PROJECT REPORT

REAL TIME TEMPERATURE AND HUMIDITY MONITORING SYSTEM

ECE692: EMBEDDED COMPUTING SYSTEMS

SPRING 2017

Under the Guidance of
Dr. NAGI NAGANATHAN

SUBMITTED BY

Sanket Shankar Kulkarni
sk2339@njit.edu
314-06-022

Raghu Vamshi Vadlakunta
rv323@njit.edu
314-13-581

Sai Sandeep Yanamandra
sy349@njit.edu
314-13-583

Acknowledgement

We wish to express our deep sense of gratitude to our Professor Dr. NAGI NAGANATHAN, New Jersey Institute of Technology for duly steering the course and for his guidance and encouragement throughout the project.

We would like to extend our sincere thanks to our Professor for clarifying each doubt of ours regarding the subject no matter how small the issues were.

Sanket Shankar Kulkarni
sk2339@njit.edu
314-06-022

Raghu Vamshi Vadlakunta
rv323@njit.edu
314-13-581

Sai Sandeep Yanamandra
sy349@njit.edu
314-13-583

Table of Contents

1.INTRODUCTION

A monitoring system generally refers to an automated system that simultaneously and continuously records one or more physical parameters such as temperature, relative humidity, wind flow, light intensity, soil moisture etc. at one or more predefined places. Continuous monitoring of any sensitive environment helps to meet security and regulatory compliance needs. Monitoring temperature and/or humidity conditions is an essential ingredient of a wide range of quality assurance applications. Monitoring deterioration would provide an early warning of incipient problems enabling the planning and scheduling of maintenance programs, hence minimizing relevant costs. Furthermore, the use of data from monitoring systems together with improved service-life prediction models leads to additional savings in life cycle costs. Temperature and humidity are key issues to be taken care of in manufacturing plants and particularly that of electronic assemblies. Lack of control over any of them will not only affect the component and equipment but also the process and the operators' comfort, all ultimately leading to loss in production . Temperature and relative humidity affects the airborne survival of viruses, bacteria and fungi. Thus environmental control in hospitals is important because of infectious disease transmission from the aerosol or airborne infection. Temperature and relative humidity plays an important role in the lifecycle of the plants. When plants have the right humidity they thrive, because they open their pores completely and so breathe deeply without threat of excessive water loss. Wireless sensor network (WSN) has revolutionized the field of monitoring and remote sensing. Wireless sensor network or wireless sensor & actuator network (WSAN) are spatially distributed sensors to monitor physical or environmental conditions such as temperature, humidity, fire etc. and to cooperatively pass their data through the network to the main location.

2.HARDWARE AND SOFTWARE STUDY

2.1 DESCRIPTION ABOUT ARDUINO

Arduino is an open-source prototyping 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. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The software, too, is open-source, and it is growing through the contributions of users worldwide.

2.1.1 WHY ARDUINO?

Thanks to its simple and accessible user experience, Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux. Teachers and students use it to build low cost scientific instruments, to prove chemistry and physics principles, or to get started with programming and robotics. Designers and architects build interactive prototypes, musicians and artists use it for installations and to experiment with new musical instruments.

Makers, of course, use it to build many of the projects exhibited at the Maker Faire, for example. Arduino is a key tool to learn new things. Anyone - children, hobbyists, artists, programmers - can start tinkering just following the step by step instructions of a kit, or sharing ideas online with other members of the Arduino community.

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Net media's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

- Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules are available in the market.
- Cross-platform - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.
- Simple, clear programming environment - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.
- Open source and extensible software - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.

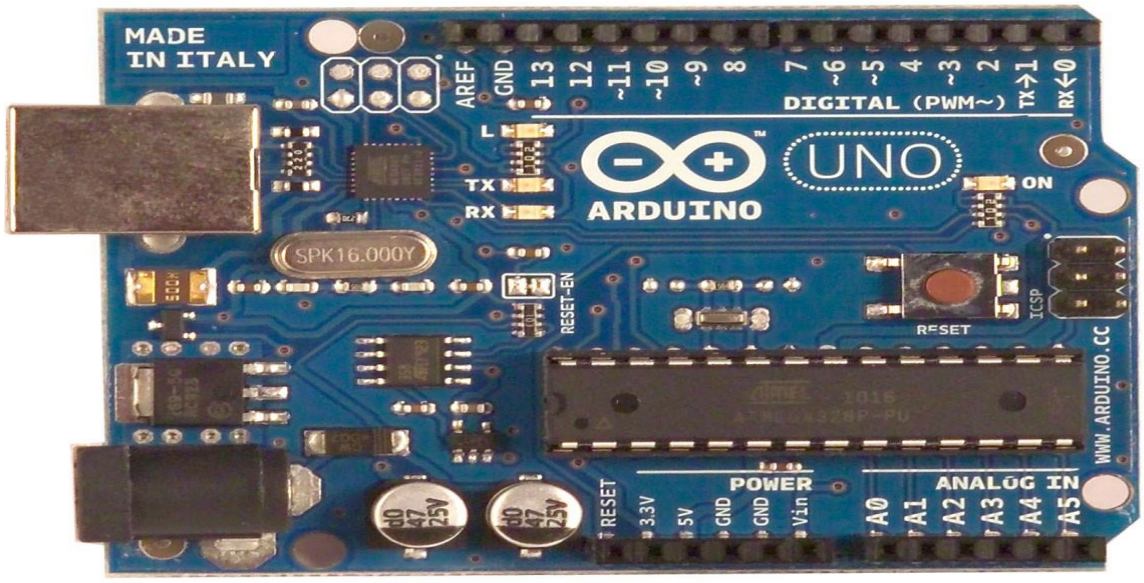
- Open source and extensible hardware - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

2.2 ARDUINO UNO

The Arduino Uno 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 crystal oscillator, 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 Atmega8U2 programmed as a USB-to-serial converter.

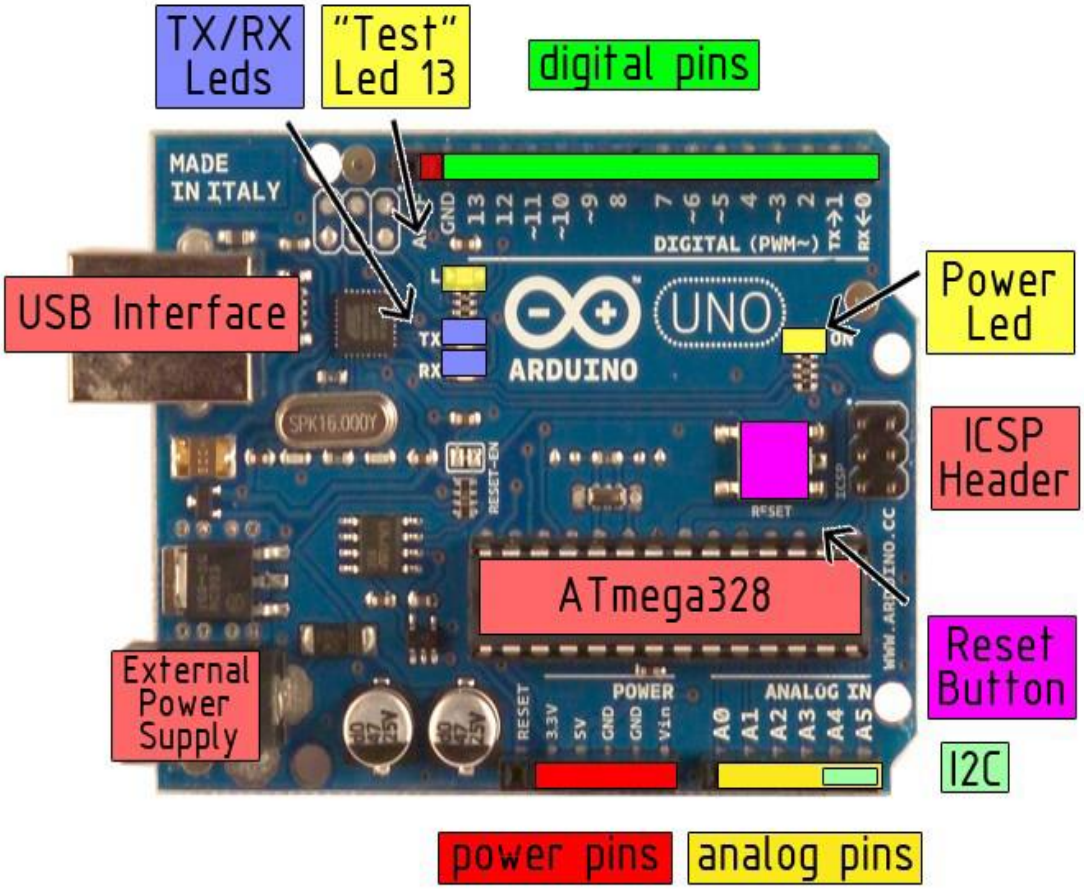
"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduno, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform

Microcontroller ATmega328



2.2.1 Technical Specifications:

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 of which 0.5 KB used by bootloader
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz



The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector. The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

VIN. The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

· **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.

· **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

· **GND.** Ground pins.

2.2.2 Memory:

The Atmega328 has 32 KB of flash memory for storing code (of which 0,5 KB is used for the bootloader); It has also 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

2.2.3 Input and Output:

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

· **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. TThese pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip .

· **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.

· **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the analogWrite() function.

· **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.

· **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off. The Uno has 6 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:

· **I2C: 4 (SDA) and 5 (SCL).** Support I2C (TWI) communication using the Wire library.

There are a couple of other pins on the board:

· **AREF.** Reference voltage for the analog inputs. Used with analogReference().

· **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

2.2.4 Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega8U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '8U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, an *.inf file is required.. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-toserial chip and USB connection to the computer (but not for serial communication on pins 0 and 1). A SoftwareSerial library allows for serial communication on any of the Uno's digital pins. The ATmega328 also support I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. To use the SPI communication, please see the ATmega328 datasheet.

2.2.5 Programming

The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno w/ ATmega328" from the **Tools > Board** menu (according to the microcontroller on your board). For details, see the reference and tutorials. The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files). You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details. The ATmega8U2 firmware source code is available . The ATmega8U2 is loaded with a DFU bootloader, which can be activated by connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2. You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader).

2.2.6 Automatic 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 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload. This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data. The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN".

2.3 V3 NodeMCU based on ESP-12E from ESP8266

The NodeMCU is an open-source firmware and development kit that helps you to prototype your IoT product within a few Lua script lines.

2.3.1 Feature

- Open-source
- Interactive
- Programmable
- Low cost
- Simple
- Smart
- WI-FI enabled

2.3.2 Arduino-like hardware IO

- Advanced API for hardware IO, which can dramatically reduce the redundant work for configuring and manipulating hardware.
- Code like ard--uino, but interactively in Lua script.

2.3.3 Nodejs style network API

- Event-driven API for network applicaitons, which faciliates developers writing code running on a 5mm*5mm sized MCU in Nodejs style.
- Greatly speed up your IOT application developing process.

2.3.4 Lowest cost WI-FI

- Less than $2 WI-FI MCU ESP8266 integrated and esay to prototyping development kit.
- We provide the best platform for IOT application development at the lowest cost.

2.3.5 Specification

- The Development Kit based on ESP8266, integates GPIO, PWM, IIC, 1-Wire and ADC all in one board.
- Power your developement in the fastest way combinating with NodeMCU Firmware!
- USB-TTL included, plug&play
- 10 GPIO, every GPIO can be PWM, I2C, 1-wire
- FCC CERTIFIED WI-FI module(Coming soon)
- PCB antenna

2.4 GSM MODULE and GSM AT COMMAND SET

GSM (Global System for Mobile Communications, originally Groupe Special Mobile), is a standard developed by the European Telecommunications Standards Institute (ETSI) to describe the protocols for second-generation (2G) digital cellular networks used by mobile phones, first deployed in Finland in July 1991.[2] As of 2014 it has become the default global standard for mobile communications - with over 90% market share, operating in over 219 countries and territories.

2G networks developed as a replacement for first generation (1G) analogy cellular networks, and the GSM standard originally described a digital, circuit-switched network optimized for full duplex voice telephony. This expanded over time to include data communications, first by circuit-switched transport, then by packet data transport via GPRS (General Packet Radio Services) and EDGE (Enhanced Data rates for GSM Evolution or EGPRS).

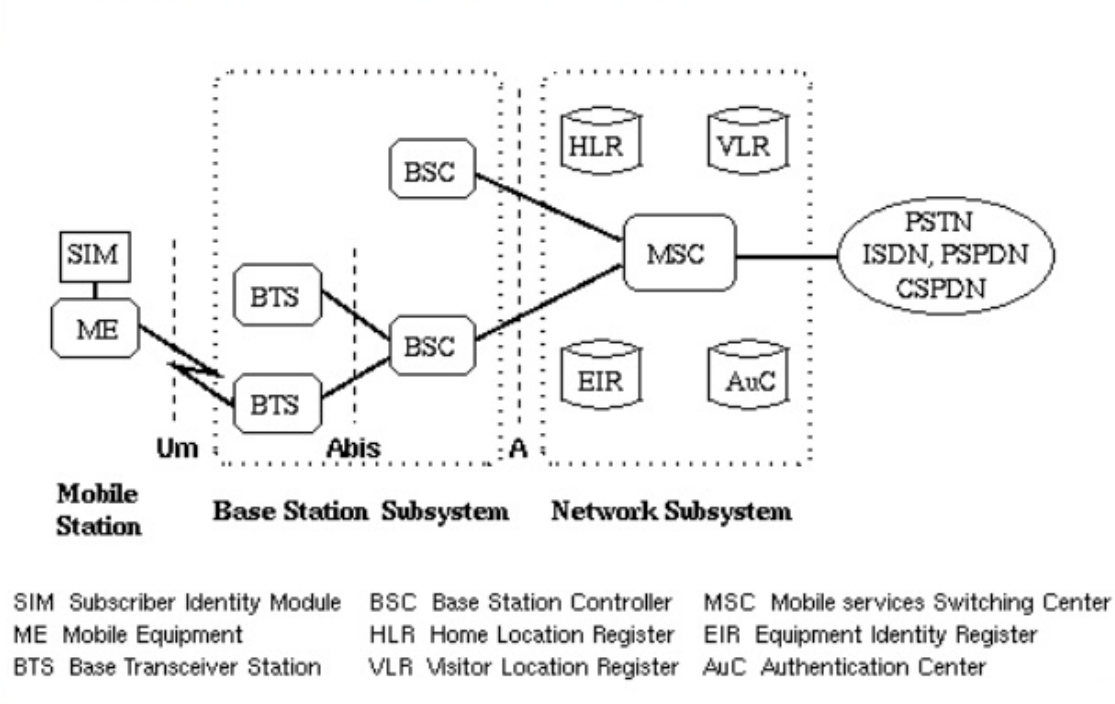
Subsequently, the 3GPP developed third-generation (3G) UMTS standards followed by fourth-generation (4G) LTE Advanced standards, which do not form part of the ETSI GSM standard.

"GSM" is a trademark owned by the GSM Association. It may also refer to the (initially) most common voice codec used, Full Rate.

A GSM modem is a wireless modem that works with a GSM wireless network. A wireless modem is like a dial-up modem. The basic difference between them is the dial-up

modem sends and receives data through a fixed telephone line while the wireless modem sends and receives data through waves. Like a GSM mobile phone, a GSM modem also requires a SIM card from a wireless carrier to operate.

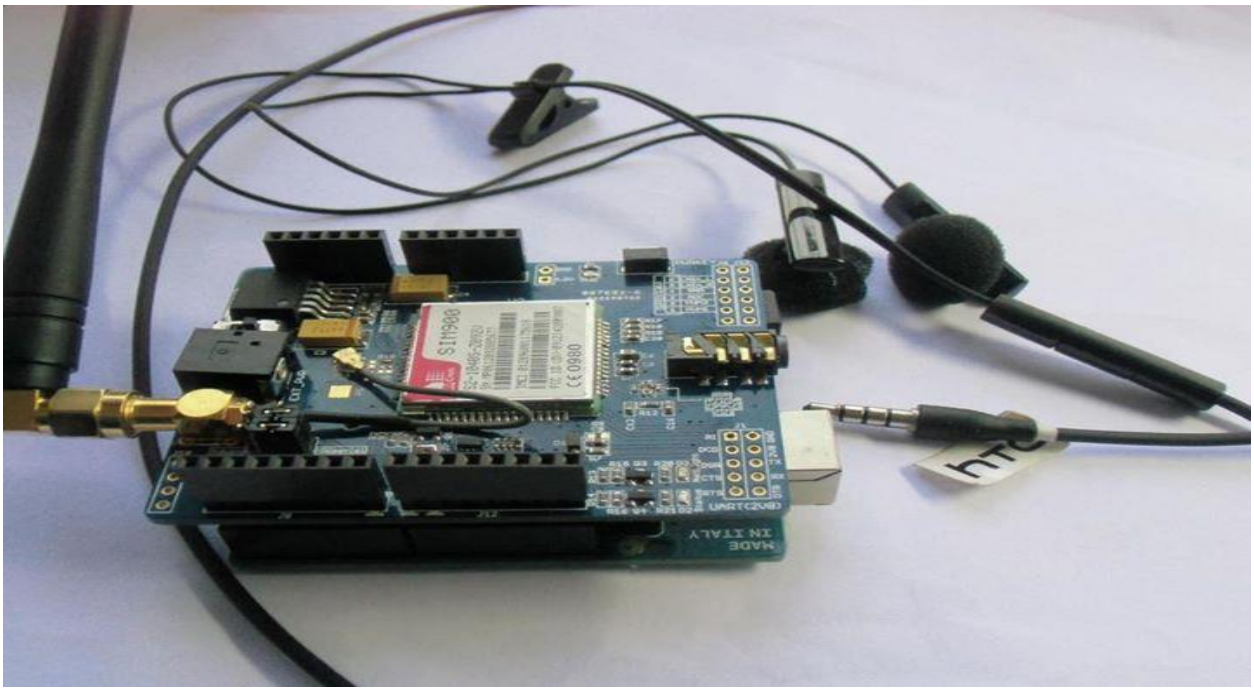
AT commands are used to control MODEMs. AT is the abbreviation for Attention. These commands come from Hayes commands that were used by the Hayes smart modems. The Hayes commands started with AT to indicate the attention from the MODEM. The dial up and wireless MODEMs (devices that involve machine to machine communication) need AT commands to interact with a computer. These include the Hayes command set as a subset, along with other extended AT commands.



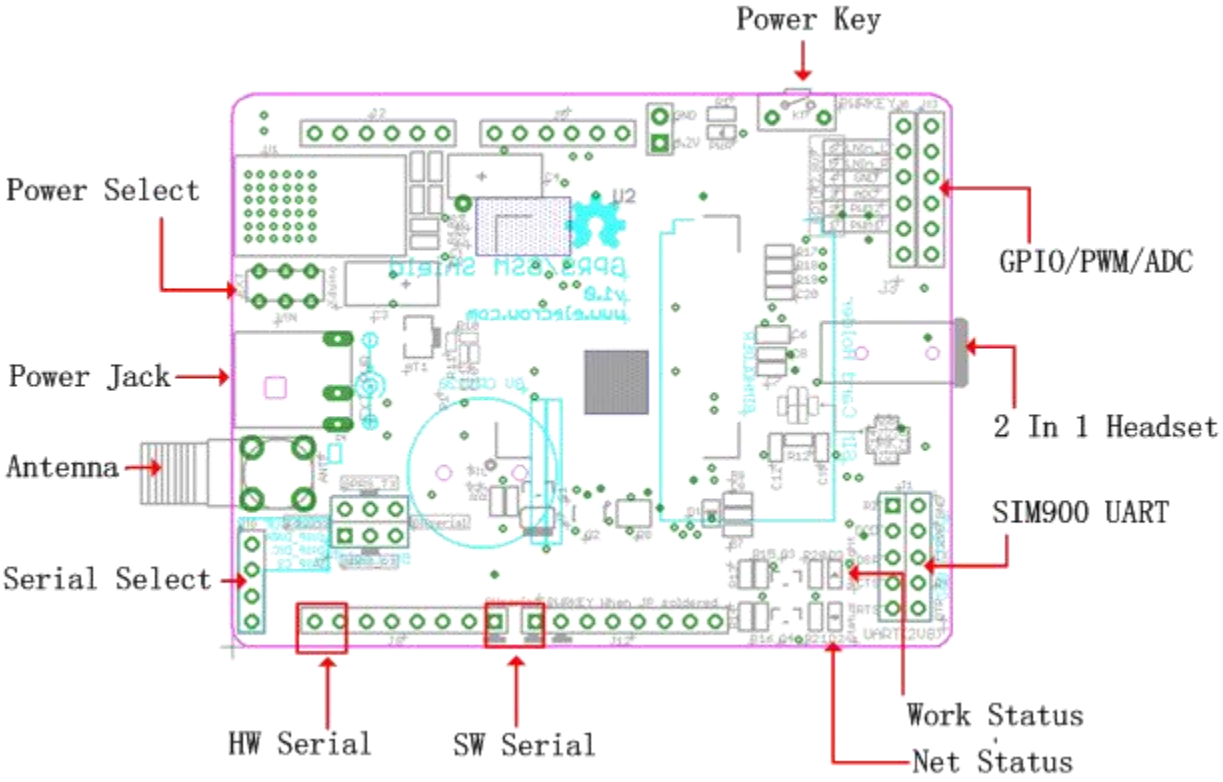
GSM Architecture (Courtesy: Google)

AT commands with a GSM/GPRS MODEM or mobile phone can be used to access following information and services:

1. Information and configuration pertaining to mobile device or MODEM and SIM card.
2. SMS services.
3. MMS services.
4. Fax services.



GSM Module SIM900 (Courtesy: Google)



GPRS Shield Interface

The Hayes subset commands are called the basic commands and the commands specific to a GSM network are called extended AT commands.

2.4.1 TYPES OF AT COMMANDS:

Fig. 4.4.3 Classification of AT Commands (Courtesy: Google)

There are four types of AT commands:

- Test commands - used to check whether a command is supported or not by the MODEM.

  SYNTAX: AT<command name>=?

  For example: ATD=?

- Read command - used to get mobile phone or MODEM settings for an operation.

  SYNTAX: AT<command name>?

  For example: AT+CBC?

- Set commands - used to modify mobile phone or MODEM settings for an operation.

  SYNTAX: AT<command name>=value1, value2, …, valueN

  Some values in set commands can be optional.

  For example: AT+CSCA=”+9876543210”, 120

- Execution commands - used to carry out an operation.

SYNTAX: AT<command name>=parameter1, parameter2, …, parameterN

The read commands are not available to get value of last parameter assigned in execution commands because parameters of execution commands are not stored.

For example: AT+CMSS=1,"+ 9876543210", 120

2.4.2 COMMONLY USED AT COMMANDS:

1) AT - This command is used to check communication between the module and the computer. For example,

AT

OK

The command returns a result code OK if the computer (serial port) and module are connected properly. If any of module or SIM is not working, it would return a result code ERROR.

2) +CMGF - This command is used to set the SMS mode. Either text or PDU mode can be selected by assigning 1 or 0 in the command.

SYNTAX: AT+CMGF=<mode>

0: for PDU mode

1: for text mode

The text mode of SMS is easier to operate but it allows limited features of SMS. The PDU (protocol data unit) allows more access to SMS services but the operator requires bit level knowledge of TPDUs. The headers and body of SMS are accessed in hex format in PDU mode so it allows availing more features.

For example,

AT+CMGF=1

OK

3) +CMGW - This command is used to store message in the SIM.

SYNTAX: AT+CMGW=" Phone number"> Message to be stored Ctrl+z As one types AT+CMGW and phone number, '>' sign appears on next line where one can type the message. Multiple line messages can be typed in this case. This is why the message is terminated by providing a 'Ctrl+z' combination. As Ctrl+z is pressed, the following information response is displayed on the screen. +CMGW: Number on which message has been stored

4) +CMGS - This command is used to send a SMS message to a phone number.

SYNTAX: AT+CMGS= serial number of message to be send.

As the command AT+CMGS and serial number of message are entered, SMS is sent to the particular SIM.

For example,

AT+CMGS=1

OK

5) ATD - This command is used to dial or call a number.

SYNTAX: ATD<Phone number>;(Enter)

For example,

ATD123456789;

7) ATH - This command is used to disconnect remote user link with the GSM module.

SYNTAX: ATH (Enter)

2.4.3 LIST OF AT COMMANDS:

The AT commands for both, GSM module and the mobile phone, are listed below. Some of these commands may not be supported by all the GSM modules available. Also there might be some commands which won't be supported by some mobile handsets.

Table 2.4.3.1: Testing

Command	Description
AT	Checking communication between the module and computer.

Table 2.4.3.2: Call Control

Command	Description
ATA	Answer command
ATD	Dial command
ATH	Hang up call
ATL	Monitor speaker loudness
ATM	Monitor speaker mode
ATO	Go on-line
ATP	Set pulse dial as default
ATT	Set tone dial as default
AT+CSTA	Select type of address
AT+CRC	Cellular result codes

Table 2.4.3.3: Data Card Control

Command	Description
ATI	Identification
ATS	Select an S-register
ATZ	Recall stored profile
AT&F	Restore factory settings
AT&V	View active configuration
AT&W	Store parameters in given profile
AT&Y	Select Set as power up option
AT+CLCK	Facility lock command
AT+COLP	Connected line identification presentation
AT+GCAP	Request complete capabilities list
AT+GMI	Request manufacturer identification
AT+GMM	Request model identification
AT+GMR	Request revision identification
AT+GSN	Request product serial number identification (IMEI)

Table 2.4.3.4: Phone Control

Command	Description
AT+CBC	Battery charge
AT+CGMI	Request manufacturer identification
AT+CGMM	Request model identification
AT+CGMR	Request revision identification
AT+CGSN	Request product serial number identification
AT+CMEE	Report mobile equipment error
AT+CPAS	Phone activity status
AT+CPBF	Find phone book entries
AT+CPBR	Read phone book entry
AT+CPBS	Select phone book memory storage
AT+CPBW	Write phone book entry
AT+CSCS	Select TE character set
AT+CSQ	Signal quality

Table 2.4.3.5: Computer Data Interface

Command	Description
ATE	Command Echo
ATQ	Result code suppression
ATV	Define response format
ATX	Response range selection
AT&C	Define DCD usage
AT&D	Define DTR usage
AT&K	Select flow control
AT&Q	Define communications mode option
AT&S	Define DSR option
AT+ICF	DTE-DCE character framing
AT+IFC	DTE-DCE Local flow control
AT+IPR	Fixed DTE rate

Table 2.4.3.6: Service

Command	Description
AT+CLIP	Calling line identification presentation
AT+CR	Service reporting control
AT+DR	Data compression reporting
AT+ILRR	DTE-DCE local rate reporting

Table 2.4.3.7: Network Communication Parameter

Command	Description
ATB	Communications standard option
AT+CBST	Select bearer service type
AT+CEER	Extended error report
AT+CRLP	Radio link protocol
AT+DS	Data compression

Table 2.4.3.8: Miscellaneous

Command	Description
A/	Re-execute command line
AT?	Command help
AT*C	Start SMS interpreter
AT*T	Enter SMS block mode protocol
AT*V	Activate V.25bis mode
AT*NOKIATEST	Test command
AT+CESP	Enter SMS block mode protocol

Table 2.4.3.9: SMS Text Mode

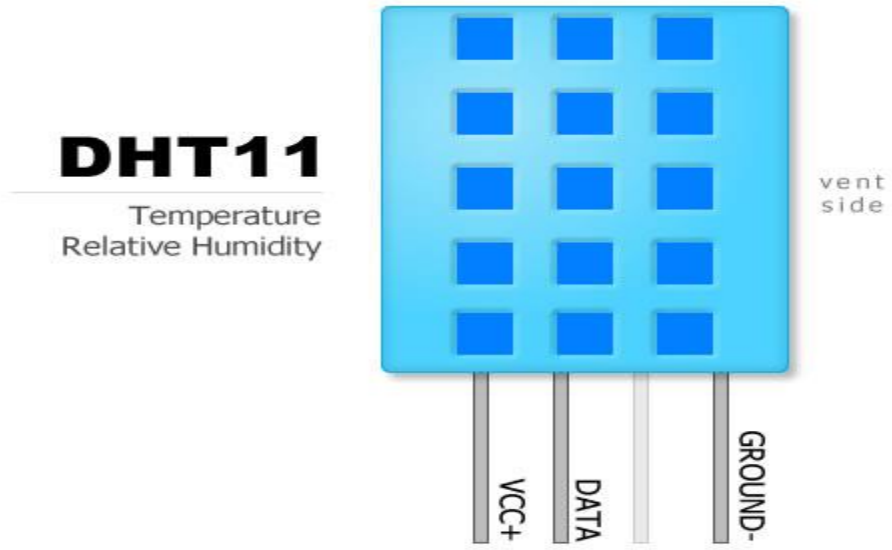
Command	Description
AT+CSMS	Select message service
AT+CPMS	Preferred message storage
AT+CMGF	Message format
AT+CSCA	Service centre address
AT+CSMP	Set text mode parameters
AT+CSDH	Show text mode parameters
AT+CSCB	Select cell broadcast message types
AT+CSAS	Save settings
AT+CRES	Restore settings
AT+CNMI	New message indications to TE
AT+CMGL	List messages
AT+CMGR	Read message
AT+CMGS	Send message
AT+CMSS	Send message from storage
AT+CMGW	Write message to memory
AT+CMGD	Delete message

Table 2.4.3.10: SMS PDU Mode

Command	Description
AT+CMGL	List Messages
AT+CMGR	Read message
AT+CMGS	Send message
AT+CMGW	Write message to memory

2.5 DHT11

. DHT 11 Sensors DHT11 digital temperature and humidity sensor is a composite Sensor contains a calibrated digital signal output of the temperature and humidity. It ensures high reliability and excellent long-term stability. This sensor is resistive-type humidity and an NTC temperature measurement component, and connects to a high performance 8-bit microcontroller, offering excellent quality, fast response, anti-interference ability and cost effectiveness.



It has a small size, low power consumption and signal transmission up-to-20. DHT11 is a 3-pin single row pin package. Its features are as follows.

It works on 5V.

Temperature range: 0 - +50 ºC.

Temperature accuracy: ± 2.0 ° C.

Humidity range: 20-95% RH.

Humidity accuracy: ± 5.0% RH

Response time: < 5 Sec

2.6 LCD

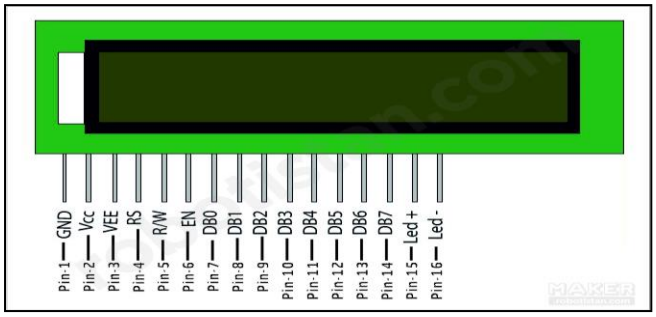


Fig 6.4: LCD Display

LCD (Liquid Crystal Display) screen is an electronic display module and find a wide range of applications. A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits. These modules are preferred over seven segments and other multi segment LEDs. The reasons being: LCDs are economical; easily programmable; have no limitation of displaying special & even custom characters (unlike in seven segments), animations and so on. A 16x2 LCD means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. This LCD has two registers, namely, Command and Data. The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD.

2.6.1 Pin Description:

Pin No	Function	Name
1	Ground (0V)	Ground
2	Supply voltage; 5V (4.7V – 5.3V)	Vcc
3	Contrast adjustment; through a variable resistor	V_{EE}
4	Selects command register when low; and data register when high	Register Select

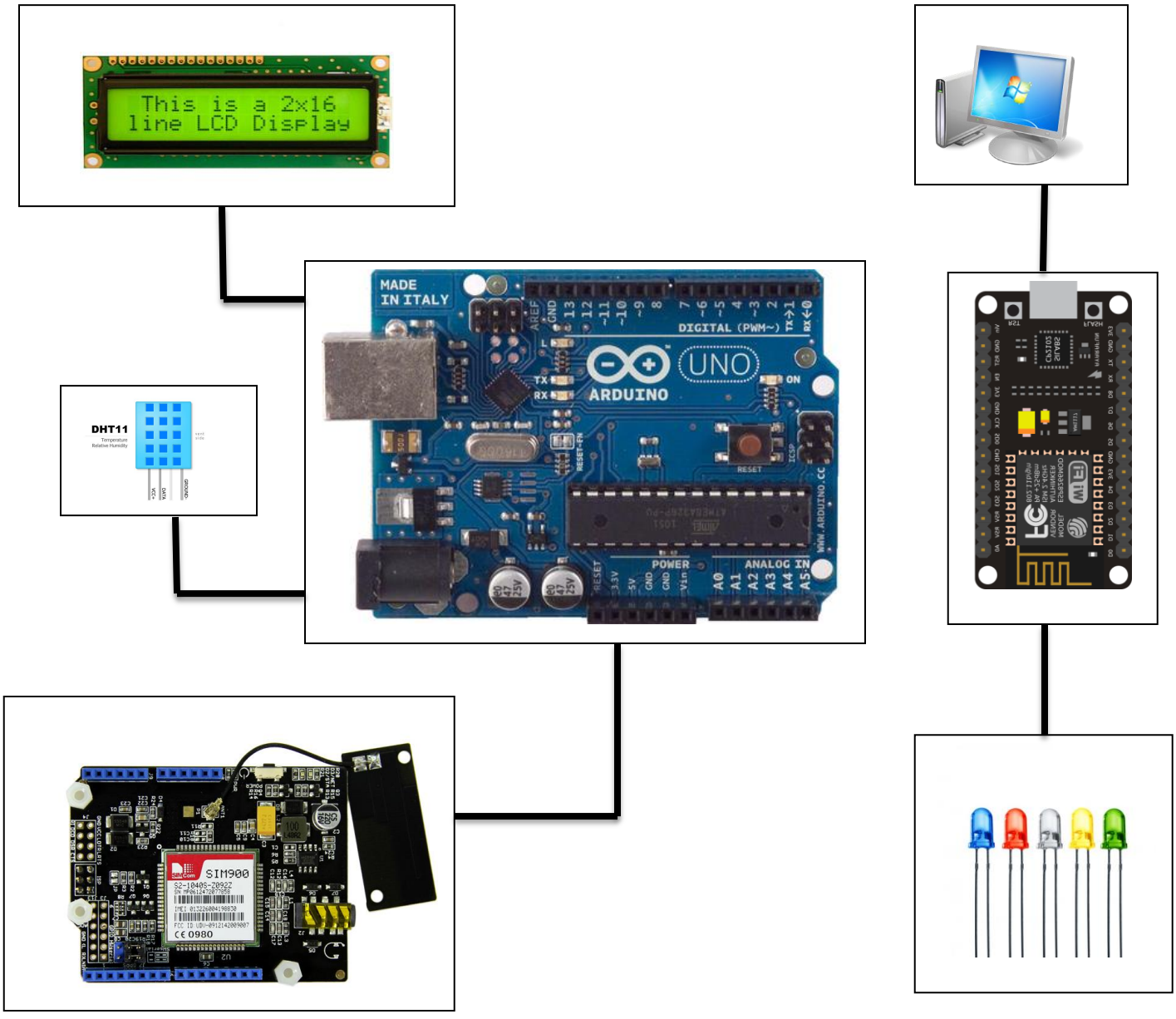
5	Low to write to the register; High to read from the register	Read/write
6	Sends data to data pins when a high to low pulse is given	Enable
7	8-bit data pins	DB0
8		DB1
9		DB2
10		DB3
11		DB4
12		DB5
13		DB6
14		DB7
15	Backlight V_{CC} (5V)	Led+
16	Backlight Ground (0V)	Led-

Table no 6.3: Pin outs of LCD

3.PROBLEM DESCRIPTION

Today monitoring environmental parameters have gained more importance due to the increasing security and regulatory compliance needs. So the measurement of such parameters becomes critically important. To do the parameter measurement of remote places, the traditional wired systems fail. Hence there is a need of next generation technology such as wireless technology. Due to the advancement in the Micro-Electromechanical systems (MEMS) technology tiny, low cost, low power, cost effective wireless modules are available and they work in such locations efficiently. Basically, in Industries, boilers may once in a while unexpectedly changes to high temperature which can influence nature and the individual who work in Industries needs to deal with it manually to cool the boilers. In compound Industries, maintainance of certain temperature for specific chemicals is compulsorily required and this must be controlled physically. This project represents the use of wireless sensor network (WSN) technology for monitoring temperature and humidity system using Arduino microcontroller, GSM/GPRS SIM900, NodeMCU V3 ESP8266, DHT11 sensor and Mobile Device. Real time temperature and humidity parameters data were displayed on LCD. Such wireless system will prove to be boon for Industries, Laboratories etc.

4.BLOCK DIAGRAM

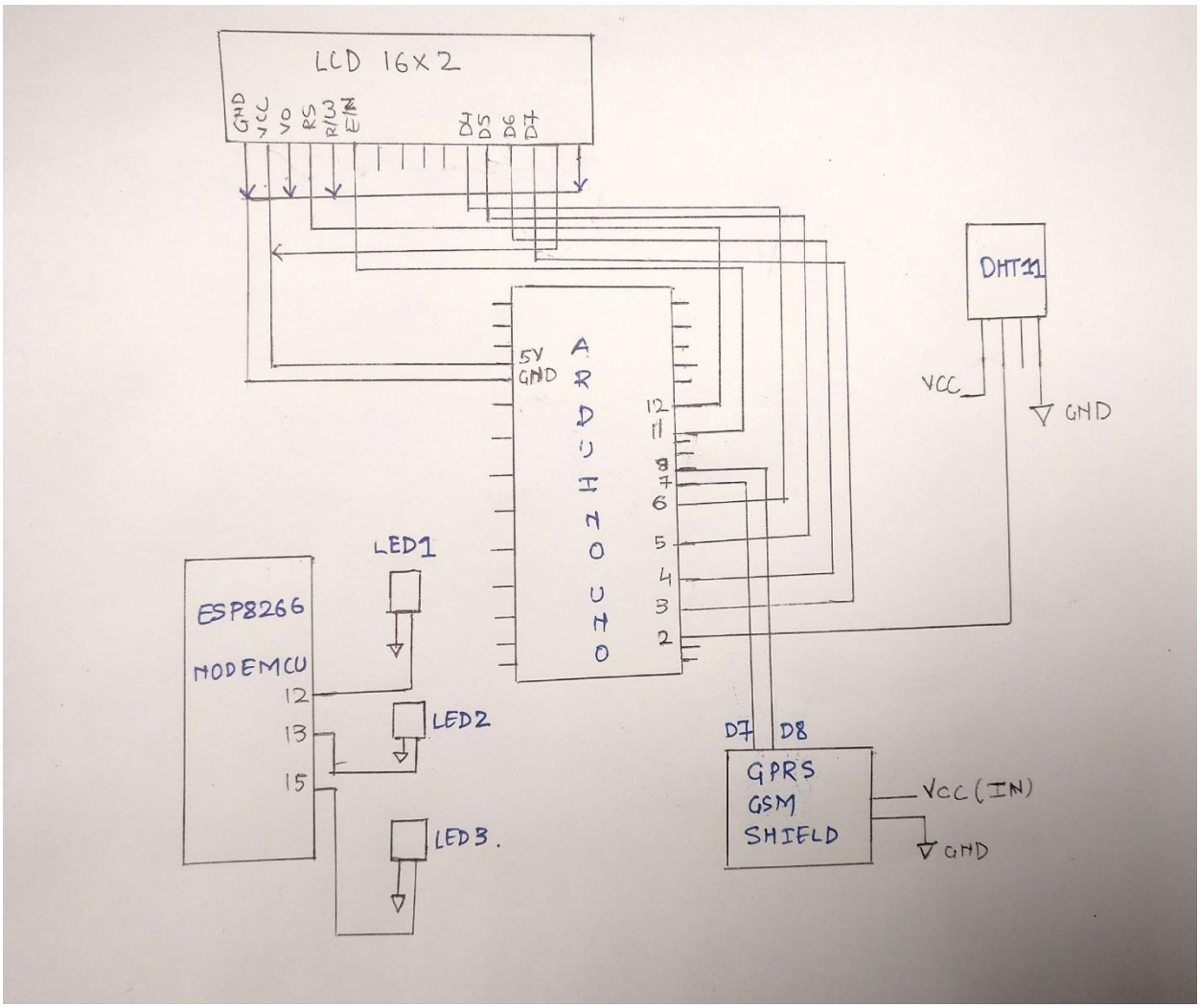


Block Diagram of the Proposed Monitoring System

5.WORKING

5.1 HARDWARE SPECIFICATION

The aim of this project is to design and develop a system which fulfils all above requirements. In this project digital humidity temperature composite (DHT11) sensor is used to sense the environmental temperature and relative Humidity. Arduino microcontroller is used to make complex computation of the parameters and then to transmit the data on LCD and sending the values wirelessly to mobile device using GSM/GPRS SIM900. By making effective use of WIFI NodeMCU ESP8266, we enter into the particular IP address and control the operation through mobile device. To achieve the function of the design and convenient machine interaction, the system is divided into 2 parts



Working Model Monitoring System

5.1.1 DATA ACQUISITION

The Control Module uses the Arduino platform due to its low cost convenience and flexibility. The information of temperature and humidity is extracted using DHT11 sensor. The DHT11 sensor senses humidity and temperature of the surrounding. This information is obtained in Arduino which in turn is displayed on the LCD for in place monitoring. Moreover this information is sent via SMS to observer even if the observer is not present at place.

5.1.2 CONTROL MODULE

This module alerts us when temperature and humidity hit certain desired range allowing we can take preventive measures as per the requirement. In this system, the data obtained via SMS is read. The Control of system which is represented by LED in our project is controlled using the control widgets provided on the Mobile Phone. Thus the action needed can be taken as soon as the data is received. This control system is application of Internet of Things which is successfully achieved by using ESP8266 Nodemcu.

5.2 SOFTWARE SPECIFICATION

ESP8266 CODE:

```
#include <ESP8266WiFi.h>
#include <ArduinoJson.h>
//#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <DHT.h>
#define DHTTYPE   DHT11
#define DHTPIN    14


const char* ssid = "We are all single";
const char* password = "kissmeiwilltellyou";

int ledPin1 = 12;// GPIO13
int ledPin2 = 13;// GPIO13
int ledPin3 = 15;// GPIO13
WiFiServer server(80);

void setup() {
  Serial.begin(115200);
  delay(10);

  pinMode(ledPin1, OUTPUT);
  digitalWrite(ledPin1, LOW);
```

```
  pinMode(ledPin2, OUTPUT);
  digitalWrite(ledPin2, LOW);

  pinMode(ledPin3, OUTPUT);
  digitalWrite(ledPin3, LOW);

  // Connect to WiFi network
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  // Start the server
  server.begin();
  Serial.println("Server started");

  // Print the IP address
  Serial.print("Use this URL to connect: ");
  Serial.print("http://");
  Serial.print(WiFi.localIP());
  Serial.println("/");

}

void loop() {
  // Check if a client has connected
  WiFiClient client = server.available();
  if (!client) {
    return;
  }

  // Wait until the client sends some data
  Serial.println("new client");
  while(!client.available()){
    delay(1);
  }
```

```
  // Read the first line of the request
  String request = client.readStringUntil('\r');
  Serial.println(request);
  client.flush();

  // Match the request
  int value = LOW;


  if (request.indexOf("/LED=ON1") != -1)  {
    digitalWrite(ledPin1, HIGH);
    value = HIGH;
    delay(4000);
    digitalWrite(ledPin1, LOW);
    value= LOW;
  }
  if (request.indexOf("/LED=ON2") != -1)  {
    digitalWrite(ledPin2, HIGH);
    value = HIGH;
    delay(4000);
    digitalWrite(ledPin2, LOW);
    value= LOW;
  }

  if (request.indexOf("/LED=ON3") != -1)  {
    digitalWrite(ledPin3, HIGH);
    value = HIGH;
    delay(4000);
    digitalWrite(ledPin3, LOW);
    value= LOW;
  }

// Set ledPin according to the request
//digitalWrite(ledPin, value);

  // Return the response
  client.println("HTTP/1.1 200 OK");
  client.println("Content-Type: text/html");
  client.println(""); //  do not forget this one
  client.println("<!DOCTYPE HTML>");
  client.println("<html>");

  client.print("Led pin is now: ");

  if(value == HIGH) {
    client.print("On");
  } else {
```

```
   client.print("Off");
  }
   if(value == LOW) {
    client.print("Off");
  } else {
    client.print("On");
  }

  client.println("<br><br>");
  client.println("<a href=\"/LED=ON1\"\"><button>Medium </button></a>");
  client.println("<a href=\"/LED=ON2\"\"><button>Low</button></a><br />");
  client.println("<a href=\"/LED=ON3\"\"><button>High</button></a><br />");
  client.println("</html>");

  delay(1);
  Serial.println("Client disonnected");
  Serial.println("");

}
```

ARDUINO CODE:

```
#include "SIM900.h"
#include <SoftwareSerial.h>
#include "sms.h"
#include "DHT.h"
#include <Wire.h>
 #include <LiquidCrystal.h>
  #include <Adafruit_LiquidCrystal.h>
  Adafruit_LiquidCrystal lcd(12, 11,6, 5, 4, 3);
SMSGSM sms;
#define DHTPIN 2

#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

int numdata;
boolean started=false;
char smsbuffer[160];
char n[20];

char sms_position;
char phone_number[20];
```

```
char sms_text[100];
int i;
float t;
float h;

void setup()
{
   Serial.begin(9600);
   lcd.begin(16, 2);
   lcd.print("Sensing Humidity");
lcd.setCursor(0, 1);
lcd.print("and temperture");
delay(2000);

  dht.begin();

   if (gsm.begin(9600))
   {
      Serial.println("\nstatus=READY");
      started=true;
   }
   else
      Serial.println("\nstatus=IDLE");

 Serial.println("DHT11 Output!");
  };

void loop()
{
  h = dht.readHumidity();

  t = dht.readTemperature();

 Serial.print("Humidity: ");

   Serial.print(h);
    lcd.begin(16, 2);
lcd.setCursor(0,0); // Sets the cursor to col 0 and row 0
lcd.print("Humidity: "); // Prints Sensor Val: to LCD
lcd.print(h);
lcd.print(" %\t");
```

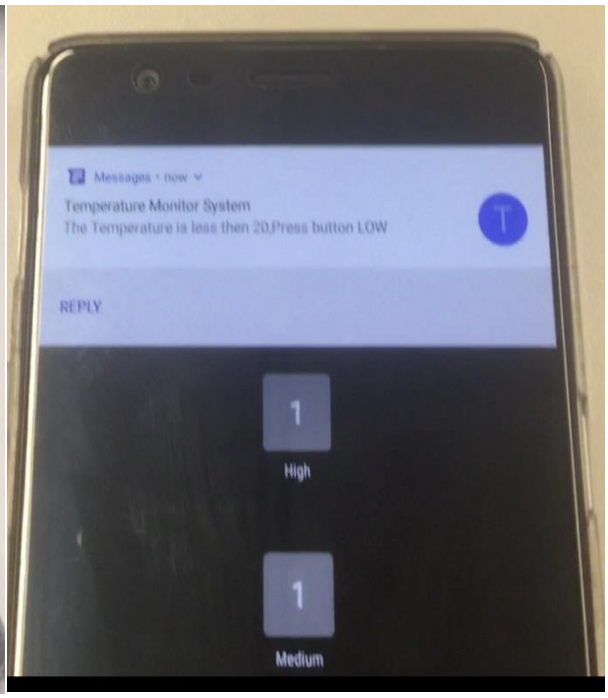
```
delay(5000);

Serial.print("Temperature: ");

Serial.print(t);
lcd.setCursor(0,0); // Sets the cursor to col 0 and row 0
lcd.print("Temperature: "); // Prints Sensor Val: to LCD
lcd.setCursor(0,1);
lcd.print(t);
lcd.print(" C\t");
  delay(5000);

  if(isnan(t) || isnan(h)){

    Serial.println("Failed to read DHT11");

  }else{
     if(t <= 20)
    {
       if (sms.SendSMS("+18628001796", "The Temperature is between 15 and 20,Press button
LOW "))
       {
         Serial.println("\nSMS sent OK.");
       }
       else
       {
         Serial.println("\nError sending SMS.");
       }
    }
     else  if(t>=21 && t <= 24)
    {
       if (sms.SendSMS("+18628001796", "The Temperature is between 20 and 25,Press button
Medium"))
       {
         Serial.println("\nSMS sent OK.");
       }
       else
       {
         Serial.println("\nError sending SMS.");
       }
    }
```

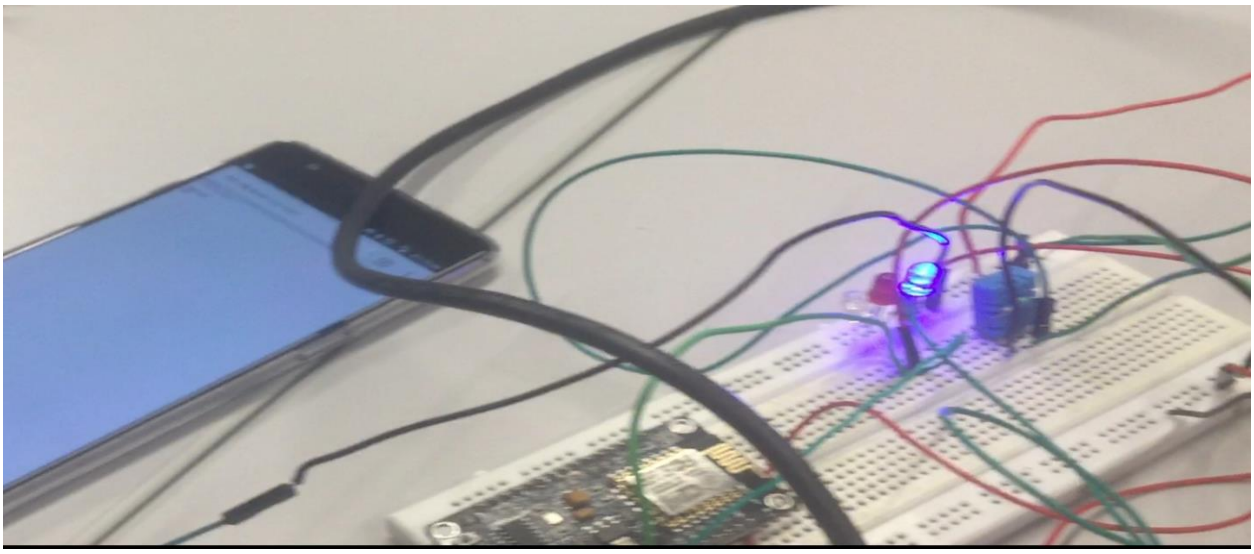
```
    else
   {
      if (sms.SendSMS("+18628001796", "The Temperature is above 25,Press button HIGH "))
      {
        Serial.println("\nSMS sent OK.");
      }
      else
      {
        Serial.println("\nError sending SMS.");
      }
   }
}
};
```

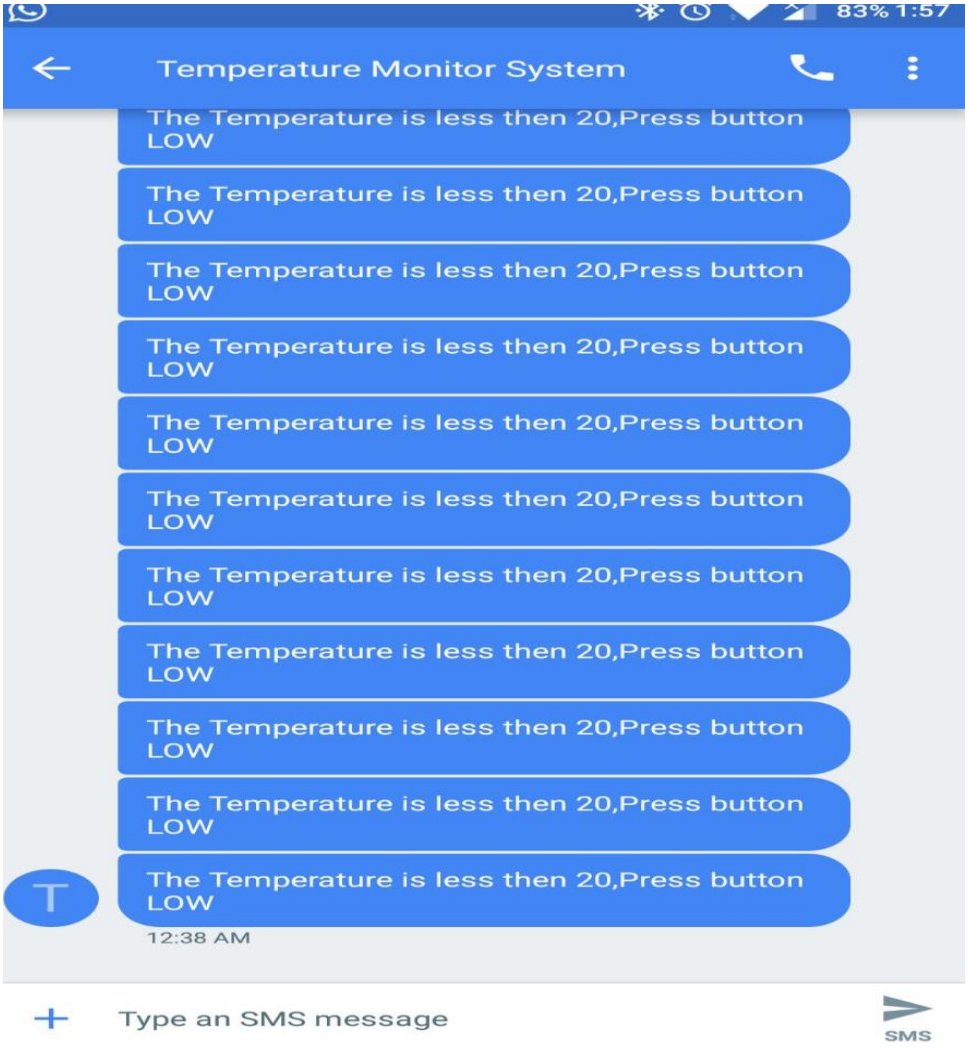
6.RESULTS


Humidity: 29.00


Temperature: 00
16.00 CE


Messages · now
Temperature Monitor System
The Temperature is less then 20,Press button LOW
REPLY
1
High
1
Medium




83% 1:57
Temperature Monitor System
The Temperature is less then 20,Press button LOW
The Temperature is less then 20,Press button LOW
The Temperature is less then 20,Press button LOW
The Temperature is less then 20,Press button LOW
The Temperature is less then 20,Press button LOW
The Temperature is less then 20,Press button LOW
The Temperature is less then 20,Press button LOW
The Temperature is less then 20,Press button LOW
The Temperature is less then 20,Press button LOW
The Temperature is less then 20,Press button LOW
The Temperature is less then 20,Press button LOW
T
12:38 AM
Type an SMS message
SMS

7.APPLICATIONS

1. Agriculture field monitoring.
2. Home automation.
3. Industrial purpose
4. Temperature and humidity are key issues to be taken care of in manufacturing plants and particularly that of electronic assemblies.
5. Environmental control in hospitals is important because of infectious disease transmission from the aerosol or airborne infection
6. Temperature and relative humidity plays an important role in the lifecycle of the plants. It can be used effectively in healthcare, agriculture, storage areas etc.
7. OIL EXPLORATION

   Today's oil drills must drill far down into the earth in their search for oil. As they drill down deep into the earth, through rocks and dirt, the temperature of the drill increases. Oil workers worry that the oil drill's bit will become too hot and break. To prevent that from happening, these oil drill bits often have a temperature sensor built inside of them. When the temperature reaches a dangerous level, that is, a level that could break the drill bit, the sensor sends an electronic signal to the oil workers to stop drilling.
8. RADIATOR OVERHEATING

   Your car contains a radiator. In it is a temperature sensor. The reason it is there is to warn you if the water that circulates in your engine becomes too hot. And that's because if it does, your engine could break and will require that you purchase a new one. The temperature sensor in your radiator measures the temperature of the radiator to the temperature gauge in your car. As the temperature of the water increases, the temperature sensor creates a larger electrical current to flow. That current flow causes the needle of your temperature gauge to move further to the right.
9. BATTERY CHARGERS

   Battery chargers are used to recharge all sorts of batteries, such as car batteries, flashlight batteries and even batteries in your computer. However, battery chargers must be designed so that they don't overcharge your battery and also so they don't undercharge your battery. Because the amount of charge a battery can store varies with temperature, the battery charger must know the battery's temperature to determine when to stop charging and when to begin charging. In these applications, the temperature sensor is used to turn on or turn off the battery charger.

8.CONCLUSION

Using wireless sensor node, the data is collected and transmitted using GSM/GPRS SIM900 module to mobile device. The Control Module uses the Arduino platform due to its low cost convenience and flexibility. The information of temperature and humidity is extracted using DHT11 sensor. The DHT11 sensor senses humidity and temperature of the surrounding. This information is obtained in Arduino which in turn is displayed on the LCD for in place monitoring. Moreover this information is sent via SMS to observer even if the observer is not present at place. This module alerts us when temperature and humidity hit certain desired range allowing we can take preventive measures as per the requirement. In this system, the data obtained via SMS is read. The Control of system which is represented by LED in our project is controlled using the control widgets provided on the Mobile Phone. Thus the action needed can be taken as soon as the data is received. This control system is application of Internet of Things which is successfully achieved by using ESP8266 Nodemcu. So we conclude that this type of monitoring system is useful for the industries where boilers may once in a while unexpectedly changes to high temperature which can influence nature and the individual who work in Industries can use this wireless monitoring system to cool the boilers and also this proposed system is useful for compound Industries for maintenance of certain temperature for specific chemicals which is compulsorily required and this can be controlled using this wireless monitoring system. The main advantage of this system is that it is less expensive and one time investment. It can be used effectively in Industries, chemical laboratories, controlling home applications etc.

9.REFERENCES

[1] TEMPERATURE AND HUMIDITY MONITORING SYSTEMS FOR FIXED STORAGE AREAS WHO TECHNICAL REPORT SERIES, NO. 961, 2011 AUGUST 2014.

[2] BUENFELD N, DAVIS R,KARMINI A,GILBERTSON A. INTELLIGENT MONITORING OF CONCRETE STRUCTURES. 666TH ED. UK: CIRIA; 2008. P. 150

[3] http://sciencing.com/uses-temperature-sensor-5997997.html

[4] https://www.arduino.cc/en/Tutorial/HelloWorld

[5] http://www.circuitstoday.com/interface-gsm-module-with-arduino

[6] https://create.arduino.cc/projecthub/jaiprak/control-led-from-web-app-using-esp8266-serial-wifi-module-cdf419

[7] http://www.instructables.com/id/Arduino-DHT11-Sensor/