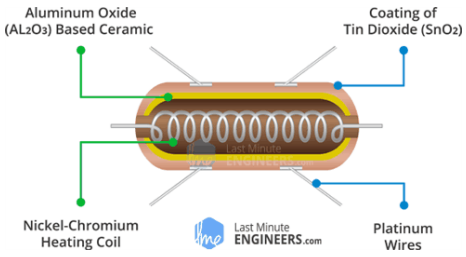
Fig 5.9 MQ2 Sensor Pinout

This is how the sensor looks like when outer mesh is removed. The star-shaped structure is formed by the sensing element and six connecting legs that extend beyond the Bakelite base. Out of six, two leads (**H**) are responsible for heating the sensing element and are connected through **Nickel-Chromium coil**, well known conductive alloy.

The remaining four leads (**A** & **B**) responsible for output signals are connected using **Platinum Wires**. These wires are connected to the body of the sensing element and convey small changes in the current that passes through the sensing element.



The tubular sensing element is made up of **Aluminum Oxide** (AL_2O_3) based ceramic and has a coating of **Tin Dioxide** (SnO_2). The Tin Dioxide is the most important material being sensitive towards combustible gases. However, the ceramic substrate merely increases heating efficiency and ensures the sensor area is heated to a working temperature constantly.



So, the Nickel-Chromium coil and Aluminum Oxide based ceramic forms a **Heating System**; while Platinum wires and coating of Tin Dioxide forms a **Sensing System**.

Working principle of Gas Sensor

When tin dioxide (semiconductor particles) is heated in air at high temperature, oxygen is adsorbed on the surface. In clean air, donor electrons in tin dioxide are attracted toward oxygen which is adsorbed on the surface of the sensing material. This prevents electric current flow.

In the presence of reducing gases, the surface density of adsorbed oxygen decreases as it reacts with the reducing gases. Electrons are then released into the tin dioxide, allowing current to flow freely through the sensor.

Hardware Overview – MQ2 Gas Sensor Module

Since MQ2 Gas Sensor is not breadboard compatible . It's very easy to use and comes with two different outputs. It not only provides a binary indication of the presence of combustible gases but also an analog representation of their concentration in air.

Fig 5.10 MQ2 Sensor Module

The analog output voltage provided by the sensor changes in proportional to the concentration of smoke/gas. The greater the gas concentration, the higher is the output voltage; while lesser gas concentration results in low output voltage. The following animation illustrates the relationship between gas concentration and output voltage. The analog signal from MQ2 Gas sensor is further fed to LM393 High Precision Comparator (soldered on the bottom of the module), of course to digitize the signal. Along with the comparator is a little potentiometer you can turn to adjust the sensitivity of the sensor. You can use it to adjust the concentration of gas at which the sensor detects it.

Calibrate MQ2 Gas Sensor Module

To calibrate the gas sensor you can hold the gas sensor near smoke/gas you want to detect and keep turning the potentiometer until the Red LED on the module starts glowing. Turn the screw clockwise to increase sensitivity or anticlockwise to decrease sensitivity.

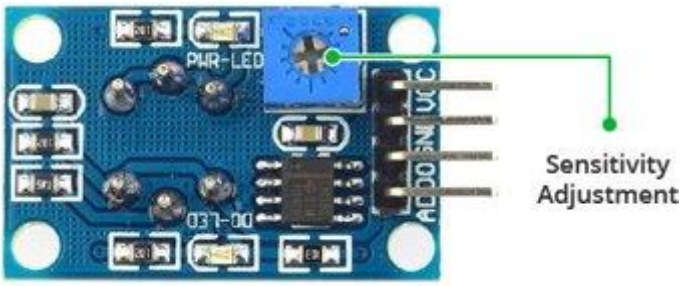


Fig 5.11 MQ 2 Sensor Module Pinouts

The comparator on the module continuously checks if the analog pin (**A0**) has hit the threshold value set by potentiometer. When it crosses the threshold, the digital pin (**D0**) will go HIGH and signal LED turns on. This setup is very useful when you need to trigger an action when certain threshold is reached. For example, when the smoke crosses a threshold, you can turn on or off a relay or instruct your robot to blow air/sprinkle water. You got the idea!

MQ2 Gas Sensor Module Pinout

Now let's have a look at the pinout.

VCC supplies power for the module. You can connect it to 5V output from your Arduino.

GND is the Ground Pin and needs to be connected to GND pin on the Arduino.

D0 provides a digital representation of the presence of combustible gases.

A0 provides analog output voltage in proportional to the concentration of smoke/gas.

5.5 Carbon Monoxide Sensor(MQ7)

Description

MQ-7 Carbon Monoxide Sensor Module[7] detects the concentrations of CO in the air and outputs its reading as an analog voltage. The sensor can measure concentrations of 10 to 10,000 ppm.The sensor can operate at temperatures from -10 to 50°C and consumes less than 150 mA at 5 V.

This MQ-7 Carbon Monoxide Sensor provides both digital and analog outputs. Threshold level for digital output can be easily adjusted using the preset on the board. MQ-7 sensor module can be easily interfaced with Micro-controllers, arduino and etc.

Fig 5.12 MQ7 Sensor Module

Electrical properties:

- Input voltage: DC5V power (current): 150mA
- DO output: TTL digital 0 and 1 (0.1 and 5V)
- AO output :0.1-0 .3 V (relatively clean), the highest concentration of voltage around 4V
- Heater voltage: 5±0.2V（AC•DC)
- Working temperature: -10~50℃ (nominal temperature: 20℃)
- Working humidity: 95%RH (nominal humidity: 65%RH)
- Loading resistance: 10K (adjustable)
- Sensitivity: ≥3% and Response time: ≤1S (preheating 3-5 minutes) and Component power consumption: ≤0.7W and Dimension: 35mm×20mm×11mm

Features:

- Wide detecting scope
- High sensitivity and fast response
- Long life and stable

5.6 433 MHz RF Transmitter Module

The 433MHz wireless module[13] is one of the cheap and easy to use modules for all wireless projects. These modules can be used only in pairs and only simplex communication is possible. Meaning the transmitter can only transmit information and the receiver can only receive it, so you can only send data from point A to B and not from B to A.

The module could cover a minimum of 3 meters and with proper antenna a power supplies it can reach upto 100 meters theoretically. But practically we can hardly get about 30-35 meters in a normal test conditions.

So if you are looking for a simple wireless communication to transmit information within a short distance then these RF pair could be the right choice.

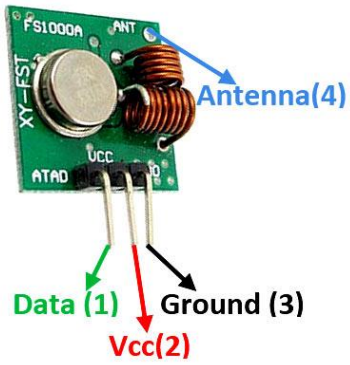


Fig 5.13 Diagram Of 433 Mhz Transmitter Module

Pin Configuration:

No:	Pin Name	Description
1	Vcc	Power supply (+5V only)
2	Data	Data to be transmitted is sent to this pin
3	Ground	Connected to the ground of the circuit
4	Antenna	Solder wire/antenna to improve range (not mandatory)

433 MHz Module Specifications:

- Wireless (RF) Simplex Transmitter and Receiver
- Transmitter Operating Voltage: +5V only
- Transmitter Operating current: 9mA to 40mA
- Operating frequency: 433 MHz
- Transmission Distance: 3 meters (without antenna) to 100 meters (maximum)
- Modulating Technique: ASK (Amplitude shift keying)
- Data Transmission speed: 10Kbps
- Circuit type: Saw resonator
- Low cost and small package

5.7 433MHz RF receiver module

433MHz RF receiver Pinout diagram

433MHz RF receiver has 6 pins, which offers 4 types of functions. The pinout diagram depicts the functionality of all these pins.[13]

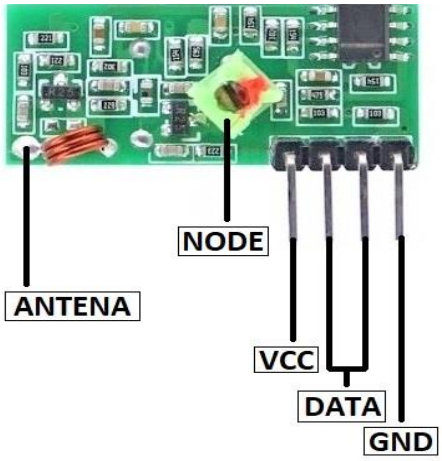


Fig 5.14 Diagram Of 433mhz RF Receiver

Pin Configurations Description

VCC Pin: VCC is the power input pin for the RF module. The power will active the internal circuit to make it functional.

GND Pin: For common ground, the RF module has only one ground pin. The module needs to use with other devices and the common ground will help the RF module to interface with external devices.

Data Pin: 433MHz RF Module has two data input pins which are internally common with each other. Only data should receive from one pin at a time.

Antenna Pin: This module has an antenna pin which helps to connect the external wire to extend the range up to 100 meters. The size of the antenna will depend on the operating frequency.

Use RF receiver

The RF receiver module may look simple to use but it is a little bit hard to receive the data from itself. The module receives the data in the form of a signal and sends it to the data pin. The data received by the module is always in an encoded form which is decodable by two methods. The first one is through programming and the second is a decoder.

433MHz RF Receiver Module Features

- The RF receiver delivers the output in an encoded form.
- The operating voltage range of the module is 5V maximum.
- The frequency of the receiver is changeable using a green node present on it.
- It is one of the cheapest receivers and has low power consumption.
- 433MHz RF module uses the ASK/OOK signal as an input.

5.8 LCD 20x4

The term LCD stands for liquid crystal display.[12] It is one kind of electronic display module used in an extensive range of applications like various circuits & devices like mobile phones, calculators, computers, TV sets, etc. These displays are mainly preferred for multi-segment light-emitting diodes and seven segments. The main benefits of using this module are inexpensive; simply programmable, animations, and there are no limitations for displaying custom characters, special and even animations, etc.

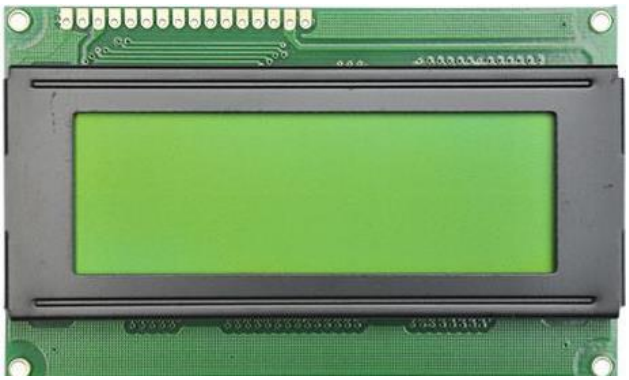


Fig. 5.15 . 20X4 LCD

The 20×4 LCD pinout is shown below.

- Pin1 (Ground/Source Pin): This is a GND pin of display, used to connect the GND terminal of the microcontroller unit or power source.
- Pin2 (VCC/Source Pin): This is the voltage supply pin of the display, used to connect the supply pin of the power source.
- Pin3 (V0/VEE/Control Pin): This pin regulates the difference of the display, used to connect a changeable POT that can supply 0 to 5V.
- Pin4 (Register Select/Control Pin): This pin toggles among command or data register, used to connect a microcontroller unit pin and obtains either 0 or 1(0 = data mode, and 1 = command mode).
- Pin5 (Read/Write/Control Pin): This pin toggles the display among the read or writes operation, and it is connected to a microcontroller unit pin to get either 0 or 1 (0 = Write Operation, and 1 = Read Operation).
- Pin 6 (Enable/Control Pin): This pin should be held high to execute Read/Write process, and it is connected to the microcontroller unit & constantly held high.
- Pins 7-14 (Data Pins): These pins are used to send data to the display. These pins are connected in two-wire modes like 4-wire mode and 8-wire mode. In 4-wire mode, only four pins are connected to the microcontroller unit like 0 to 3, whereas in 8-wire mode, 8-pins are connected to microcontroller unit like 0 to 7.
- Pin15 (+ve pin of the LED): This pin is connected to +5V
- Pin 16 (-ve pin of the LED): This pin is connected to GND.

Features of LCD 20x4

- The operating voltage of this LCD is 4.7V-5.3V
- It includes two rows where each row can produce 16-characters.
- The utilization of current is 1mA with no backlight
- Every character can be built with a 5×8 pixel box
- The alphanumeric LCDs alphabets & numbers
- Is display can work on two modes like 4-bit & 8-bit
- These are obtainable in Blue & Green Backlight
- It displays a few custom generated characters

Registers of LCD

A 20×4 LCD has two registers like data register and command register. The RS (register select) is mainly used to change from one register to another. When the register set is '0', then it is known as command register. Similarly, when the register set is '1', then it is known as data register.[12]

Command Register

The main function of the command register is to store the instructions of command which are given to the display. So that predefined tasks can be performed such as clearing the display, initializing, set the cursor place, and display control. Here commands processing can occur within the register.

Data Register

The main function of the data register is to store the information which is to be exhibited on the LCD screen. Here, the ASCII value of the character is the information which is to be exhibited on the screen of LCD. Whenever we send the information to LCD, it transmits to the data register, and then the process will be starting there. When register set =1, then the data register will be selected.

Important command codes for LCD:-

Sr. No.	Hex Code	Command to LCD instruction Register
1	01	Clear display screen
2	02	Return home
3	04	Decrement cursor (shift cursor to left)
4	06	Increment cursor (shift cursor to right)
5	05	Shift display right
6	07	Shift display left
7	08	Display off, cursor off
8	0A	Display off, cursor on
9	0C	Display on, cursor off
10	0E	Display on, cursor blinking
11	0F	Display on, cursor blinking
12	10	Shift cursor position to left
13	14	Shift cursor position to right
14	18	Shift the entire display to the left
15	1C	Shift the entire display to the right
16	80	Force cursor to beginning (1st line)
17	C0	Force cursor to beginning (2nd line)
18	38	2 lines and 5×7 matrix

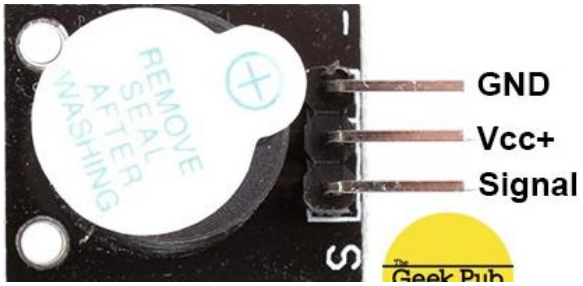
5.9 Buzzer

Active Buzzer Module KY-012 Arduino module, it produces a single-tone sound when signal is high. To produce different tones use the KY-006 Passive Buzzer module. The **KY-012 active piezo buzzer** is a 3-pin module that creates an audible sound at 2.5 kHz without the need for pulse width modulation (PWM) or any additional complex code. The only requirement is to set the signal pin to **HIGH**.

Fig 5.16 Buzzer

DEVICE PINOUT AND SCHEMATICS

This module has three pins: GND, Vcc+, and Signal.



KY-012 Specifications

The KY-012 Active Buzzer module consists of an active piezoelectric buzzer, it generates a sound of aproximately 2.5kHz when signal is high.

Operating Voltage	3.5V ~ 5.5V
Maximum Current	30mA / 5VDC
Resonance Frequency	2500Hz ± 300Hz
Minimum Sound Output	85Db @ 10cm
Working Temperature	-20°C ~ 70°C [-4°F ~ 158°F]
Storage Temperature	-30°C ~ 105°C [-22°F ~ 221°F]
Dimensions	18.5mm x 15mm [0.728in x 0.591in]

Chapter 6-: System Operation

This chapter consist of all about the operation of the system . This chapter will highlight the work for the system operator and will consist of the measure and procedure to be taken while using the system

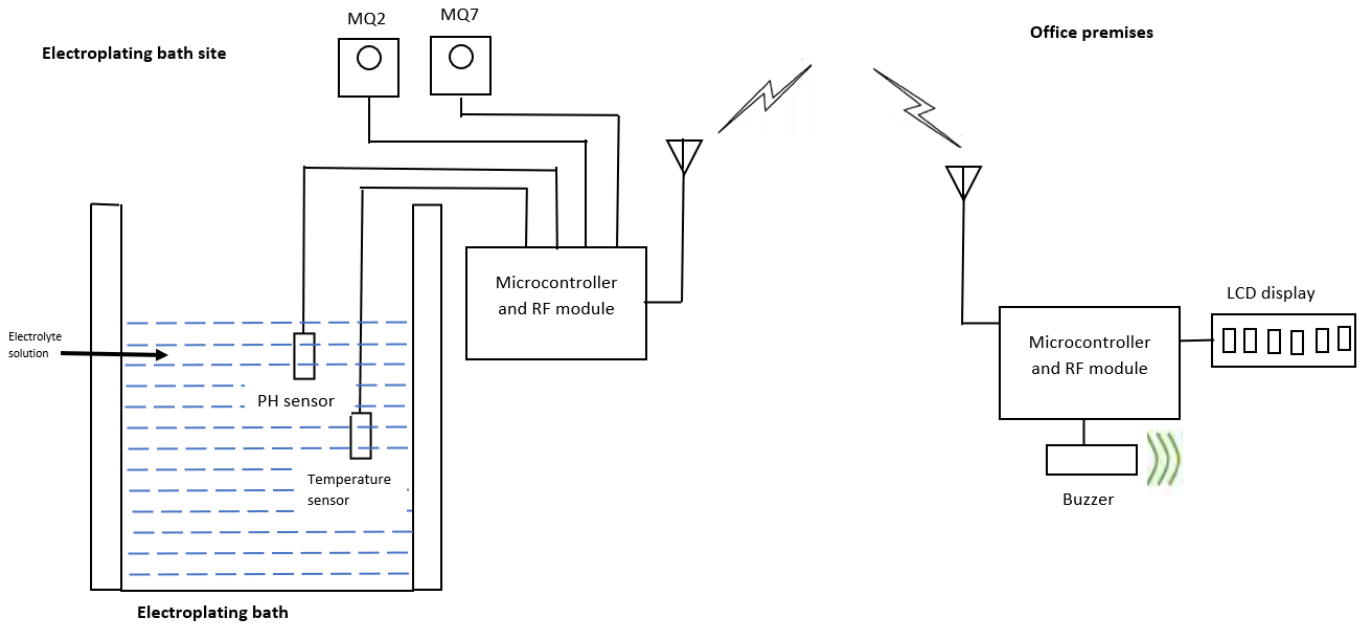


Fig 6.1 System operating in work mode

6.1 Calibration Guidelines

Basically whole system does not need the calibration but a few sensors need to be calibrate according to the use of the electrolyte solution you are using, and what material you are electroplating.

6.1.1 Temperature and MQ series sensors

This sensors doesn't need to be calibrated. They need to set the threshold value by using the pot given on the sensor itself. According to it you need to put the threshold value in the program which will in turn acuate the buzzer indicating the measured value reached beyond the threshold limits.

6.1.2 PH sensor

- **PH limit setting**

There is another pot that acts like a limit switch. Basically, the D0 pin on the sensor board will supply 3.3V to the pin until a preset PH value (that you set with the limit pot) is reached, at this point a red LED will light up and the pin will go down to about 0V. I did not play with this much but suppose it can be handy if you want to activate a buzzer or something if a certain PH is reached, it will work great on an Arduino digital port – that will go high from about 2V up. This will work if the PH value goes higher than the set value. If you want it to trigger something when the PH goes lower, you need to monitor the digital pin to trigger when the digital pin goes low. You will unfortunately not be able to set this limit between two values, either if the pH goes up to high or if the PH drop to low. Programmatically it can be possible to do an upper and lower limit.[6]

- **Connecting and calibrating the PH probe.**

The hard part is over and this offset does not have to be set again, even if you change PH probes. We have PH probes available here: PH probe Electrode BNC connector

Here is a couple of things to know about PH probes:

- The probes readings change over time and need to be calibrated every now and again to make sure the value is still the same and be adjusted if it did change.

- You need at least one PH buffer solution to calibration your PH probe. They are available at many different PH values, A buffer solution of 6.86 and 4.01 is most common as it covers the range of most applications. If you are only going to use one buffer solution make sure its value is near the value range you will use in your normal tests – if it is pool water a buffer solution of 6.86 is usually near enough.
- Buffers come in pre-made solutions or as a powder. I prefer the powder because it is cheaper and does not have an expiration date. The powder is easy to make up as well, I suppose it depends on the power you will use, the one I use you add the powder to 250ml distilled water and stir until all powder is dissolved. It will last about a month once you added water to it.
- A PH probe **takes some time** to get to the right value, allow it to be in the liquid you want to measure for at least two minutes or longer, it does not mean it will be stable at one ph value, it will jump around a bit between 3 or 4 values but on the last digit, for example, between 6.84 – 6.88
- PH values differ in different temperatures, although that might sound cumbersome, in the temperature range between 10 – 30 degrees Celcius the PH does not differ and from 30 degrees Celcius it goes up with about a pH of 0.01 to 50 degrees Celcius that is about 0.06. In most uses, it will be below 30 degrees Celcius and temperature do not have to be calculated in.

6.2 Precautionary measures

- Do not expose this sensor to water and frost.
- Applying a voltage higher than 5V or applying the voltage to the wrong pins may damage the sensor.
- Exposing the sensor to a high concentration of gases for a long time may have a negative effect on its performance.
- Shaking or vibrating the sensor may decrease its accuracy.
- Do not let the measuring terminal box (containing the arduino) to be in direct contact with the electrolyte solution present in the bath
- Do not power up the arduino until you hav made the proper conection
- Make the use of proper standard ideal solution while calibrating the ph sensor
- Make sure that BNC connecter is properly shorted to eash other while setting offset of ph sensor[details given in chapter no 3]
- According to the requirement only set the set the threshold value of the mq2 and mq7 sensor otherwise you may get wrong indication.
- Make sure that code has been properly uploaded in the arduino
- And very important keep at least 200m distance between transmitter and receiver module for proper data transfer

6.3 Standard operating Procedure (SOP)

Step1-: before operating the system ensure that you have followed all the calibration and precautionary measures.

Step 2 -: Put all the sensors at their actual site (mq2and mq7 at the bath site and temperature and ph sensor in the electrolyte solution)

Step 3-: Upload the program in the arduino and power on it

Step4-: Wait for some delay and see the lcd display if it showing the readings and if not then follow the calibration and precautionary steps again and implement it .

Step5-: Now the system is in its work mode. While implementation the system to another bath you need to follow the steps given above

Chapter 7-: Hardware Simulation

This chapter consist of the simulation done before implementing the actual circuits which gives us the advantages such as

- It is much faster to build the circuit in the simulator than in real life
- If it does not work at first, no harm done. It is easy to adjust and improve.
- You can access any node in the circuit with a click of the mouse, which makes debugging much faster.
- You can try components that you do not physically have.

7.1 Simulation software -: Proteus [15]

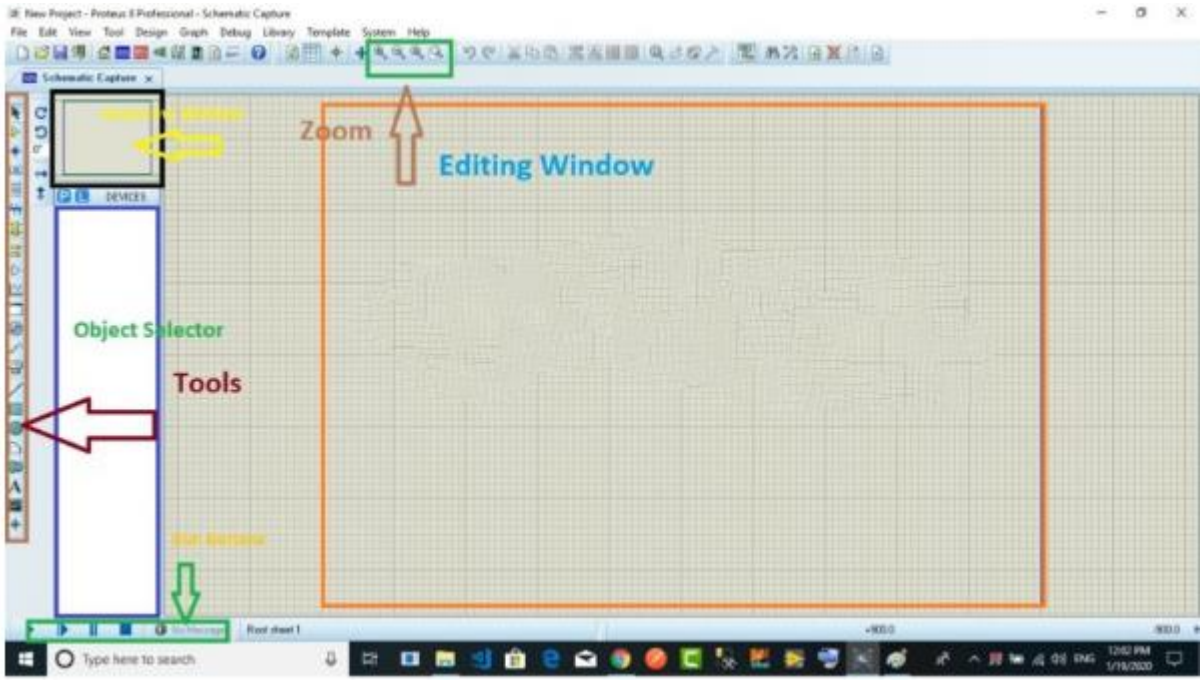


Fig 7.1 proteus window

Proteus is used to simulate, design and drawing of electronic circuits. It was invented by the Labcenter electronic.By using proteus you can make two-dimensional circuits designs as well.With the use of this engineering software, you can construct and simulate different electrical and electronic circuits on your personal computers or laptops.There are numerous benefits to simulate circuits on proteus before make them practically.Designing of circuits on the proteus takes less time than practical construction of the circuit.The possibility of error is less in software simulation such as loose connection that takes a lot of time to find out connections problems in a practical circuit.Circuit simulations provide the main feature that some components of circuits are not practical then you can construct your circuit on proteus.There is zero possibility of burning and damaging of any electronic component in proteus.

The electronic tools that are very expensive can easily get in proteus such as an oscilloscope.Using proteus you can find different parents of circuits such as current, a voltage value of any component and resistance at any instant which is very difficult in a practical circuit.

7.2 Simulation circuit

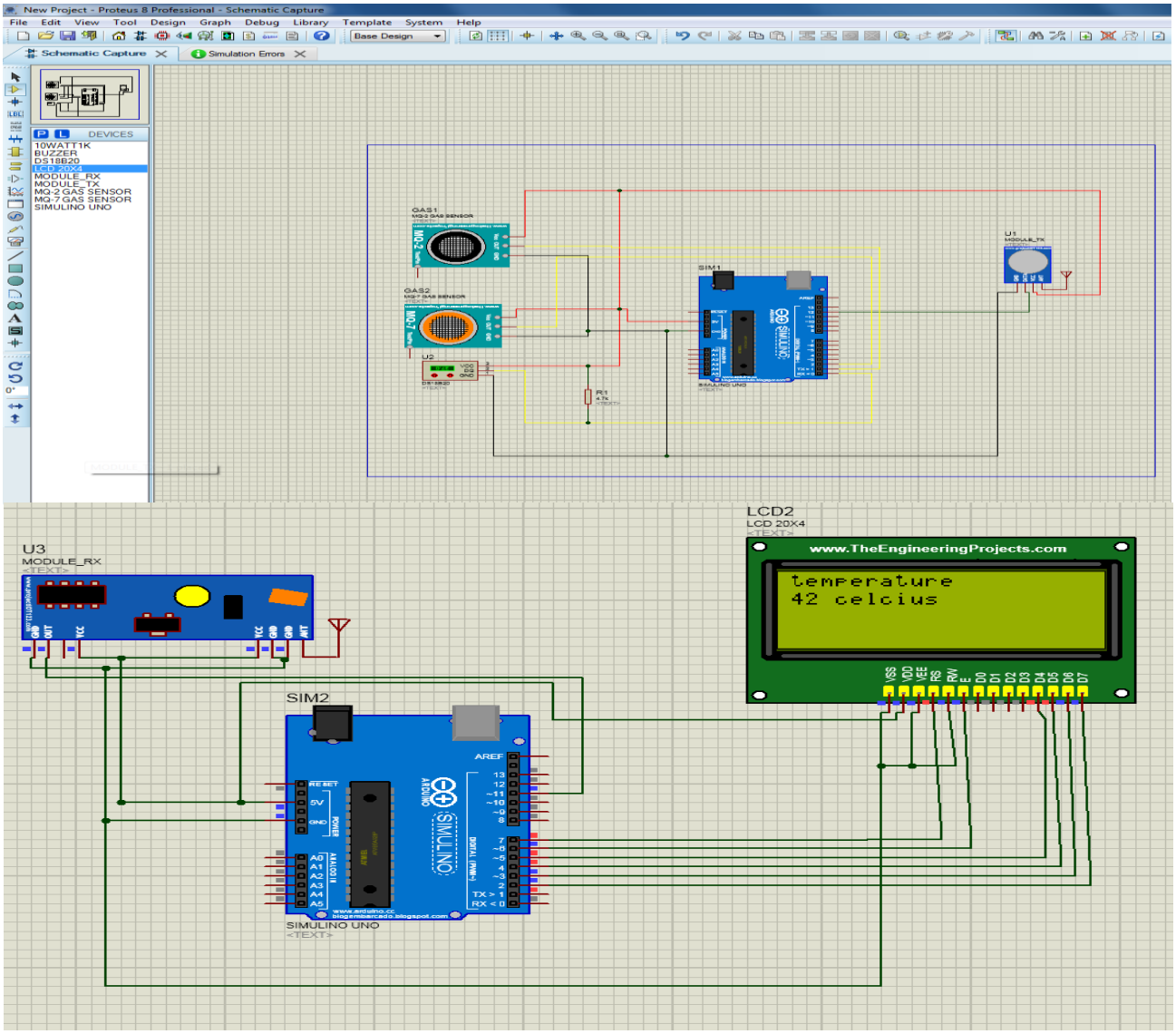


Fig 7.2 hardware simulation

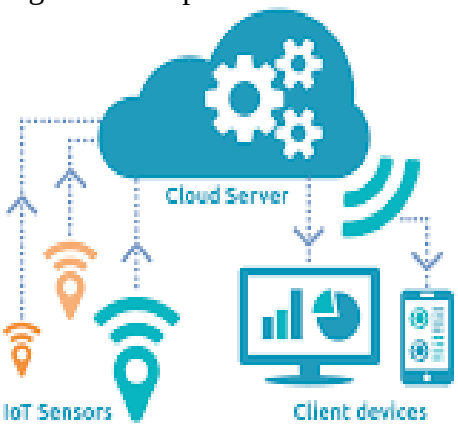
How to upload the code in arduino in proteus

- Write the code in the arduino ide
- Check whether the compilation option is checked to generate the hex file. It can be found from **File —> Preferences.**
- Compile the code and copy the hex file-path.
- Double click on the Arduino board to insert the hex file of code.
- After inserting a hex file, you can start the simulation by pressing the play key.
- Now the circuit will simulate and give the output as per program

Chapter 8-: Future Scope

1. The performance of the system can be future improves in terms of operating speed, memory, memory capacity and by using the arduino mega. The number of channels can be increase to interface more number of sensors so whole plant can be monitored
2. If the advancement made not only the one bath but whole plant can be monitored and can be controlled automatically eg if the Ph value of certain bath goes beyond the level controller wil automatically dilute the electrolyte solution by actuating solonied valve of water and if goes below the level controller will add the electrolyte solution in it. In similar way it I applicable to temperature and other parameters.[b][5]

3. In terms it can also be made compliance to IOT where the data collected can be send to cloud and can be monitored by Apk application or software. This can be done by interfacing the wsp8266 Wifi module which will give the internet access for the data storage
4. It can also be modified with the data logger to store the measure sensor data over a period of time
5. A speaking voice alarm could be used instead of the normal buzzer so it will dictate the exact value measured
6. This system can be connected to communication device such as modems, cellular phones or satellite terminal to enable the remote collection of recorded data or alarming of certain parameters
7. A self desister control provision can be implemented by taking considering the accident probability and chances making the system efficient to desister management
8. Owing towards the current industrial revolution movement industry 4.0 electroplating plant can be be given a remote access where electroplating plant can be made independent and can access from the anywhere. It will be a step towards the automation.



Referances

Links

1. https://www.explainthatstuff.com/electroplating.html#:~:text=Electroplating%20involves%20passing%20an%20electric,battery%20or%20other%20power%20supply.
2. https://sensorex.com/blog/2017/06/13/improving-electroplating-process-using-ph-control/
3. http://spectrum-india.com/
4. https://m.economictimes.com/definition/systems-design/amp
5. https://analyticsindiamag.com/about/
6. https://www.botshop.co.za/how-to-use-a-ph-probe-and-sensor/
7. https://create.arduino.cc/projecthub/electropeak/how-to-calibrate-use-mq9-gas-sensor-w-arduino-e93cb1
8. https://forum.arduino.cc/t/pull-up-resister-with-ds18b20-why/84593/11
9. https://microcontrollerslab.com/433mhz-rf-receiver-module-pinout-applications-arduino-examples/
10. https://components101.com/asset/sites/default/files/component_datasheet/433%20MHz%20RF%20Transmitter%20Module_0.pdf
11. http://www.varesano.net/blog/fabio/what%20arduino%20why%20we%20choose%20it%20what%20can%20we%20do%20it#:~:text=Arduino%20is%20a%20great%20tool,to%20a%20computer%20using%20USB.
12. https://www.elprocus.com/interface-lcd-liquid-crystal-display-using-arduino/
13. https://github.com/sparkfun/SparkFun_RadioHead_Arduino_Library#:~:text=It%20provides%20a%20complete%20object,a%20range%20of%20embedded%20microprocessors.&text=RadioHead%20consists%20of%202%20main%20sets%20of%20classes%3A%20Drivers%20and%20Managers
14. https://www.pjrc.com/teensy/td_libs_OneWire.html#:~:text=OneWire%20Library,be%20used%20with%20this%20library.&text=OneWire%20communicates%20with%201%2Dwire%20devices
15. https://www.labcenter.com/downloads/
16. https://www.arduino.cc/

Papers

a. How to Write a Problem Statement by A C Sheffield
b. Internet of Things (IOT): Research Challenges and Future Applications by Dr AbdelRahman H. Hussein

Bibliography

Title of Book Author Publication

1. Electrical and Electronic Measurements and instrumentation Sawhyen, A.K. Dhanpat Rai andSons, New Delhi, 2005, ISBN: 13-9788177000160

2. The AVR Microcontroller and embedded Systems using Assembly and C Muhammad Ali Mazidi MicroDigitalEd.com ISBN- 13:078-0997925968

3. ARM Assembly Language Programming and Architecture Muhammad Ali Mazidi, Sarmad Naimi MicroDigitalEd.com ISBN- 13:978-0997925906

Appendix

Offset setting sketch for PH sensor

```
void setup() {
 // initialize serial communication at 9600 bits per second:
 Serial.begin(9600);
}

// the loop routine runs over and over showing the voltage on A0
void loop() {
 // read the input on analog pin 0:
 int sensorValue = analogRead(A0);
 // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V):
 float voltage = sensorValue * (5.0 / 1023.0);
 // print out the value you read:
 Serial.println(voltage);
 delay(300);
```