CHAPTER 1

INTRODUCTION

1.1. Introduction

The proliferation of Internet of Things (IoT) devices such as smartphones, sensors, cameras. It is possible to collect massive amount of data for localization and tracking of Aircraft Collisions. An aircraft collision system is designed is to track the signal. Aircraft intelligent system better suits in all areas for tracking signal which is close to the aircraft device. It is necessary to have an intelligent system which identifies collision before it happens and informs the driver to alert or regulate the speed of the aircraft and to move if it is very close to collision.

This project is an advanced wireless communication by which it detects if any aircraft is coming nearer to another aircraft by giving an LED indication as well as buzzer sound. In this project we are using Microcontroller as its Controller.

In this project Ultrasonic Distance sensor is placed to track various frequencies emitted by radio stations. The system is provided with LED and buzzer. If user moves the aircraft, antenna tracks for a corresponding signal the rotation of antenna is based on the signal tracked by Ultrasonic Sound waves. Once the signal is matched LED blinks with a buzzer sound and the data is displayed on the LCD.

In this way we can track the signal and can avoid the collision with the desired frequency signal and these information passing to the both the aircrafts using Zigbee communication also pass the live updates to the ground station using Wi-Fi Internet communication. This project uses regulated 5V, 750mA power supply. 7805 three terminal voltage regulator is used for voltage regulation. Bridge type full wave rectifier is used to rectify the ac output of secondary of 230/18V step down transformer.

1.2. Existing Technology

In existing system, there is no specific proper system communicate in air between two aircrafts, so that some of the aircraft accidents occurs due to route missing and distance measures when fails radar signals.

1.3. Proposed System

To avoid aircraft collision while mobile networks, we used Zigbee network for communication between two aircrafts. Zigbee communication not fails due to telephone network, its secure network. By using ultrasonic sensors aircrafts measures the distance of the object clearly and communicate to ground station clearly using wireless technology.

1.4. Block Diagram

Aircraft 1:

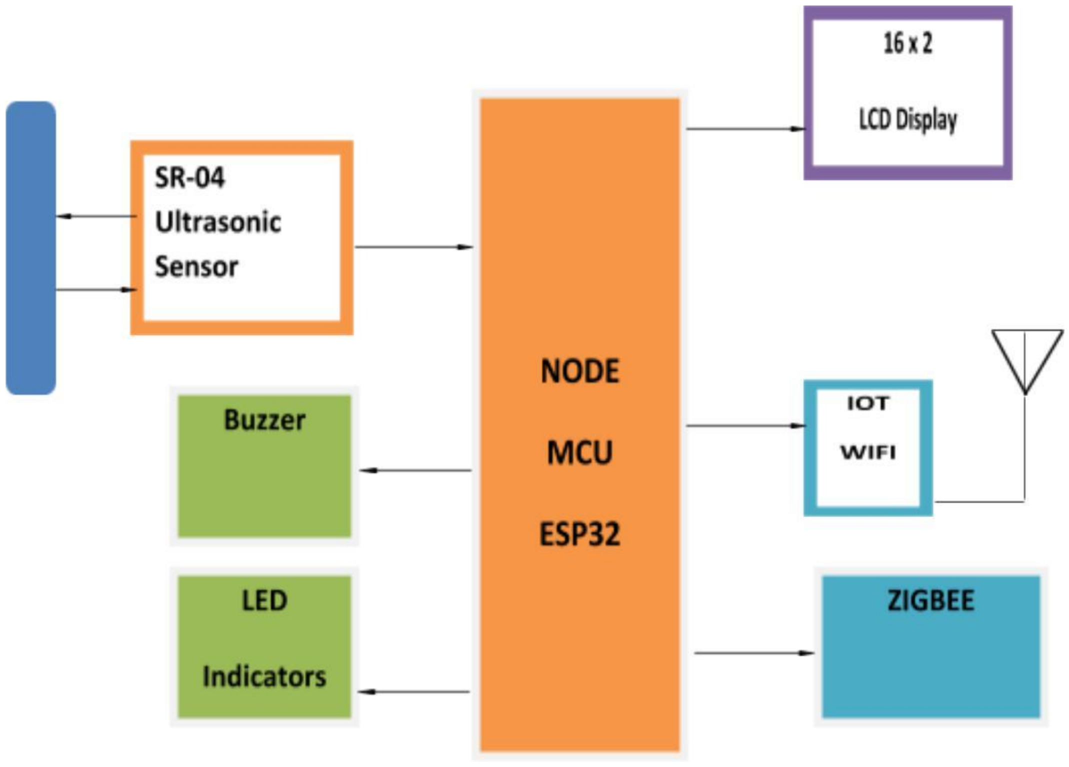


Fig.1. Block Diagram of aircraft 1

Aircraft 2:

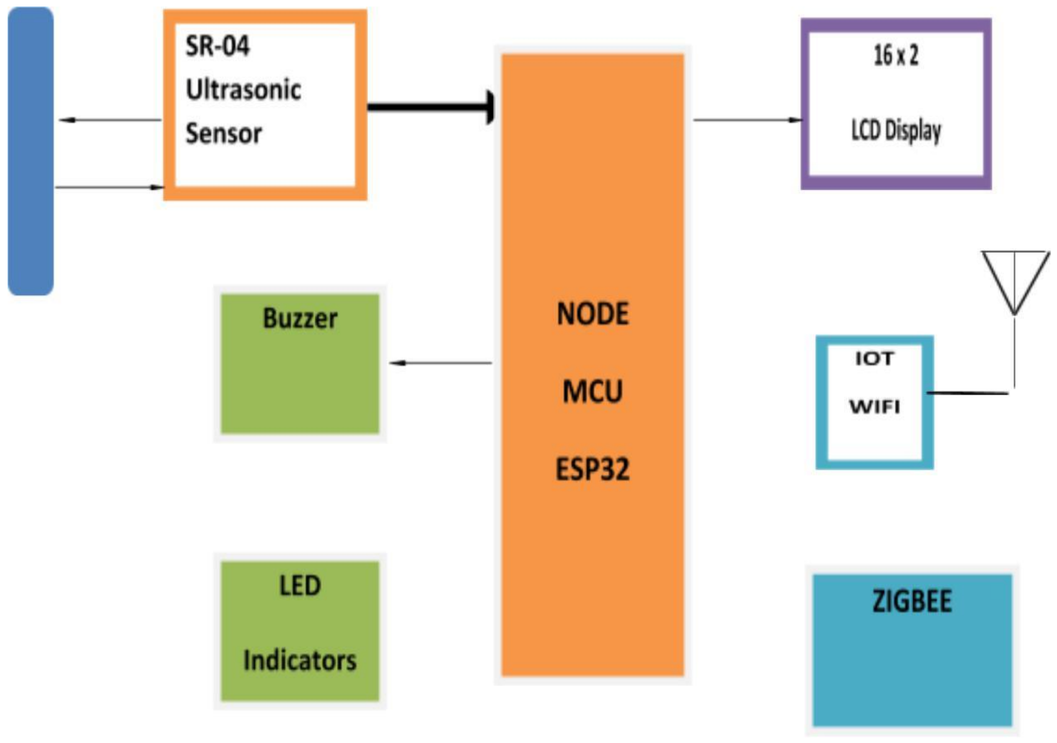


Fig.2. Block Diagram of aircraft 2

CHAPTER 2
EMBEDDED SYSTEM

2.1. Introduction

An embedded system can be defined as a computing device that does a specific focused job. Appliances such as the air conditioner, VCD player, DVD player, printer, fax machine, mobile phone etc. are examples of embedded systems. Each of these appliances will have a processor and special hardware to meet the specific requirement of the application along with the embedded software that is executed by the processor for meeting that specific requirement. The embedded software is also called “firmware”. The desktop/laptop computer is a general purpose computer. You can use it for a variety of applications such as playing games, word processing, accounting, software development and so on. In contrast, the software in the embedded systems is always fixed listed below:

Embedded systems do a very specific task, they cannot be programmed to do different things. Embedded systems have very limited resources, particularly the memory. Generally, they do not have secondary storage devices such as the CDROM or the floppy disk. Embedded systems have to work against some deadlines. A specific job has to be completed within a specific time. In some embedded systems, called real-time systems, the deadlines are stringent. Missing a deadline may cause a catastrophe loss of life or damage to property. Embedded systems are constrained for power. As many embedded systems operate through a battery, the power consumption has to be very low. Some embedded systems have to operate in extreme environmental conditions such as very high temperatures and humidity.

Application Areas:

Nearly 99 per cent of the processors manufactured end up in embedded systems. The embedded system market is one of the highest growth areas as these systems are used in very market segment- consumer electronics, office automation, industrial automation, biomedical engineering, wireless communication, data communication, telecommunications, transportation, military and so on.

ⅰ. Consumer appliances

At home we use a number of embedded systems which include digital camera, digital diary, DVD player, electronic toys, microwave oven, remote controls for TV and air conditioner, VCO player, video game consoles, video recorders etc.

Today's high-tech car has about 20 embedded systems for transmission control, engine spark control, air conditioning, navigation etc. Even wristwatches are now becoming embedded systems. The palmtops are powerful embedded systems using which we can carry out many general-purpose tasks such as playing games and word processing.

ⅱ. Office Automation

The office automation products using embedded systems are copying machine, fax machine, key telephone, modem, printer, scanner etc.

ⅲ. Industrial Automation

Today a lot of industries use embedded systems for process control. These include pharmaceutical, cement, sugar, oil exploration, nuclear energy, electricity generation and transmission. The embedded systems for industrial use are designed to carry out specific tasks such as monitoring the temperature, pressure, humidity, voltage, current etc., and then take appropriate action based on the monitored levels to control other devices or to send information to a centralized monitoring station. In hazardous industrial environment, where human presence has to be avoided, robots are used, which are programmed to do specific jobs. The robots are now becoming very powerful and carry out many interesting and complicated tasks such as hardware assembly.

ⅳ. Medical Electronics

Almost every medical equipment in the hospital is an embedded system. These equipment include diagnostic aids such as ECG, EEG, blood pressure measuring devices, X-ray scanners; Equipment used in blood analysis, radiation, colonoscopy, endoscopy etc. Developments in medical electronics have paved way for more accurate diagnosis of diseases.

v . Computer Networking

Computer networking products such as bridges, routers, Integrated Services Digital Networks (ISDN), Asynchronous Transfer Mode (ATM), X.25 and frame relay switches are embedded systems which implement the necessary data communication protocols. For example, a router interconnects two networks.

The two networks may be running different protocol stacks. The router's function is to obtain the data packets from incoming pores, analyse the packets and send them towards the destination after doing necessary protocol conversion. Most networking equipment, other than the end systems (desktop computers) we use to access the networks, are embedded systems.

vi . Telecommunications

In the field of telecommunications, the embedded systems can be categorized as subscriber terminals and network equipment. The subscriber terminals such as key telephones, ISDN phones, terminal adapters, web cameras are embedded systems. The network equipment includes multiplexers, multiple access systems, Packet Assemblers Dissemblers (PADs), sate11ite modems etc. IP phone, IP gateway, IP gatekeeper etc. are the latest embedded systems that provide very low-cost voice communication over the Internet.

vii . Wireless Technologies

Advances in mobile communications are paving way for many interesting applications using embedded systems. The mobile phone is one of the marvels of the last decade of the 20th century. It is a very powerful embedded system that provides voice communication while we are on the move. The Personal Digital Assistants and the palmtops can now be used to access multimedia service over the Internet. Mobile communication infrastructure such as base station controllers, mobile switching centers are also powerful embedded systems.

viii. Insemination

Testing and measurement are the fundamental requirements in all scientific and engineering activities. The measuring equipment we use in laboratories to measure parameters such as weight, temperature, pressure, humidity, voltage, current etc. are all embedded systems. Test equipment such as oscilloscope, spectrum analyser, logic analyser, protocol analyser, radio communication test set etc. are embedded systems built around powerful processors. Thank to miniaturization, the test and measuring equipment are now becoming portable facilitating easy testing and measurement in the field by field personnel.

ix . Security

Security of persons and information has always been a major issue. We need to protect our homes and offices; And also the information we transmit and store. Developing embedded systems for security applications is one of the most lucrative businesses nowadays. Security devices at homes, offices, airports etc. for authentication and verification are embedded systems. Encryption devices are nearly 99 per cent of the processors that are manufactured end up in~ embedded systems. Embedded systems find applications in every industrial segment- consumer electronics, transportation, avionics, biomedical engineering, manufacturing, process control and industrial automation, data communication, telecommunication, defence, security etc. Used to encrypt the data/voice being transmitted on communication links such as telephone lines. Biometric systems using fingerprint and face recognition are now being extensively used for user authentication in banking applications as well as for access control in high security buildings.

x . Finance

Financial dealing through cash and cheques are now slowly paving way for transactions using smart cards and ATM (Automatic Teller Machine, also expanded as Any Time Money) machines. Smart card, of the size of a credit card, has a small micro-controller and memory; And it interacts with the smart card reader! ATM machine and acts as an electronic wallet. Smart card technology has the capability of ushering in a cashless society.

2.2. Overview of Embedded System Architecture

Every embedded system consists of custom-built hardware built around a Central Processing Unit (CPU). This hardware also contains memory chips onto which the software is loaded. The software residing on the memory chip is also called the ‘firmware’. The embedded system architecture can be represented as a layered architecture as shown in Fig. The operating system runs above the hardware, and the application software runs above the operating system.

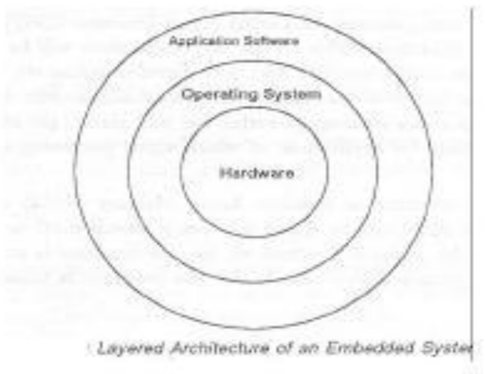


Fig.3. Layered Architecture of an embedded system

The same architecture is applicable to any computer including a desktop computer. However, there are significant differences. It is not compulsory to have an operating system in every embedded system. For small appliances such as remote control units, air conditioners, toys etc., there is no need for an operating system and you can write only the software specific to that application. For applications involving complex processing, it is advisable to have an operating system. In such a case, you need to integrate the application software with the operating system and then transfer the entire software on to the memory chip. Once the software is transferred to the memory chip, the software will continue to run for a long time you don’t need to reload new software. Now, let us see the details of the various building blocks of the hardware of an embedded system. As shown in Fig. the building blocks are

- Central Processing Unit (CPU)
- Memory (Read-only Memory and Random Access Memory)
- Input Devices
- Output devices
- Communication interfaces & Application specific circuitry

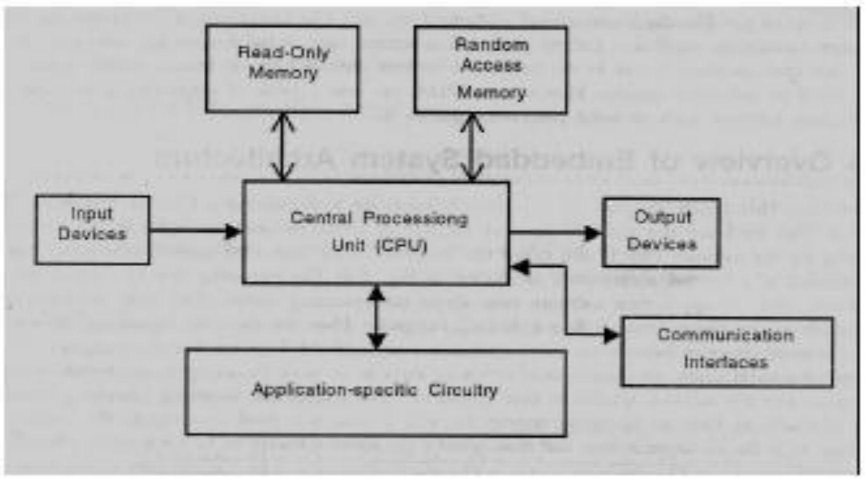


Fig.4. Hardware of embedded system

i . Central Processing Unit (CPU)

The Central Processing Unit (processor, in short) can be any of the following: microcontroller, microprocessor or Digital Signal Processor (DSP). A micro-controller is a low-cost processor. Its main attraction is that on the chip itself, there will be many other components such as memory, serial communication interface, analog to digital converter etc. So, for small applications, a microcontroller is the best choice as the number of external components required will be very less. On the other hand, microprocessors are more powerful, but you need to use many external components with them. D5P is used mainly for applications in which signal processing is involved such as audio and video processing.

ii . Memory

The memory is categorized as Random Access 11emory (RAM) and Read Only Memory (ROM). The contents of the RAM will be erased if power is switched off to the chip, whereas ROM retains the contents even if the power is switched off. So, the firmware is stored in the ROM. When power is switched on, the processor reads the ROM; The program is program is executed.

iii. Input Devices

Unlike the desktops, the input devices to an embedded system have very limited capability. There will be no keyboard or a mouse, and hence interacting with the embedded system is no easy task. Many embedded systems will have a small keypad you press one key to give a specific command. A keypad may be used to input only the digits. Many embedded systems used in process control do not have any input device for user interaction; They take inputs from sensors or transducers 1'fnd produce electrical signals that are in turn fed to other systems.

iv. Output Devices

The output devices of the embedded systems also have very limited capability. Some embedded systems will have a few Light Emitting Diodes (LEDs) to indicate the health status of the system modules, or for visual indication of alarms. A small Liquid Crystal Display (LCD) may also be used to display some important parameters.

v. Communication Interfaces

The embedded systems may need to, interact with other embedded systems at they may have to transmit data to a desktop. To facilitate this, the embedded systems are provided with one or a few communication interfaces such as RS232, RS422, RS485, Universal Serial Bus (USB), and IEEE 1394, Ethernet etc.

vi. Application Specific Circuitry

Sensors, transducers, special processing and control circuitry may be required fat an embedded system, depending on its application. This circuitry interacts with the processor to carry out the necessary work. The entire hardware has to be given power supply either through the 230 volts main supply or through a battery. The hardware has to design in such a way that the power consumption is minimized.

2.3. MICROCONTROLLER

A Microcontroller (or MCU) is a computer-on-a-chip used to control electronic devices. It is a type of microprocessor emphasizing self-sufficiency and cost-effectiveness, in contrast to a general-purpose microprocessor (the kind used in a PC). A typical microcontroller contains all the memory and interfaces needed for a simple application, whereas a general purpose microprocessor requires additional chips to provide these functions.

A microcontroller is a single integrated circuit with the following key features:

- Central processing unit - ranging from small and simple 8-bit processors to sophisticated 32- or 64-bit processors.
- Input/output interfaces such as serial ports.
- RAM for data storage.
- ROM, EEPROM or Flash memory for program storage.
- Clock generator - often an oscillator for a quartz timing crystal, resonator or RC circuit.

Microcontrollers are inside many kinds of electronic equipment (see embedded system). They are the vast majority of all processor chips sold. Over 50% are "simple" controllers, and another 20% are more specialized digital signal processors (DSPs).
A typical home in a developed country is likely to have only one or two general-purpose microprocessors but somewhere between one and two dozen microcontrollers. A typical mid-range vehicle has as many as 50 or more microcontrollers. They can also be found in almost any electrical device: washing machines, microwave ovens, telephones etc.

2.3.1. NodeMCU

2.3.1.1. Introduction to NodeMCU

The module is mainly based on ESP8266 that is a low-cost Wi-Fi microchip incorporating both a full TCP/IP stack and microcontroller capability. It is introduced by manufacturer Espressif Systems – A manufacturer based in Shanghai, China.

Arduino Modules and Microcontrollers have always been a great choice to incorporate automation into the relevant project. But these modules come with a little drawback as they don’t feature a built-in Wi-Fi capability, subsequently, we need to add external Wi-Fi protocol into these devices to make them compatible with the internet channel. This is where NodeMCU comes handy that incorporates a built-in Wi-Fi support, giving an easy pathway to design IoT applications as per your technical requirements.

NodeMCU is an open source firmware and development kit that plays a vital role in designing your own IoT product using a few Lua script lines.Multiple GPIO pins on the board allow you to connect the board with other peripherals and are capable of generating PWM, I2C, SPI, and UART serial communications.

The interface of the module is mainly divided into two parts including both Firmware and Hardware where former runs on the ESP8266 Wi-Fi SoC and later is based on the ESP-12 module.

The firmware is based on Lua – A scripting language that is easy to learn, giving a simple programming environment layered with a fast scripting language that connects you with a well-known developer community.

The open source firmware gives you the flexibility to edit, modify and rebuilt the existing module and keep changing the entire interface until you succeed in optimizing the module as per your requirements.

- USB to UART converter is added on the module that helps in converting USB data to UART data which mainly understands the language of serial communication.

  Instead of the regular USB port, Micro USB port is included in the module that connects it with the computer for dual purposes: programming and powering up the board.

- The board incorporates status LED that blinks and turns off immediately, giving you the current status of the module if it is running properly when connected with the computer.

The ability of module to establish a flawless Wi-Fi connection between two channels makes it an ideal choice for incorporating it with other embedded devices like Raspberry Pi.

2.3.1.2. NodeMCU Pinout

NodeMCU comes with a number of GPIO Pins. Following figure shows the Pinout of the board. There is a candid difference between Vin and VU where former is the regulated voltage that may stand somewhere between 7 to 12 V while later is the power voltage for USB that must be kept around 5 V.

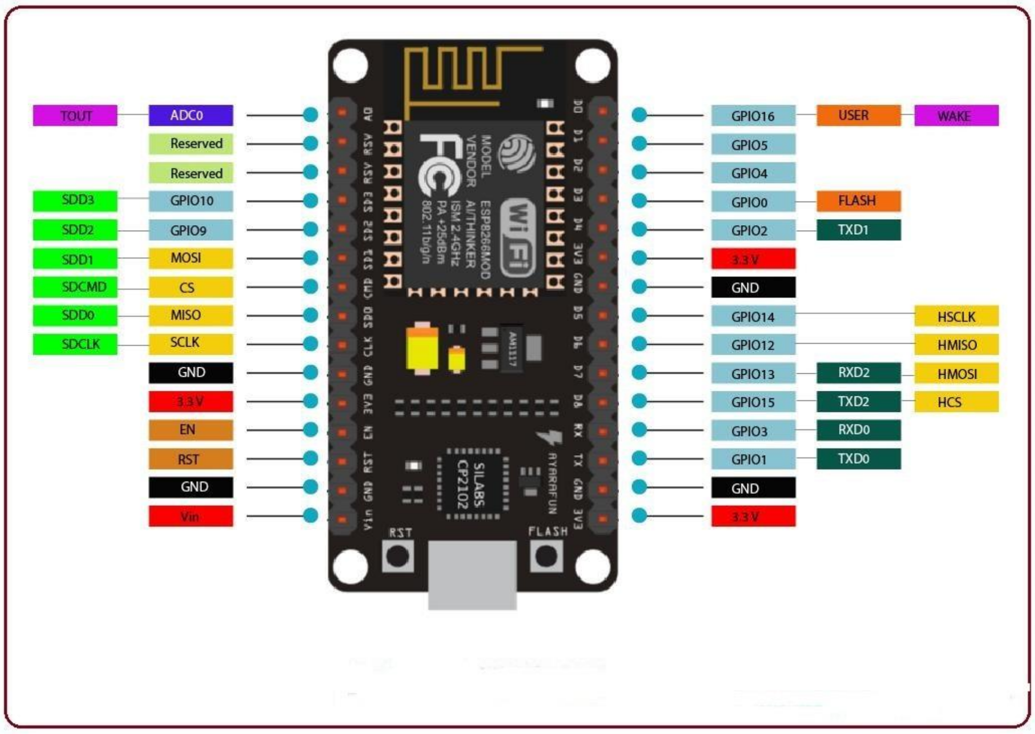


Fig.5. Pin Diagram of NodeMCU

2.3.1.3. NodeMCU Features

- Open source
- Arduino like hardware
- Status LED

- Micro USB port
- Reset/Flash buttons
- Interactive and Programmable
- Low cost
- ESP8266 with inbuilt Wi-Fi
- USB to UART converter
- GPIO pins

As mentioned above, a cable supporting micro USB port is used to connect the board. As you connect the board with a computer, LED will flash. You may need some drivers to be installed on your computer if it fails to detect the NodeMCU board.

Note: We use Arduino IDE software for programming this module. It is important to note that the pin configuration appearing on the board is different from the configuration we use to program the board on the software i.e. when we write code for targeting pin 16 on the Arduino IDE, it will actually help is laying out the communication with the D0 pin on the module.

Following figure the shows the pin configuration to use in Arduino IDE.

2.3.1.4. How to Power NodeMCU?

You can see from the pinout image above, there are five ground pins and three 3 pins on the board. The board can be powered up using the following three ways.

- USB Power, It proves to an ideal choice for loading programs unless the project you aim to design requires separate interface i.e. disconnected from the computer.
- Provide 3.3V, this is another great option to power up the module. If you have your own off board regulator, you can generate an instant power source for your development kit.

- Power Vin, This is a voltage regulator that comes with the ability to support up to 800 mA. It can handle somewhere between 7 to 12 V. You cannot power the devices operating at 3.3 V, as this regulator unable to generate as low as 3.3V.

CHAPTER 3
REGULATED POWER SUPPLY

The power supply section is the section which provide +5V for the components to work. IC LM7805 is used for providing a constant power of +5V.

The AC voltage, typically 220V, is connected to a transformer, which steps down that AC voltage down to the level of the desired DC output. A diode rectifier then provides a full-wave rectified voltage that is initially filtered by a simple capacitor filter to produce a DC voltage. This resulting DC voltage usually has some ripple or AC voltage variation.

A regulator circuit removes the ripples and also retains the same DC value even if the input DC voltage varies, or the load connected to the output DC voltage changes. This voltage regulation is usually obtained using one of the popular voltage regulator IC units.

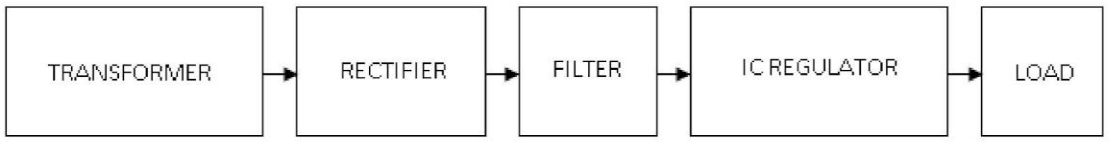


Fig.6. Block Diagram of Power Supply

3.1. Transformer

Transformers convert AC electricity from one voltage to another with little loss of power. Transformers work only with AC and this is one of the reasons why mains electricity is AC.

Step-up transformers increase voltage, step-down transformers reduce voltage. Most power supplies use a step-down transformer to reduce the dangerously high mains voltage (230V in India) to a safer low voltage.

The input coil is called the primary and the output coil is called the secondary. There is no electrical connection between the two coils; Instead they are linked by an alternating magnetic field created in the soft iron core of the transformer. Transformers waste very little power so the power out is (almost) equal to the power in. Note that as voltage is stepped down current is stepped up.

The transformer will step down the power supply voltage (0-230V) to (0- 6V) level. Then the secondary of the potential transformer will be connected to the bridge rectifier, which is constructed with the help of PN junction diodes. The advantages of using bridge rectifier are it will give peak voltage output as DC.

3.2. Rectifier

There are several ways of connecting diodes to make a rectifier to convert AC to DC. The bridge rectifier is the most important and it produces full wave varying DC. A full wave rectifier can also be made from just two diodes if a centre tap transformer is used, but this method is rarely used now that diodes are cheaper. A single diode can be used as a rectifier but it only uses the positive (+) parts of the AC wave to produce half wave varying DC

3.2.1. Bridge Rectifier

When four diodes are connected as shown in figure, the circuit is called as bridge rectifier. The input to the circuit is applied to the diagonally opposite corners of the network, and the output is taken from the remaining two corners. Let us assume that the transformer is working properly and there is a positive potential, at point A and a negative potential at point B. the positive potential at point A will forward bias D3 and reverse bias D4.

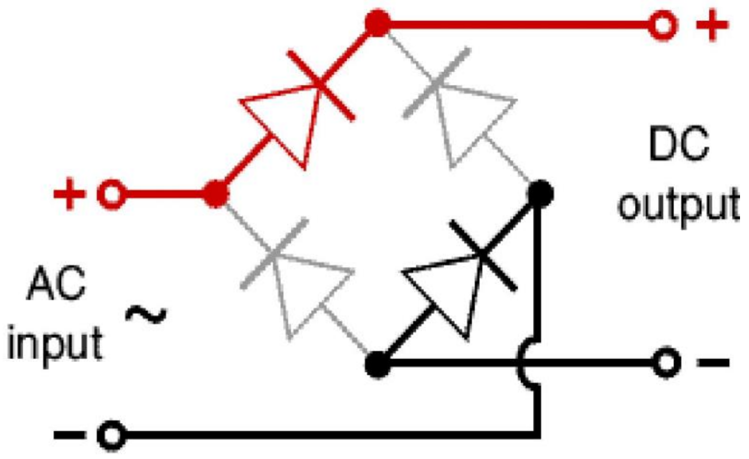


Fig.7. Bridge Rectifier

The negative potential at point B will forward bias D1 and reverse D2. At this time D3 and D1 are forward biased and will allow current flow to pass through them; D4 and D2 are reverse biased and will block current flow.

One advantage of a bridge rectifier over a conventional full-wave rectifier is that with a given transformer the bridge rectifier produces a voltage output that is nearly twice that of the conventional full-wave circuit.

i. The main advantage of this bridge circuit is that it does not require a special centre tapped transformer, thereby reducing its size and cost.
ii. The single secondary winding is connected to one side of the diode bridge network and the load to the other side as shown below.
iii. The result is still a pulsating direct current but with double the frequency.

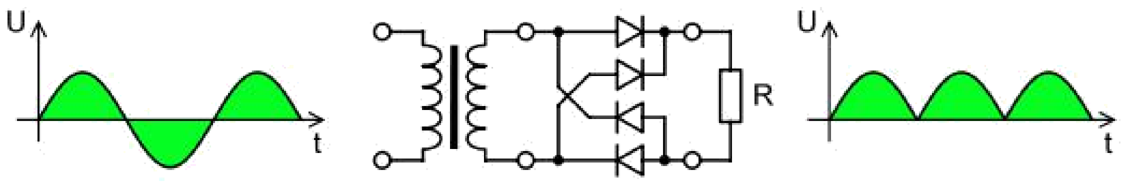


Fig.8. Output Waveform of pulsating DC

3.3. Filter

Smoothing is performed by a large value electrolytic capacitor connected across the DC supply to act as a reservoir, supplying current to the output when the varying DC voltage from the rectifier is falling. The capacitor charges quickly near the peak of the varying DC, and then discharges as it supplies current to the output.

3.4. Voltage Regulators

Voltage regulators comprise a class of widely used ICs. Regulator IC units contain the circuitry for reference source, comparator amplifier, control device, and overload protection all in a single IC. IC units provide regulation of either a fixed positive voltage, a fixed negative voltage. The regulators can be selected for operation with load currents from hundreds of milli amperes to tens of amperes, corresponding to power ratings from milli watts to tens of watts.

A fixed three-terminal voltage regulator has an unregulated dc input voltage, Vi, applied to one input terminal, a regulated dc output voltage, Vo, from a second terminal, with the third terminal connected to ground.

The series 78 regulators provide fixed positive regulated voltages from 5 to 24 volts. Similarly, the series 79 regulators provide fixed negative regulated voltages from 5 to 24 volts. Voltage regulator ICs are available with fixed (typically 5, 12 and 15V) or variable output voltages. They are also rated by the maximum current they can pass. Negative voltage regulators are available, mainly for use in dual supplies. Most regulators include some automatic protection from excessive current ('overload protection') and overheating ('thermal protection').

Many of the fixed voltage regulator ICs has 3 leads and look like power transistors, such as the 7805 +5V 1Amp regulator. They include a hole for attaching a heat sink if necessary.

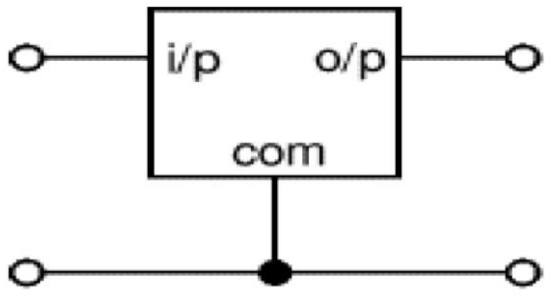


Fig.9. Regulator pin diagram

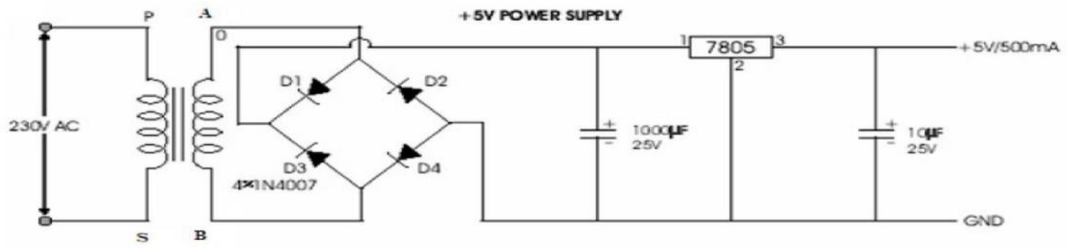


Fig.10. Circuit Diagram of Power Supply

3.5. Light Emitting Diode (LED)

LED is abbreviation of Light Emitting Diode. It’s nothing, but just a combination of semiconductors which emits light when current pass through it. Over the years, semiconductor technology has advanced to bigger heights, Light Emitting Devices have also been a part of this revolution and as a result, now we have LED which give better illumination with low power consumption.

3.5.1. Types of LED

There are many types of LEDs available in the market. As you can see on above pic there is different LEDs according to our requirement and there has been many other are too available depending upon different parameters and LEDs are choose according to parameters like space required by it, size, intensity, colors, etc. Typical LEDs are in size of 3mm, 5mm and 8mm.

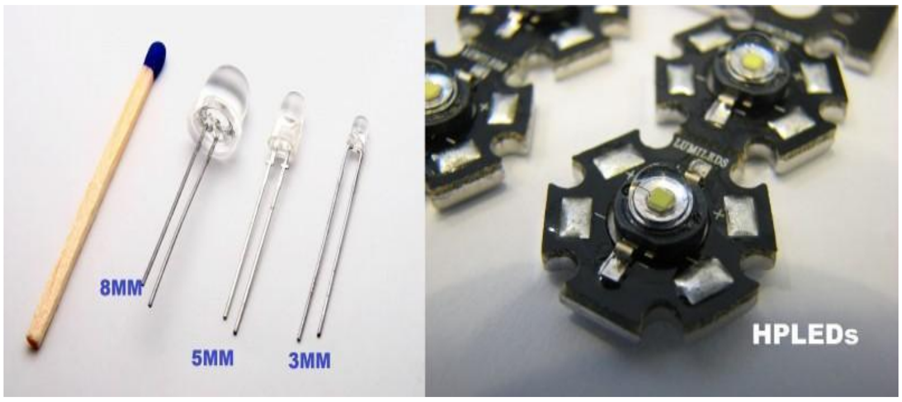


Fig.11. Types of LED

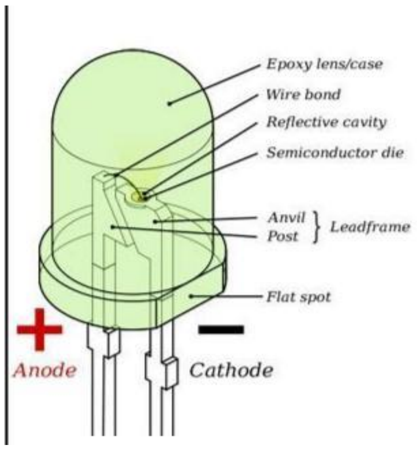


Fig.12. Internal structure of LED

3.5.2. Operating parameters & circuit symbol

Above figures show basic elements inside the LED and circuit symbol which helps in interfacing LED. Typical current ratings ranges from around 1 mA to above 20 mA and voltage is about colors.

- 1.9 to 2.1 V for red, orange and yellow,
- 3.0 to 3.4 V for green and blue,
- 2.9 to 4.2 V for violet, pink, purple and white.
- 5V and 12V LEDs are incorporate a suitable series resistor for direct connection to a 5V or 12V supply.

3.5.3. Applications

LED is everywhere because it’s an indicating component used in many areas. Just look around, if u can’t find even single LED, you are not on earth. We can find LED in light bulbs, television etc.

CHAPTER 4
BUZZER

4.1. Introduction

A buzzer or beeper is a signalling device, usually electronic, typically used in automobiles, household appliances such as a microwave oven, or game shows.

It most commonly consists of a number of switches or sensors connected to a control unit that determines if and which button was pushed or a preset time has lapsed, and usually illuminates a light on the appropriate button or control panel, and sounds a warning in the form of a continuous or intermittent buzzing or beeping sound. Initially this device was based on an electromechanical system which was identical to an electric bell without the metal gong (which makes the ringing noise).

Often these units were anchored to a wall or ceiling and used the ceiling or wall as a sounding board. Another implementation with some AC-connected devices was to implement a circuit to make the AC current into a noise loud enough to drive a loudspeaker and hook this circuit up to a cheap 8-ohm speaker. Nowadays, it is more popular to use a ceramic-based piezoelectric sounder like a Sonalert which makes a high-pitched tone. Usually these were hooked up to “driver” circuits which varied the pitch of the sound or pulsed the sound on and off.

In game shows it is also known as a “lockout system,” because when one person signals (“buzzes in”), all others are locked out from signalling. Several game shows have large buzzer buttons which are identified as “plungers”.

Fig.13. Buzzer

4.2. USES

- Annunciator panels
- Electronic metronomes
- Game shows
- Microwave ovens and other household appliances
- Sporting events such as basketball games
- Electrical alarms

CHAPTER 5
LIQUID CRYSTAL DISPLAY

5.1. Introduction

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.

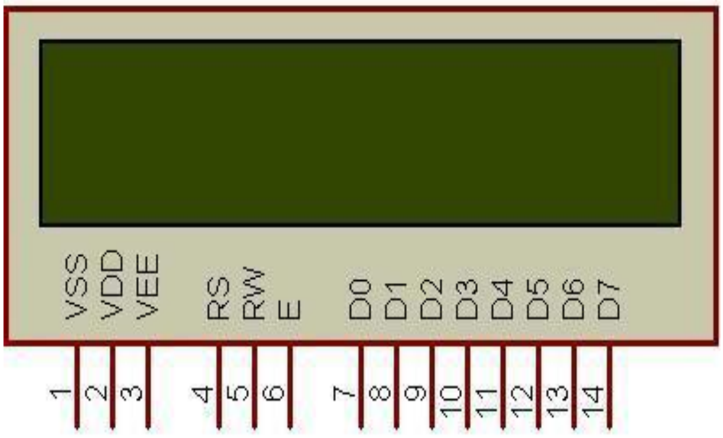


Fig.14. 16x2 LCD

The most commonly used Character based LCDs are based on Hitachi's HD44780 controller or other which are compatible with HD44580.

5.2. Pin Description

Most LCDs with 1 controller has 14 Pins and LCDs with 2 controller has 16 Pins (two pins are extra in both for backlight LED connections). Pin description is shown in the table.

5.3. Data/Signals/Execution of LCD

Coming to data, signals and execution. LCD accepts two types of signals, one is data, and another is control. These signals are recognized by the LCD module from status of the RS pin. Now data can be read also from the LCD display, by pulling the R/W pin high. As soon as the E pin is pulsed, LCD display reads data at the falling edge of the pulse and executes it, same for the case of transmission.

LCD display takes a time of 39-43µS to place a character or execute a command. Except for clearing display and to seek cursor to home position it takes 1.53ms to 1.64ms. Any attempt to send any data before this interval may lead to failure to read data or execution of the current data in some devices. Some devices compensate the speed by storing the incoming data to some temporary registers.

5.3.1. Instruction Register (IR) and Data Register (DR)

There are two 8-bit registers in HD44780 controller Instruction and Data register. Instruction register corresponds to the register where you send commands to LCD e.g LCD shift command, LCD clear, LCD address etc. and Data register is used for storing data which is to be displayed on LCD.

When send the enable signal of the LCD is asserted, the data on the pins is latched in to the data register and data is then moved automatically to the DDRAM and hence is displayed on the LCD.

Data Register is not only used for sending data to DDRAM but also for CGRAM, the address where you want to send the data, is decided by the instruction you send to LCD. We will discuss more on LCD instruction set further in this tutorial.

Table.1. Pin Configuration table for a 16X2 LCD character display

Pin Number	Symbol	Function
1	Vss	Ground Terminal
2	Vcc	Positive Supply
3	Vdd	Contrast adjustment
4	RS	Register Select; 0→Instruction Register, 1→Data Register
5	R/W	Read/write Signal; 1→Read, 0→ Write
6	E	Enable; Falling edge
7	DB0	Bi directional data bus, data transfer is performed once, thru DB0 to DB7, in the case of interface data length is 8-bits; and twice, through DB4 to DB7 in the case of interface data length is 4-bits. Upper four bits first then lower four bits.
8	DB1	
9	DB2	
10	DB3	
11	DB4	
12	DB5	
13	DB6	
14	DB7	
15	LED-(K)	Back light LED cathode terminal
16	LED+(A)	Back Light LED anode terminal

5.3.2. Commands and Instruction set

Only the instruction register (IR) and the data register (DR) of the LCD can be controlled by the MCU. Before starting the internal operation of the LCD, control information is temporarily stored into these registers to allow interfacing with various MCUs, which operate at different speeds, or various peripheral control devices.

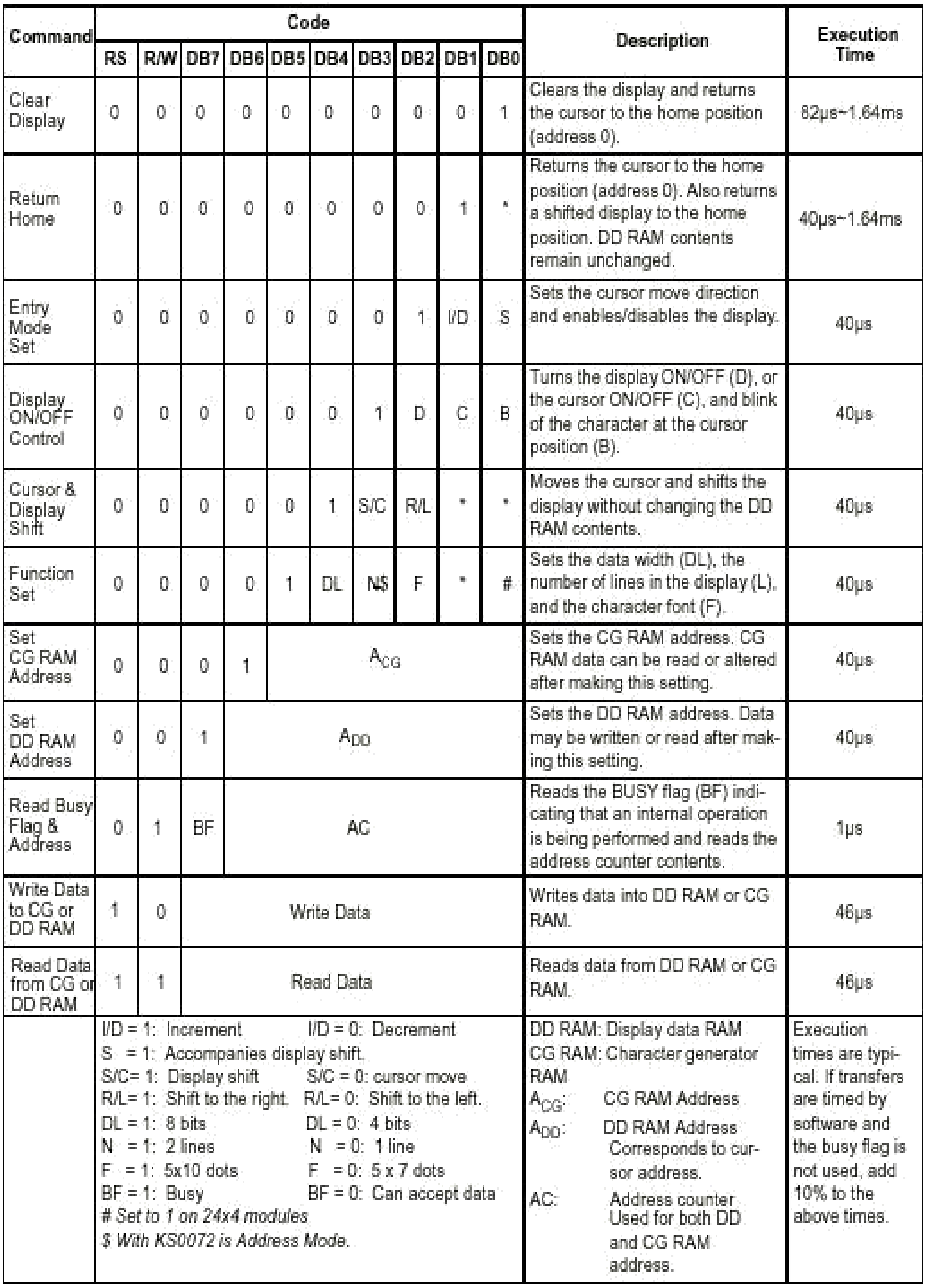
The internal operation of the LCD is determined by signals sent from the MCU. These signals, which include register selection signal (RS), read/write signal (R/W), and the data bus (DB0 to DB7), make up the LCD instructions (Table 3). There are four categories of instructions that:

- Designate LCD functions, such as display format, data length, etc.

- Set internal RAM addresses
- Perform data transfer with internal RAM
- Perform miscellaneous functions

Although looking at the table you can make your own commands and test them. Table 2 is a brief list of useful commands which are used frequently while working on the LCD.

Table.2. Showing various LCD Command Description



Command	Code										Description	Execution Time
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears the display and returns the cursor to the home position (address 0).	82μs~1.64ms
Return Home	0	0	0	0	0	0	0	0	1	*	Returns the cursor to the home position (address 0). Also returns a shifted display to the home position. DD RAM contents remain unchanged.	40μs~1.64ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets the cursor move direction and enables/disables the display.	40μs
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	Turns the display ON/OFF (D), or the cursor ON/OFF (C), and blink of the character at the cursor position (B).	40μs
Cursor & Display Shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves the cursor and shifts the display without changing the DD RAM contents.	40μs
Function Set	0	0	0	0	1	DL	N$	F	*	#	Sets the data width (DL), the number of lines in the display (L), and the character font (F).	40μs
Set CG RAM Address	0	0	0	1	A_{CG}						Sets the CG RAM address. CG RAM data can be read or altered after making this setting.	40μs
Set DD RAM Address	0	0	1	A_{DD}							Sets the DD RAM address. Data may be written or read after making this setting.	40μs
Read Busy Flag & Address	0	1	BF	AC							Reads the BUSY flag (BF) indicating that an internal operation is being performed and reads the address counter contents.	1μs
Write Data to CG or DD RAM	1	0	Write Data								Writes data into DD RAM or CG RAM.	46μs
Read Data from CG or DD RAM	1	1	Read Data								Reads data from DD RAM or CG RAM.	46μs
	I/D = 1: Increment I/D = 0: Decrement S = 1: Accompanies display shift. S/C= 1: Display shift S/C = 0: cursor move R/L= 1: Shift to the right. R/L= 0: Shift to the left. DL = 1: 8 bits DL = 0: 4 bits N = 1: 2 lines N = 0: 1 line F = 1: 5x10 dots F = 0: 5 x 7 dots BF = 1: Busy BF = 0: Can accept data *# Set to 1 on 24x4 modules* *$ With KS0072 is Address Mode.*										DD RAM: Display data RAM CG RAM: Character generator RAM A_{CG}: CG RAM Address A_{DD}: DD RAM Address Corresponds to cursor address. AC: Address counter Used for both DD and CG RAM address.	Execution times are typical. If transfers are timed by software and the busy flag is not used, add 10% to the above times.

5.3.3. List of Command

Table.3. Frequently Used Commands And Instructions For LCD

No.	Instruction	Hex	Decimal
1	Function Set: 8-bit, 1 Line, 5x7 Dots	0x30	48
2	Function Set: 8-bit, 2 Line, 5x7 Dots	0x38	56
3	Function Set: 4-bit, 1 Line, 5x7 Dots	0x20	32
4	Function Set: 4-bit, 2 Line, 5x7 Dots	0x28	40
5	Entry Mode	0x06	6
6	Display off Cursor off (clearing display without clearing DDRAM content)	0x08	8
7	Display on Cursor on	0x0E	14
8	Display on Cursor off	0x0C	12
9	Display on Cursor blinking	0x0F	15
10	Shift entire display left	0x18	24
12	Shift entire display right	0x1C	30
13	Move cursor left by one character	0x10	16
14	Move cursor right by one character	0x14	20
15	Clear Display (also clear DDRAM content)	0x01	1

* DDRAM address given in LCD basics section see Figure 2,3,4

** CGRAM address from 0x00 to 0x3F, 0x00 to 0x07 for char1 and so on.

5.4. Liquid crystal displays interfacing with Controller

The LCD standard requires 3 control lines and 8 I/O lines for the data bus.

• 8 data pins D7:D0

Bi-directional data/command pins.

Alphanumeric characters are sent in ASCII format.

• RS: Register Select

RS = 0 -> Command Register is selected

RS = 1 -> Data Register is selected

• R/W: Read or Write

0 -> Write, 1 -> Read

• E: Enable (Latch data)

Used to latch the data present on the data pins.

A high-to-low edge is needed to latch the data.

CHAPTER 6
ULTRASONIC SENSOR

6.1. Introduction

Ultrasonic sensors are industrial control devices that use sound waves above 20,000 Hz, beyond the range of human hearing, to measure and calculate distance from the sensor to a specified target object.

6.2. Features of ultrasonic sensors

- Devices with TEACH-IN functionality for fast and simple installation.
- ULTRA 3000 software for improved adaptation of sensors to applications.
- Adjustable sensitivity to the sound beam width for optimized adjustment of the sensor characteristics according to the application.
- Temperature compensation - compensates for sound velocity due to varying air temperatures.
- Synchronization input to prevent crosstalk interference when sensors are mounted within close proximity of each other.
- Sensors with digital and/ or analog outputs.

6.3. Description

Ultrasonic sensors use electrical energy and a ceramic transducer to emit and receive mechanical energy in the form of sound waves. Sound waves are essentially pressure waves that travel through solids, liquids and gases and can be used in industrial applications to measure distance or detect the presence or absence of targets. Ultrasonic sensors (also known as transrecivers when they both send and receive) work on a principle similar to radar or sonar which evaluate attributes of a target by interpreting the echoes from radio or sound waves respectively. Ultrasonic sensors generate high frequency sound waves and evaluate the echo which is received back by the sensor.Sensors calculate the time interval between sending the signal and receiving the echo to determine the distance to an object.

This technology can be used for measuring: wind speed and direction (anemometer), fullness of a tank, and speed through air or water. For measuring speed or direction a device uses multiple detectors and calculates the speed from the relative distances to particulates in the air or water. To measure the amount of liquid in a tank, the sensor measures the distance to the surface of the fluid. Further applications include: humidifiers, sonar, medical ultrasonography, burglar alarms, and non-destructive testing.

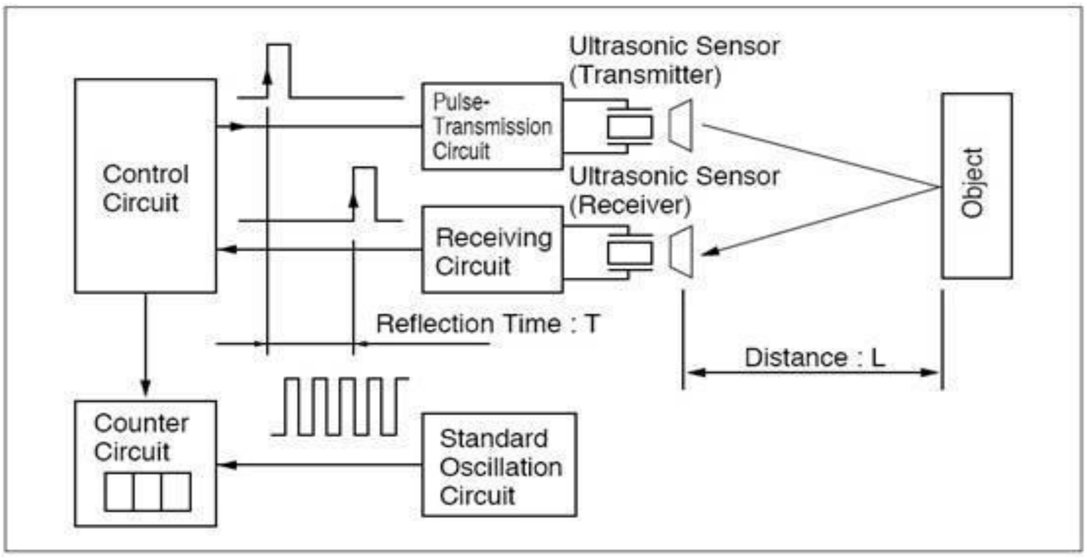

Ultrasonic Sensor (Transmitter)
Pulse-Transmission Circuit
Control Circuit
Ultrasonic Sensor (Receiver)
Receiving Circuit
Object
Reflection Time : T
Distance : L
Counter Circuit
Standard Oscillation Circuit

Fig.15. Block Diagram of Ultrasonic Sensor

Systems typically use a transducer which generates sound waves in the ultrasonic range, above 20,000 hertz, by turning electrical energy into sound, then upon receiving the echo turn the sound waves into electrical energy which can be measured and displayed.

The technology is limited by the shapes of surfaces and the density or consistency of the material. For example foam on the surface of a fluid in a tank could distort a reading.

6.4. Transducers

Sound field of a non focusing 4 MHz ultrasonic transducer with a near field length of N=67mm in water. The plot shows the sound pressure at a logarithmic db-scale.

Sound pressure field of the same ultrasonic transducer (4MHz, N=67mm) with the transducer surface having a spherical curvature with the curvature radius R=30mm

An ultrasonic transducer is a device that converts energy into ultrasound, or sound waves above the normal range of human hearing. While technically a dog whistle is an ultrasonic transducer that converts mechanical energy in the form of air pressure into ultrasonic sound waves, the term is more apt to be used to refer to piezoelectric transducers that convert electrical energy into sound. Piezoelectric crystals have the property of changing size when a voltage is applied, thus applying an alternating current (AC) across them causes them to oscillate at very high frequencies, thus producing very high frequency sound waves. The location at which a transducer focuses the sound, can be determined by the active transducer area and shape, the ultrasound frequency and the sound velocity of the propagation medium.

The example shows the sound fields of an unfocused and a focusing ultrasonic transducer in water.

Range: This ultrasonic rangefinder can measure distances up to 2.5 meters at an accuracy of 1 centimetre.

6.4.1. Working

The sensor has a ceramic transducer that vibrates when electrical energy is applied to it. The vibrations compress and expand air molecules in waves from the sensor face to a target object. A transducer both transmits and receives sound. The ultrasonic sensor will measure distance by emitting a sound wave and then "listening" for a set period of time, allowing for the return echo of the sound wave bouncing off the target, before retransmitting.

Microcontroller and the ultrasonic transducer module HC-SR04 forms the basis of this circuit. The ultrasonic module sends a signal to the object, then picks up its echo and outputs a wave form whose time period is proportional to the distance. The microcontroller accepts this signal, performs necessary processing and displays the corresponding distance on the 3 digit seven segment display. This circuit finds a lot of application in projects like automotive parking sensors, obstacle warning systems, terrain monitoring robots, industrial distance measurements etc.

It has a resolution of 0.3cm and the ranging distance is from 2cm to 500cm. It operates from a 5V DC supply and the standby current is less than 2mA. The module transmits an ultrasonic signal, picks up its echo, measures the time elapsed between the two events and outputs a waveform whose high time is modulated by the measured time which is proportional to the distance.

The supporting circuits fabricated on the module makes it almost stand alone and what the programmer need to do is to send a trigger signal to it for initiating transmission and receive the echo signal from it for distance calculation.

The HR-SR04 has four pins namely Vcc, Trigger, Echo, GND and they are explained in detail below.

6.4.2. Pin Description

1) VCC : 5V DC supply voltage is connected to this pin.
2) Trigger : The trigger signal for starting the transmission is given to this pin. The trigger signal must be a pulse with 10uS high time. When the module receives a valid trigger signal it issues 8 pulses of 40KHz ultrasonic sound from the transmitter. The echo of this sound is picked by the receiver.
3) Echo : At this pin, the module outputs a waveform with high time proportional to the distance.
4) GND : Ground is connected to this pin.

The transmitter part of the circuit is build around IC1(NE 555).The IC1 is wired as an astable multi vibrator operating at 40KHz.The output of IC1 is amplifier the complementary pair of transistors (Q1 & Q2) and transmitted by the ultrasonic transmitter K1.The push button switch S1 is used the activate the transmitter.

The receiver uses an ultrasonic sensor transducer (K2) to sense the ultrasonic signals. When an ultrasonic signal is falling on the sensor, it produces a proportional voltage signal at its output. This weak signal is amplified by the two stage amplifier circuit comprising of transistors Q3 and Q4.The output of the amplifier is rectified by the diodes D3 & D4.The rectified signal is given to the inverting input of the op-amp which is wired as a comparator. Whenever there is an ultrasonic signal falling on the receiver, the output of the comparator activates the transistors Q5 & Q6 to drive the relay. In this way the load connected via the relay can be switched. The diode D5 is used as a free-wheeling diode.

6.4.3. Detectors

Since piezoelectric crystal generate a voltage when force is applied to them, the same crystal can be used as an ultrasonic detector. Some systems use separate transmitter and receiver components while others combine both in a single piezoelectric transceiver.

Alternative methods for creating and detecting ultrasound include magnetostriction and capacitive actuation.

6.5. Application

Ultrasonic sensors use sound waves rather than light, making them ideal for stable detection of uneven surfaces, liquids, clear objects, and objects in dirty environments. These sensors work well for applications that require precise measurements between stationary and moving objects.

Ultrasonic sensor provides a very low-cost and easy method of distance measurement. This sensor is perfect for any number of applications that require you to perform measurements between moving or stationary objects. Naturally, robotics applications are very popular but it is also find in product which is useful in security systems or as an infrared replacement if so desired.

ⅰ. Use in medicine

Medical ultrasonic transducers (probes) come in a variety of different shapes and sizes for use in making pictures of different parts of the body. The transducer may be passed over the surface of the body or inserted into a body opening such as the rectum or vagina. Clinicians who perform ultrasound guided procedures often use a probe positioning system to hold the ultrasonic transducer.

ⅱ. Use in industry

Ultrasonic sensors are used to detect the presence of targets and to measure the distance to targets in many automated factories and process plants. Sensors with an on or off digital output are available for detecting the presence of objects, and sensors with an analog output which varies proportionally to the sensor to target separation distance are commercially available. Other types of transducers are used in commercially available ultrasonic cleaning devices. An ultrasonic transducer is affixed to a stainless steel pan which is filled with a solvent (frequently water or isopropanol) and a square wave is applied to it, imparting vibrational energy on the liquid.

iii. Application in Industries

- Measurement of dynamically changing diameters, distance, heights, depths.
- Counting number of units.

CHAPTER 7
SOFTWARE SPECIFICATION

The Arduino Integrated Development Environment or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them.

7.1. WRITING SKETCHES

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors.

The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom right hand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

NB: Versions of the Arduino Software (IDE) prior to 1.0 saved sketches with the extension .pde. It is possible to open these files with version 1.0, you will be prompted to save the sketch with the .ino extension on save.

Verify

Checks your code for errors compiling it.

Upload

Compiles your code and uploads it to the configured board.

See uploading below for details.

Note: If you are using an external programmer with your board, you can hold down the "shift" key on your computer when using this icon. The text will change to "Upload using Programmer"

New Creates a new sketch.

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

Note: due to a bug in Java, this menu doesn't scroll; if you need to open a sketch late in the list, use the File | Sketchbook menu instead.

Save Saves your sketch.

Serial Monitor Opens the serial monitor.

Additional commands are found within the five menus: File, Edit, Sketch, Tools, Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

ⅰ . File

- *New* Creates a new instance of the editor, with the bare minimum structure of a sketch already in place.
- *Open* Allows to load a sketch file browsing through the computer drives and folders.
- *Open Recent* Provides a short list of the most recent sketches, ready to be opened.
- *Sketchbook* Shows the current sketches within the sketchbook folder structure; Clicking on any name opens the corresponding sketch in a new editor instance.
- *Examples* Any example provided by the Arduino Software (IDE) or library shows up in this menu item. All the examples are structured in a tree that allows easy access by topic or library.
- *Close* Closes the instance of the Arduino Software from which it is clicked.

- *Save* Saves the sketch with the current name. If the file hasn't been named before, a name will be provided in a "Save as” window.
- *Saveas...* Allows to save the current sketch with a different name.

- *PageSetup* It shows the Page Setup window for printing.

- *Print*
  Sends the current sketch to the printer according to the settings defined in Page Setup.
- *Preferences*
  Opens the Preferences window where some settings of the IDE may be customized, as the language of the IDE interface.
- *Quit*
  Closes all IDE windows. The same sketches open when Quit was chosen will be automatically reopened the next time you start the IDE.

ii . EDIT

- *Undo/Redo*
  Goes back of one or more steps you did while editing; When you go back, you may go forward with Redo.
- *Cut*
  Removes the selected text from the editor and places it into the clipboard.
- *Copy*
  Duplicates the selected text in the editor and places it into the clipboard.
- *Copy for Forum* Copies the code of your sketch to the clipboard in a form suitable for posting to the forum, complete with syntax coloring.
- *Copy as HTML* Copies the code of your sketch to the clipboard as HTML, suitable for embedding in web pages.
- *Paste*
  Puts the contents of the clipboard at the cursor position, in the editor.
- *Select All* Selects and highlights the whole content of the editor.
- *Comment/Uncomment*
  Puts or removes the // comment marker at the beginning of each selected line.

- *Increase/Decrease Indent* :Adds or subtracts a space at the beginning of each selected line, moving the text one space on the right or eliminating a space at the beginning.
- *Find*

  Opens the Find and Replace window where you can specify text to search inside the current sketch according to several options.
- *Find Next*: Highlights the next occurrence - if any - of the string specified as the search item in the Find window, relative to the cursor position.
- *Find Previous* : Highlights the previous occurrence - if any - of the string specified as the search item in the Find window relative to the cursor position.

iii. SKETCH

- *Verify/Compile*

  Checks your sketch for errors compiling it; it will report memory usage for code and variables in the console area.
- *Upload*

  Compiles and loads the binary file onto the configured board through the configured Port.
- *Upload Using Programmer*

  This will overwrite the bootloader on the board; you will need to use Tools > Burn Bootloader to restore it and be able to Upload to USB serial port again. However, it allows you to use the full capacity of the Flash memory for your sketch. Please note that this command will NOT burn the fuses. To do so a *Tools -> Burn Bootloader* command must be executed.
- *Export Compiled Binary*

  Saves a .hex file that may be kept as archive or sent to the board using other tools.
- *Show Sketch Folder*

  Opens the current sketch folder.
- *Include Library*

  Adds a library to your sketch by inserting #include statements at the start of your code. For more details, see libraries below. Additionally, from this menu item you can access the Library Manager and import new libraries from .zip files.

- *Add File...*

  Adds a source file to the sketch (it will be copied from its current location). The new file appears in a new tab in the sketch window. Files can be removed from the sketch using the tab menu accessible clicking on the small triangle icon below the serial monitor one on the right side o the toolbar.

  ⅳ. TOOLS

- *Auto Format*

  This formats your code nicely: i.e. indents it so that opening and closing curly braces lineup and that the statements inside curly braces are indented more.

- *Archive Sketch*

  Archives a copy of the current sketch in .zip format. The archive is placed in the same directory as the sketch.

- *Fix Encoding & Reload*

  Fixes possible discrepancies between the editor charmap encoding and other operating systems char maps.

- *Serial Monitor*

  Opens the serial monitor window and initiates the exchange of data with any connected board on the currently selected Port. This usually resets the board, if the board supports Reset over serial port opening.

- *Board*

  Select the board that you're using. See below for descriptions of the various boards.

- *Port*

  This menu contains all the serial devices (real or virtual) on your machine. It should automatically refresh every time you open the top-level tools menu.

- *Programmer*

  For selecting a hardware programmer when programming a board or chip and not using the on board USB-serial connection. Normally you won't need this, but if you're burning a bootloader to a new microcontroller, you will use this.

- *Burn Bootloader*

  The items in this menu allow you to burn a bootloader onto the microcontroller on an Arduino board. This is not required for normal use of an Arduino or Genuino board but is useful if you purchase a new ATMEGA microcontroller (which normally come without a bootloader). Ensure that you've selected the correct board from the Boards menu before burning the bootloader on the target board. This command also set the right fuses.

v. HELP

Here you find easy access to a number of documents that come with the Arduino Software (IDE). You have access to Getting Started, Reference, this guide to the IDE and other documents locally, without an internet connection. The documents are a local copy of the online ones and may link back to our online website.

- *Find in Reference*

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

vi. SKETCHBOOK

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

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

Tabs, Multiple Files, and Compilation

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

vii. UPLOADING

Before uploading your sketch, you need to select the correct items from the Tools > Board and Tools > Port menus. The boards are described below. On the Mac, the serial port is probably something like /dev/tty. usb modem 241 (for an Uno or Mega 2560 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/ttyACM ,/dev/ttyUSB or similar. Once you've selected the correct serial port and board, press the upload button in the toolbar or select the Upload item from the Sketch menu. Current Arduino boards will reset automatically and begin the upload. With older boards (pre-Diecimila) that lack auto-reset, you'll need to press the reset button on the board just before starting the upload. On most boards, you'll see the RX and TX LEDs blink as the sketch is uploaded. The Arduino Software (IDE) will display a message when the upload is complete, or show an error.

When you upload a sketch, you're using the Arduino bootloader, a small program that has been loaded on to the microcontroller on your board. It allows you to upload code without using any additional hardware. The bootloader is active for a few seconds when the board resets; then it starts whichever sketch was most recently uploaded to the microcontroller. The bootloader will blink the on-board (pin 13) LED when it starts (i.e. when the board resets).

viii. LIBRARIES

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from the Sketch >

Import Library menu. This will insert one or more #include statements at the top of the sketch and compile the library with your sketch. Because libraries are uploaded to the board with your sketch, they increase the amount of space it takes up. If a sketch no longer needs a library, simply delete its #include statements from the top of your code.

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

ix. THIRD-PARTY HARDWARE

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

For details on creating packages for third-party hardware, see the Arduino IDE 1.5 3rd party Hardware specification.

x. SERIAL MONITOR

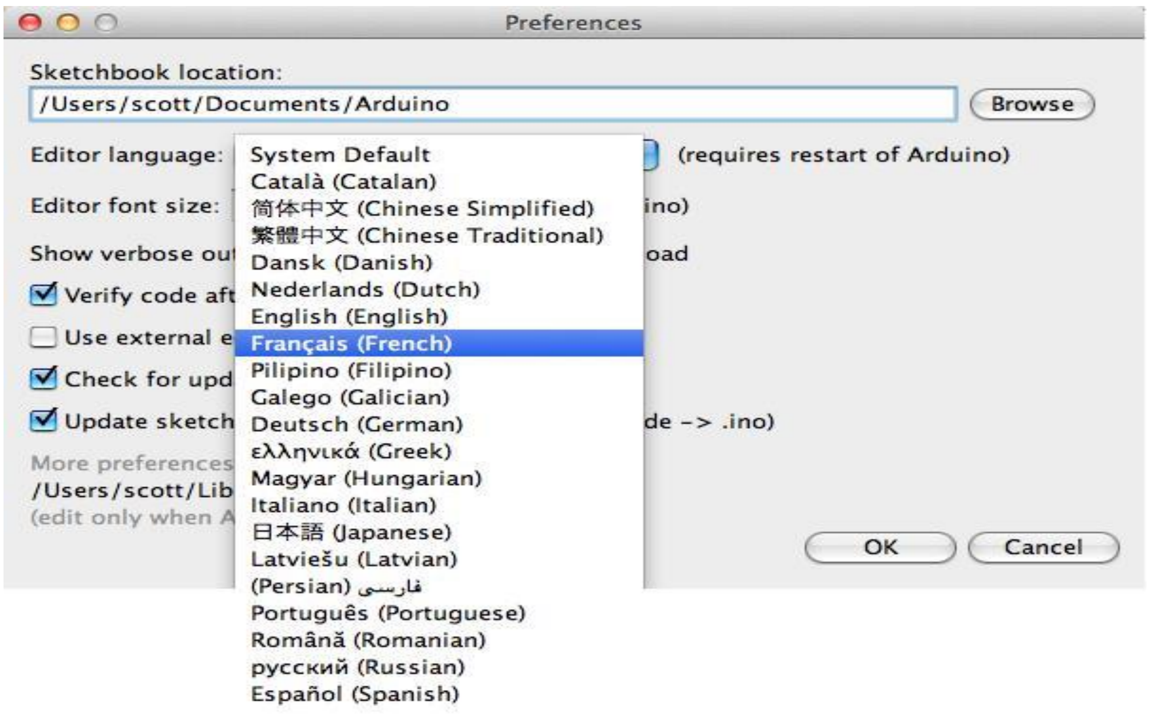
Displays serial data being sent from the Arduino or Genuino board (USB or serial board). To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down that matches the rate passed to Serial.begin in your sketch. Note that on Windows, Mac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

You can also talk to the board from Processing, Flash, Max MSP, etc (see the interfacing page for details).

xi. PREFERENCES

Some preferences can be set in the preferences dialog (found under the Arduino menu on the Mac, or File on Windows and Linux). The rest can be found in the preferences file, whose location is shown in the preference dialog.

7.2. LANGUAGE SUPPORT



Preferences

Sketchbook location:
/Users/scott/Documents/Arduino
Browse

Editor language: (requires restart of Arduino)

- System Default
- Català (Catalan)
- 简体中文 (Chinese Simplified)
- 繁體中文 (Chinese Traditional)
- Dansk (Danish)
- Nederlands (Dutch)
- English (English)
- [✓ Français (French)]
- Pilipino (Filipino)
- Galego (Galician)
- Deutsch (German)
- ελληνικά (Greek)
- Magyar (Hungarian)
- Italiano (Italian)
- 日本語 (Japanese)
- Latviešu (Latvian)
- (Persian) فارسی
- Português (Portuguese)
- Română (Romanian)
- русский (Russian)
- Español (Spanish)

Editor font size: ino)

Show verbose ou oad

☑ Verify code aft

☐ Use external e

☑ Check for upd

☑ Update sketch de -> .ino)

More preferences
/Users/scott/Lib
(edit only when A

OK Cancel

Fig.17. The Arduino Software (IDE) has been translated into 30+ different language

Since version 1.0.1 , the Arduino Software (IDE) has been translated into 30+ different languages. By default, the IDE loads in the language selected by your operating system. (Note: On Windows and possibly Linux, this is determined by the locale setting which controls currency and date formats, not by the language the operating system is displayed in.)

If you would like to change the language manually, start the Arduino Software (IDE) and open the Preferences window. Next to the Editor Language there is a drop down menu of currently supported languages. Select your preferred language from the menu, and restart the software to use the selected language. If your operating system language is not supported, the Arduino Software (IDE) will default to English.

You can return the software to its default setting of selecting its language based on your operating system by selecting System Default from the Editor Language drop-down. This setting will take effect when you restart the Arduino Software (IDE). Similarly, after changing your operating system's settings, you must restart the Arduino Software (IDE) to update it to the new default language.

7.3. BOARDS

The board selection has two effects: it sets the parameters (e.g. CPU speed and baud rate) used when compiling and uploading sketches; and sets and the file and fuse settings used by the burn bootloader command. Some of the board definitions differ only in the latter, so even if you've been uploading successfully with a particular selection you'll want to check it before burning the bootloader. You can find a comparison table between the various boards here.

Arduino Software (IDE) includes the built in support for the boards in the following list, all based on the AVR Core. The Boards Manager included in the standard installation allows to add support for the growing number of new boards based on different cores like Arduino Due, Arduino Zero, Edison, Galileo and so on.

- *Arduino Yùn*

  An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
- *Arduino/Genuino Uno*

An ATmega328 running at 16 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.

- *Arduino Diecimila or Duemilanove w/ ATmega168*

  An ATmega168 running at 16 MHz with auto-reset.
- *Arduino Nano w/ ATmega328*

  An ATmega328 running at 16 MHz with auto-reset. Has eight analog inputs.
- *Arduino/Genuino Mega 2560*

  An ATmega2560 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.
- *Arduino Mega*

  An ATmega1280 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.
- *Arduino Mega ADK*

  An ATmega2560 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.
- *Arduino Leonardo*

  An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
- *Arduino/Genuino Micro*

  An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
- *Arduino Esplora*

  An ATmega32u4 running at 16 MHz with auto-reset.
- *Arduino Mini w/ ATmega328*

An ATmega328 running at 16 MHz with auto-reset, 8 Analog In, 14 Digital I/O and 6 PWM.

- *LilyPad Arduino*

  An ATmega168 or ATmega132 running at 8 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.

- *Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328*

  An ATmega328 running at 16 MHz with auto-reset. Equivalent to Arduino Duemilanove or Nano w/ ATmega328; 6 Analog In, 14 Digital I/O and 6 PWM.

- *Arduino NG or older w/ ATmega168*

  An ATmega168 running at 16 MHz *without* auto-reset. Compilation and upload is equivalent to Arduino Diecimila or Duemilanove
  w/ ATmega168, but the bootloader burned has a slower timeout (and blinks the pin 13 LED three times on reset); 6 Analog In, 14 Digital I/O and 6 PWM.

- *Arduino Robot Control*

  An ATmega328 running at 16 MHz with auto-reset.

- *Arduino Robot MOTOR*

  An ATmega328 running at 16 MHz with auto-reset.

- *Arduino Gemma*

An ATtiny85 running at 8 MHz with auto-reset, 1 Analog In, 3 Digital I/O and 2 PWM

7.4. The Compilation Process

The Arduino code is actually just plain old c without all the header part (the includes and all). When you press the 'compile' button, the IDE saves the current file as arduino.c in the 'lib/build' directory then it calls a makefile contained in the 'lib' directory. This makefile copies arduino.c as prog.c into 'lib/tmp' adding 'wiringlite.inc' as the beginning of it. This operation makes the arduino/wiring code into a proper c file (called prog.c). After this, it copies all the files in the 'core' directory into 'lib/tmp'. These files are the implementation of the various arduino/wiring commands adding to these files adds commands to the language. The core files are supported by pascal stang's procyon avr-lib that is contained in the 'lib/avrlib' directory. At this point the code contained in lib/tmp is ready to be compiled with the c compiler contained in 'tools'. If the make operation is successful then you'll have prog.hex ready to be downloaded into the processor.

CHAPTER 8
ZIGBEE

8.1. Introduction

ZigBee is a specification for a suite of high level communication protocols used to create personal area networks built from small, low power digital radios. ZigBee is based on an IEEE 802.15 standard. Though low powered, ZigBee devices often transmit data over longer distances by passing data through intermediate devices to reach more distant ones, creating a mesh network; i.e., a network with no centralized control or high-power transmitter/receiver able to reach all of the networked devices. The decentralized nature of such wireless ad hoc networks make them suitable for applications where a central node can't be relied upon.

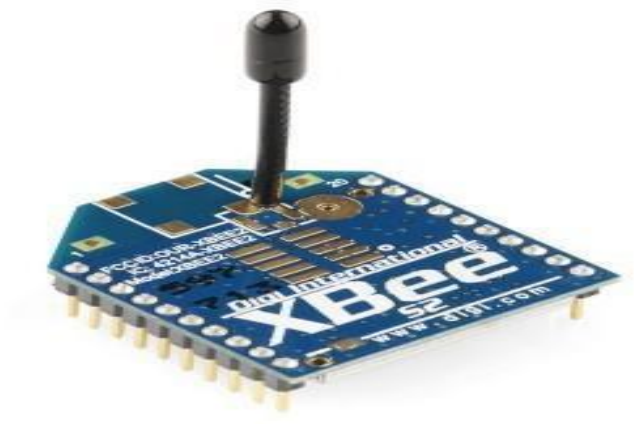


Fig.18. Zigbee Module

8.2. ZigBee Protocol features

- Low duty cycle - Provides long battery life
- Support for multiple network topologies: Static, dynamic, star and mesh
- Direct Sequence Spread Spectrum (DSSS)
- Up to 65,000 nodes on a network
- 128-bit AES encryption – Provides secure connections between devices
- Collision avoidance

- Clear channel assessment
- Retries and acknowledgements
- Support for guaranteed time slots and packet freshness

8.3. Specifications

The core ZigBee specification defines Zig Bee's smart, cost-effective and energy-efficient mesh network. It's an innovative, self-configuring, self-healing system of redundant, low-cost, very low-power and even battery-free nodes that enable Zig Bee's unique flexibility, mobility and ease of use.

8.4. Description

ZigBee is used in applications that require a low data rate, long battery life, and secure networking. ZigBee has a defined rate of 250 kbit/s, best suited for periodic or intermittent data or a single signal transmission from a sensor or input device. Applications include wireless light switches, electrical meters with in-home-displays, traffic management systems, and other consumer and industrial equipment that requires short-range wireless transfer of data at relatively low rates. The technology defined by the ZigBee specification is intended to be simpler and less expensive than other WPANs, such as Bluetooth or Wi-Fi.

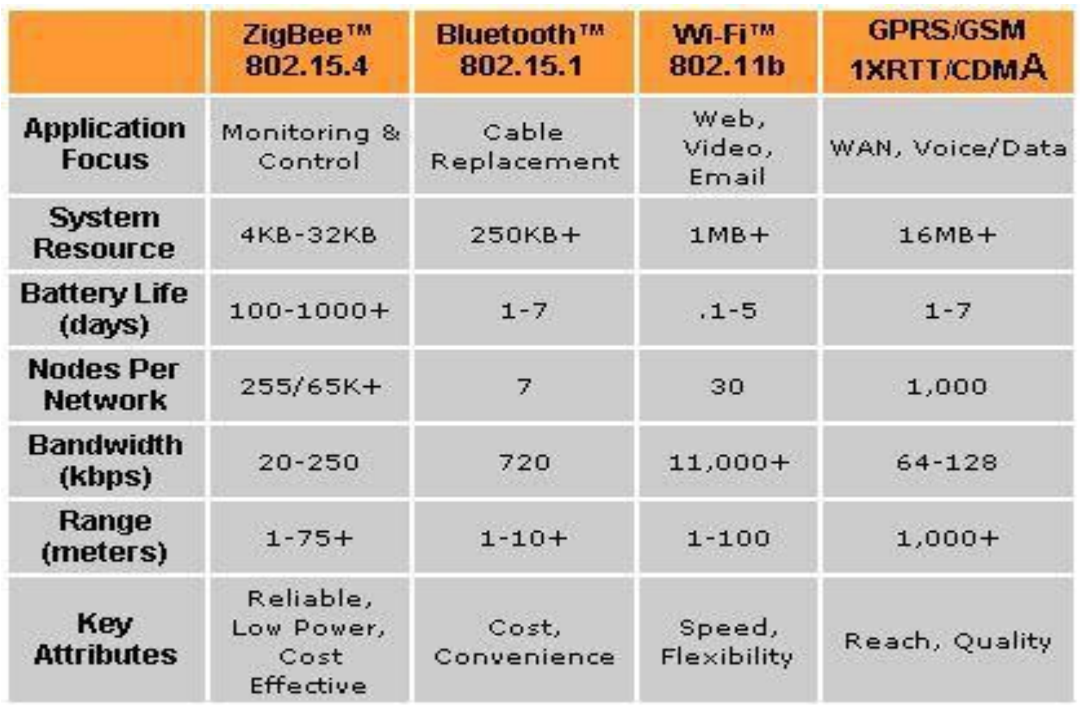
Since ZigBee can be used almost anywhere, is easy to implement and needs little power to operate, the opportunity for growth into new markets, as well as innovation in existing markets, is limitless. Here are some facts about ZigBee:

- With hundreds of members around the globe, ZigBee uses the 2.4 GHz radio frequency to deliver a variety of reliable and easy-to-use standards anywhere in the world.
- Consumer, business, government and industrial users rely on a variety of smart and easy-to-use ZigBee standards to gain greater control of everyday activities.

- With reliable wireless performance and battery operation, ZigBee gives you the freedom and flexibility to do more.

- ZigBee offers a variety of innovative standards smartly designed to help you be green and save money.

Table.4. Comparison between ZigBee, Bluetooth, WiFi, GSM/GPRS



	ZigBee™ 802.15.4	Bluetooth™ 802.15.1	Wi-Fi™ 802.11b	GPRS/GSM 1XRTT/CDMA
Application Focus	Monitoring & Control	Cable Replacement	Web, Video, Email	WAN, Voice/Data
System Resource	4KB-32KB	250KB+	1MB+	16MB+
Battery Life (days)	100-1000+	1-7	.1-5	1-7
Nodes Per Network	255/65K+	7	30	1,000
Bandwidth (kbps)	20-250	720	11,000+	64-128
Range (meters)	1-75+	1-10+	1-100	1,000+
Key Attributes	Reliable, Low Power, Cost Effective	Cost, Convenience	Speed, Flexibility	Reach, Quality

8.5. USES

ZigBee protocols are intended for embedded applications requiring low data rates and low power consumption. The resulting network will use very small amounts of power individual devices must have a battery life of at least two years to pass ZigBee certification.

Typical application areas include :

- Home Entertainment and Control — Home automation, smart lighting, advanced temperature control, safety and security, movies and music
- Industrial control
- Embedded sensing
- Medical data collection
- Smoke and intruder warning
- Building automation

CHAPTER 9
THINGSPEAK

9.1. Introduction

ThingSpeak is an IoT analytics platform service that allows you to aggregate, visualize, and analyze live data streams in the cloud. You can send data to ThingSpeak™ from your devices, create instant visualizations of live data, and send alerts using web services like Twitter® and Twilio®. With MATLAB® analytics inside ThingSpeak, you can write and execute MATLAB code to perform preprocessing, visualizations, and analyses. ThingSpeak enables engineers and scientists to prototype and build IoT systems without setting up servers or developing web software.

ThingSpeak is an IoT analytics platform service that lets you collect and store sensor data in the cloud and develop Internet of Things applications.

• The ThingSpeak service also lets you perform online analysis and act on your data. Sensor data can be sent to ThingSpeak from any hardware that can communicate using a REST API.

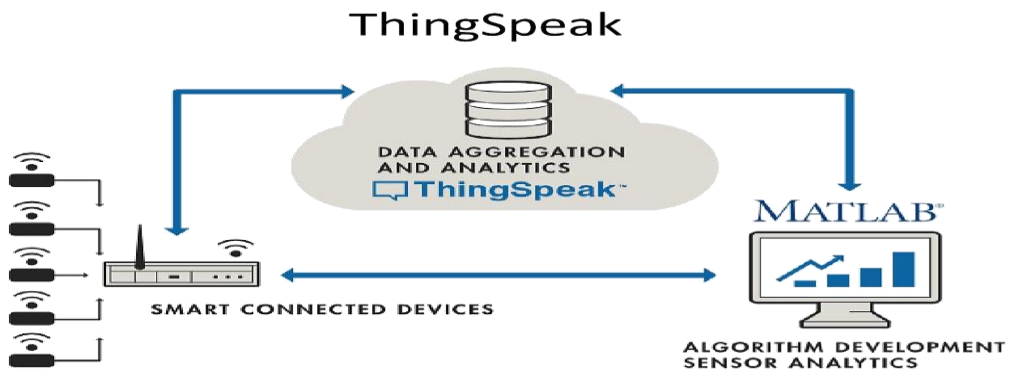


Fig 19. Working of ThingSpeak

9.2. Collect data into new channel

This example shows how to create a new channel to collect analyzed data. You read data from the public ThingSpeak channel 12397 - Weather Station, and write it into your new channel. To learn how to post data to a channel from devices, see the device Examples and the API Reference.

Step 1- Create a Channel

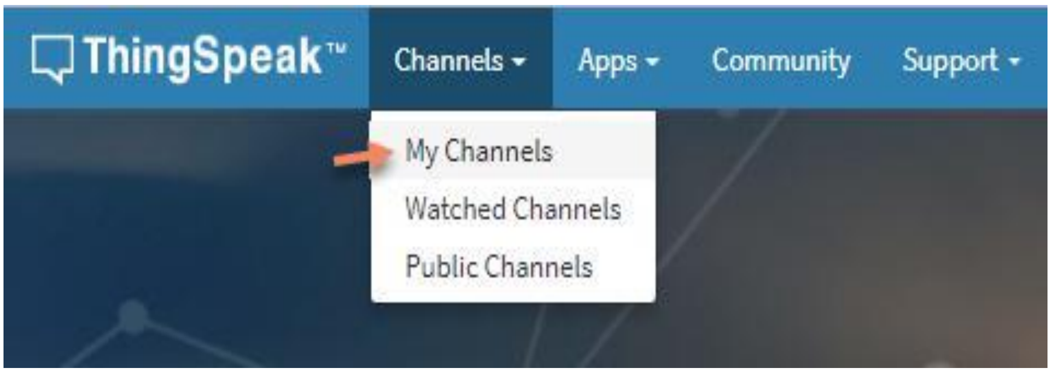
1. Sign In to ThingSpeak™ using your MathWorks® Account, or create a new MathWorks account.
2. Click Channels > MyChannels.
3. 

Fig.20.ThingSpeak Channel

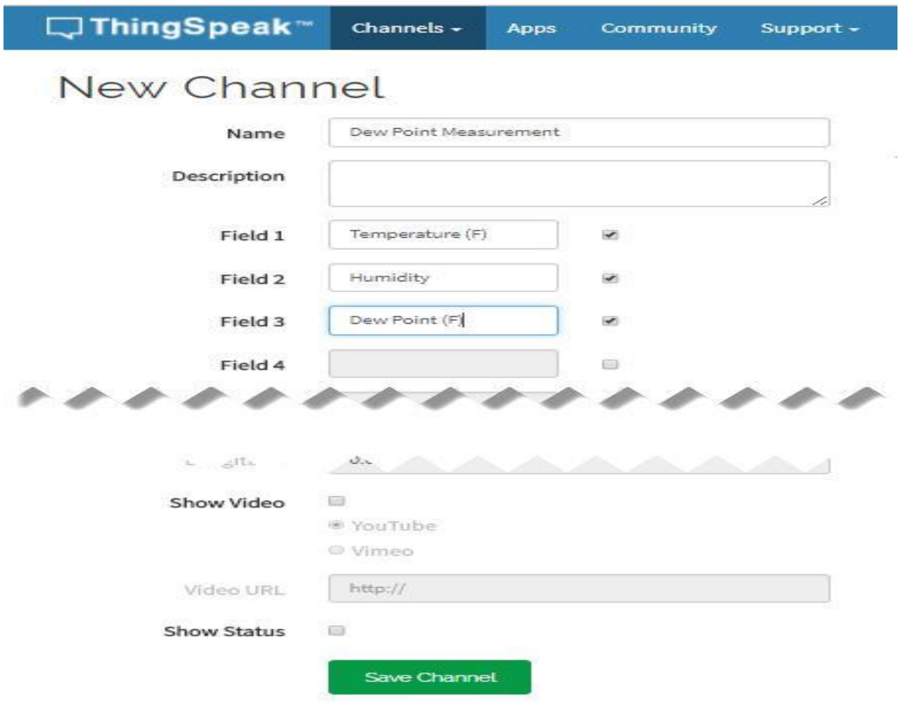
4. On the Channels page, click New Channel.
5. Check the boxes next to Fields 1–3. Enter these channel setting values:

    Name: Dew Point Measurement

    Field 1: Temperature (F)

    Field 2: Humidity

    Field 3: Dew Point

6. 

New Channel

Name	Dew Point Measurement	
Description		
Field 1	Temperature (F)	✓
Field 2	Humidity	✓
Field 3	Dew Point (F)	✓
Field 4		☐
Show Video	☐ [✓ YouTube] Vimeo	
Video URL	http://	
Show Status	☐	

Fig.21. ThingSpeak New Channel creation

7. Click Save Channel at the bottom of the settings.

8. You now see these tabs:

Private View: This tab displays information about your channel that only you can see.

Public View: If you choose to make your channel publicly available, use this tab to display selected fields and channel visualizations.

Channel Settings: This tab shows all the channel options you set at creation. You can edit, clear, or delete the channel from this tab.

Sharing: This tab shows channel sharing options. You can set a channel as private, shared with everyone (public), or shared with specific users.

API Keys: This tab displays your channel API keys. Use the keys to read from and write to your channel.

Data Import/Export: This tab enables you to import and export channel data.

Next Steps

Your channel is available for future use by clicking Channels > My Channels.

In the next example, Analyze Your Data, you use the temperature and humidity data from the public WeatherStation channel to calculate the dew point data. Then you can write the temperature, humidity, and calculated dew point data to Fields 1, 2 and 3, respectively, of your Dew Point Measurement channel. For advanced weather analysis with MATLAB® and ThingSpeak, see Arduino Weather Station Data Analysis on MakerZone.

Step2 - Analyze your Data

This example shows how to read temperature and humidity data from ThingSpeak channel 12397, which collects weather-related data from an Arduino® device. You write the temperature and humidity data into your Dew Point Measurement channel, along with the calculated dew point data. Then use ThingSpeak™ to visualize the results on your channel.

Prerequisite Steps

This example requires that you have already performed these steps:

Sign In either to your MathWorks® Account or ThingSpeak account, or create a new MathWorks account.

Create a Channel as your Dew Point Measurement channel.

Read Data from a Channel

Read the humidity and temperature from the public WeatherStation channel Fields 3 and 4, and write that data to Fields 2 and 1, respectively, of your Dew Point Measurement channel. Dew point is calculated and written to Field 3.

Use MATLAB® Analysis to read, calculate, and write your data.

1. Go to the Apps tab, and click MATLAB Analysis.

2. Click New. Select the Custom template, and click Create.

3. In the Name field, enter Dew Point Calculation.

4. In the MATLAB Code field, enter the following lines of code.

    1. Save the public Weather Station channel ID and your Dew Point Measurement channel ID to variables

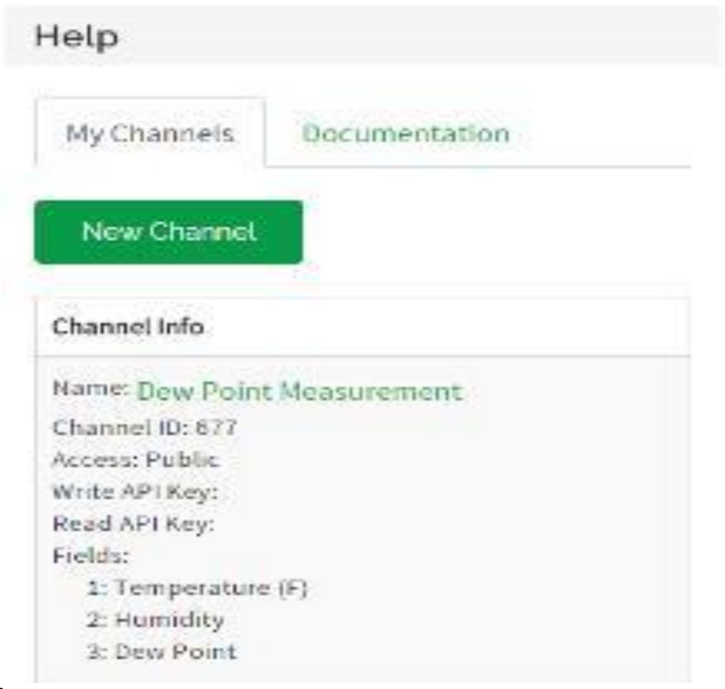
    2. readChId = 12397;

    3. writeChId = 671; % replace with your channel number

    4. Save your Write API Key to a variable.

    5. writeKey = 'F6CSCVKX42WFZN9Y'; % Replace with your channel write key

6. To find your Channel ID and Write API Key, refer to Channel Info on the My Channels tab.

7. 

Channel Info

Name: Dew Point Measurement
Channel ID: 677
Access: Public
Write API Key:
Read API Key:
Fields:
1: Temperature (F)
2: Humidity
3: Dew Point

Fig.22. ThingSpeak API key generation

8. Read the latest 20 points of temperature data with timestamps and humidity data from the public Weather Station channel into variables.

[temp,time] = thingSpeakRead(readChId,'Fields',4,'NumPoints',20);

9. humidity = thingSpeakRead(readChId,'Fields',3,'NumPoints',20);

Calculate the Dew Point:

Add the following MATLAB code to calculate the dew point using temperature and humidity readings:

1. Convert the temperature from Fahrenheit to Celsius.
2. tempC = (5/9)*(temp-32);
3. Specify the constants for water vapor (b) and barometric pressure (c).
4. b = 17.62;
5. c = 243.5;
6. Calculate the dew point in Celsius.

   gamma = log(humidity/100) + b*tempC./(c+tempC);

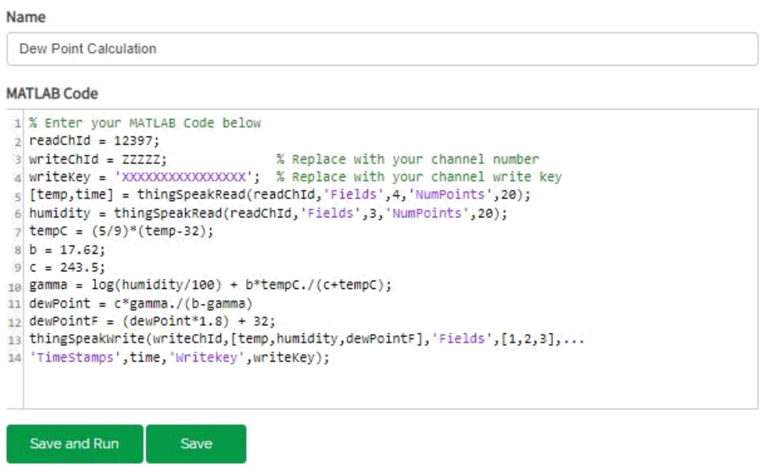
7. dewPoint = c*gamma./(b-gamma)
8. Convert the result back to Fahrenheit.
9. dewPointF = (dewPoint*1.8) + 32;
10. Write data to your Dew Point Measurement channel. This code posts all the available data in one operation and includes the correct timestamps.

thingSpeakWrite(writeChId,[temp,humidity,dewPointF],'Fields',[1,2,3],...

11. 'TimeStamps',time,'Writekey',writeKey);

12. The full block of code now appears as:

13



Dew Point Calculation

MATLAB Code

```
% Enter your MATLAB Code below
readChId = 12397;
writeChId = ZZZZZ;              % Replace with your channel number
writeKey = 'XXXXXXXXXXXXXXXX';  % Replace with your channel write key
[temp,time] = thingSpeakRead(readChId,'Fields',4,'NumPoints',20);
humidity = thingSpeakRead(readChId,'Fields',3,'NumPoints',20);
tempC = (5/9)*(temp-32);
b = 17.62;
c = 243.5;
gamma = log(humidity/100) + b*tempC./(c+tempC);
dewPoint = c*gamma./(b-gamma)
dewPointF = (dewPoint*1.8) + 32;
thingSpeakWrite(writeChId,[temp,humidity,dewPointF],'Fields',[1,2,3],...
'TimeStamps',time,'Writekey',writeKey);
```

Fig.23. Matlab code

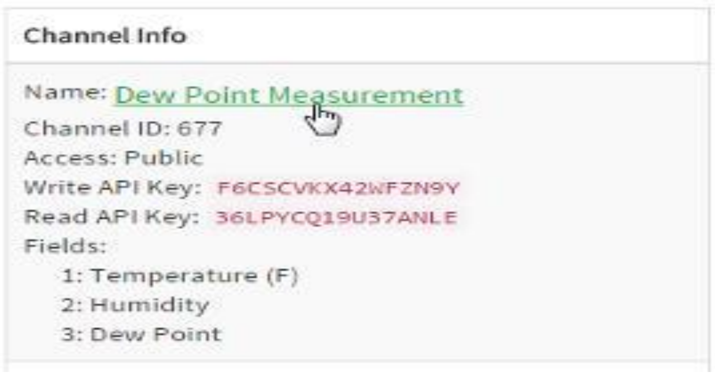
14. See the Full Code

15. Click Save and Run to validate and process your code.

16. Any errors in the code are indicated in the Output field.

17. To see if your code ran successfully, click your Dew Point Measurement channel link in the Channel Info panel.

18.

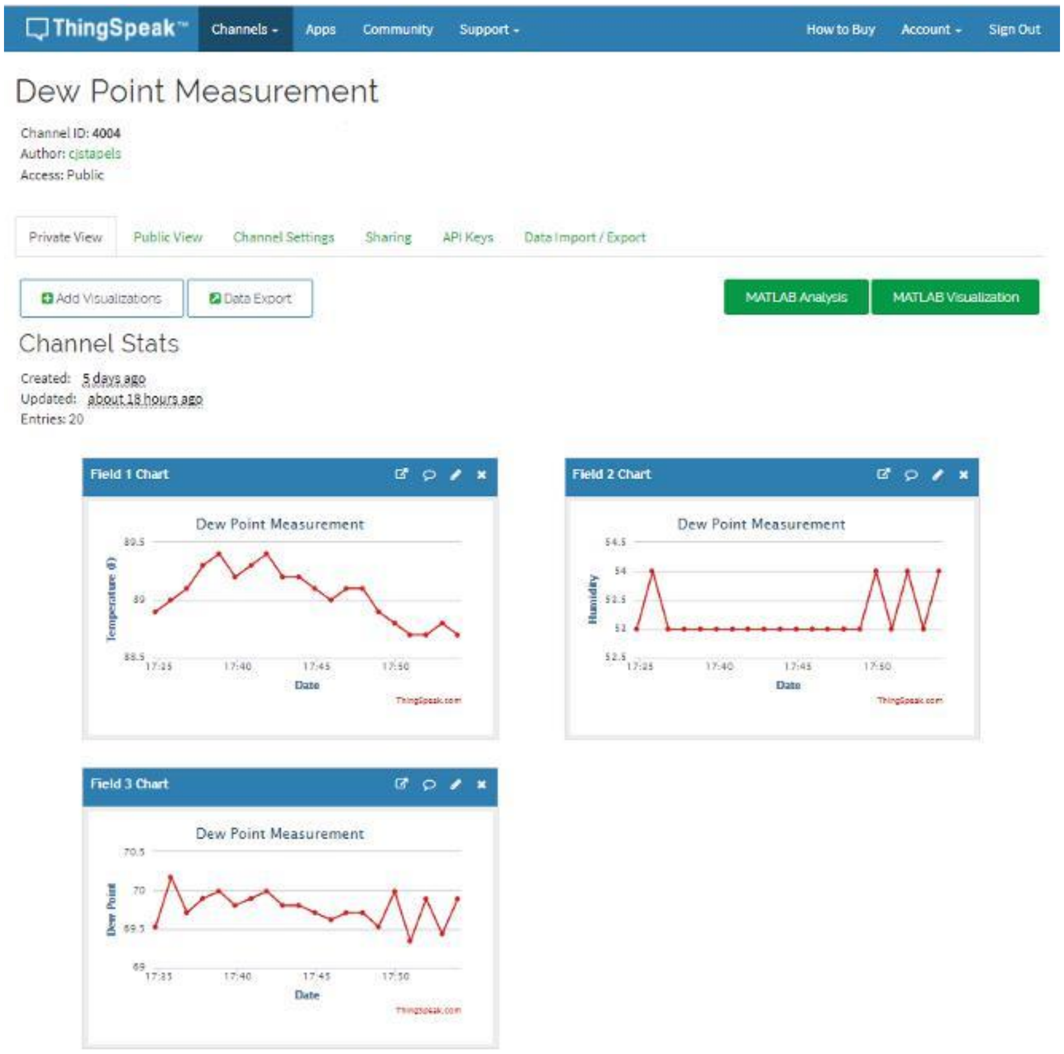


Channel Info

Name: Dew Point Measurement
Channel ID: 677
Access: Public
Write API Key: F6CSCVKX42WFZN9Y
Read API Key: 36LPYCQ19U37ANLE
Fields:
1: Temperature (F)
2: Humidity
3: Dew Point

Fig.24. API keys for write and read

The Dew Point Measurement channel now shows charts with channel data from each Field.



Dew Point Measurement

Channel ID: 4004
Author: cjstapels
Access: Public

Channel Stats

Created: 5 days ago
Updated: about 18 hours ago
Entries: 20

Fig.25. Representation of DATA

Schedule Code :

Use the TimeControl app to schedule the dew point calculation in your MATLAB Analysis code. Schedule it to read data from the weather station every 30 minutes and calculate the dew point.

1. Scroll to the bottom of your MATLAB Analysis Dew Point Calculation page. Click TimeControl to open the app with MATLAB Analysis preselected in the Actions field and the Dew Point Calculation as the Code to execute.

2. Name your new TimeControl Dew Point TC

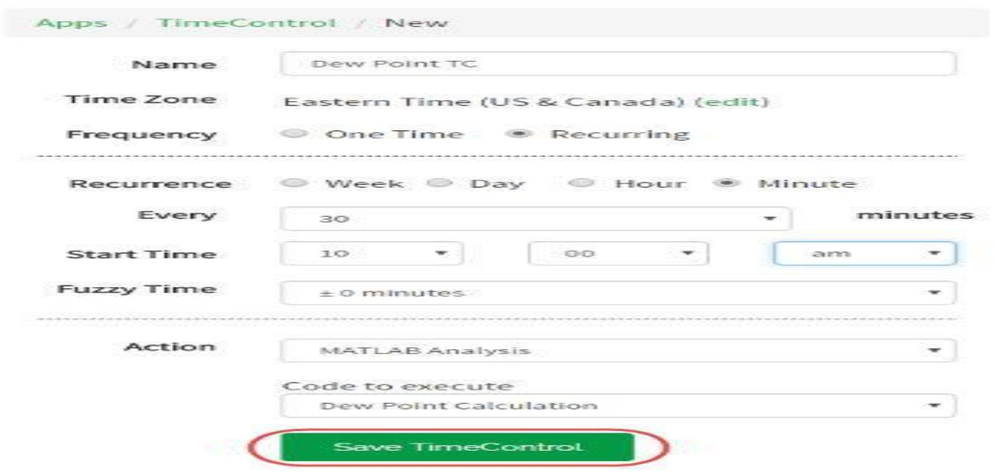
3. Choose Recurring in the Frequency field.

4. Choose Minute in the Recurrence field.

5. Select 30 in the every — minutes field.

6. Keep the Start Time at the default value.

7. Verify that the Action is MATLAB Analysis, and the Code to execute is your Dew Point Calculation.
8. Click Save TimeControl



Apps / TimeControl / New

Name	Dew Point TC
Time Zone	Eastern Time (US & Canada) (edit)
Frequency	One Time [✓ Recurring]
Recurrence	Week Day Hour [✓ Minute]
Every	30 minutes
Start Time	10 : 00 am
Fuzzy Time	± 0 minutes
Action	MATLAB Analysis
Code to execute	Dew Point Calculation

Save TimeControl

Fig.26. Setup - time and date

CHAPTER 10
RESULTS

In this project, we are using esp32 Microcontroller, LCD, Zigbee, Led array, Buzzer. In this project, we can identify the aircraft coming towards the other aircraft. By this the pilot can take certain measurements to avoid collisions. There is a Zigbee which is used for the communication between two aircrafts. This Zigbee is connected to the microcontroller. Whenever the Aircraft 1 is approaching aircraft 2 then a signal will be sent to the microcontroller and a buzzer will be activated, LED is in ON condition and "S1 is nearer to S2" message will be displayed on the LCD screen of Aircraft 1 Whenever both the aircrafts are out of range then "safe mode" will be displayed on the LCD screen.

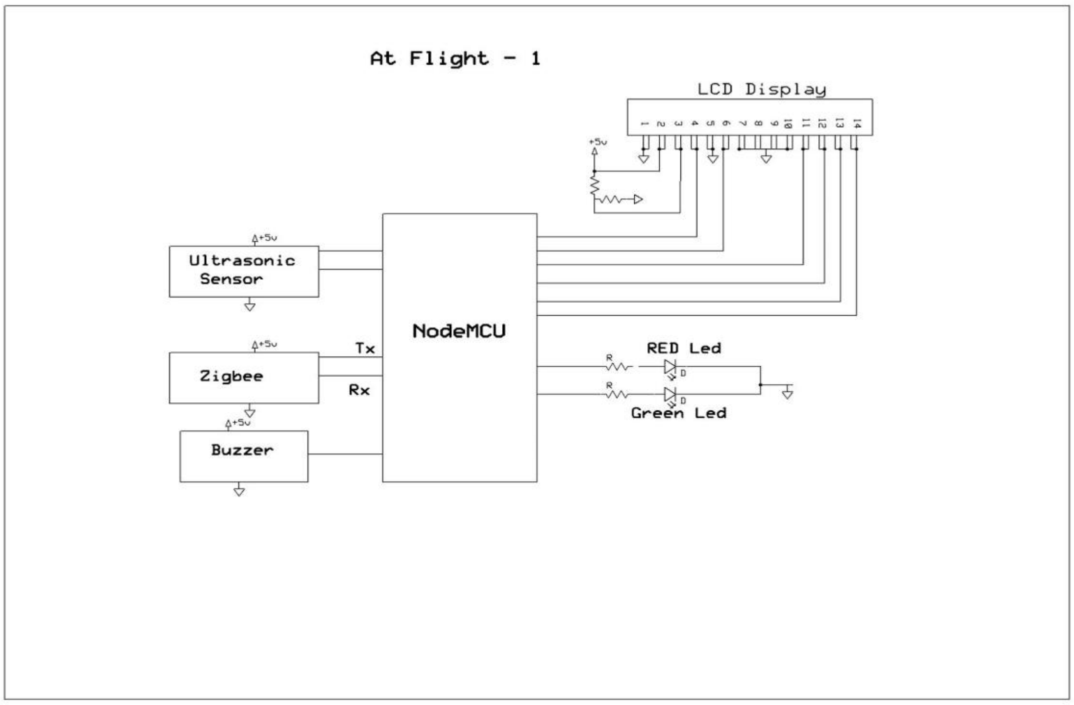


Fig.27. Schematic circuit diagram of aircraft 1

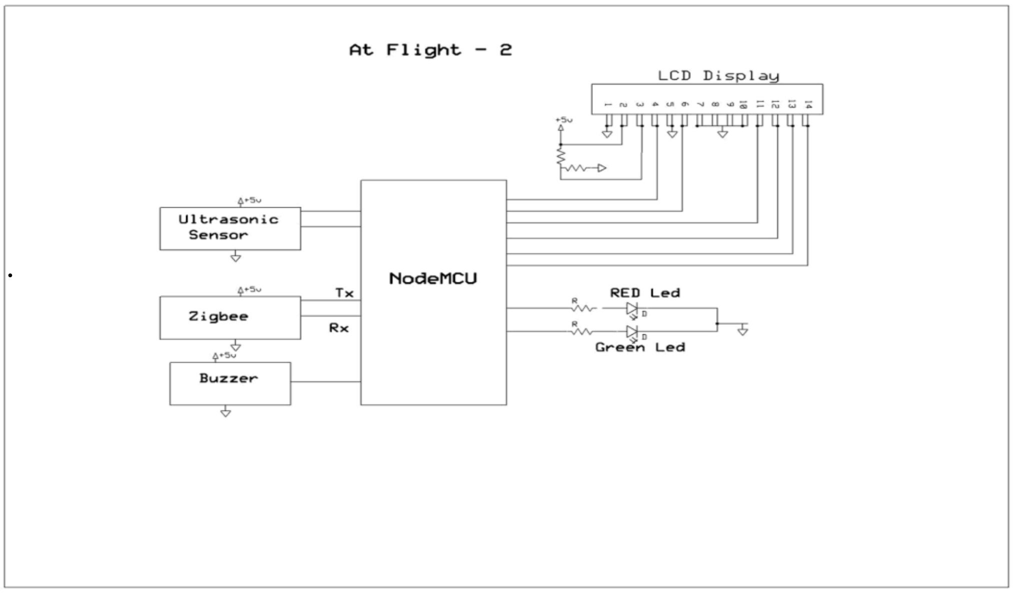


Fig.28. Schematic circuit diagram of aircraft 2

The data of the flight s1 and s2 are collected and transmitted to the ThingSpeak IoT analytics platform. In this platform the data is graphically plotted with the help matlab support. In our project we used three fields to store our data. The field 1 shows the data gathered from flight s1, field 2 shows the data gathered from flight s2 and the field 3 shows the emergency alert. This can be accessed from anywhere in the world if we know there webpage address. The results are as follows

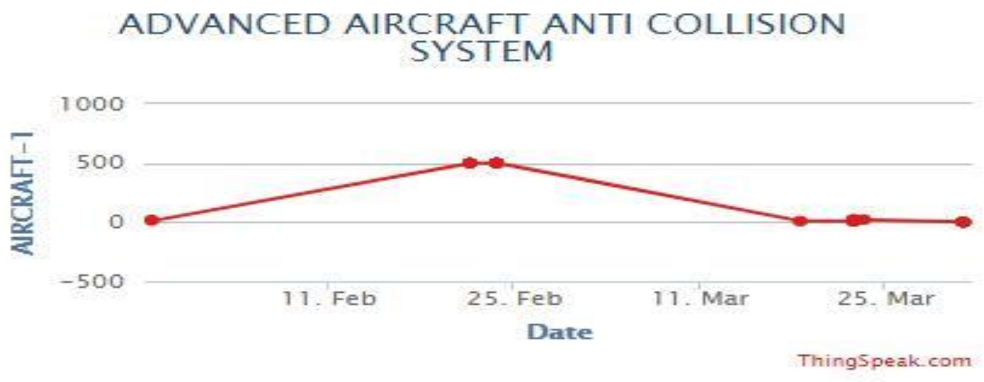


Fig.29. Flight parameters of aircraft 1 (Field 1)

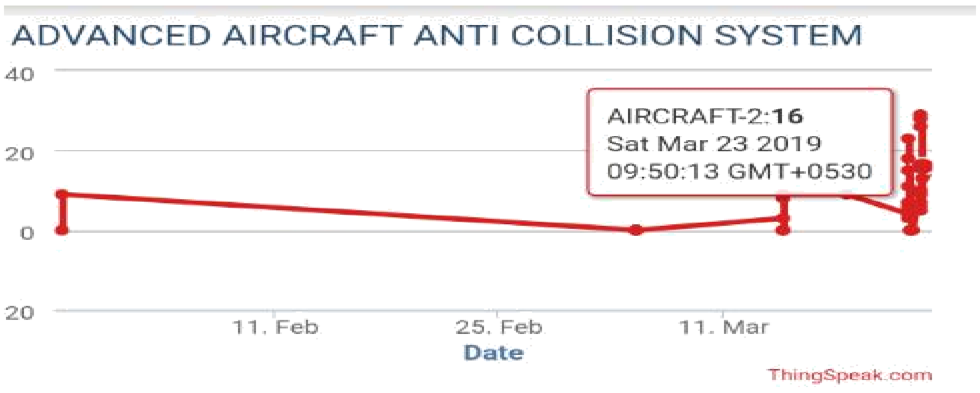


Fig.30. Flight parameters of aircraft 2 (Field 2)

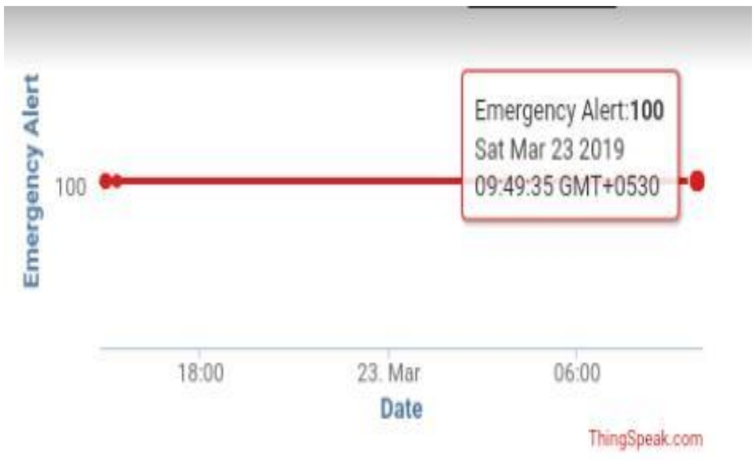


Fig.31. Emergency Alert (Field 3)

CHAPTER 11
CONCLUSION

Hence we have designed and implemented successfully the Aircraft anti-collision system using Zigbee in embedded systems using internet of thing. Use of Zigbee Module in Place of heavy radar system will be helpful in reducing the complexity as well as the maintenance of the system.

By creating the communication between two aircraft can increase the avoidance of collision of aircrafts. Due to the transmission of the aircraft parameters through an IoT module to the base station leads to major improvement in Aircraft safety two aircrafts S1&S2. The advantage of having this kind of system will have high accuracy and it also reduces the man power. This system is not light sensitive, not as sensitive to weather/environmental conditions.

CHAPTER 12
REFERENCES

[1] academia.edu, The NODEMCU Microcontroller- Architecture, Programming and Applications

[2] Raj Kamal (2004), " Embedded Systems. Architecture, Programming and design, International Edition ", New Delhi:McGraw-Hill.

[3]Barr, Micheal(1999),Programming Embedded system C , Sebastopol, C.A:

O Reilly.

[4]Thingspeak official website https://thingspeak.com/ [5]2011
IEEE/AIAA 30th Digital Avionics Systems Conference

[6]Sugano, Masashi, et al. "Indoor localization system using RSSI measurement of wireless sensor network based on ZigBee standard." Target 538 (2007): 050.

[7] Borenstein, Johann, and Yoram Koren. "Obstacle avoidance with ultrasonic sensors." Robotics and Automation

[8] Drumm, A. C., etal. "Remotely Piloted Vehicles in civil airspace: requirements and analysis methods for the traffic alert and collision avoidance system (TCAS) and see-and-avoid systems." Digital Avionics Systems Conference, 2004. DASC 04. The 23rd. Vol. 2. IEEE, 2008.

CHAPTER 13

APPENDIX

```
// https://thingspeak.com/channels/689629 -- PUBLIC VIEW PAGE



#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <LiquidCrystal.h>
#include <String.h>

LiquidCrystal lcd (D0,D1,D2,D3,D4,D5);

// WiFi parameters to be configured
const char* ssid = "project12"; // Hoofdlettergevoelig
const char* password = "project123456"; // Hoofdlettergevoelig

const char* http_site = "api.thingspeak.com";
const int http_port = 80;
long duration = 0;
unsigned int distance = 0;

String link =
"http://api.thingspeak.com/update?api_key=WXGBY1HU6V57QD3F&fi
eld2=";
String iot = "\0";

const int trigPin = 12;  //D6
const int echoPin = 13;  //D7



#define gled 15
#define rled 1

void setup(void) {
  Serial.begin(9600);
Serial.println("power up");
lcd.begin(16,2);

pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
pinMode(rled, OUTPUT);
pinMode(gled, OUTPUT);

lcd.clear();
lcdstring(0,0,"   WELCOME TO   ");
lcdstring(0,1,"ADVANCED AIRCRAFT");
delay(2000);
lcd.clear();
lcdstring(0,0," ANTI-COLLISION ");
lcdstring(0,1,"     SYSTEM     ");
delay(1500);




  Serial.print("power up");
  WiFi.begin(ssid, password); // Connect to WiFi

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
    lcd.clear();
    lcdstring(0,0," CONNECTING  TO ");
lcdstring(0,1,ssid);
delay(1500);
  }

  // Verbonden.
  Serial.println("OK!");

  // Access Point (SSID).
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());

    lcd.clear();
lcdstring(0,0,"WI-FI CONNECTED");
lcdstring(0,1,"***************");
delay(500);

  // IP adres.
  Serial.print("IP: ");
  Serial.println(WiFi.localIP());

  // Signaalsterkte.
  long rssi = WiFi.RSSI();
  Serial.print("Signaal sterkte (RSSI): ");
  Serial.print(rssi);
  Serial.println(" dBm");
  Serial.println("");

delay(500);
digitalWrite(gled,HIGH);
digitalWrite(rled,HIGH);
  delay(500);
digitalWrite(gled,LOW);
digitalWrite(rled,LOW);
delay(500);
digitalWrite(gled,HIGH);
digitalWrite(rled,HIGH);
  delay(500);
digitalWrite(gled,LOW);
digitalWrite(rled,LOW);
}

void loop() {

Serial.println("loop Started");

digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
```

```
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distance= duration*0.034/2;
Serial.print("Distance: ");
Serial.println(distance);
delay(100);

lcd.clear();
lcdstring(0,0,"DISTANCE = ");
lcdstring(11,0,String(distance));
delay(200);

if (distance <30)
{

digitalWrite(gled,LOW);
digitalWrite(rled,HIGH);
delay(50);
 // lcd.clear();
  lcdstring(0,1,"AIRCRAFT DETECTED");
  //lcdstring(0,1,"sending to server");
  delay(1500);
  send_data ();
}
else
{

digitalWrite(rled,LOW);
digitalWrite(gled,HIGH);
delay(50);
}

}

void send_data () {

lcd.clear();
lcdstring(0,0,"AIRCRAFT DETECTED");
lcdstring(0,1,"sending to server");
delay(100);

iot = link + String(distance);



 if(WiFi.status()== WL_CONNECTED){ //Check WiFi connection status

   HTTPClient http; //Declare an object of class HTTPClient
   //
http.begin("http://orangewebtools.com/Irrigation/moisture.php?a=32");
//Specify request destination

 Serial.println(iot);
   http.begin(iot);
    http.begin("\n");
    int httpCode = http.GET(); //Send the request
    if (httpCode > 0) { //Check the returning code
```

```
      String payload = http.getString(); //Get the request response payload
      Serial.println(payload); //Print the response payload
    }
    http.end(); //Close connection
  } else {
    Serial.println("Error in WiFi connection");
  }
  lcd.clear();
lcdstring(0,0,"  DATA SENDING  ");
lcdstring(0,1,"  --COMPLETED--  ");
delay(1);
delay(3000); //Send a request every 30 seconds

}
void lcdstring (unsigned int a, unsigned int b, String data)
{
  // lcd.clear();
  lcd.setCursor(a, b);
  lcd.print(data);
}




}
```