

A
Project
Report on
**IoT based smart garbage monitoring system in cities-To
detect metal or non-metal**

Submitted in partial fulfilment of Requirements for the award of

**BACHELOR OF ENGINEERING IN
ELECTRONICS AND COMMUNICATIONS**

CHINNALA VENKATESHWARLU -38130041
CHINNI NAGA BADRI KISHORE -38130042

Under the guidance of
Dr. I.REXILINE SHEEBA, M.E.Ph.D
ASSOCIATE PROFESSOR

**Department of electronics and communications and
engineering**

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

CHENNAI, TAMILNADU – 600119

MARCH-2022





SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of CH.Venkateshwarlu (38130041) and CH.Naga Badri Kishore (38130042) who carried out the project entitled "**IoT based smart garbage monitoring system in cities-To detect metal or non-metal**" under my supervision from December 2022 to May 2022.

Internal Guide

Dr. I.REXILINE SHEEBA, M.E.Ph.D

Head Of The Department

Dr. Ravi Kumar M.E., Ph.D.

Submitted for Viva-voce Examination held on.....

Internal Examiner

External Examiner

DECLARATION

I **CHINNALA VENKATESHWARLU** and **CHINNI NAGA BADRI KISHORE** we are here by declare that the project report entitled IoT based smart garbage monitoring system in cities-To detect metal or non-metal done by us under the guidance of **Dr. I.REXILINE SHEEBA** ,associate professor, **M.E.Ph.D**

Date: 01/03/2022.

Place: Chennai

signature of the candidate

CH. Venkateshwarlu

CH. Naga Badri kishore

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of SATHYABAMA for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to Dr. **M Nandita**, Dean, School of Electronics and Dr. **Ravi Kumar**, Head of the Department, Dept. of Electronics for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Guide **Dr.I. REXILINE SHEEEBA**, **associate professor** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Electronics who were helpful in many ways for the completion of the project.

ABSTRACT

Waste management becomes a challenge faced not only by the developing nations, but also the developed and advanced countries. The overall waste management involves three main types of entities: users who generate waste, waste collectors & city administration, stakeholders.

It is very important to have a smart way of managing waste, so that not only the waste status is notified in-time when to be collected, but also, all the stakeholders are made aware in timely fashion that what type of waste in what quantity is coming up at what particular time.

To introduce IoT Garbage monitoring system to separate metallic and non-metallic components etc. To stop overflow of garbage in bins. The IoT Garbage monitoring system is a very innovative system which help to keep the cities clean. The implementation of smart garbage management system using sensors, microcontrollers, WIFI module and with the help of motors, it assures the cleaning of trash bins soon when the garbage level reaches its maximum.

This reduce the unhygienic condition and avoids health issues, here in case of wet waste, we are converting it to manure, hence total number of trips of garbage collection vehicle is reduced, thereby overall expenditure associated with the garbage collection will also be minimize.

TABLE OF CONTENT

CHAPTER	TITLE	PAGE NO
1.	INTRODUCTION.....	1
2.	LITERATURE SURVEY	2
3.	AIM AND OBJECTIVES	5
3.1	EXISTING SYSTEM.....	5
	PROPOSED SYSTEM.....	5
3.2	BLOCK DIAGRAM.....	6
4.	EXPERIMENTAL OR MATERIALS	7
	AND METHODS USED	
4.2	NODE MCU.....	10
5.	ARDUINO UNO & IT'S PROGRAMMING	35
5.2	ARDUINO SOFTWARE IDE.....	36
5.3	TOOLS.....	41
5.4	LIBRARIES.....	44
6.	UBIDOTS EXPLORER.....	49
7.	CONCLUSION.....	58
8.	REFERENCES.....	59

CHAPTER-I

1.1 INTRODUCTION:

As the world is in a stage of up gradation, there is one stinking problem [1] we have to deal with. Garbage! In our daily life, we see the pictures of garbage bins being overfull and all the garbage spills out. This leads to the number of diseases as large number of insects and mosquitoes breed on it. A big challenge in the urban cities is solid waste management not only in India but for most of the countries in the world. Hence, such a system has to be build which can eradicate this problem or at least reduce it to the minimum level. The project gives us one of the most efficient ways to keep our environment clean and green. The smart city [2] concept is still new in India, although it has received a lot of attention in few years when our present prime minister gave the idea of building 100 smart cities throughout India. Now, with the upcoming large number of smart cities, large numbers of responsibilities are also required to be fulfilled. The prime need of a smart lifestyle begins with cleanliness and cleanliness begins with dustbin. A society will get its waste dispatched properly only if the dustbins are placed well and collected well. The main problem in the current waste management system in most of the Indian cities is the unhealthy status of dustbins. In this paper we have tried to upgrade the trivial but vital component of the urban waste management system, i.e. dustbin. Now with the rise of technology it is high time that we should use technology for waste management systems [2]. As we have seen that technology with analytics has made the world a better place to live by its application in the field of genetics, insurance, marketing, engineering, banking etc in past many years. So, in this paper we have integrated analytics and electronics in order to create optimal changes in the conventional methodology of waste collection with the large amount of data that is being produced by the smart bin networks.

CHAPTER -2

2.1 LITERATURE SURVEY

As we know, the large bins are present in each locality and serve as the centre point of garbage of that particular locality. The garbage collection team collects the garbage from these central bins in their trucks. Our model of hardware is going to be applied in these central bins and thus making them smart bins. For this we have divided the dustbin into three different levels according to the level of garbage filled. Accordingly, the text messages indicating the levels are being sent to the central office, which acts as a data warehouse for all the level data being sent by the different bins. The central office of waste management department now will be able to track the level of every dustbin getting filled up just by sitting in their office at real time. This information will now guide them efficiently to take up the action of sending the trucks to empty the dustbin whose levels are significant. The hardware has used some very basic electronic components to make out the complete system at the best possible cost. The built hardware can be implemented at any dustbin irrespective of its size and height and thus is portable with any bins.

This is not an original idea, for the implementation of smart garbage bin; the idea has existed for many years, After the IoT field finding its grip in our lives. This is, however an original plan for designing a smart garbage bin with weight sensor, IR sensor and Wi-Fi module for transmission of data. [1]. A State of the Art review on Internet of Things by P. Suresh, Vijay. Daniel, R.H. Aswathy, Dr. V. Parthasarathy. It gave the idea of IoT subject and addition details about IoT. The proper smart environment and various applications.

[2].Internet of Things: Challenges and state-of-the-art solutions in Internet-scale Sensor Information Management and Mobile analytics by Arkady Zaslavsky, Dimitrios Georgakopoulos. This paper gave us the details about mobile analysis and sensor information management that will help in data segregation of various dustbins. [3]Top-k Query based dynamic scheduling for IoT-enabled small city waste collection by Theodoros Anagnostopoulos, Arkady Zaslavsky, Alexey Medvedev, Sergei Khoruzhnicov. It gave us the concept of dynamic scheduling required for the cleaning of dustbin and the Top-k query led us to priority based cleaning of dustbins [4]City Garbage collection indicator using RF(Zigbee) and GSM technology. This paper gave the details for the module required for the

transmission of the data to the receiver side and also the main channel follow of the project. Initially we used GSM technology for our project but later on decided to us Wi-Fi module for the ease of data transmission.

[2].Internet of Things: Challenges and state-of-the-art solutions in Internet-scale Sensor Information Management and Mobile analytics by Arkady Zaslavsky, Dimitrios Georgakopoulos. This paper gave us the details about mobile analysis and sensor information management that will help in data segregation of various dustbins. [3]Top-k Query based dynamic scheduling for IoT-enabled small city waste collection by Theodoros Anagnostopoulos, Arkady Zaslavsky, Alexey Medvedev, Sergei Khoruzhnicov. It gave us the concept of dynamic scheduling required for the cleaning of dustbin and the Top-k query led us to priority based cleaning of dustbins [4]City Garbage collection indicator using RF(Zigbee) and GSM technology. This paper gave the details for the module required for the transmission of the data to the receiver side and also the main channel follow of the project. Initially we used GSM technology for our project but later on decided to us Wi-Fi module for the ease of data transmission.

Literature survey:

Title 1:

Smart Waste Management System using IOT

Author: Tejashree Kadus, Pawankumar Nirmal, Kartikey Kulkarni

The paper is based on the concept of Automation used in waste management system under the domain of Cleanliness and Hygiene. Dumping garbage onto the streets and in public areas is a common synopsis found in all developing countries and this mainly end up affecting the environment and creating several unhygienic conditions. In order to deal with these problems Smart netbin is an ideology put forward which a combination of hardware and software technologies is i.e. connecting Wi-Fi system to the normal dustbin in order to provide free internet facilities to the user for a particular period of time.

Title 2:

Waste Management Improvement in Cities using IoT

Author: Shivam Jagtap,Aditya Gandh,Raviraj Bochare,Ashwinkumar Patil.

Garbage collection is one of the most critical problems faced by Municipal Corporation. While implementing the waste management in cities the biggest challenge is the management of waste in cost optimal way with high performance. The current process of collecting the waste, separating it and transporting the containers everyday which is a complicated process. This paper deals with the concept of waste management and the smart system for waste management with higher benefits to the society.

Title 3: Smart Waste Management using Internet-of-Things

Author: Chitluri Sai Srikanth, Tadvaka Bhupathi Rayudu, Javvaji Radhika, Raju Anitha

To avoid each such condition, we plan to propose a reaction for this issue "Sharp Garbage Bin", which will alert and set up the maintained individual when the waste compartment will fill. By then message will be send to the gotten a handle on individual to amass the hardship from the specific territory. The announced individual will send the message from his web application to the waste specialists by sending a SMS. In this undertaking we use strain check to know the burglary of the report. This will diminish the flood of the waste compartment and suitably keeping nature clean.

CHAPTER -3

3.1 Aim and Objectives

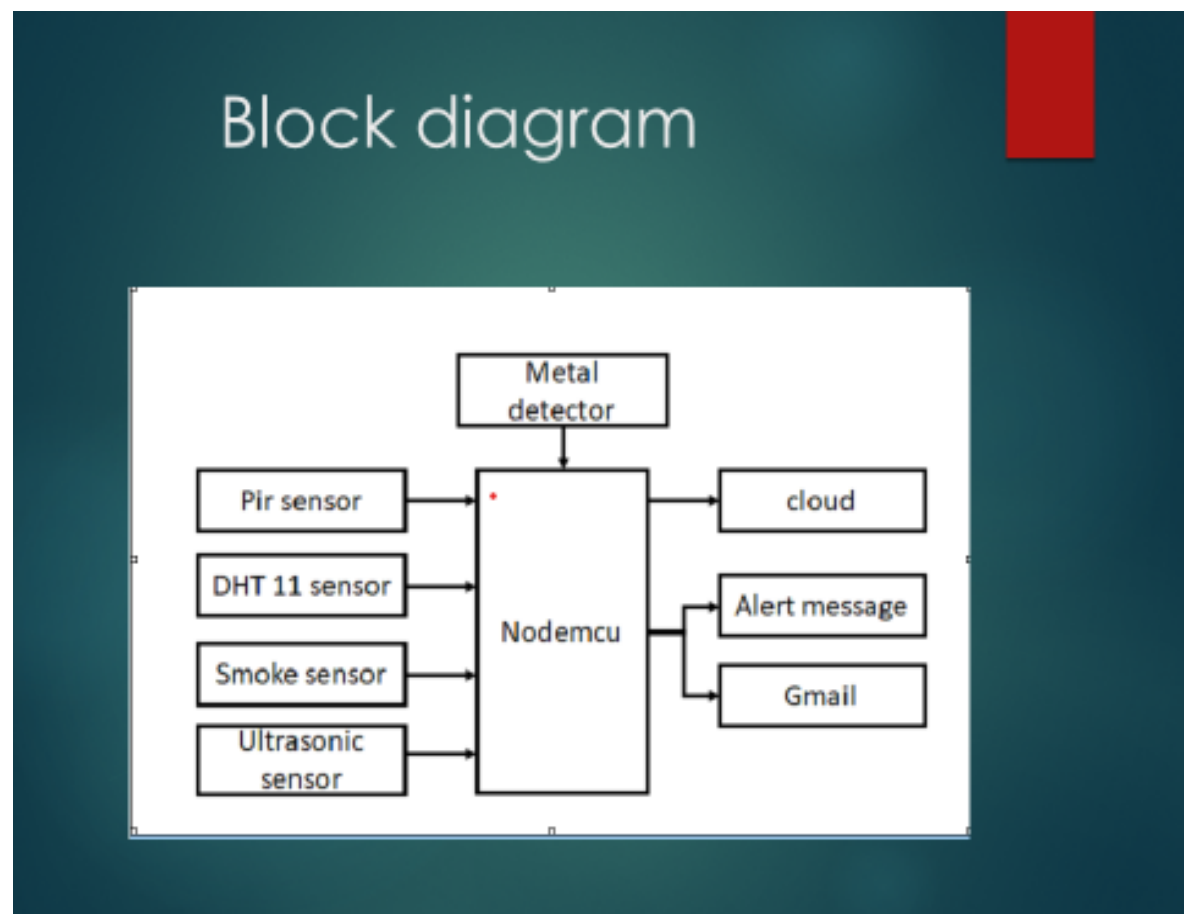
EXISTING SYSTEM:

- In a smart bin using some kind of sensor like IR and Ultrasonic sensor is used to identify the bin is filled or not.
- If it is filled means it sends message to garbage management contractor or office in IOT technology or GSM based technology.
- This is already existing technology.
- This bin can separates dry waste, wet waste and e-waste.

PROPOSED SYSTEM;

- In this proposed system it divides the waste into dry waste, wet waste, e-waste and metal or non metal.
- Using ultrasound sensor used to detect the level of waste in garbage bin. When the bin filled with 50% then alert messages send to the ward member, and it will filled with 90% the alert message send to the government.
- The bin is continuously monitoring the level of garbage if it's full means it gives alert to Government Corporation. When a corporation separate the waste for recycle process.
- After the recycling process is completed when a government organization gives notification to some corporate company, NGO etc.
- It is also have another advantage to reduce environmental pollution and it increase the financial support in government organization.

3.2 Block diagram:



HARDWARE USED:

- Nodemcu

- ultrasonic sensor
- DHT11 sensor
- smoke sensor
- metal detector

SOFTWARE USED:

Arduino c

CHAPTER -4

4.1 Communication

Microcontrollers depend on a host computer for developing and compiling programs. The software used on the host computer is known as an integrated development environment, or IDE. For the Arduino, the development environment is based on the open source Processing platform (www.processing.org) which is described by its creators as a “programming language and environment for people who want to program images, animation, and interactions.” The Arduino programming language leverages an open source project known as Wiring (wiring.org.co). The Arduino language is based on good old- fashioned C. If you are unfamiliar with this language, don’t worry; it’s not hard to learn, and the Arduino IDE provides some feedback when you make mistakes in your programs.

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The

RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A Software Serial library allows for serial communication on any of the Uno's digital pins. The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. For SPI communication, use the SPI library.

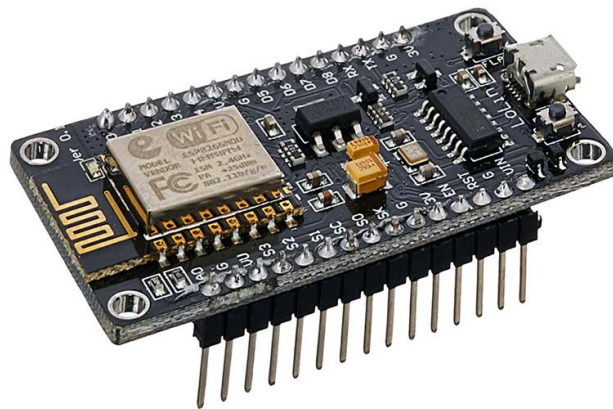
As you go through the list of programming statements available in the Arduino IDE (choose Help->Reference), you might think there isn't much power for doing things like running servos, operating stepper motors, reading potentiometers, or displaying text on an LCD. Like most any language based on C, the Arduino supports the notion of "libraries" code

Repositories that extend core programming functionality. Libraries let you re-use code without having to physically copy and paste it into all your programs. The standard Arduino software installation comes with several libraries you may use, and you can download others from the Arduino support pages and from third-party websites that publish Arduino library code. A good example of a library you'll use with the Robot and likely many other robot projects is Servo. This library allows you to connect one or more hobby R/C servos to the Arduino's digital I/O pins. The Servo library comes with the standard Arduino installation package Library->Servo. This adds the line `#include <Servo.h>` Which tells the Arduino IDE that you wish to include the Servo library in your sketch. With the functionality of the library now available to you, you can use its various functions to control one or more servos. For example, you can use the write function to rotate a servo to a specific position, from 0 to 180 degrees. The following code `myServo.write(90);` Moves a servo to its midpoint, or 90 degree position. Structurally, Arduino sketches are very straightforward and are pretty easy to read and understand. The Arduino program contains two main parts: `setup ()` and `loop ()`. These are programming functions that do what their names suggest: `setup ()` sets up the Arduino hardware, such as specifying which I/O lines you plan to use, and whether

They are inputs or outputs. The loop () function is repeated endlessly when the Arduino is operating.

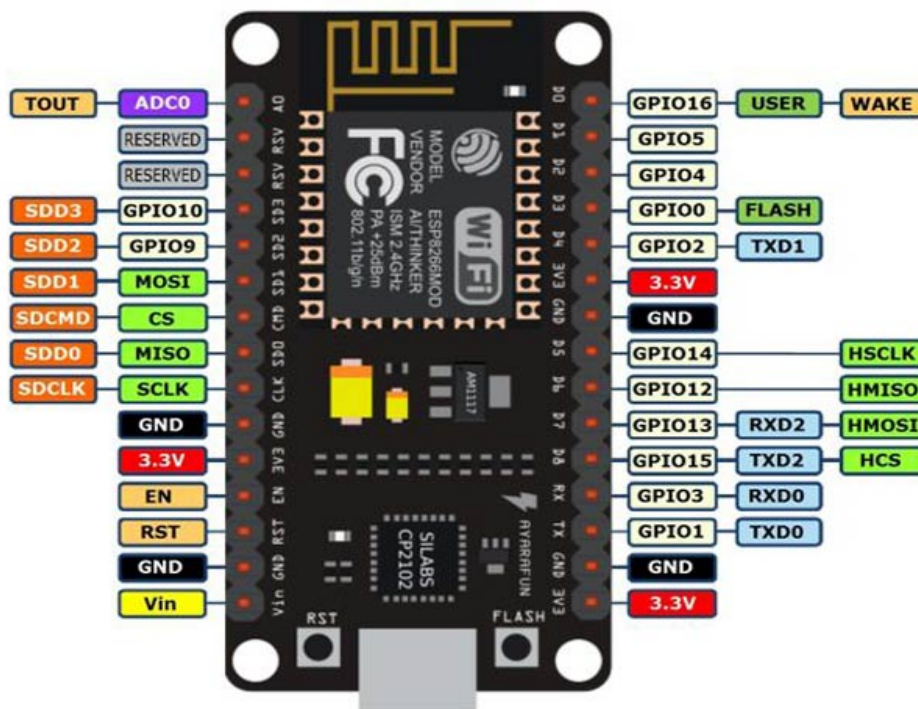
Arduino IDE (Integrated development environment) is used to write the program and dump into the Arduino board

NodeMCU ESP8266



Pin Category	Name	Description
Power	Micro-USB, 3.3V, GND, Vin	Micro-USB: NodeMCU can be powered through the USB port 3.3V: Regulated 3.3V can be supplied to this pin to power the board GND: Ground pins Vin: External Power Supply
Control	EN, RST	The pin and the button resets the microcontroller

Pins		
Analog Pin	A0	Used to measure analog voltage in the range of 0-3.3V
GPIO Pins	GPIO1 to GPIO16	NodeMCU has 16 general purpose input-output pins on its board
SPI Pins	SD1, CMD, SD0, CLK	NodeMCU has four pins available for SPI communication.
UART Pins	TXD0, RXD0, TXD2, RXD2	NodeMCU has two UART interfaces, UART0 (RXD0 & TXD0) and UART1 (RXD1 & TXD1). UART1 is used to upload the firmware/program.
I2C Pins		NodeMCU has I2C functionality support but due to the internal functionality of these pins, you have to find which pin is I2C.



4.2 Node MCU ESP8266

NodeMCU is an open-source Lua based firmware and **development board** specially targeted for IoT based Applications. It includes firmware that runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module.

NodeMCU Development Board Pinout Configuration

NodeMCU ESP8266 Specifications & Features

- Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
- Operating Voltage: 3.3V
- Input Voltage: 7-12V
- Digital I/O Pins (DIO): 16
- Analog Input Pins (ADC): 1
- UARTs: 1
- SPIs: 1
- I2Cs: 1
- Flash Memory: 4 MB
- SRAM: 64 KB
- Clock Speed: 80 MHz
- USB-TTL based on CP2102 is included onboard, Enabling Plug n Play
- PCB Antenna
- Small Sized module to fit smartly inside your IoT projects

Other Espressif Boards

ESP8266, ESP12E, ESP32

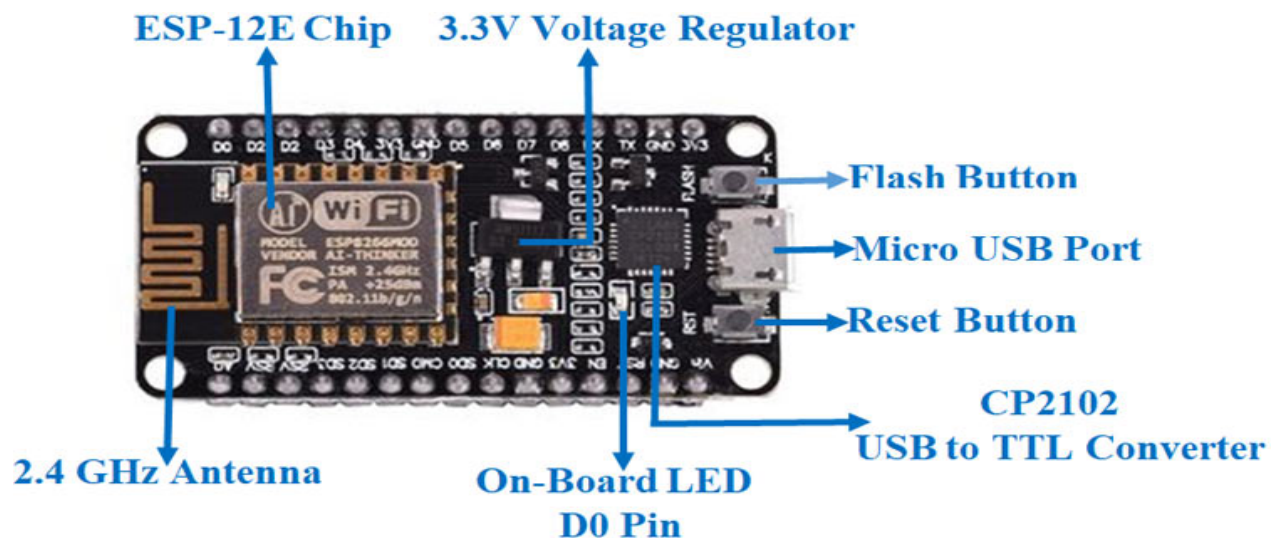
Other Development Boards

Arduino, Raspberry Pi, PIC Development Board, AVR Development Board, MSP430 Launchpad, Intel Edison, Beagle Bone

Brief About NodeMCU ESP8266

The **NodeMCU ESP8266 development board** comes with the ESP-12E module containing ESP8266 chip having Tensilica Xtensa 32-bit LX106 RISC microprocessor. This microprocessor supports RTOS and operates at 80MHz to 160 MHz adjustable clock frequency. NodeMCU has 128 KB RAM and 4MB of Flash memory to store data and programs. Its high processing power with in-built Wi-Fi / Bluetooth and Deep Sleep Operating features make it ideal for IoT projects.

NodeMCU can be powered using Micro USB jack and VIN pin (External Supply Pin). It supports UART, SPI, and I2C interface.



Programming Nodemcu ESP8266 with Arduino IDE

The Nodemcu Development Board can be easily programmed with Arduino IDE since it is easy to use.

Programming Nodemcu with the Arduino IDE will hardly take 5-10 minutes. All you need is the Arduino IDE, a USB cable and the NodeMCU board itself. You can check this [Getting Started Tutorial for NodeMCU](#) to prepare your Arduino IDE for NodeMCU.

Uploading your first program, once Arduino IDE is installed on the computer, connect the board with the computer using the USB cable. Now open the Arduino IDE and choose the correct board by selecting Tools>Boards>NodeMCU1.0 (ESP-12E Module), and choose the correct Port by selecting Tools>Port. To get it started with the NodeMCU board and blink the built-in LED, load the example code by selecting Files>Examples>Basics>Blink. Once the example code is loaded into your IDE, click on the 'upload' button given on the top bar. Once the upload is finished, you should see the built-in LED of the board blinking.

Applications of NodeMCU

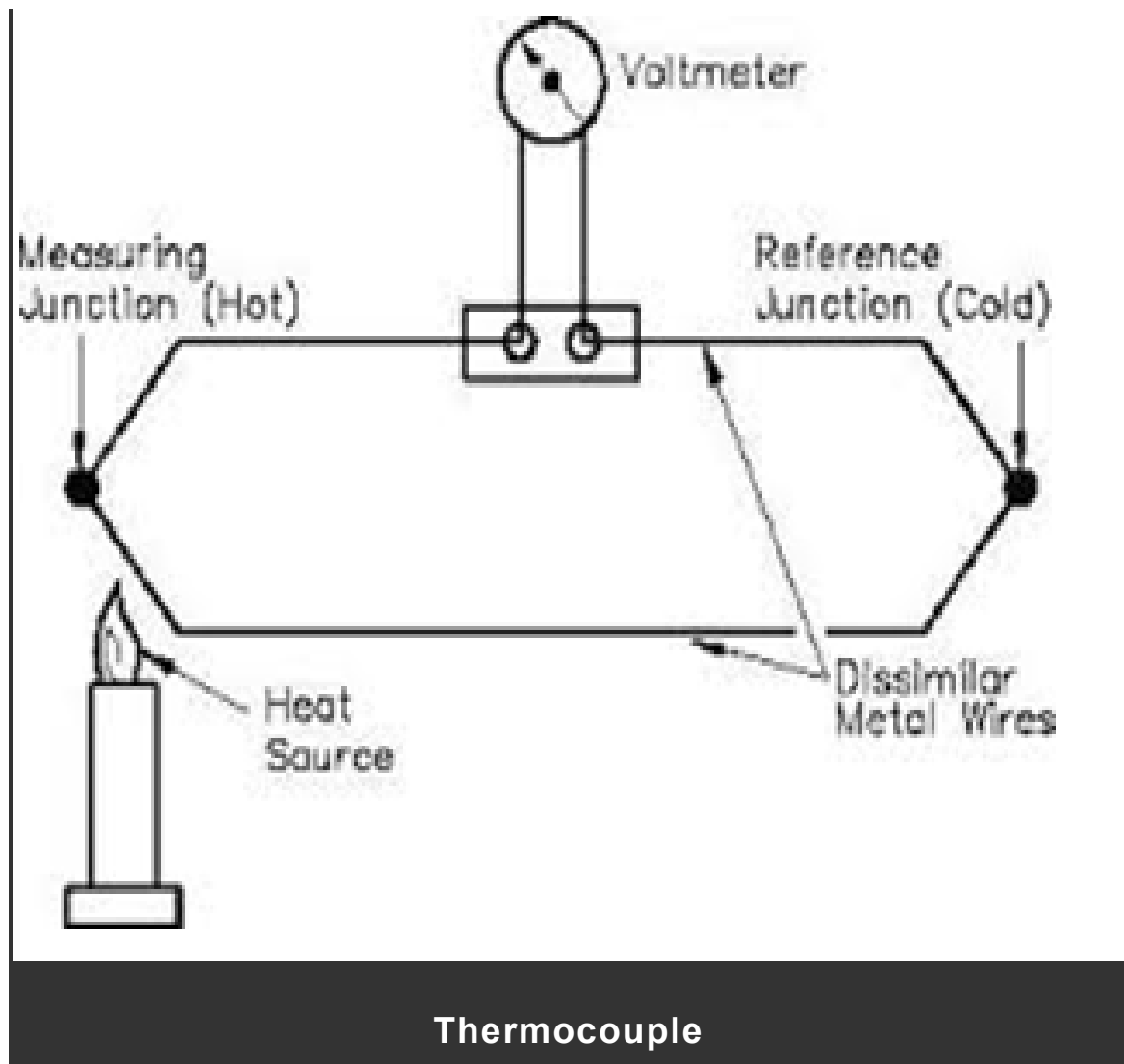
- Prototyping of IoT devices
- Low power battery operated applications
- Network projects
- Projects requiring multiple I/O interfaces with Wi-Fi and Bluetooth functionalities

4.3 TEMPERATURE SENSOR

Temperature is the most often-measured environmental quantity. This might be expected since most physical, electronic, chemical, mechanical, and biological systems are affected by temperature. Certain chemical reactions, biological processes, and even electronic circuits perform best within limited temperature ranges. Temperature is one of the most commonly measured variables and it is therefore not surprising that there are many ways of sensing it. Temperature sensing can be done either through direct contact with the heating source, or remotely, without direct contact with the source using radiated energy instead. There are a wide variety of temperature sensors on the market today, including Thermocouples, Resistance Temperature Detectors (RTDs), Thermistors, Infrared, and Semiconductor Sensors.

THERMOCOUPLE:

It is a type of temperature sensor, which is made by joining two dissimilar metals at one end. The joined end is referred to as the HOT JUNCTION. The other end of these dissimilar metals is referred to as the COLD END or COLD JUNCTION. The cold junction is actually formed at the last point of thermocouple material. If there is a difference in temperature between the hot junction and cold junction, a small voltage is created. This voltage is referred to as an EMF (electro-motive force) and can be measured and in turn used to indicate temperature.



The RTD is a temperature sensing device whose resistance changes with temperature. Typically built from platinum, though devices made from nickel or copper are not uncommon, RTDs can take many different shapes like wire wound, thin film. To measure the resistance across an RTD, apply a constant current, measure the resulting voltage, and determine the RTD resistance. RTDs exhibit fairly linear resistance to temperature curves over their operating regions, and any nonlinearity are highly predictable and repeatable. The PT100 RTD evaluation board uses surface mount RTD to measure temperature. An external 2, 3 or 4-wire PT100 can also be associated with measure temperature in remote areas. The RTDs are biased using a constant current source. So as to reduce self-heat due to power dissipation, the current magnitude is moderately low. The

circuit shown in figure is the constant current source uses a reference voltage, one amplifier, and a PNP transistor.

Thermistors:

Similar to the RTD, the thermistor is a temperature sensing device whose resistance changes with temperature. Thermistors, however, are made from semiconductor materials. Resistance is determined in the same manner as the RTD, but thermistors exhibit a highly nonlinear resistance vs. temperature curve. Thus, in the thermistors operating range we can see a large resistance change for a very small temperature change. This makes for a highly sensitive device, ideal for set-point applications.

Semiconductor sensors:

They are classified into different types like Voltage output, Current output, Digital output, Resistance output silicon and Diode temperature sensors. Modern semiconductor temperature sensors offer high accuracy and high linearity over an operating range of about 55°C to $+150^{\circ}\text{C}$. Internal amplifiers can scale the output to convenient values, such as $10\text{mV}/^{\circ}\text{C}$. They are also useful in cold-junction compensation circuits for wide temperature range thermocouples. A brief details about this type of temperature sensor are given below.

SENSOR ICS

There are a wide variety of temperature sensor ICs that are available to simplify the broadest possible range of temperature monitoring challenges. These silicon temperature sensors differ significantly from the above mentioned types in a couple of important ways. The first is operating temperature range. A temperature sensor IC can operate over the nominal IC temperature range of -55°C to $+150^{\circ}\text{C}$. The second major difference is functionality.

A silicon temperature sensor is an integrated circuit, and can therefore include extensive signal processing circuitry within the same package as the sensor. There is no need to add compensation circuits for temperature sensor Ics. Some of these are analogue circuits with either voltage or current output.

Others combine analogue-sensing circuits with voltage comparators to provide alert functions. Some other sensor ICs combine analogue-sensing circuitry with digital input/output and control registers, making them an ideal solution for microprocessor-based systems.

Digital output sensor usually contains a temperature sensor, analog-to-digital converter (ADC), a two-wire digital interface and registers for controlling the IC's operation. Temperature is continuously measured and can be read at any time. If desired, the host processor can instruct the sensor to monitor temperature and take an output pin high (or low) if temperature exceeds a programmed limit. Lower threshold temperature can also be programmed and the host can be notified when temperature has dropped below this threshold. Thus, digital output sensor can be used for reliable temperature monitoring in microprocessor-based systems.

Above temperature sensor has three terminals and required Maximum of 5.5 V supply. This type of sensor consists of a material that performs the operation according to temperature to vary the resistance. This change of resistance is sensed by circuit and it calculates temperature. When the voltage increases then the temperature also rises. We can see this operation by using a diode.

Temperature sensors directly connected to microprocessor input and thus capable of direct and reliable communication with microprocessors. The sensor unit can communicate effectively with low-cost processors without the need of A/D converters.

An example for a temperature sensor is **LM35**. The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius temperature. The LM35 is operates at -55° to $+120^{\circ}\text{C}$. The basic centigrade temperature sensor ($+2^{\circ}\text{C}$ to $+150^{\circ}\text{C}$) is shown in figure below.

Features of LM35 Temperature Sensor:

- ☛ Calibrated directly in $^{\circ}\text{C}$ Celsius (Centigrade)

- ☛ Rated for full I -55° to $+150^{\circ}\text{C}$ range
- ☛ Suitable for remote applications
- ☛ Low cost due to wafer-level trimming
- ☛ Operates from 4 to 30 volts
- ☛ Low self-heating,
- ☛ $\pm 1/4^{\circ}\text{C}$ of typical nonlinearity

4.4 Operation of LM35:

The LM35 can be connected easily in the same way as other integrated circuit temperature sensors. It can be stuck or established to a surface and its temperature will be within around the range of 0.01°C of the surface temperature.

- ☛ This presumes that the ambient air temperature is just about the same as the surface temperature; if the air temperature were much higher or lower than the surface temperature, the actual temperature of the LM35 die would be at an intermediate temperature between the surface temperature and the air temperature.

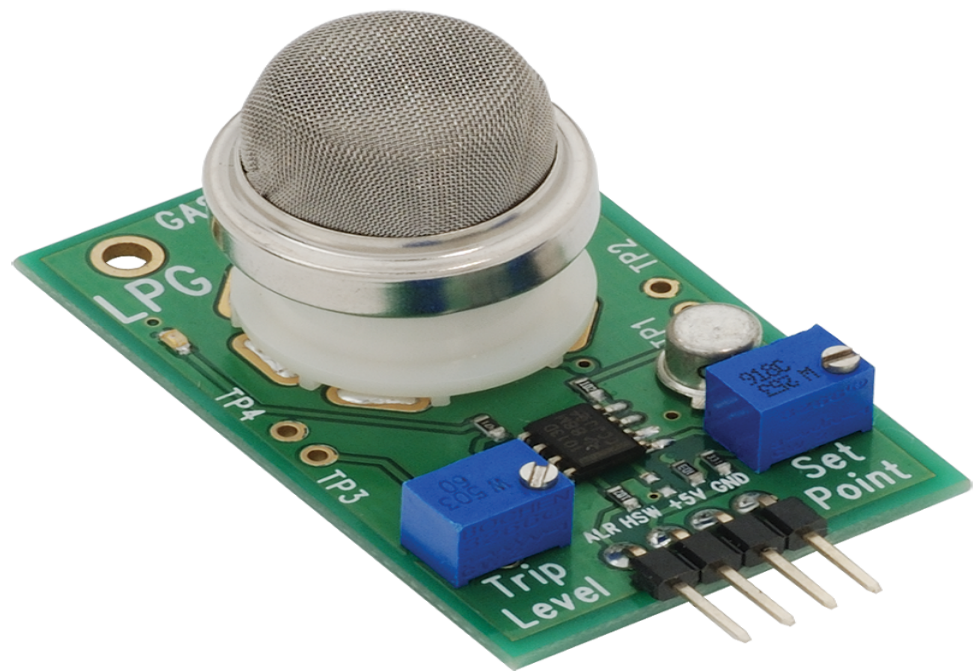
The temperature sensors have well known applications in environmental and process control and also in test, measurement and communications. A digital temperature is a sensor, which provides 9-bit temperature readings. Digital temperature sensors offer excellent precise accuracy, these are designed to read from 0°C to 70°C and it is possible to achieve $\pm 0.5^{\circ}\text{C}$ accuracy. These sensors completely aligned with digital temperature readings in degree Celsius.

smoke sensor module

In current technology scenario, monitoring of gases produced is very important. From home appliances such as air conditioners to electric chimneys and safety systems at industries monitoring of gases is very crucial. **Gas**

sensors are very important part of such systems. Small like a nose, gas sensors spontaneously react to the gas present, thus keeping the system updated about any alterations that occur in the concentration of molecules at gaseous state.

Gas sensors are available in wide specifications depending on the sensitivity levels, type of gas to be sensed, physical dimensions and numerous other factors. This Insight covers a **methane gas sensor** that can sense gases such as ammonia which might get produced from methane. When a gas interacts with this sensor, it is first ionized into its constituents and is then adsorbed by the sensing element. This adsorption creates a potential difference on the element which is conveyed to the processor unit through output pins in form of current. What is this sensing element? Is it kept in some chamber or is kept exposed? How does it get current and how it is taken out? Let's find out in this Insight!!!



The **gas sensor module** consists of a steel exoskeleton under which a sensing element is housed. This sensing element is subjected to current through connecting leads. This current is known as heating current through it, the gases coming close to the sensing element get ionized and are absorbed by the

sensing element. This changes the resistance of the sensing element which alters the value of the current going out of it.

Image shows externals of a standard gas sensor module: a steel mesh, copper clamping ring and connecting leads. The top part is a stainless steel mesh which takes care of the following:

Filtering out the suspended particles so that only gaseous elements are able to pass to insides of the sensor.

- 🌐 Protecting the insides of the sensor.
- 🌐 Exhibits an anti-explosion network that keeps the sensor module intact at high temperatures and gas pressures.

In order to manage above listed functions efficiently, the steel mesh is made into two layers. The mesh is bound to rest of the body via a copper plated clamping ring.

The connecting leads of the sensor are thick so that sensor can be connected firmly to the circuit and sufficient amount of heat gets conducted to the inside part. They are casted from copper and have tin plating over them. Four of the six leads (A, B, C, D) are for signal fetching while two (1, 2) are used to provide sufficient heat to the sensing element.

The pins are placed on a Bakelite base which is a good insulator and provides firm gripping to the connecting leads of the sensor.

The top of the gas sensor is removed off to see the internals parts of the sensor: sensing element and connection wiring. The hexapod structure is constituted by the sensing element and six connecting legs that extend beyond the Bakelite base.

Using a ceramic substrate increases the heating efficiency and tin oxide, being sensitive towards adsorbing desired gas' components (in this case methane and its products) suffices as sensing coating.

The leads responsible for heating the sensing element are connected through Nickel-Chromium, well known conductive alloy. Leads responsible for output signals are connected using platinum wires which convey small changes

in the current that passes through the sensing element. The platinum wires are connected to the body of the sensing element while Nickel-Chromium wires pass through its hollow structure.

While other wires are attached to the outer body of the element, Nickel-Chromium wires are placed inside the element in a spring shaped. Image shows coiled part of the wire which is placed on the inside of the hollow ceramic

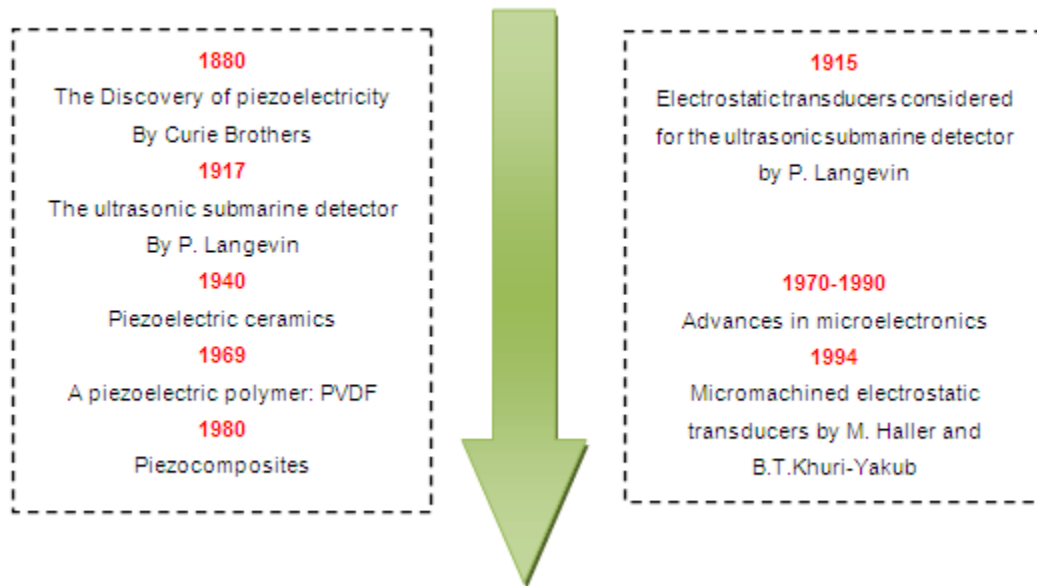
Image shows the ceramic with tin dioxide on the top coating that has good adsorbing property. Any gas to be monitored has specific temperature at which it ionizes. The task of the sensor is to work at the desired temperature so that gas molecules get ionized. Through Nickel-chromium wire, the ceramic region of the sensing element is subjected to heating current. The heat is radiated by the element in the nearby region where gases interact with it and get ionized. Once, ionized, they are absorbed by the tin dioxide. Adsorbed molecules change the resistance of the tin dioxide layer. This changes the current flowing through the sensing element and is conveyed through the output leads to the unit that controls the working of the gas sensor.

4.5 Ultrasonic (UV) Sensors or Ultrasound Sensors

Bats are wonderful creatures. Blind from the eyes and yet a vision so precise that could distinguish between a moth and a broken leaf even when flying at full speed. No doubt the vision is sharper than ours and is much beyond human capabilities of seeing, but is certainly not beyond our understanding. **Ultrasonic ranging** is the technique used by bats and many other creatures of the animal kingdom for navigational purposes. In a bid to imitate the ways of nature to obtain an edge over everything, we humans have not only understood it but have successfully imitated some of these manifestations and harnessed their potential to the greatest extent.

History

Ultrasonic Transducers: Historical Perspective



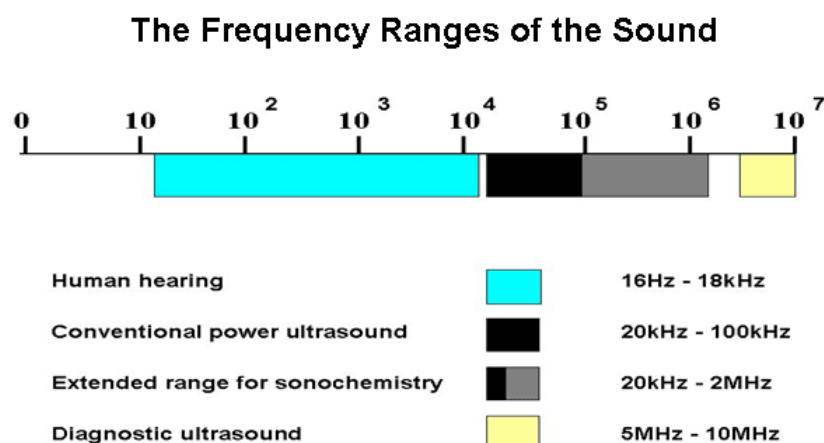
The history dates back to 1790, when Lazzaro Spallanzani first discovered that bats maneuvered in flight using their hearing rather than sight. Jean-Daniel Colladon in 1826 discovered sonography using an underwater bell, successfully and accurately determining the speed of sound in water. Thereafter, the study and research work in this field went on slowly until 1881 when Pierre Curie's discovery set the stage for modern ultrasound transducers. He found out the relationship between electrical voltage and pressure on crystalline material. The unfortunate Titanic accident spurred rigorous interest into this field as a result of which Paul Langevin invented the hydrophone to detect icebergs. It was the first ultrasonic transducer. The hydrophone could send and receive low frequency sound waves and was later used in the detection of submarines in the World War 1.

On a note parallel to the SONAR, medical research also started taking interest in ultrasonics. In late 1930's Dr. Karl Dussik used a technique called hyperphonography which recorded echoes of ultrasonic waves on a sensitive paper. This technique was used to produce ultrasound pictures of the brain to help detect tumors and marked the birth of ultrasound imaging. After that, many scientists like Ian Donald, Douglas Howry, Joseph Holmes, John Wild and John Reid improved upon the various aspects of ultrasonic sensors in the medical field

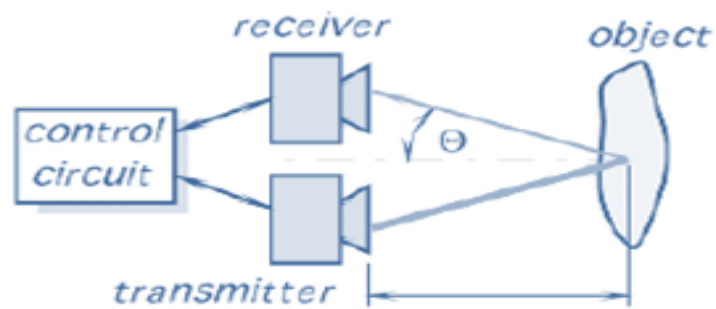
which enabled diagnosis of stomach cancers, ovarian cysts, detection of twin pregnancies, tumors etc. Industry too did not waste time in jumping on to the bandwagon and soon developed techniques like ultrasonic welding and non destructive testing at the outset of the 1960s.

How Ultrasonic Sensors work?

Ultrasonic sensors are devices that use electrical–mechanical energy transformation, the mechanical energy being in the form of ultrasonic waves, to measure distance from the sensor to the target object. Ultrasonic waves are longitudinal mechanical waves which travel as a succession of compressions and rarefactions along the direction of wave propagation through the medium. Any sound wave above the human auditory range of 20,000 Hz is called ultrasound. Depending on the type of application, the range of frequencies has been broadly categorized as shown in the figure below:



When ultrasonic waves are incident on an object, diffused reflection of the energy takes place over a wide solid angle which might be as high as 180 degrees. Thus some fraction of the incident energy is reflected back to the transducer in the form of echoes and is detected.



The distance to the object (L) can then be calculated through the speed of ultrasonic waves (v) in the medium by the relation

$$L = \frac{v t \cos \theta}{2}$$

Where 't' is the time taken by the wave to reach back to the sensor

and 'θ' is the angle between the horizontal and the path taken as shown in the figure. If the object is in motion, instruments based on Doppler shift are used. Get all the details about internal structure and working of an ultrasound sensor at [Insight-How Ultrasonic Sensors Work](#).

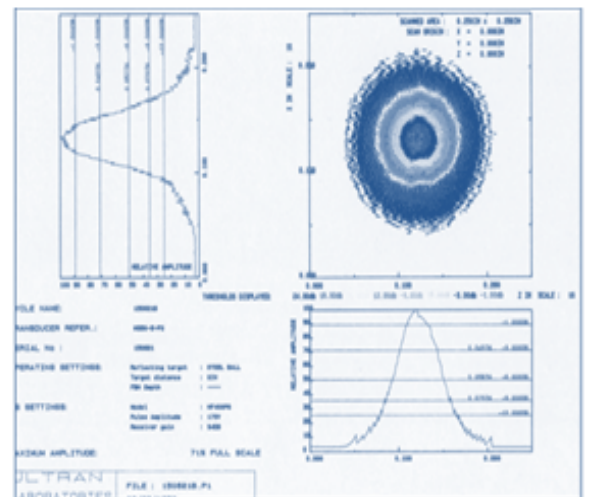
4.6 Generating Ultrasonic Waves

For the generation of such mechanical waves, movement of some surface like a diaphragm is required which can then induce the motion to the medium in front of it in the form of compression and rarefaction. Piezoelectric materials operating in the motor mode and magnetostrictive materials have been widely employed in the generation of ultrasonic waves at frequency ranges of 1-20 MHz and 20-40 kHz respectively. The sensors employ piezoelectric ceramic transducers which flex when an electric signal is applied to them. These are connected to an electronic oscillator whose output generates the oscillating voltages at the required frequency. Materials like Lead Zirconate Titanate are popular piezoelectric materials used in medical ultrasound imaging. For best results, the frequency of the applied oscillations must be equal to the natural

frequency of the ceramic, which produces oscillations readily through resonance. It offers maximum sensitivity and efficiency when operated at resonance.

Piezoelectricity being a reversible phenomenon produces electrical voltages when ultrasonic waves reflect back from the target and impinge upon the ceramic structure. In this way, a transducer may work both as a transmitter and a receiver in pulsed mode. When continuous measurement of distances is required, separate transducers may be used for transmission and reception. The sensors when used in industry are generally employed in arrays which may be mechanical arrays consisting of oscillating or rotating sensors, or electronic arrays which may be linear, curved or phased. To visualize the output of an ultrasonic sensor, displays of different kind are used whose shape depends on the type of transducer array used and the function. A sectorized Field of View is produced by mechanical arrays and curved and phased electronic arrays, while a linear field is generated by linear arrays. The display modes may be linear graphical plotting with amplitude on y-axis and time on x-axis called Amplitude mode or A-mode, or intensity modulated B-scans where the brightness of a spot indicates the amplitude of reflected waves. Other modes include M-mode, Doppler (D) Mode etc.

The parameterization of these sensors is generally done by monitoring the reflected and transmitted signals from the lateral and axial motion of transducer while keeping the target fixed in a specific medium (water in general). The sound beam diverges rapidly, hence care is taken that the transducer produces the smallest possible beams. The narrower the beam pattern, the more sensitive the sensor is. However, the angle possible between the transducer and the surface increases with the beam width. The beam patterns of the kind shown below are observed:



Axial and Cross Sectional beam profiles

The parameters on which the performance of an ultrasonic sensor is measured include bandwidth, attenuation, dynamic range and resolution like grayscale, axial and lateral resolution. Other parameters are Nominal Frequency, Peak Frequency, Bandwidth center Frequency, Pulse Width, sensitivity and Signal to Noise Ratio (SNR).

Importance of Ultrasonic Sensors

There are a variety of sensors based on other physical transduction principles like the optical range finding sensors and the microwave based devices too. Then why should one use ultrasonic transducers in the first place, given that the speed of sound is very slow than the speed of electromagnetic waves? The answer lies in the question itself. Because the EM waves based devices are too fast. Being slower than the EM waves, the time taken by ultrasonic waves is much longer than that taken by the latter and hence its measurement can be done more easily and less expensively. Because these are based on sound waves rather than EM waves, these would work in places where the latter would not.

For example, in the case of clear object detection and measurement of liquid levels or high glare environments, light based sensors would suffer greatly because of the transmittance of the target or the translucence of the propagating

media. Ultrasonic devices being based upon sound propagation would remain practically unaffected.

These also function well in wet environments where optical beams may suffer from refraction from the water droplets in the environment. On account of range and accuracy, the ultrasonic sensors may lie in between two EM wave based sensors, the Infrared rangefinders on the lower end and the LIDARs on the upper end. Not as accurate or long distance as the LIDARs, the Ultrasonic rangefinders fare better than the IR rangefinders which are highly susceptible to ambient conditions and require recalibration when environment changes. Further these devices offer advantage in medical imaging as compared to MRI or X-Ray scans due to inexpensiveness and portability. No harmful effects of ultrasonic waves at the intensity levels used have been detected in contrast to X-rays or radioactivity based methods and is particularly suited for imaging soft tissues.

Problems & Concerns

However, Ultrasonic sensors too aren't free of all the problems. The speed of sound in a medium increases as the temperature of the medium increases. Thus even when the target has remained in the same place, it may now seem that it has shifted to a place closer to the sensor. Air currents due to varied reasons may disturb the path of the wave which could lead to 'Missed Detection' or a wrong measurement.

Acoustic noise like high pitched sounds created due to whistling or hissing of valves and pneumatic devices at the frequency close to the operating frequency may interfere with the output of the sensor. Electrical noise also affects the performance of the sensor. These may generate artifacts which are not a true representation of the imaged object. Just like the vision starts to blur when the distance of the object from the eye gets too small for the eyes to see it, ultrasonic devices also have a 'dead zone' where the sensor cannot reliably make measurements. This happens due to a phenomenon called ringing which is the

continuous vibration of the transducer after emitting the pulse. Thus when the distance is too small, the transducer has not yet come to rest to be able to differentiate between the vibration due to the incident radiation or the oscillation from the electrical excitation. The dangers of Ultrasonic waves are also well founded. If the intensity is too high, it can cause human tissues to heat and may cause ruptures in people exposed to it. Ethical issues like fetus identification and resulting abortions in medical field are also a widespread concern.

Applications

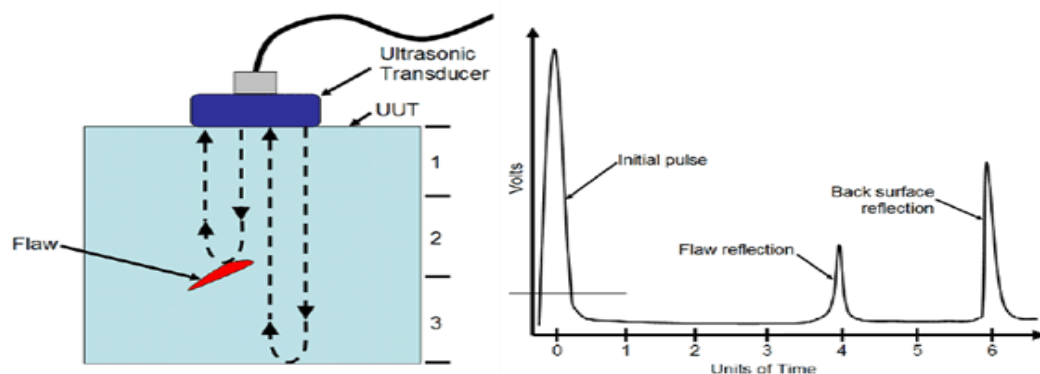
The applications of ultrasonic sensors can be classified on the basis of the property that they exploit.

These can be summarized as:

Domain	Parameter	Applications
Time	Time-of-Flight, Velocity	Density, Thickness, Flaw Detection, Anisotropy, Robotics, Remote Sensing etc.
Attenuation	Fluctuations in reflected and Transmitted Signals	Defect characterization, microstructures, interface analysis
Frequency	Ultrasonic Spectroscopy	Microstructure, grain size, porosity, phase analysis.

Image	Time-of-Flight, velocity, attenuation mapping in Raster C-Scan or SARs	Surface and internal Defect imaging, density, velocity, 2D and 3D imaging.
-------	--	--

Research has been going on to overcome the problems of ultrasonic sensors, particularly in medical imaging where it is known as ultrasound. The artifacts of ultrasonic sensors like Acoustic shadowing and Acoustic Enhancement are being exploited to characterize tissues which allow the differentiation between solid and cystic tissues. The industry too has reaped the benefits from ultrasonic sensors in applications like plastic welding, jewelry cleaning, remote sensing and telemetry, assisted parking systems etc. Robotics has been known to use ultrasonic rangefinders as a favorite tool for distance ranging and mapping. Even the fashion industry is using ultrasonic sensors in hair styling like hair extension implants.



Flaw Detection Using Ultrasonic Sensors

4.7 Future

Non destructive testing and flaw detection uses ultrasonic waves in various modes like the Longitudinal (L-wave) mode and the Shear (S-wave) mode to detect flaws in materials. With the advances in Science, new materials offering increased performance at lower voltages like the capacitive micromachined ultrasonic transducers (CMUTs) are being developed which are expected to have higher bandwidth and greater potential for integration with electronic circuits.

These devices provide non-invasive measures for the detection of problems in all kinds of materials, be it a living tissue or non-living manufactured goods. With a healthy history of being able to detect many problems which otherwise left the doctors dazed and the problem untreated, ultrasonic sensors do offer a lot of promise even in the coming times. The environmental and psychological effects of exposure to EM-radiation being rigorously being put under the scanner, ultrasonic applications are expected to thrive and offer a substantial alternative to the contemporary technologies.

ME 445 INTEGRATED MANUFACTURING TECHNOLOGIES

EXPERIMENT 1

"PROXIMITY SENSORS"

4.8 OBJECTIVE

Increasing automation of complex production systems necessitates the use of components which are capable of acquiring and transmitting information relating to the production process. Sensors fulfill these requirements and have therefore in the last few years become increasingly important components in measuring and in open and closed loop technology. Sensors provide information to a controller in the form of individual process variables.

Proximity sensors are the most basic data acquisition devices in automation. They measure / detect physical input such as temperature, pressure, force, length, and proximity of an object. Transducers are typically a sensorial system capable of signal processing, equipped with electronic instrumentation. Position sensors give a "yes" or "no" response according to the place of the object.

The aim of this experiment is to illustrate the aspects of different types of proximity sensors, their properties, and to compare them. For this, a setup table containing Magnetic, Inductive, Capacitive, and Optical sensors is used. A positioning slide coupled with a vernier caliper is used to measure switching distances.

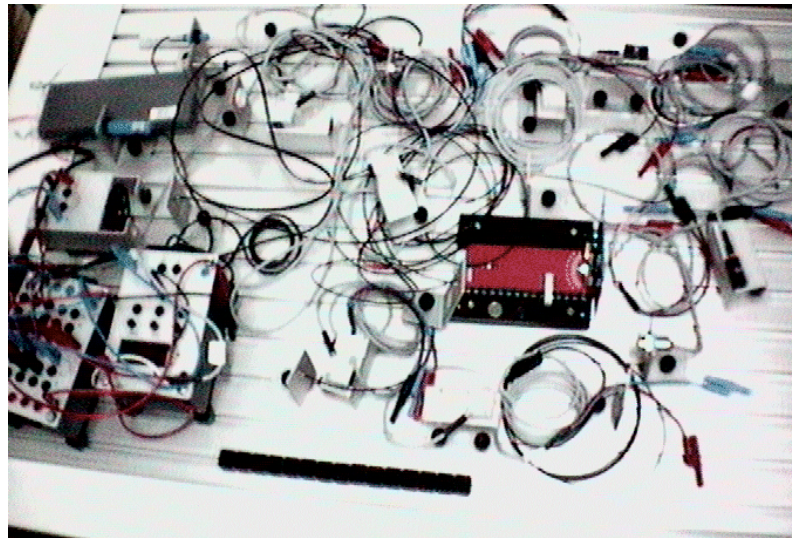


Figure: Proximity sensors setup table

GENERAL INFORMATION

Sensors are the first of the four milestones of Automation:

1. Sensing
2. Signal Processing
3. Planning and Response
4. Memory

They usually convert some physical data into a voltage difference for further processing by a Computer, PLC or I/O Card. The advantages of proximity sensors are:

- They determine the geometrical positions automatically and sensitively.
- They do not need of a direct contact with the workpiece.
- They do not have movable parts that can wear out.
- They are usually equipped with electronic circuits for failure protection.
- They have various types that can be used under different situations.
- They provide the secure working of the process.
- They are used for the system failure analysis.

Their typical usage areas are:

- Automotive industry,
- Packaging industry,
- Printing and paper industry
- Ceramic industry
- Wood-working industry
- Food processing industry

CATEGORIES

According to I/O processing:

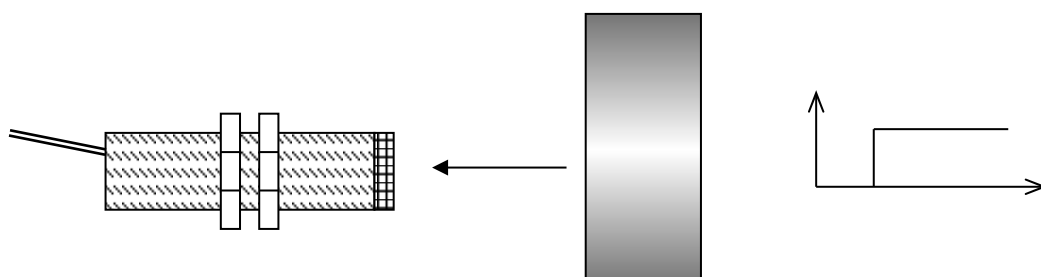
- Binary: Convert a physical measurement value to a binary code (in the form of ON/OFF signals in a selected voltage range)
- Analog: Convert a physical measurement into an analog signal (e.g. temperature readings to variable voltage differences)

According to physical considerations:

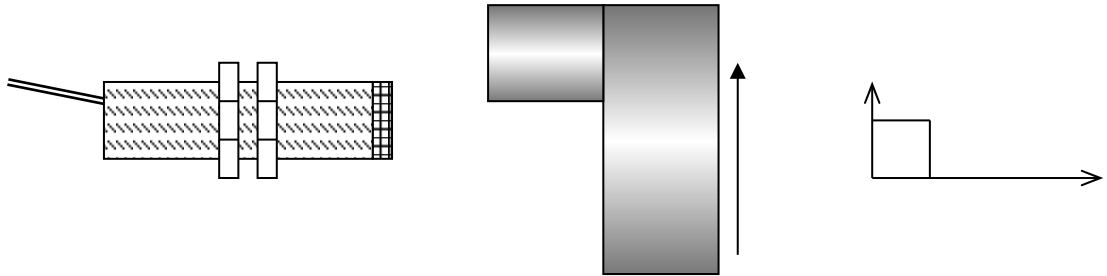
- Mechanical switches
- Magnetic (with/without contacts, pneumatic output)
- Inductive (inductive sensors)
- Capacitive (capacitive sensors)
- Optical (light barriers, reflection sensors)
- Ultrasonic (ultrasonic barriers, ultrasonic sensors)
- Pneumatic (back-pressure nozzles, air reflection sensors, air barriers)

TYPICAL USAGE

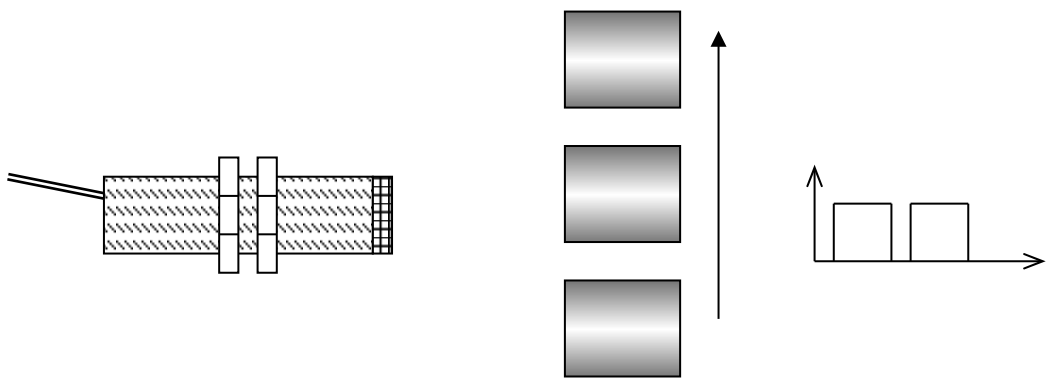
Detecting whether an object exists in a defined position:



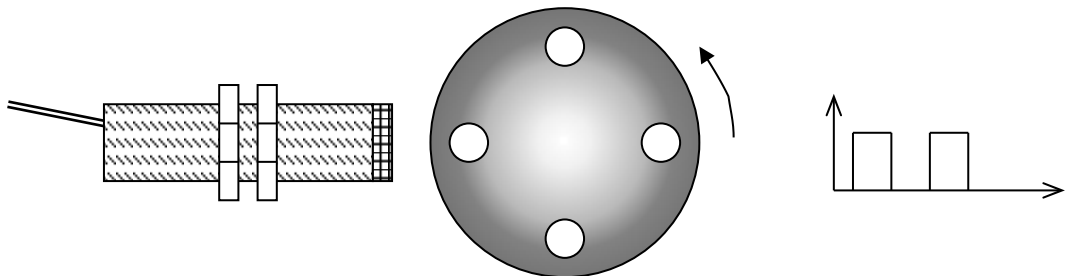
Positioning of an object:



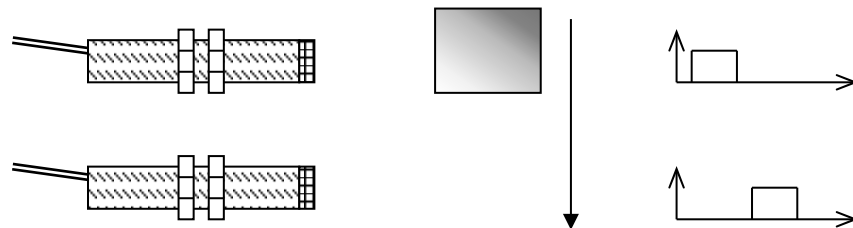
Counting the number of parts:



Determining the rotational speed:



Determining the linear speed:



TYPES

1. Mechanical switches:

Mechanical switches are simple GO/NoGO indicators. They have physical contact with the object, usually coupled with relays and contactors to drive a circuit. Widely used in the industry to mark the end-start points of cylinders, pistons, linear and rotary drives, to sense doors. They are less sensitive and have lower maximum switching frequency compared to proximity switches. Because of the physical contact with the object, they require maintenance and replacement.

2. Magnetic Proximity Switches:

Magnetic switches (also called as Reed-contacts) use the distortion of the magnetic field. If a ferromagnetic material (Fe-Ni compound) comes in the vicinity, the magnetic field distorts and gives an input to the switch. Thus, they are only sensitive to ferromagnetic materials and magnetic fields. Dirt and humidity is of little importance. They preserve high hysteresis (undefinite range of physical input). They are widely used in pairs of machine parts such as piston-cylinder arrangements.

3. Inductive Proximity switches:

Inductive proximity switches also work on the principles of magnetic fields and induction. They response to conductive materials, typically metals. The tabular data on switching distance depends on mild steel (usually Fe37); thus, a

reduction coefficient must be defined for different metals. For the metals such as Cr-Ni, brass, aluminum, and copper this value must be modified with the experimental reduction coefficient found usually in the range of 0.25-0.9. Also the reduction coefficient depends on the size of the measured object. They are widely used in the mass production lines and conveyors to detect metallic workpieces, moving parts of machinery, for measuring linear, rotational speeds, presses, and encoders.

4. Capacitive Proximity switches:

Unlike the magnetic and inductive types, capacitive proximity switches response to all types of materials. The reduction coefficient is determined experimentally in the range of 0.1 to 1 (metals =1 and water =1). Note that liquids can also be detected by capacitive switches. They are very sensitive to environmental factors such as dust, dirt and humidity. Therefore they can be used to distinguish object properties such as color, thickness, water column height, and vibration. Sample application areas are in production lines and conveyors to count workpieces, sense packaging defects etc.

5. Optical Proximity switches:

Optical proximity switches use the presence of visible (with wavelength of 660nm -red-) or invisible (with wavelength of 880nm -ultra-red-), light for input. They give a NPN or PNP output to the circuit. Here, instead of the reduction coefficient the *operation reserve* is defined as the ratio of signal intensity in the input of the sensor to the required intensity for switching. Note that in correct working conditions, operation reserve must have a value of greater than one. The operation reserve depends on ambient conditions such as dust, dirt, ambient light color and intensity, distance from part, reflect-angle etc.

Optical sensors are divided into two main parts:

- Light sensors (can be equipped with fiber-optic cabling for long distance transmission, may use ambient light or the light produced in a coupled unit)
- Reflected light sensors (can be equipped with fiber-optic cabling for long distance transmission, uses the reflected light produced in the same unit from the part or a reflector sheet)

Optical sensors have a relatively greater switching distance. Therefore they may be used in detecting surface irregularities, failure detection, detection of transmissive surfaces, colors etc. Fiber optic cabling for transmission also gives a flexibility to use small units at difficult locations.

6. Ultrasonic Proximity switches:

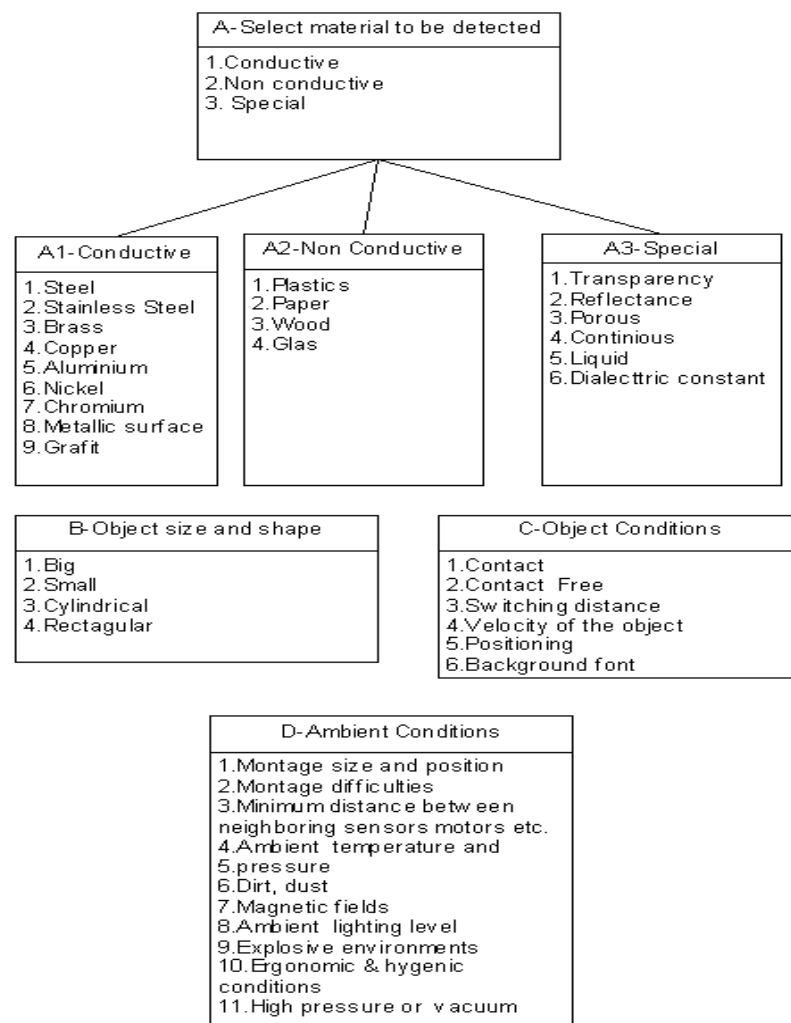
They use the reflected sound power for input. Note that above the sensors stated here, ultrasonic proximity switches have the greatest switching distance and frequency. Therefore, they are used to detect distant objects with very high speeds. They are usually insensitive to ambient conditions and should be preferred in very extreme conditions, while they are very expensive.

7. Pneumatic Proximity switches:

They use the reflected back-pressure supplied from a nozzle at or distant from the switch unit. Generally preferred in the areas of:

- Very dirty and dusty places,
- At high temperatures,
- In the vicinity of explosive materials where electrical currents may be dangerous,
- At places where intensive magnetic fields are present, in the vicinity of big motors, pumps, turbines etc.
- The sensor unit and nozzle unit may be built in one package or as different units. Can be used to drive a pneumatic piston directly.

SELECTION CRITERIA:



4.9 PROTECTION CLASSES

The protection classes of the mechanical elements are defined in DIN 40050. For example, IP67 represents a device with protection against contact and foreign material according to 6 (Table A1) and against water and humidity according to 7 (Table A2).

First digit	Protection Class
0	No special protection
1	Protection against solid objects larger than 50 mm diameter. Unprotected against forced contacts (eg. via hand). Should be kept apart from the body
2	Protection against solid objects larger than 12 mm diameter. Should be kept apart from the fingers
3	Protection against solid objects larger than 2.5 mm diameter. Should be kept apart from the devices (wire, hand tools etc.)
4	Protection against solid objects larger than 1 mm diameter. Should be kept apart from the devices (wire, hand tools etc.)
5	Protection against hazardous dust accumulation. Dust protection is not totally achieved, but inner dust accumulation does not affect functioning of the device. Full protection against forced contact.
6	Full protection against dust accumulation. Full protection against forced contact.

Table A1: Protection against dust & forced contact.

Second digit	Protection Class
0	No special protection
1	Protection against vertically tipping water. The water has no hazardous effects (tipping water).

2	Protection against vertically tipping water at 15 ° to the normal of the device surface. The water has no hazardous effects (inclined tipping water).
3	Protection against water tipping at 60 ° to the normal of the device. The water has no hazardous effects (sprinkling water)
4	Protection against water from any direction to the device. The water has no hazardous effects (flowing water)
5	Protection against water from a nozzle coming from any direction to the device. The water has no hazardous effects (flowing water)
6	Protection against water forced water coming from any direction to the device. The water has no hazardous effects (forced water)
7	Protection against water in case of immersion at certain pressure for a specific time Leakage of the water into the device is avoided.
8	Full protection against water in case of immersion for a predetermined period of time (permanent immersion).

Table A2: Protection against water

CHAPTER-5

5.1 ARDUINO UNO AND ITS PROGRAMMING

Arduino is a tool for making computers that can sense and control more of the physical world than your desktop computer. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board.

Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other physical outputs. Arduino projects can be stand-alone, or they can be communicate with software running on your computer. The boards can be assembled by hand or purchased preassembled; the open-source IDE can be downloaded for free.

The Arduino programming language is an implementation of Wiring, a similar physical computing platform, which is based on the Processing multimedia programming environment.

5.2 Arduino Software (IDE)



Writing Sketches

- File
- Edit
- Sketch
- Tools
- Help
- 📁 Sketchbook
- 📁 Tabs, Multiple Files, and Compilation
- 📁 Uploading
- 📁 Libraries
- 📁 Third-Party Hardware
- 📁 Serial Monitor
- 📁 Preferences
- 📁 Language Support
- 📁 Boards

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.

Writing Sketches

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom righthand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

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.



SerialMonitor

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.

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.

OpenRecent

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.

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.

☐ *SelectAll*

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/DecreaseIndent*

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.

FindNext

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.

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.

UploadUsingProgrammer

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.

ExportCompiledBinary

Saves a .hex file that may be kept as archive or sent to the board using other tools.

ShowSketchFolder

Opens the current sketch folder.

IncludeLibrary

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.

■ *AddFile...*

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 of the toolbar.

5.3 Tools

■ *AutoFormat*

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

■ *ArchiveSketch*

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

■ *FixEncoding&Reload*

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

■ *SerialMonitor*

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 onboard USB-serial connection. Normally you won't need this, but if you're burning a bootloader to a new microcontroller, you will use this.

BurnBootloader

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.

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.

Sketchbook

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

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

Tabs, Multiple Files, and Compilation

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

Uploading

Before uploading your sketch, you need to select the correct items from the Tools > Board and Tools > Port menus. The boards are described below. On the Mac, the serial port is probably something like /dev/tty.usbmodem241 (for an Uno or Mega2560 or Leonardo) or /dev/tty.usbserial-1B1 (for a Duemilanove or earlier USB board), or /dev/tty.USA19QW1b1P1.1 (for a serial board connected with a Keyspan USB-to-Serial adapter). On Windows, it's probably COM1 or COM2 (for a serial board) or COM4, COM5, COM7, or higher (for a USB board) - to find out, you look for USB serial device in the ports section of the Windows Device Manager. On Linux, it should be /dev/ttyACMx , /dev/ttyUSBx or similar. Once you've selected the correct serial port and board, press the upload button in the toolbar or select the Upload item from the Sketch menu. Current Arduino boards will reset automatically and begin the upload. With older boards (pre-Diecimila) that lack auto-reset, you'll need to press the reset button on the board just before starting the upload. On most boards, you'll see the RX and TX LEDs blink as the sketch is uploaded. The Arduino Software (IDE) will display a message when the upload is complete, or show an error.

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

5.4 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.

To write your own library, see this tutorial.

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.

Serial Monitor

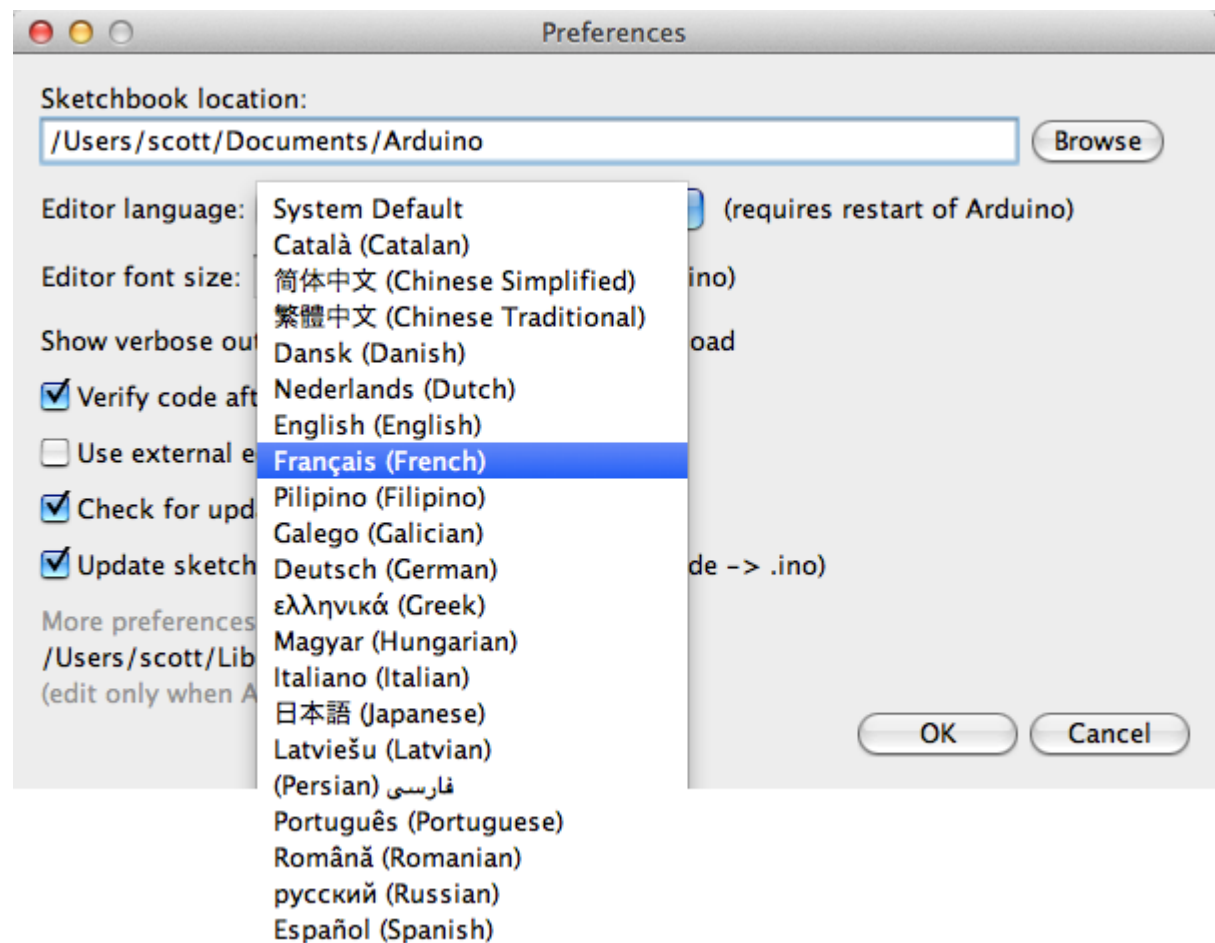
This displays serial sent from the Arduino or Genuino board over USB or serial connector. To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down menu that matches the rate passed to `Serial.begin` in your sketch. Note that on Windows, Mac or Linux the board will reset (it will rerun your sketch) when you connect with the serial monitor. Please note that the Serial Monitor does not process control characters; if your sketch needs a complete management of the serial communication with control characters, you can use an external terminal program and connect it to the COM port assigned to your Arduino board.

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

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.

Language Support



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 dropdown 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.

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.

ArduinoYùn

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

Arduino/GenuinoUno

An ATmega328P running at 16 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.

ArduinoDiecimilaorDuemilanovew/ ATmega168

An ATmega168 running at 16 MHz with auto-reset.

ArduinoNanow/ ATmega328P

An ATmega328P running at 16 MHz with auto-reset. Has eight analog inputs.

Arduino/GenuinoMega2560

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

ArduinoMega

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

■ *ArduinoMegaADK*

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

■ *ArduinoLeonardo*

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

■ *Arduino/GenuinoMicro*

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

■ *ArduinoEsplora*

An ATmega32u4 running at 16 MHz with auto-reset.

■ *ArduinoMiniw/ ATmega328P*

An ATmega328P running at 16 MHz with auto-reset, 8 Analog In, 14 Digital I/O and 6 PWM.

■ *ArduinoEthernet*

Equivalent to Arduino UNO with an Ethernet shield: An ATmega328P running at 16 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.

■ *ArduinoFio*

An ATmega328P running at 8 MHz with auto-reset. Equivalent to Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328P, 6 Analog In, 14 Digital I/O and 6 PWM.

■ *ArduinoBTw/ ATmega328P*

ATmega328P running at 16 MHz. The bootloader burned (4 KB) includes codes to initialize the on-board bluetooth module, 6 Analog In, 14 Digital I/O and 6 PWM..

■ *LilyPad ArduinoUSB*

An ATmega32u4 running at 8 MHz with auto-reset, 4 Analog In, 9 Digital I/O and 4 PWM.

■ *LilyPad Arduino*

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

■ *ArduinoPro/Mini(5V,16 MHz)w/ ATmega328P*

An ATmega328P running at 16 MHz with auto-reset. Equivalent to Arduino Duemilanove or Nano w/ ATmega328P; 6 Analog In, 14 Digital I/O and 6 PWM.

■ *ArduinoNGorolderw/ 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.

■ *ArduinoRobotControl*

An ATmega328P running at 16 MHz with auto-reset.

■ *ArduinoRobotMotor*

An ATmega328P running at 16 MHz with auto-reset.

■ *ArduinoGemma*

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

For instructions on installing support for other boards, see third-party hardware above.

Last revision 2015/09/07 by SM

The text of the Arduino getting started guide is licensed under a Creative Commons Attribution-ShareAlike 3.0 License. Code samples in the guide are released into the public domain.

CHAPTER – 6

UBIDOTS EXPLORER:

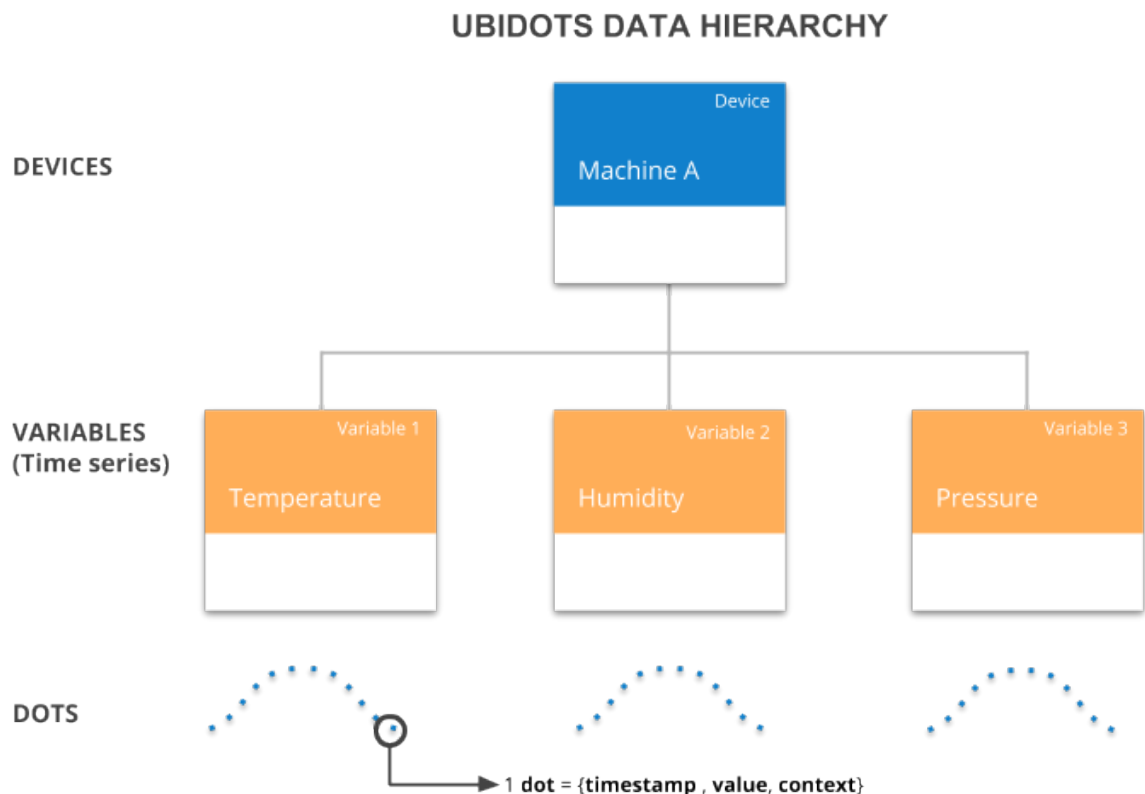
6.1 Introduction

Welcome to the Ubidots REST API hardware docs. Here you would reference how to send or retrieve data from your hardware devices using our supported communication protocols: **HTTP**, **MQTT** and **TCP/UDP**.

The purpose of this section is to help you understand what happens in the backstage when communicating with Ubidots, so you can replicate this in your firmware. For this reason, we avoid the use of examples using libraries. However, if you'd like to use one of our libraries (Python, C, and more), check out our compatible devices and API clients sections.

How Ubidots Works?

Every time a device updates a sensor value in a variable, a data-point or "**dot**" is created. Ubidots stores dots that come from your devices inside variables, and these stored dots have corresponding **timestamps**:



Each dot contains these items:

Item	Description	Mandatory
Value	A numerical value. Ubidots accepts up to 16 floating-point length numbers.	YES
timestamp	Unix Epoch time, in milliseconds. If not specified, then our servers will assign one upon reception.	NO
context	An arbitrary collection of key-value pairs. Mostly used to store the latitude and longitude coordinates of GPS devices.	NO

Values

A numerical value. Ubidots accepts up to 16 floating-point length numbers.

```
{"value" : 34.87654974}
```

Timestamps

A **timestamp**, as best described [here](#), is a way to track time as a running total of seconds. This count starts at the Unix Epoch on January 1st, 1970 at UTC. Therefore, the unix time stamp is merely the number of seconds between a particular date and the Unix Epoch. Please keep in mind that when you send data to Ubidots, you must set the timestamp in milliseconds; also, if you retrieve a dot's timestamp, it will be in milliseconds.

```
"timestamp" : 1537453824000
```

The above timestamp corresponds to Thursday, September 20, 2018 2:30:24 PM.

PRO-TIP: A useful tool to convert between Unix timestamps and human-readable dates is [Epoch Converter](#).

Context

Numerical values are not the only data type supported; you can also store *string* or *char* data types inside what we call **context**. The **context** is a key-

value object that allows you to store not only numerical but also string values. An example use of the **context** could be:

```
"context" : {"status" : "on", "weather" : "sunny"}
```

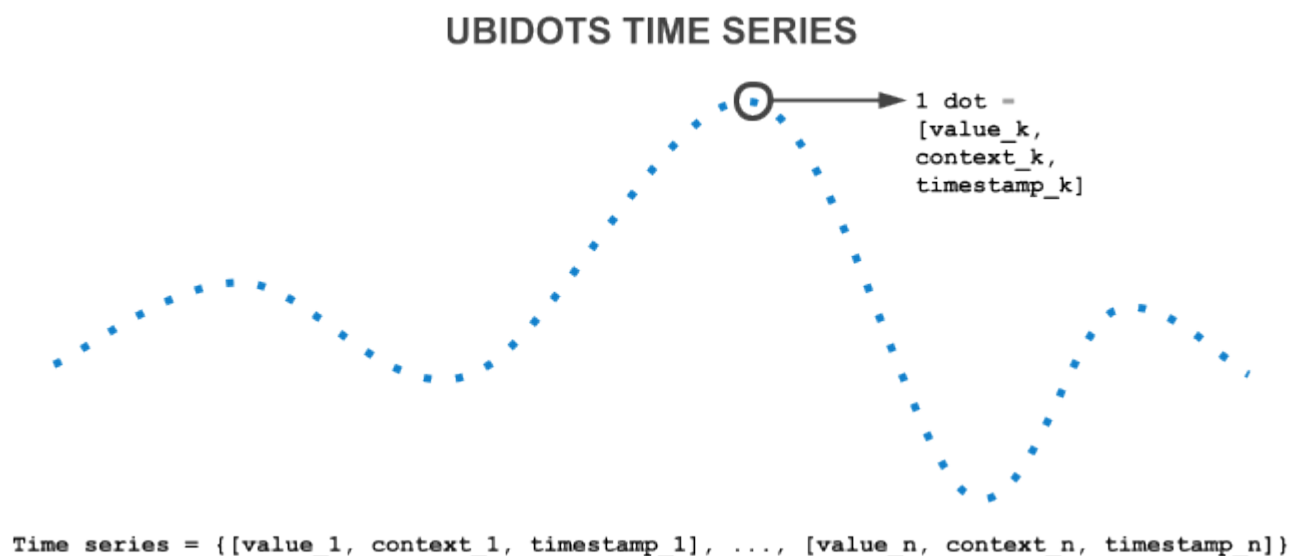
A **context** is commonly used to store the latitude and longitude coordinates of your device for GPS/tracking application use cases. All Ubidots maps uses the *lat* and *lng* keys from a dot's **context** to extract the coordinates of your device, in that way you just need to send a single dot with the coordinates values in the variable context to plot a map instead of sending separately both latitude and longitude in two different variables. Below you can find a typical **context** with coordinates values:

```
"context" : {"lat":-6.2, "lng":75.4, "weather" : "sunny"}
```

Please note that you can mix both string and numerical values in the **context**. If your application is for geo-localization purposes, make sure that the coordinates are set in decimal degrees.

Time Series

Based on the above, we can illustrate an Ubidots time series like this:



Ubidots is an agnostic platform, this means that it does not really care what hardware device you are using, as long as you're able to interact with us through at least one of these protocols:

- [HTTP](#)
- [MQTT](#)
- [TCP/UDP](#)

HTTP

HTTP is the main internet protocol for information exchange in the Internet. It also supports data encryption, which is usually called *secure HTTP* or *HTTPS*. When you use a web browser (MS Edge, Firefox, Google Chrome, etc) you are sending and receiving packages using *HTTP* or *HTTPS*. Because of this, it was not surprising to see *HTTPS* become a popular option for data exchange within IoT. When a developer is to implement *HTTP* communication, then he should look for a *RESTful Application Programming Interface*, or *REST API*, which exposes all the endpoints required to interact with a third party application (like Ubidots). In this section, you can reference our REST API to understand how to send and retrieve data from our servers.

Note: Our goal with this section is to help you understand what's happening behind when communicating with Ubidots, enabling you to replicate it within your own firmware. Because of this, we avoid using our custom libraries in all the examples.

HTTP requests

The following methods are specified within the *HTTP* standard:

HTTP Method	Description
GET	Used to retrieve information
POST	Used to create a new element
PATCH	Used to update existing elements
DELETE	Used to delete existing elements

HTTP is a **request/response** protocol, this means that every request that you make is answered by the server. This response includes a number (response code) and a body. For example, when you make a request to retrieve a file on a webpage "(e.g. *"Get me the file 'webside.html'"*)", you build a GET request. If the

request is correct, the server will typically return a 200 response code, along with the file requested (body).

Under the same logic, when a device sends a GET request to the Ubidots' server (i.e. requesting the last value of a variable), then the server sends back a response with the status of the request (response code), and a body, which would be the a value with its context and timestamp.

An *HTTP* request also needs the parameters below:

- **Host:** Specifies the server you will be making HTTP requests to.
- **Path:** This is typically the remaining portion of the URL that specifies the resource you want to consume, be it a variable or a device. For example, if an API endpoint is: `industrial.api.ubidots.com/api/v1.6/devices/my-device` then the path would be `/api/v1.6/devices/my-device`
- **Headers:** Define the operating parameters of the *HTTP* request such as authentication, Content-Type, Content-Length, etc.
- **Body/payload:** In the case of POST and PATCH requests, this is the data sent by your device to the server. GET requests typically do not have a body because they are meant to request data, not to send data.

Ubidots accepts data as *JavaScript Object Notation* or *JSON*. *JSON* is a typical *HTTP* data type, it is a collection of name/value pairs. In various programming languages, this is treated as an object, record, struct, dictionary, hash table, keyed list, or associative array. It is also human readable and language independent. An example of a *JSON* data type that Ubidots accepts can be referenced below:

```
{"temperature": {"value":10, "timestamp": 1534881387000, "context": {"machine": "1st floor"}}}
```

A typical *HTTP* request to Ubidots should be set as below:

```
POST {PATH} HTTP/1.1<CR><LN>
```

```
Host: {HOST}<CR><LN>
```

```
User-Agent: {USER_AGENT}<CR><LN>
```

```
X-Auth-Token: {TOKEN}<CR><LN>
```

```
Content-Type: application/json<CR><LN>
```

```
Content-Length: {PAYLOAD_LENGTH}<CR><LN><CR><LN>
```

```
{PAYLOAD}
```

<CR><LN>

Where:

- {PATH}: Path to the resource to consume
Example: /api/v1.6/variables/ to get user's variables information.
- {HOST}: Host URL.
Example: industrial.api.ubidots.com
- {USER_AGENT}: An optional string used to identify the type of client, be it by application type, operating system, software vendor or software version of the requesting user agent.
Examples: ESP8266/1.0, Particle/1.2
- {TOKEN}: Unique key that authorizes your device to ingest data inside your Ubidots account.
- {PAYLOAD_LENGTH}: The number of characters of your payload.
Example: The payload {"temperature": 20} will have a content-length of 19.
- {PAYLOAD}: Data to send.
Example: {"temperature": 20}

PRO TIP: The easy way of handle *HTTP* and *HTTPs* requests is by using **cURL**, a command line tool. To learn how to install cURL on Windows, MacOSX or Linux please refer to this [guide](#).

API URLs

API access can be over *HTTP* or secure HTTP, *HTTPs*, using the following endpoints based on your Ubidots Account type.

Security Note: We strongly advise to use *HTTPs* to make sure your data travels encrypted, avoiding the exposure of your API token and/or sensor data.

HTTP

Ubidots Account	Endpoint	Port
Educational	http://things.ubidots.com	80
Industrial	http://industrial.api.ubidots.com	80

HTTPs

Ubidots Account	Endpoint	Port
Educational	https://things.ubidots.com	443
Industrial	https://industrial.api.ubidots.com	443

Ubidots supports SSL v1.1, TLS v1.2 and v1.3. You can download our root certificates in different formats:

- 📄 [PEM file](#): A certificate chain with two root certificates from our certificate authorities (CAs).
- 📄 [DER file](#): Same as the PEM file, with an alternative encoding.
- 📄 [CRT file](#): Same as the PEM file, with a different extension. Often referred to as .crt, .cert or .cer.

NOTE: To avoid future inconveniences, we advise to **use both certificates** in your device's auth routines. While the second is the current valid certificate, we will use the first after 2021.

Fixed IPs

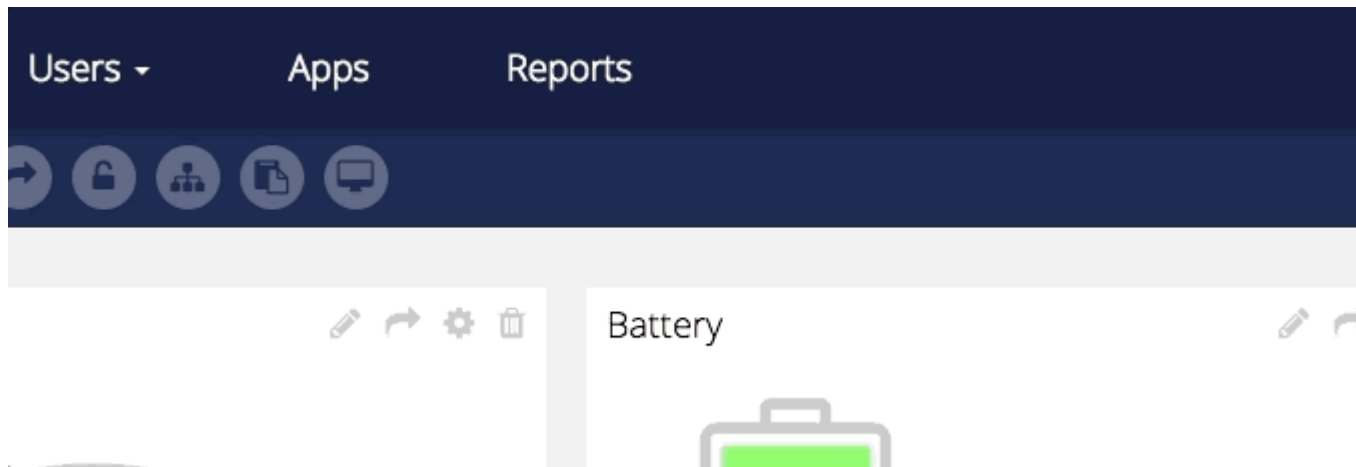
Ubidots also supports fixed IPs address for data ingestion, but we highly advise to our users to implement their firmware routines using the dns endpoints instead of fixed IPs, as these last ones may change in the future.

Ubidots Account	ipv4	ipv6	Port
Educational	50.23.124.68	2607:f0d0:2101:39::2	80/443
Industrial	169.55.61.243	2607:f0d0:3:b::2	80/443

HTTP Authentication

Every request requires a *token*. A *token* is an unique key that authorizes your device to ingest data inside your Ubidots account.

The easiest way to get yours is clicking on "**API Credentials**" under your profile tab:



Also, 6 hour valid *tokens* can be generated using your account *API-KEY*. If you desire to learn how to do it, please refer the [Ubidots REST Software API reference](#).

There are two ways to send a *token* in a request:

1. Send the *token* as parameter in the endpoint, setting it with the `?token=` expression. This is the easiest one to implement but the most unsecure method to send your authentication credentials.
2. Send the *token* as a request header. For this, use the X-Auth-Token header.

Security Note: While sending your *token* as a URL parameter can be quite comfortable in some devices, we only advise to do this for prototyping stages. For production stages, we strongly advise sending the token using the X-Auth-Token header.

```
GET
https://industrial.api.ubidots.com/api/
```

Send data

Our REST API allows you send data to the platform in two different ways:

- Send data to a **device**
- Send data to a **variable**

If you're looking for more advanced API capabilities, such as editing a device name, or listing all of your devices, please refer to our Ubidots Software REST API Reference.

Send data to a device

Request structure:

```
POST /api/v1.6/devices/{DEVICE_LABEL} HTTP/1.1<CR><LN>
Host: {Host}<CR><LN>
User-Agent: {USER_AGENT}<CR><LN>
X-Auth-Token: {TOKEN}<CR><LN>
Content-Type: application/json<CR><LN>
Content-Length: {PAYLOAD_LENGTH}<CR><LN><CR><LN>
{PAYLOAD}
<CR><LN>
```

Expected Response:

```
HTTP/1.1 200 OK<CR><LN>
Server: nginx<CR><LN>
Date: Tue, 04 Sep 2018 22:35:06 GMT<CR><LN>
Content-Type: application/json<CR><LN>
Transfer-Encoding: chunked<CR><LN>
Vary: Cookie<CR><LN>
Allow: GET, POST, HEAD, OPTIONS<CR><LN><CR><LN>
{PAYLOAD_LENGTH_IN_HEXADECIMAL}<CR><LN>
{"{VARIABLE_LABEL}": [{"status_code": 201}]}<CR><LN>
0<CR><LN>
```

The easiest way to **send values to Ubidots** is by specifying your Device label in the request *path* and making a POST request to it.

Replace the {DEVICE_LABEL} with the unique label of the desired device to send data to. If the device does not exist in your Ubidots account, it will be automatically created. Device labels can contain any alphanumerical character, without blank spaces, i.e *weather-station*, *my-home*, *esp8266*.

To set the variables values, just create a *JSON* data payload as below

N° Variables	JSON structure
One	<code>{"variable-label": VALUE}</code>
Multiple	<code>{"variable-label-1": {"value": VALUE, "timestamp": TIMESTAMP, "context": {"key": VALUE}}, ... , "variable-label- n": {"value": VALUE, "timestamp": TIMESTAMP, "context": {"key": VALUE}}}</code>

timestamp and *context* parameters are optional.

NOTE 1:: By default, the Ubidots backend will take the latitude and longitude values inside any variable labeled as **"gps"**, **"location"**, **position** or **"coordinates"** automatically to get your device coordinates in all of the map widgets. Alternatively, you change the default location variable.

NOTE 2: All timestamps must be in milliseconds. If you set a timestamp for a variable value, it will be stored with the date represented by it. You can easily convert dates to timestamps values [here](#).

CONCLUSION

The smart basket helps us reduce pollution. Many times, the garbage can over flowing and a lot of animals like the dog or the mouse enter in or near the trash can. This creates a bad scene. Even some birds also try to remove garbage from the trash. This project can avoid such situations. And the the message can be sent directly to the cleaning vehicle instead of the contractor's office. This project is also helpful in the government project of “SWACHH BHARAT ABHIYAN”.This project can also be used in the “SMART CITY”.

REFERENCES

- [1] Daniel V., Puglia P.A., and M. Puglia (2007). "RFID-A Guide to Radio Frequency Identification", Technology Research Corporation.
- [2] Flora, A. (2009). "Towards a clean environment: A proposal on sustainable and integrated solid waste management system for university Kebangsaan Malaysia". Report from Alam Flora.

- [3] Gogoi.L (2012). "Solid Waste Disposal and its Health Implications in Guwahati City: A Study in Medical Geography", Lambert Academic Publishing, Germany, ISBN 978-3-8454-0149-2.
- [4] Hannan, M., A., Arebey, M., Basri, H. (2010). "Intelligent Solid Waste Bin monitoring and Management System", Australian Journal of Basic and Applied Sciences, 4(10): 5314-5319, 2010, ISSN 1991-8178.
- [5] Md. Liakot Ali, Mahbubul Alam, Md. Abu Nayeem Redwanur Rahaman, (2012). "RFID based E-monitoring System for Municipal Solid Waste Management", International Conference on Electrical and Computer Engineering, Pg 474-477.