Chapter no 3-: System Hardware Design

In the last chapter we seen 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

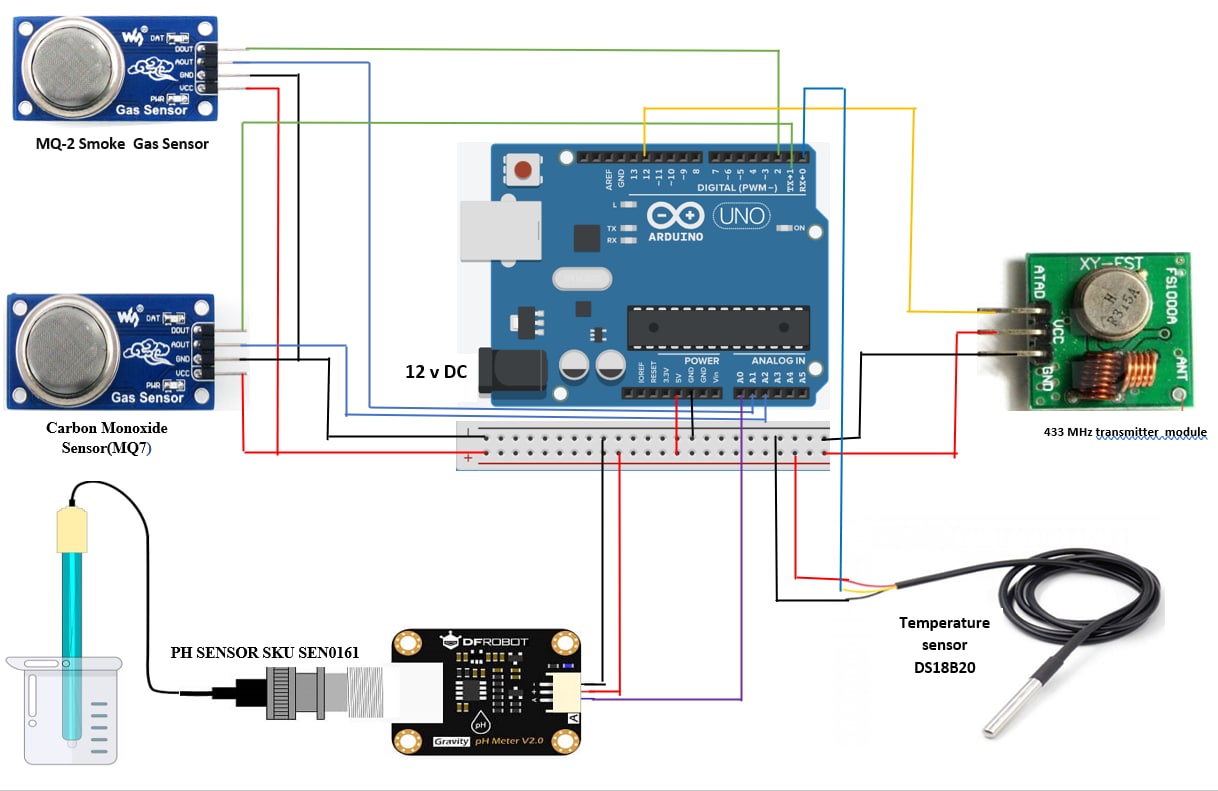
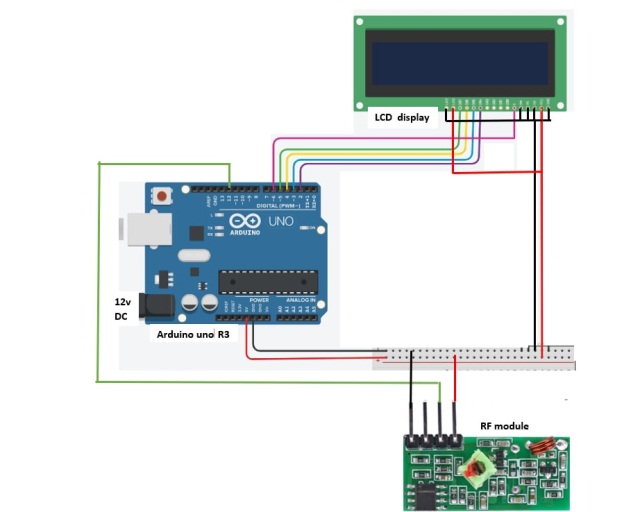
**Circuit diagram**

Figure Sensing and Transmitting terminal

Figure Reciving and Display terminal

**Design of the system elements**

**Microcontroller**

In this project we have use the arduino uno R3 as the controller. Description of the controller has been given in the hardware description chapter. But here we will see the criteria for selection of controller

**Interfacing the Transducer to controller -:** It consist all the sensors used in the design

* **PH sensor module (**SKU.SEN0161**)**

This sensor is directly compatible with arduino and can be interfaced without any extra circuitry added but while interfacing it it 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, we want a 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,You 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). I took 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

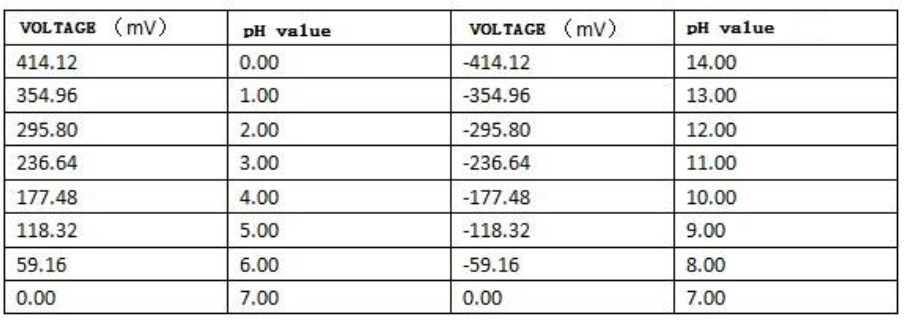
There are two ways you can do the adjustment.

* 1. If you have a multimeter handy you can measure the value of the PO pin and adjust the offset potentiometer until  PO measures 2.5V.
  2. second way is to write the sketch given in the appendix A then., open serial monitor and view the reading there. All this sketch does is to print the volts it receivesfrom 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



* **Gas sensors (**MQ2 and MQ7 **)-:**

both the sensors has interfacing procedure is same In order to get correct and accurate data, you 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.

You 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). In clean air, after preheating, upload the following code ( appendix A) and wait for about 15 minutes until R0 reaches a fixed value.

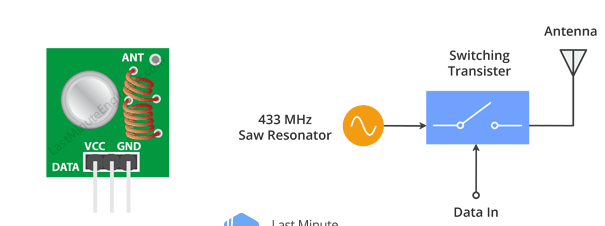
* **Temperature sensor (DS18B20)**

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.

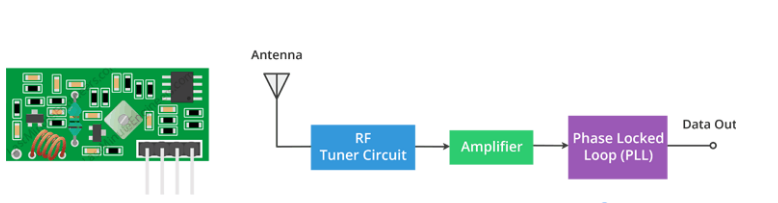
It has a 3 pins out of which two are VCC and GND which is connected to arduino’s +5V and GND , and DQ (data pin) is connected to the digital pin 0 of arduino .

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

* **433MHz 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](https://en.wikipedia.org/wiki/Phase-locked_loop)) which enables the decoder to “lock” onto a stream of digital bits which gives better decoded output and noise immunity.

**Connecting 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.

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.

* **Liquid Crystal Display**

The LCD has a lot of pins (16 pins in total) that we’ll show you how to wire up. But, the good news is that not all these pins are necessary for us to connect to the Arduino.

We know that there are 8 Data lines that carry raw data to the display. But, HD44780 LCDs are designed in a way that we can 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, we will be interfacing LCD using 4-bit mode and hence we need only 6 pins: RS, EN, D7, D6, D5, and D4 to talk to the LCD.

Now, let’s 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.

**Note** 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 you write the data in just one go. However, in 4-bit mode you 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.

* **Arduino Uno R3 (microcontroller development board).**

In the system arduino uno r3 Board which contain ATmega328p micro-controllerintegrated on I 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.

* t 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 the the 12 V DC supply or 9 V DC battery which saves the power supply cost in the project. However we are going to 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 we are going to communicate 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.

.