Project report on

IOT SYSTEM FOR CAR

Under the Guidance of

Prof. Nikhil Tiwari

Prof. Megha Gupta

By

Mr. Pranav Tiwari	**TE EXTC-B (44)**
Mr. Chinmay Tompe	**TE EXTC-B (45)**
Mr. Pranay Vora	**TE EXTC-B(51)**

Department of

Electronics and Telecommunication Engineering,

University of Mumbai.

Academic Year: 2017-2018.

ACKNOWLEDGEMENT

During this semester long project several individuals have provided many forms of help and support.However,it would not have been possible without the kind support and help of these individuals and our college.We would like to extend our sincere thanks to all of them.

We are highly indebted to Prof. Nikhil Tiwari and Prof. Megha Gupta for their guidance and constant supervision as well as providing necessary information regarding the project and also supporting us in completion of our project in time.

Also we are thankful to the R&D Cell of TCET for providing us with components when required.

We would also like to thank our college Lab Assistants and Technical Assistants Mr.Dinesh Kanswal, Mr.Chandresh Yadav,Ms.Jinal Rathod and Ms.Kinjal Joshi for providing the necessary facilities and equipments that could be used by us during the project.

Last but not the least, we would like to express our gratitude towards our parents and classmates for their kind cooperation and encouragement which helped us in completion of our project.

Mr. Pranav Tiwari	TE EXTC-B (44)
Mr. Chinmay Tompe	TE EXTC-B (45)
Mr. Pranay Vora	TE EXTC-B (51)

CERTIFICATE

This is to certify that,

Mr. Pranav Tiwari	**TE EXTC-B (44)**
Mr. Chinmay Tompe	**TE EXTC-B (45)**
Mr. Pranay Vora	**TE EXTC-B (51)**

has satisfactorily completed the project according to Mini Project-1 curriculum as per the University of Mumbai at Thakur College of Engineering & Technology on partial fulfilment of the T.E.(Electronics & Telecommunication Engineering) course for Semester –V.

Internal Guides:

Prof. Nikhil Tiwari

Prof. Megha Gupta

Internal Examiner   External Examiner

ABSTRACT

Considering the direction in which everyday technologies are headed i.e. the interconnectedness of objects over the internet which we call IOT. However, IOT is not just connecting devices to the internet but it also involves a certain level of autonomy given to those devices. IOT involves a network of sensors and actuators which provide raw data.When we have such systems, it becomes important to use platforms that process this raw data ,present and provide analysis.Through this project we have tried to understand such platforms through a basic system of sensors.Interfacing sensors with platforms such as Thingspeak and NodeRed will provide us with the basics for an IOT system with data analysis.

INDEX

LIST OF FIGURES

CHAPTER-01

INTRODUCTION

1.1 System:

A system is an arrangement in which all its unit assemble work together according to a set of rules. It can also be defined as a way of working, organizing or doing one or many tasks according to a fixed plan. For example, a watch is a time displaying system. Its components follow a set of rules to show time. If one of its parts fails, the watch will stop working. So we can say, in a system, all its subcomponents depend on each other.

1.2 Embedded Systems:

As its name suggests, Embedded means something that is attached to another thing. An embedded system can be thought of as a computer hardware system having software embedded in it. An embedded system can be an independent system or it can be a part of a large system. An embedded system is a microcontroller or microprocessor based system which is designed to perform a specific task. For example, a fire alarm is an embedded system; it will sense only smoke.

An embedded system has three components:

· It has hardware.

· It has application software.

· It has Real Time Operating system (RTOS) that supervises the application software and provide mechanism to let the processor run a process as per scheduling by following a plan to control the latencies. RTOS defines the way the system works. It sets the rules during the execution of application program. A small scale embedded system may not have RTOS.

So we can define an embedded system as a Microcontroller based, software driven, reliable, real-time control system.

1.3 Processors in a System:

Processor is the heart of an embedded system. It is the basic unit that takes inputs and produces an output after processing the data. For an embedded system designer, it is necessary to have the knowledge of both microprocessors and microcontrollers.

A processor has two essential units:

- Program Flow Control Unit (CU) .
- Execution Unit (EU) .

The CU includes a fetch unit for fetching instructions from the memory. The EU has circuits that implement the instructions pertaining to data transfer operation and data conversion from one form to another.

The EU includes the Arithmetic and Logical Unit (ALU) and also the circuits that execute instructions for a program control task such as interrupt, or jump to another set of instructions.

A processor runs the cycles of fetch and executes the instructions in the same sequence as they are fetched from memory.

1.4 Microprocessor:

A **microprocessor** is a computer processor which incorporates the functions of a computer's central processing unit (CPU) on a single integrated circuit (IC),[1] or at most a few integrated circuits.[2] The microprocessor is a multipurpose, clock driven, register based, digital-integrated circuit which accepts binary data as input, processes it according to instructions stored in its memory, and provides results as output. Microprocessors contain both combinational logic and sequential digital logic. Microprocessors operate on numbers and symbols represented in the binary numeral system.

1.5 Microcontroller:

A **microcontroller** (or **MCU** for *microcontroller unit*) is a small computer on a single integrated circuit. In modern terminology, it is similar to, but less sophisticated than, a system on a chip or SoC; anSoC may include a microcontroller as one of its components. A microcontroller contains one or more CPUs (processor cores) along with memory and programmable input/output peripherals. Program memory in the form of Ferroelectric RAM, NOR flash or OTP ROM is also often included on chip, as well as a small amount of RAM. Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications consisting of various discrete chips.

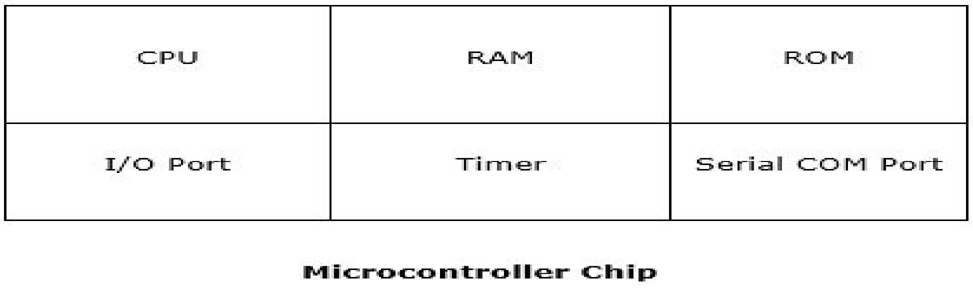
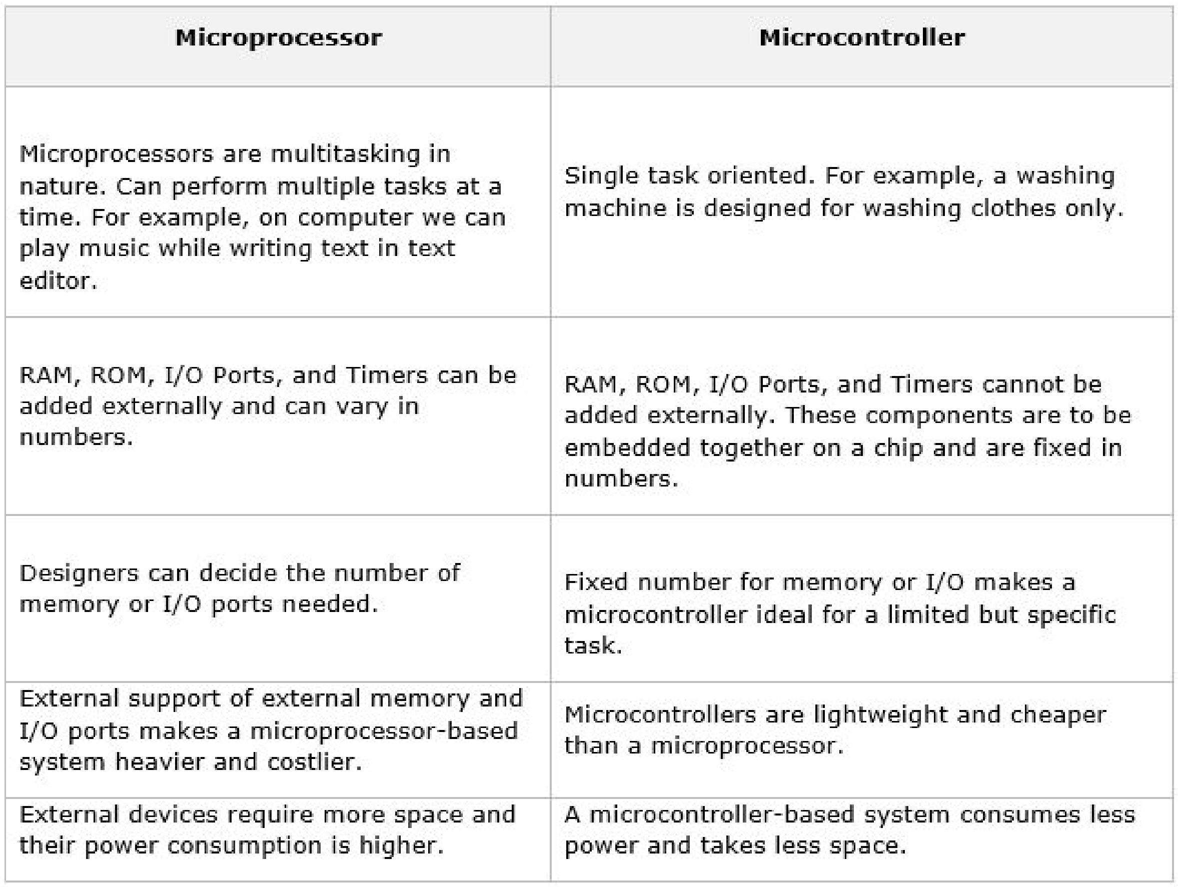


Figure 1.1: Microcontroller chip

1.6 Microprocessor vs Microcontroller:

Let us now take a look at the most notable differences between a microprocessor and a microcontroller.



Microprocessor	Microcontroller
Microprocessors are multitasking in nature. Can perform multiple tasks at a time. For example, on computer we can play music while writing text in text editor.	Single task oriented. For example, a washing machine is designed for washing clothes only.
RAM, ROM, I/O Ports, and Timers can be added externally and can vary in numbers.	RAM, ROM, I/O Ports, and Timers cannot be added externally. These components are to be embedded together on a chip and are fixed in numbers.
Designers can decide the number of memory or I/O ports needed.	Fixed number for memory or I/O makes a microcontroller ideal for a limited but specific task.
External support of external memory and I/O ports makes a microprocessor-based system heavier and costlier.	Microcontrollers are lightweight and cheaper than a microprocessor.
External devices require more space and their power consumption is higher.	A microcontroller-based system consumes less power and takes less space.

Figure 1.2: Comparision uc and uP

1.7 Internet of Things:

Internet of Things (IoT) describes an emerging trend where a large number of embedded devices (things) are connected to the Internet. These connected devices communicate with people and other things and often provide sensor data to cloud storage and cloud computing resources where the data is processed and analyzed to gain important insights. Cheap cloud computing power and increased device connectivity is enabling this trend.

IoT solutions are built for many vertical applications such as environmental monitoring and control, health monitoring, vehicle fleet monitoring, industrial monitoring and control, and home automation.

At a high level, many IoT systems can be described using the diagram below:

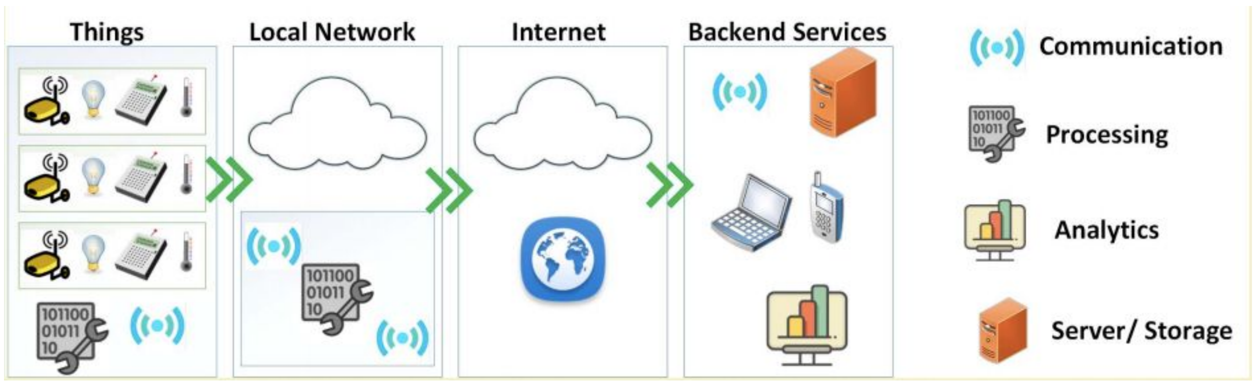


Fig 1.3 IOT system block diag

On the left, we have the smart devices (the "things" in IoT) that live at the edge of the network. These devices collect data and include things like wearable devices, wireless temperatures sensors, heart rate monitors, and hydraulic pressure sensors, and machines on the factory floor.

In the middle, we have the cloud where data from many sources is aggregated and analyzed in real time, often by an IoT analytics platform designed for this purpose.

The right side of the diagram depicts the algorithm development associated with the IoT application. Here an engineer or data scientist tries to gain insight into the collected data by performing historical analysis on the data. In this case, the data is pulled from the IoT platform into a desktop software environment to enable the engineer or scientist to prototype algorithms that may eventually execute in the cloud or on the smart device itself.

An IoT system includes all these elements. ThingSpeak fits in the cloud part of the diagram and provides a platform to quickly collect and analyze data from internet connected sensors.

1.7.2Advantages of IoT:

1. Data: The more the information, the easier it is to make the right decision. Knowing what to get from the grocery while you are out, without having to check on your own, not only saves time but is convenient as well.

2. Tracking: The computers keep a track both on the quality and the viability of things at home. Knowing the expiration date of products before one consumes them improves safety and quality of life. Also, you will never run out of anything when you need it at the last moment.

3. Time: The amount of time saved in monitoring and the number of trips done otherwise would be tremendous.

4. Money: The financial aspect is the best advantage. This technology could replace humans who are in charge of monitoring and maintaining supplies.

1.7.3 IoT components:

The following illustration shows the basic structure of an IOT:

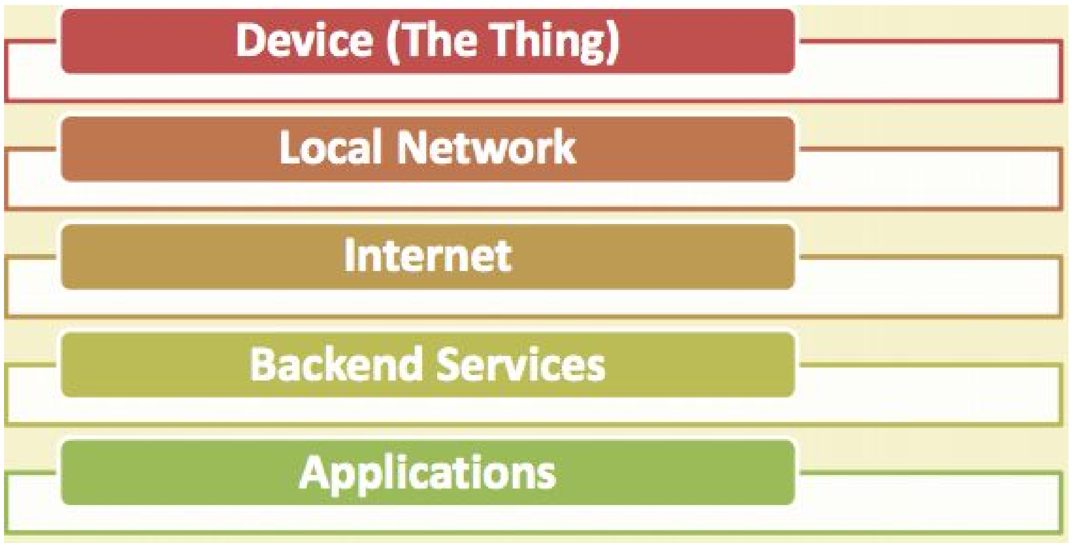


Fig 1.4 IOT components

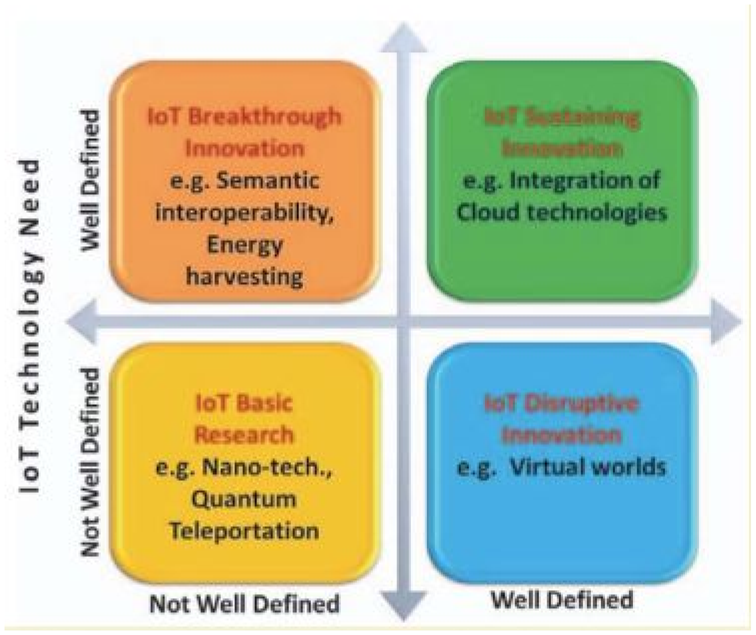

IoT Technology Need
Well Defined
Not Well Defined
IoT Breakthrough Innovation
e.g. Semantic interoperability, Energy harvesting
IoT Sustaining Innovation
e.g. Integration of Cloud technologies
IoT Basic Research
e.g. Nano-tech., Quantum Teleportation
IoT Disruptive Innovation
e.g. Virtual worlds
Not Well Defined
Well Defined

Figure 1.5: Interdependency of domains

Functional Components of IoT

➔ Component for interaction and communication with other IoT devices

➔ Component for processing and analysis of operations

➔ Component for Internet interaction

➔ Components for handling Web services of applications

➔ Component to integrate application services

➔ User interface to access IoT

1.7.4 Functionality-based IoT Protocol Organization

- Connectivity (6LowPAN, RPL)
- Identification (EPC, uCode, IPv6, URIs)
- Communication
- Transport (WiFi, Bluetooth, LPWAN)
- Discovery (Physical Web, mDNS, DNS
- Data Protocols (MQTT, CoAP, AMQP, Websocket, Node)

- Semantic (JSON-LD, Web Thing Model)
- Multi -layer Frameworks (Alljoyn, IoTivity, Weave, Homekit)
- Communication Protocols

The following communication protocols have immediate importance to consumer and industrial IoTs:

- IEEE 802.15.4
- Zigbee
- 6LoWPAN
- Wireless HART
- Z-Wave
- ISA 100
- Bluetooth
- NFC
- RFID

1.8 VEHILE TO VEHICLE (V to V-CONNECTED VEHICLES)

- Vehicles equipped with Sensors
- Networking and communicating devices
- Capable of :
- Communicating with other devices within the vehicle
- Communicating with other similar vehicles
- Communicating with fixed infrastructure
- Vehicle-to-Everything (V2X) Paradigm

Main component of future Intelligent Transportation System (ITS).

- Enables vehicles to wirelessly share a diverse range of information.
- Information sharing may be with other vehicles, pedestrians, or fixed infrastructures (mobile towers, parking meters, etc.)
- Allows for traffic management, ensuring on-road and off-road safety,
- mobility for traveling.

CHAPTER NO-02

ABOUT THE PROJECT

2.1 Problem Definition:

1) The government has been advising its citizens not to drink and drive for decades, but we are not following it properly and facing serious accident issues in the world.
2) Also when the car is handed over to a driver, car owners are not making attention towards cars different parameters and drivers habits of drinking alcohol.
3) For accident alert or for security from theft we can track our car location.
4) So it will become important to track a real time location and different parameters of car, which can be easily implemented by iot concept.

2.2 Problem Solution:

1) So,we are going to make a small prototype for tracking a real time location of car and alert system for verification of driver, whether he is drunk or not.

CHAPTER NO-3

PROJECT REQUIREMENTS

PART A: HARDWARE

3.1 ALCOHOL DETECTION SENSOR

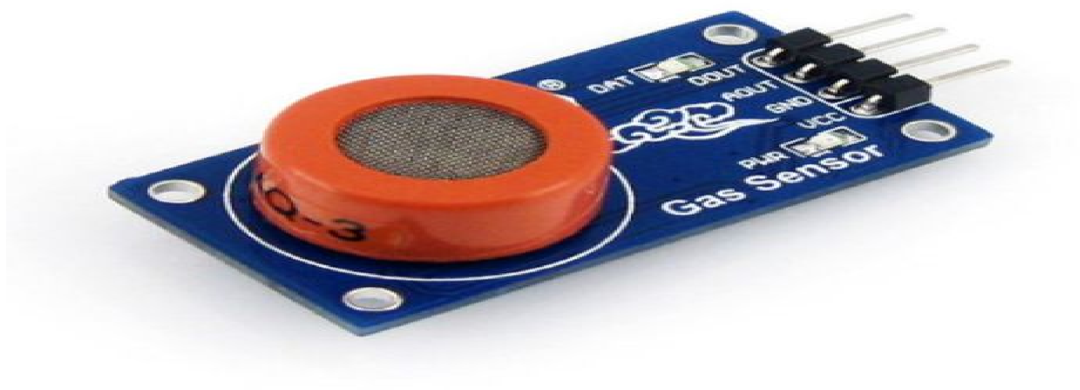


Figure 3.1: MQ -3

3.1.2 BRIEF IDEA

In this project, we will go over how to build an alcohol sensor with an arduino.

The alcohol sensor we will use is the MQ-3 sensor. This is a sensor that is not only sensitive to alcohol, particularly ethanol, which is the type of alcohol which is found in wine, beer, and liquor.

This type of sensor circuit can be used as a breathalyzer to check a person's blood alcohol level. Just as we exhale carbon dioxide when we breathe out, we also will breathe out some alcohol if we have alcohol in our blood. Any alcometer device can measure this alcohol content.

The more ethanol in your blood, the more there is in the air on exhalation. This alcohol content gives a good indication for if a person is drunk and how drunk they are.The amount of alcohol exhaled into the air is proportional to the amount of alcohol which will be found in a person's blood. Alcometers use a built-in formula to estimate blood alcohol content from exhaled air alcohol content.

For different countries, the level of alcohol in the blood that defines a person as over the limit for driving varies. The range ranges from 0.01 to 0.10. Most countries have a limit of about 0.05. For example, Greece, Greenland, and Iceland all have limits of 0.05. Canada has a higher limit set at 0.08. In the United States, it is also 0.08. This means that if the alcometer reading measures above this, the person can receive a DUI.

3.1.3 **FEATURES**

- 5V operation
- Simple to use
- LEDs for output and power
- Output sensitivity adjustable
- Analog output 0V to 5V
- Digital output 0V or 5V
- Low Cost
- Fast Response
- Stable and Long Life
- Good Sensitivity to Alcohol Gas
- Both Digital and Analog Outputs
- On-board LED Indicator
- It is a simple drive circuit

3.1.4 TECHNICAL DATA

- Concentration : 0.05 mg/L ~ 10 mg/L Alcohol
- Operating Voltage : 5V ±0.1
- Current Consumption : 150mA
- Operation Temperature : -10°C ~ 70°C

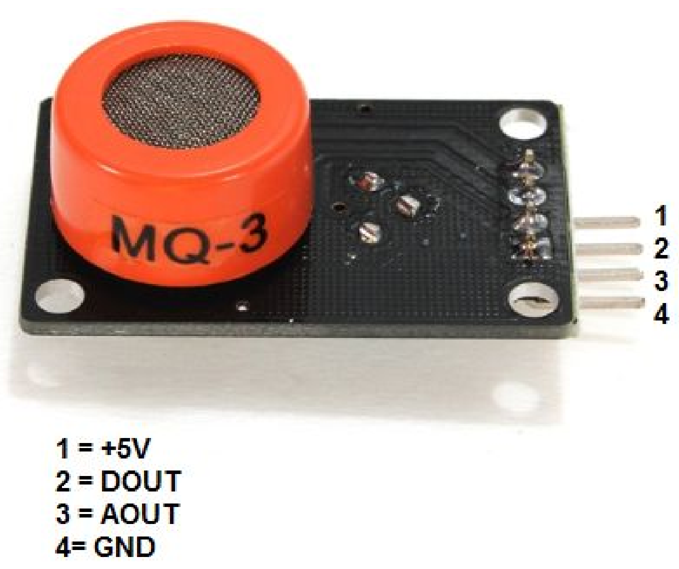


Fig 3.2 Pin out of MQ3

3.1.5 What is an Alcohol Sensor?

An alcohol sensor detects the attentiveness of alcohol gas in the air and an analog voltage is an output reading. The sensor can activate at temperatures ranging from -10 to 50° C with a power supply is less than 150 Ma to 5V. The sensing range is from 0.04 mg/L to 4 mg/L, which is suitable for breathalyzers.

3.1.6 Basic Pin Configuration Of Alcohol Sensor

The MQ-3 alcohol gas sensor consists of total 6-pins including A, H, B and the other three pins are A, H, B out of the total 6-pins we use only 4 pins. The two pins A, H are used for the heating purpose and the other two pins are used for the ground and power. There is a heating system inside the sensor, which is made up of aluminium oxide, tin dioxide. It has heat coils to produce heat, and thus it is used as a heat sensor. The below diagram shows the pin diagram and the configuration of the MQ-3 alcohol sensor.

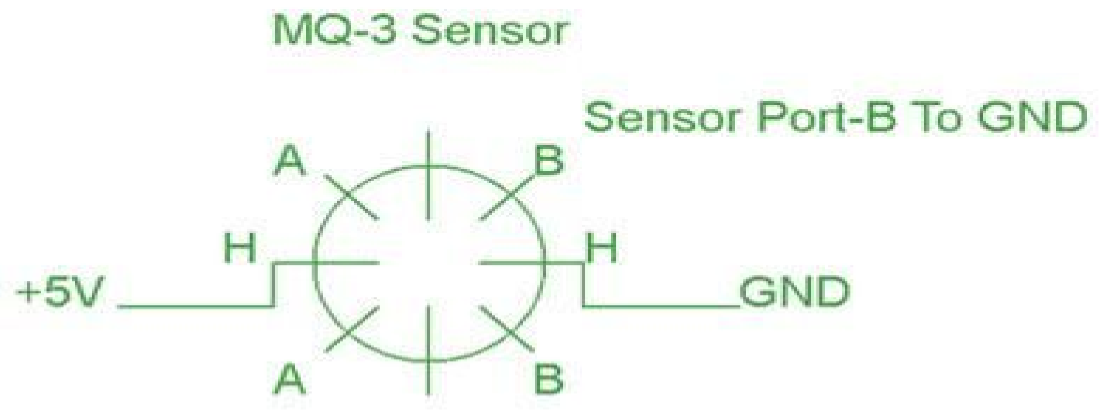


Fig 3.3 Pin Configuration Of Alcohol Sensor

3.1.7 MQ-3 Alcohol Sensor Circuit Schematic (Interfacing with an Arduino)

The alcohol sensor circuit we will build with an MQ-3 sensor integrated with an arduino is shown below.

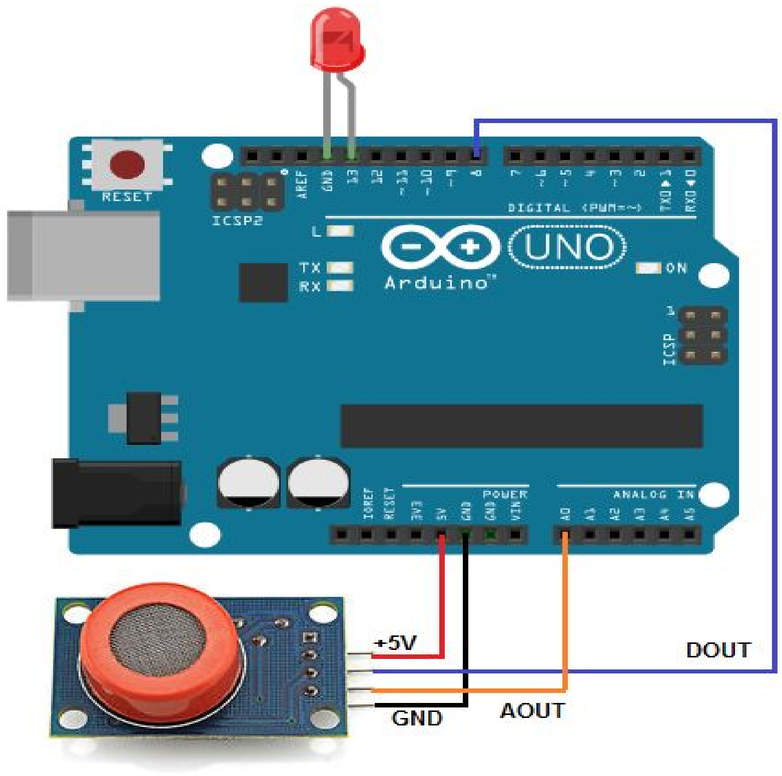


Fig 3.5: Interfacing with Arduino

Components Needed

- MQ-3 alcohol sensor
- Arduino
- LED

If you buy the complete board, there are 4 leads which need to be connected.

There 4 leads are +5V, AOUT, DOUT, and GND.

The +5V and GND leads establishes power for the alcohol sensor.

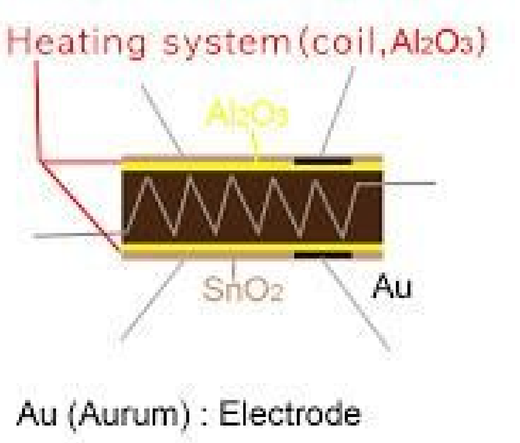
The other 2 leads are AOUT (analog output) and DOUT (digital output). How the sensor works is the terminal AOUT gives an analog voltage output in proportion to the amount of alcohol the sensor detects. The more alcohol it detects, the greater the analog voltage it will output. Conversely, the less alcohol it detects, the less analog voltage it will output. If the analog voltage reaches a certain threshold, it will send the digital pin DOUT high. Once this DOUT pin goes high, the arduino will detect this and will trigger the LED to turn on, signaling that the alcohol threshold has been reached and is now over the limit. How you can change this threshold level is by adjusting the potentiometer to either raise or lower the level.

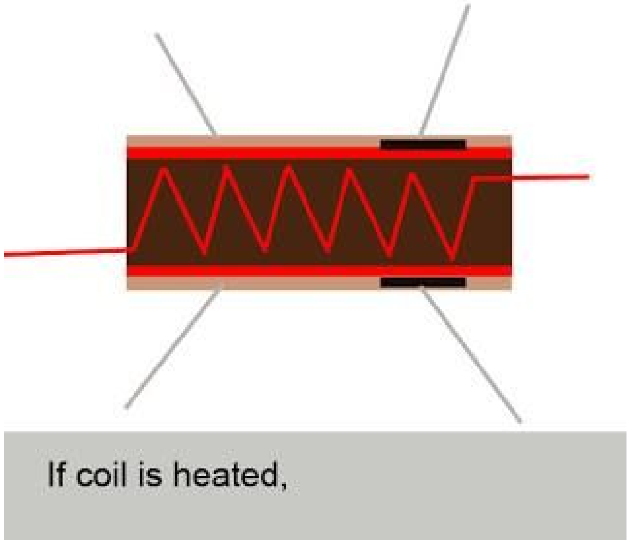
3.1.8 Working Principle

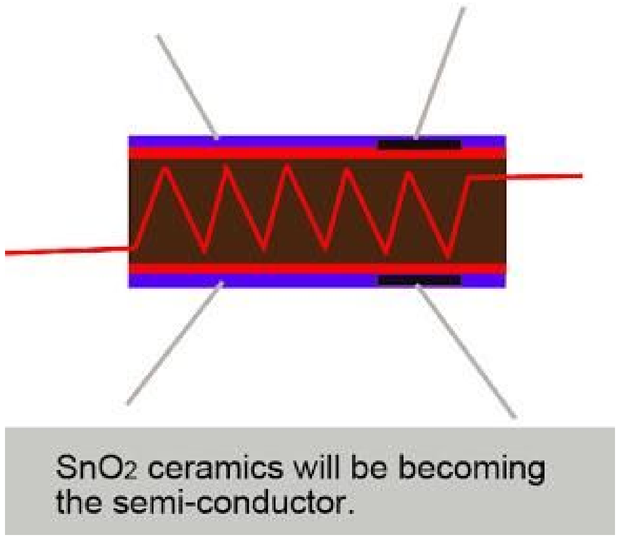
The core system is the cube. As you can see in this cross-sectional view, basically, it is an Alumina tube cover by SnO2, which is tin dioxide. And between them there is an Aurum electrode, the black one. And also you can see how the wires are connected. So, why do we need them? Basically, the alumina tube and the coils are the heating system, the yellow, brown parts and the coils in the picture.

Working Process

If the coil is heated up,







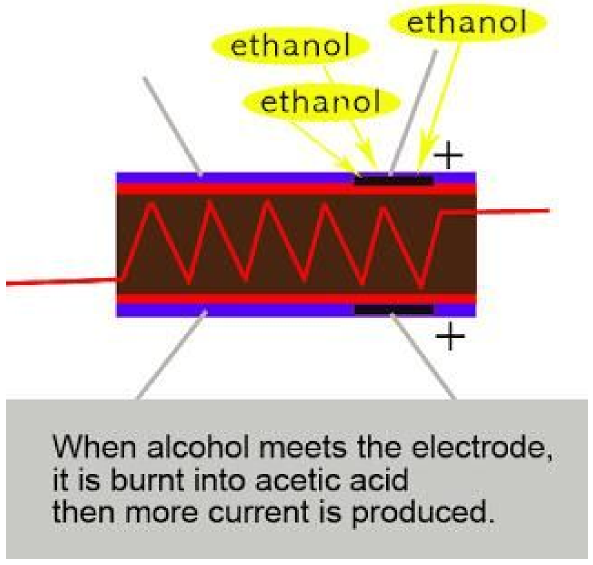

ethanol
ethanol
ethanol
+
+
When alcohol meets the electrode, it is burnt into acetic acid then more current is produced.

Fig 3.6 Working process of MQ3

SnO2 ceramics will become the semi - conductor, so there are more movable electrons, which means that it is ready to make more current flow.
Then, when the alcohol molecules in the air meet the electrode that is between alumina and tin dioxide, ethanol burns into acetic acid then more current is produced. So the more alcohol molecules there are, the more current we will get. Because of this current change, we get the different values from the sensor.

3.1.9 Understanding the Chemistry

When the user exhales into a breath analyzer, any ethanol present in their breath is oxidized to acetic acid at the anode

at the anode :

$CH_3CH_2OH(g) + H_2O(l) \rightarrow CH_3CO_2H(l) + 4H^+(aq) + 4e^-$.

At the cathode, atmospheric oxygen is reduced:

$O_2(g) + 4H^+(aq) + 4e^- \rightarrow 2H_2O(l)$.

The overall reaction is the oxidation of ethanol to acetic acid and water.

$CH_3CH_2OH(l) + O_2(g) \rightarrow CH_3COOH(l) + H_2O(l)$.

The electrical current produced by this reaction is measured and used to determine resistance, which corresponds to the different levels of intoxication that the microcontroller will determine.

3.1.10 CAUTION

You don't have to drink alcohol in order to test this sensor. Mouthwash, such as Listerine, contains alcohol. If you gargle mouthwash for a few seconds and then breathe into the sensor, the readings should jump and register. This is all that's needed to test. I take no responsibility for any drunkenness.

3.1.11 Applications

This module can be applied to vehicle alcohol gas alarm, portable alcohol gas detection device and etc

3.2 UBLOX NEO 6MV2 GPS SENSOR

3.2.1 SPECIFICATION (FEATURES)

1.**Supply Voltage**:2.7 to 3.6V

2.**Supply current**:67 mA

3.**Antenna gain**:50 dB

4.**Operating temperature:**-40 to 85°C

5.**Antenna Type:**Passive and active antenna

6.**Interfaces**:UART,USB,SPI,DDC

7.**Sensitivity-**

- **Tracking & Navigation:**-160 dBm
- **Reacquisition:**-160 dBm
- **Cold Start (Autonomous):**-146 dBm

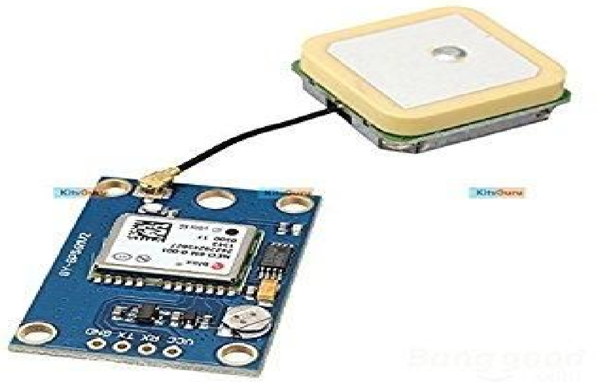


Figure 3.7: Ublox Neo 6MV2

3.2.2 PIN DESCRIPTION

- **Vcc**-Supply Voltage
- **Gnd**-Ground pin
- **TX and RX**-These 2 pins acts as an UART interface for communication

3.2.3 INTERFACING GY –GPS6MV2 WITH ARDUINO

Hardware connections

The connections are made as follows:

- Vcc to 5V
- Gnd to Gnd
- TX to Digital pin 3
- RX to Digital pin 4

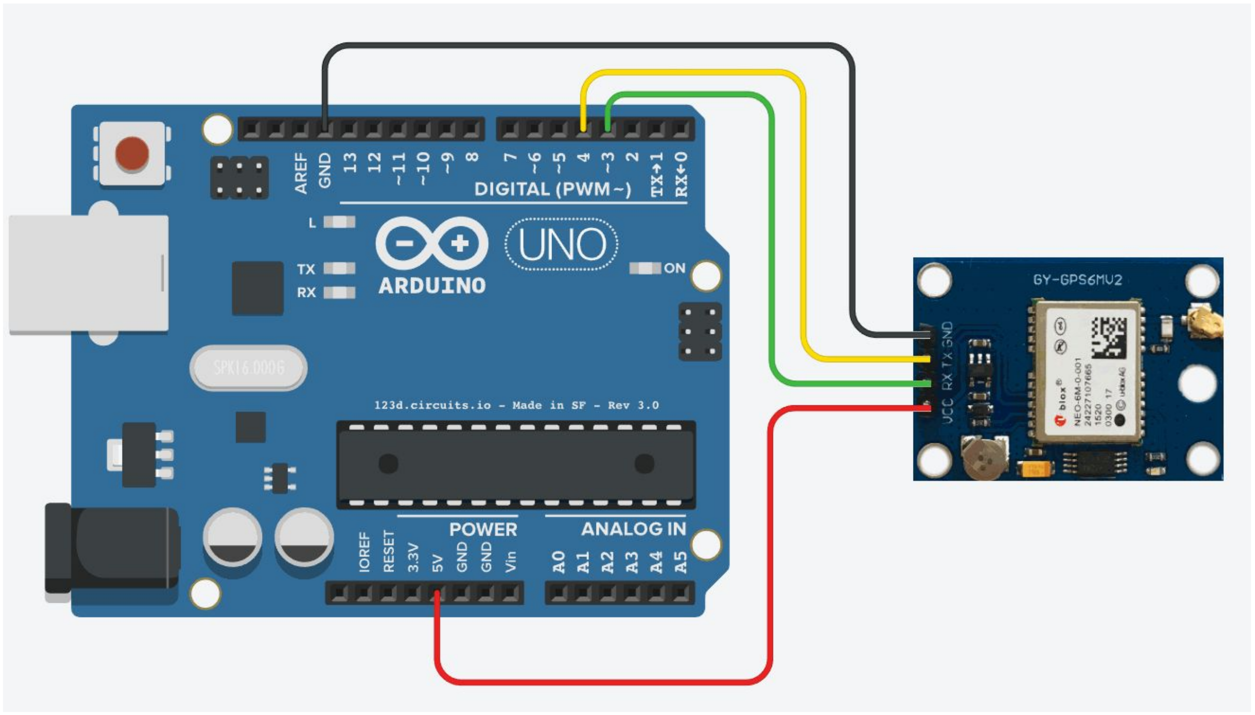


Fig 3.8 Interfacing Neo 6MV2 with Arduino

3.2.4 WORKING

The NEO6MV2 GPS module comes with 4 connections: RX, TX, VCC and GND, which is quite easy to incorporate with using SoftwareSerial on an Arduino Uno or a serial interface on an Arduino Mega. There is only one little problem: The module uses 3v3 logic, which is not compatible with the 5v supplied by Arduino. However, a simple **voltage divider** can solve this issue.

The optimal solution was to use a voltage divider circuit. After few calculations it was decided to use 4.7K as R1 and 10K as R2. Given input voltage of 5V, mentioned voltage divider will give around 3.4V which is well above the minimum logic level.

3.2.5 CAUTION

Be careful and check if input voltage of 5V can be directly applied with the board / module being used). But I/O maximum logic level is rated at 3.6V which might be a problem while connecting the device with Arduino (which works with 5V logic level).

GPS receivers are Electrostatic Sensitive Devices (ESD) and require special precautions when handling. Particular care must be exercised when handling patch antennas, due to the risk of electrostatic charges. In addition to standard ESD safety practices, the following measures should be taken into account whenever handling the receiver: Unless there is a galvanic coupling between the local GND (i.e. the work table) and the PCB GND, then the first point of contact when handling the PCB shall always be between the local GND and PCB GND. Before mounting an antenna patch, connect ground of the device GND Local GND When handling the RF pin, do not come into contact with any charged capacitors and be careful when contacting materials that can develop charges (e.g. patch antenna ~10pF, coax cable ~50- 80pF/m, soldering iron, …) ESD Sensitive! RF_IN To prevent electrostatic discharge through the RF input do not touch the mounted patch antenna. When soldering RF connectors and patch antennas to the receiver's RF pin, make sure to use an ESD safe soldering iron (tip).

3.2.6 About GPS library:TINYGPS++

The library arises from the need to simplify the use of the module, using the NMEA standart. It does not generate excess memory consumption, it is implemented with own functions to handle the pointers of the buffers.

CONSTRUCTORS	**Notes**
Gpsneo ()	Rx , Tx and baudrate by default
Gpsneo (rx, tx)	Rx and Tx set by user. Baudrate by default
Gpsneo (rx, tx, baudrate)	Rx, Tx, baudrate set by user.

Methods	**Notes**
Google (char *link)	Gives a link for google maps.
getDataGPRMC ()	To see the parameters look at the examples
convertLongitude (char *longitude,char *destination)	Converts the longitude given by the read NMEA, to grades.

convertLatitude (char *longitude,char *destination)	Converts the latitude given by the read NMEA, to grades.

How to get data with getDataGPRMC()

$GPRMC - Recommended minimum specific GPS/Transit data
an example of a normal sentence:

- $GPRMC,081836,A,3751.65,S,14507.36,E,000.0,360.0,130998,011.3,E*62
- $GPRMC,225446,A,4916.45,N,12311.12,W,000.5,054.7,191194,020.3,E*68
- 225446 Time of fix 22:54:46 UTC
- A Navigation receiver warning (A = OK, V = warning)
- 4916.45,N Latitude 49 deg. 16.45 min North = 49°16'45"
- 12311.12,W Longitude 123 deg. 11.12 min West = 123°11'12"
- 000.5 Speed over ground, Knots
- 054.7 Course Made Good, True
- 191194 Date of fix 19 November 1994
- 020.3,E Magnetic variation 20.3 deg East
- *68 mandatory checksum

3.3 Arduino Uno:

An Arduino is actually a microcontroller based kit which can be either used directly by purchasing from the vendor or can be made at home using the components, owing to its open source hardware feature. It is basically used in communications and in controlling or operating many devices. It was founded by Massimo Banzi and David Cuartielles in 2005.

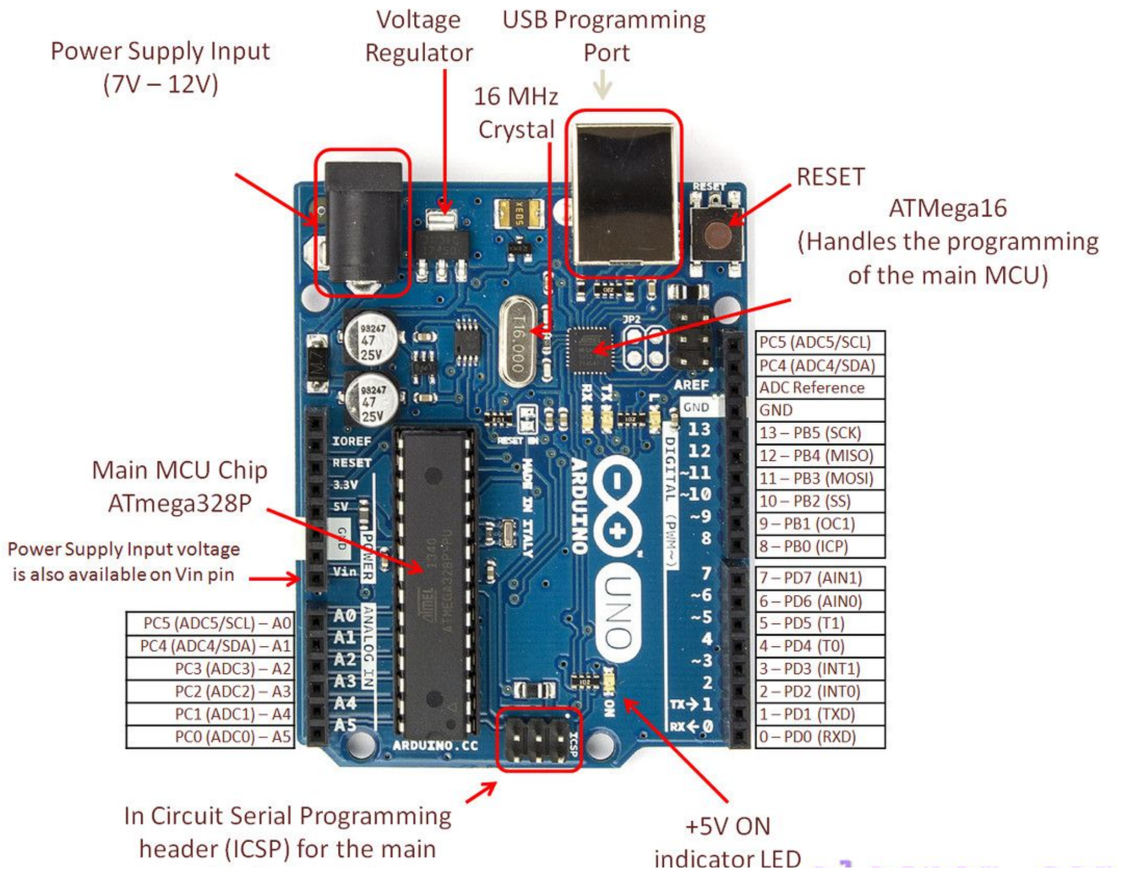


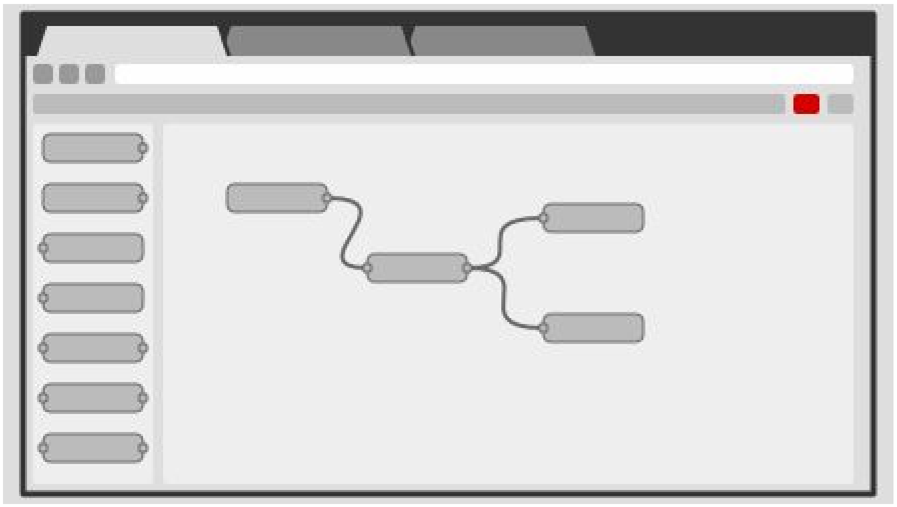
Figure 3.9: Arduino board

Reasons why Arduino is being preferred these days

1. It is inexpensive
2. It comes with an open source hardware feature which enables users to develop their own kit using already available one as a reference source.
3. The Arduino software is compatible with all types of operating systems like Windows, Linux, and Macintosh etc.
4. It also comes with open source software feature which enables experienced software developers to use the Arduino code to merge with the existing programming language libraries and can be extended and modified.
5. It is easy to use for beginners.
6. We can develop an Arduino based project which can be completely stand alone or projects which involve direct communication with the software loaded in the computer.

7. It comes with an easy provision of connecting with the CPU of the computer using serial communication over USB as it contains built in power and reset circuitry.

PART B: SOFTWARE



3.4 NODE-RED

3.4.1 What is Node-Red?

Node-RED is a flow-based programming tool, original developed by IBM's Emerging Technology Services team and now a part of the JS Foundation.

Browser-based flow editing

Node-RED provides a browser-based flow editor that makes it easy to wire together flows using the wide range of nodes in the palette. Flows can be then deployed to the runtime in a single-click.
JavaScript functions can be created within the editor using a rich text editor.
A built-in library allows you to save useful functions, templates or flows for re-use.

Editor UI Widgets About npm

- npm is the package manager for Node.js. It was created in 2009 as an open source project to help JavaScript developers easily share packaged modules of code.
- The npm Registry is a public collection of packages of open-source code for Node.js, front-end web apps, mobile apps, robots, routers, and countless other needs of the JavaScript community.
- npm is the command line client that allows developers to install and publish those packages.
- npm, Inc. is the company that hosts and maintains all of the above.

Built on Node.js

The light-weight runtime is built on Node.js, taking full advantage of its event-driven, non-blocking model. This makes it ideal to run at the edge of the network on low-cost hardware such as the Raspberry Pi as well as in the cloud.
With over 225,000 modules in Node's package repository, it is easy to extend the range of palette nodes to add new capabilities.

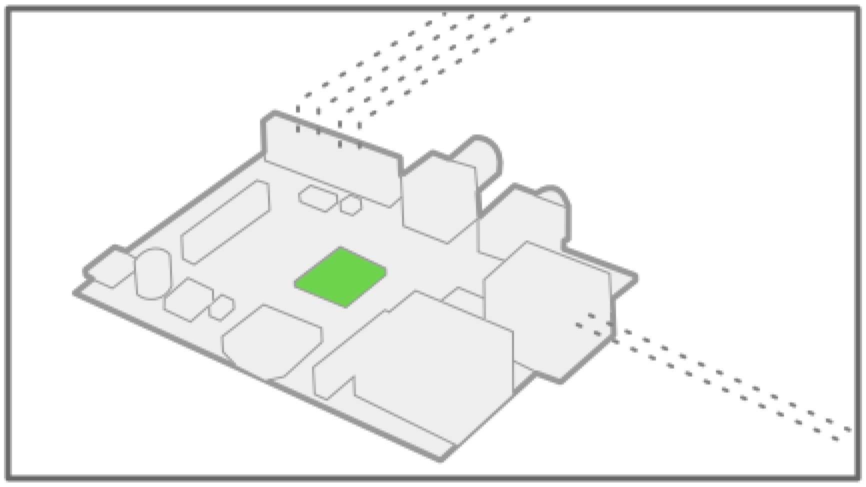


Fig 3.10 Node red and Arduino

3.4.3 API Reference

Admin HTTP API

This HTTP-based API can be used to remotely administer the runtime. It is used by the Node-RED Editor and command-line admin tool.

Runtime API

This API can be used when embedding Node-RED into another application.

Storage API

This API provides a pluggable way to configure where the Node-RED runtime stores data.

A set of jQuery widgets that can be used within a node's edit template. *Since 0.14*

3.4.4 Interacting with Arduino

There are several ways to interact with an Arduino using Node-RED. They all assume the Arduino is connected to the host computer via a USB serial connection.

Note: you can't use both the Arduino IDE and the Arduino nodes at the same time as they will conflict. You will need to stop Node-RED running if you wish re-program the Arduino from the IDE.

Serial

As the Arduino appears as a Serial device, the Serial in/out nodes can be used to communicate with it. In the case if you program the Arduino with the IDE, as you can then send and receive input over the serial port to interact with your creation. Just make sure you set the serial port speed (baud rate) to be the same at both ends.

3.5 Thingspeak:

3.5.1 About ThingSpeak0

ThingSpeak™ is an IoT analytics platform service that allows you to aggregate, visualize and analyze live data streams in the cloud. ThingSpeak provides instant visualizations of data posted by your devices to ThingSpeak. With the ability to execute MATLAB® code in ThingSpeak you can perform online analysis and processing of the data as it comes in. ThingSpeak is often used for prototyping and proof of concept IoT systems that require analytics.

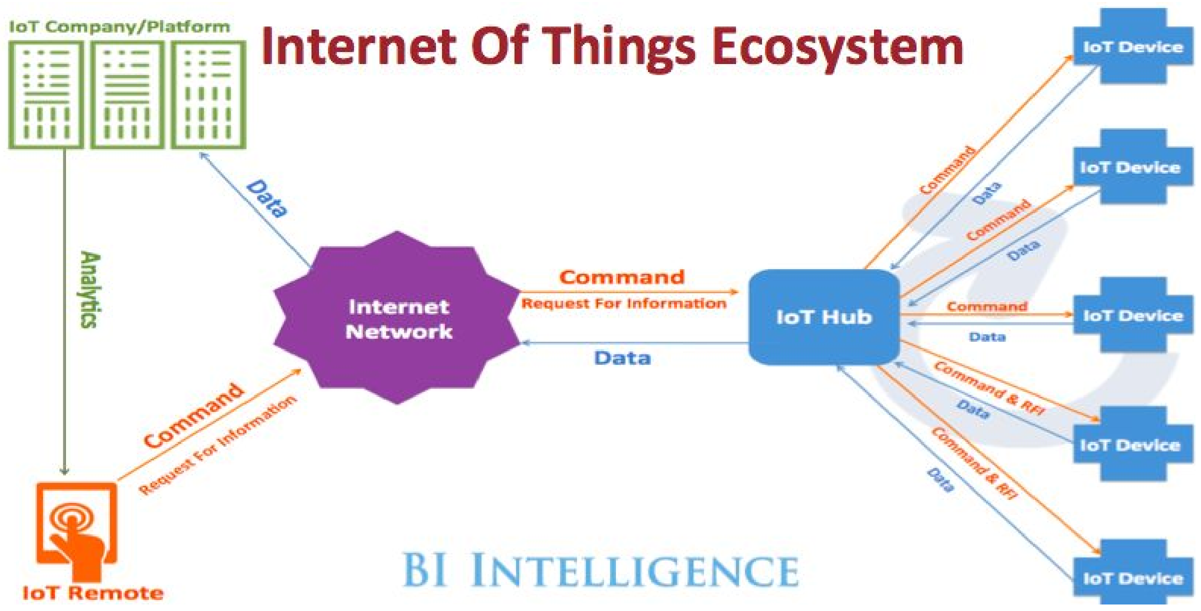


Fig 3.11 Thingspeak system

ThingSpeak Key Features

ThingSpeak allows you to aggregate, visualize and analyze live data streams in the cloud. Some of the key capabilities of ThingSpeak include the ability to:

- Easily configure devices to send data to ThingSpeak using popular IoT protocols.
- Visualize your sensor data in real-time.
- Aggregate data on-demand from third-party sources.
- Use the power of MATLAB to make sense of your IoT data.
- Run your IoT analytics automatically based on schedules or events.
- Prototype and build IoT systems without setting up servers or developing web software.
- Automatically act on your data and communicate using third-party services like Twilio® or Twitter®.

3.5.2 Product Description

The Internet of Things (IoT) provides access to a broad range of embedded devices and web services. ThingSpeak is an IoT platform that enables you to collect, store, analyze, visualize, and act on data from sensors or actuators, such as Arduino®, Raspberry Pi™, BeagleBone Black, and other hardware. For example, with ThingSpeak™ you can create sensor-logging applications, location-tracking applications, and a social network of things with status updates, so that you could have your home thermostat control itself based on your current location.

ThingSpeak acts as the IoT platform for data collection and analytics that serves as a bridge connecting edge node devices such as temperature and pressure sensors to collect data and data exploratory analysis software to analyze data. ThingSpeak serves as the data collector which collects data from edge node devices and also enables the data to be pulled into a software environment for historical analysis of data.

The primary element of ThingSpeak activity is the *channel*, which contains data fields, location fields, and a status field. After you create a ThingSpeak channel, you can write data to the channel, process and view the data with MATLAB® code, and react to the data with tweets and other alerts. The typical ThingSpeak workflow lets you:

1. Create a Channel and collect data
2. Analyze and Visualize the data
3. Act on the data using any of several Apps

3.5.3Channels and Charts API

Use the REST and MQTT APIs to update ThingSpeak™ channels and to chart numeric data stored in channels

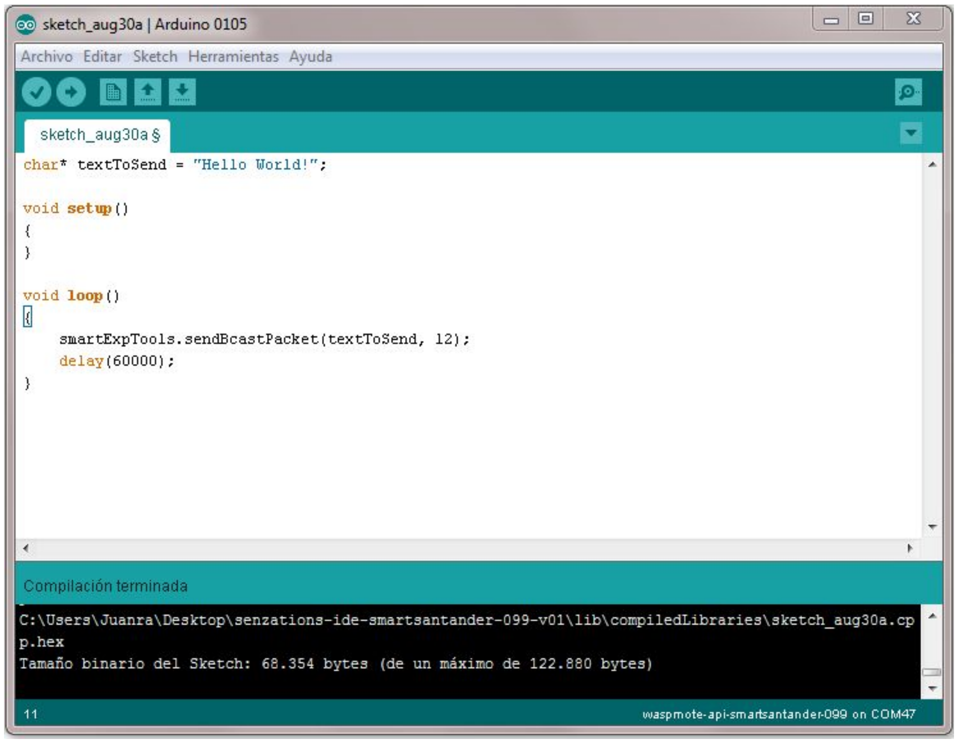
ThingSpeak is an IoT platform that uses channels to store data sent from apps or devices. With the settings described in Channel Configurations, you create a channel, and then send and retrieve data to and from the channel. You can make your channels public to share data. Using the REST API calls such as GET, POST, PUT, and DELETE, you can create a channel and update its feed, update an existing channel, clear a channel feed, and delete a channel. You can also use the MQTT Publish method to update a channel feed. Learn more about when to Choose Between REST API and MQTT API while updating a channel.

MATLAB® analysis and visualization apps enable you to explore and view your channel data. ThingSpeak enables you to interact with social media, web services, and devices.

3.6 Arduino IDE

The Arduino IDE is a cross-platform Java application that serves as a code editor and compiler and is also capable of transferring firmware serially to the board.

The development environment is based on Processing, an IDE designed to introduce programming to artists unfamiliar with software development. The programming language is derived from Wiring, a C-like language that provides similar functionality for a more tightly restricted board design, whose IDE is also based on Processing.


sketch_aug30a | Arduino 0105
Archivo Editar Sketch Herramientas Ayuda
sketch_aug30a §

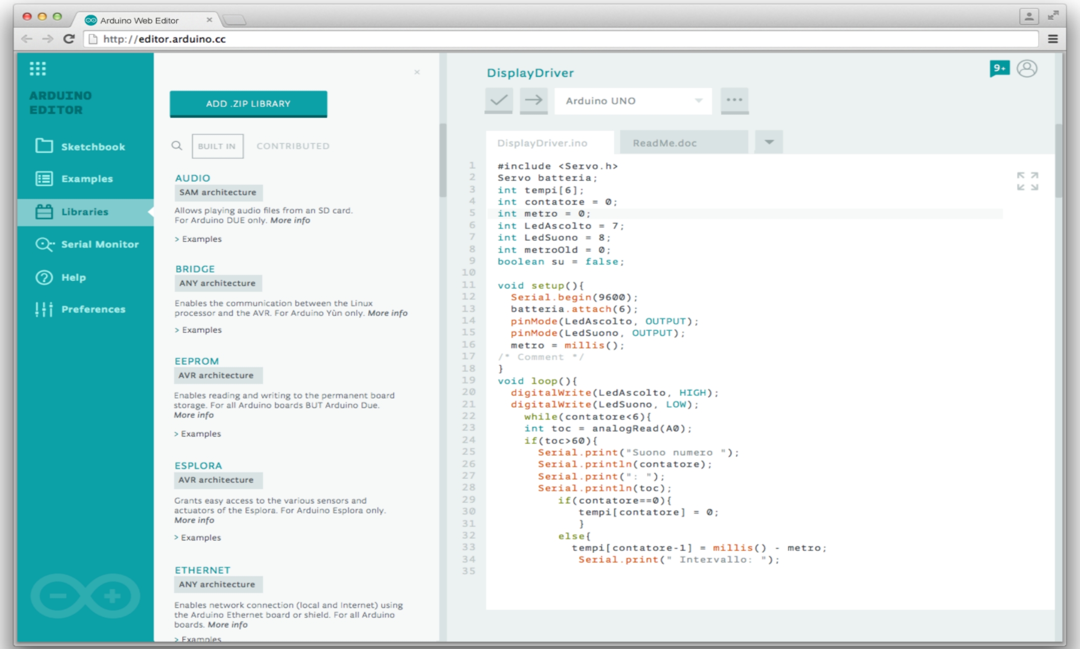
```
char* textToSend = "Hello World!";

void setup()
{
}

void loop()
{
    smartExpTools.sendBcastPacket(textToSend, 12);
    delay(60000);
}
```

Compilación terminada

```
C:\Users\Juanra\Desktop\senzations-ide-smartsantander-099-v01\lib\compiledLibraries\sketch_aug30a.cp
p.hex
Tamaño binario del Sketch: 68.354 bytes (de un máximo de 122.880 bytes)
```

11
waspmote-api-smartsantander-099 on COM47

Fig 3.12 Arduino IDE software



AUDIO
SAM architecture
Allows playing audio files from an SD card. For Arduino DUE only. *More info*
> Examples

BRIDGE
ANY architecture
Enables the communication between the Linux processor and the AVR. For Arduino Yùn only. *More info*
> Examples

EEPROM
AVR architecture
Enables reading and writing to the permanent board storage. For all Arduino boards BUT Arduino Due. *More info*
> Examples

ESPLORA
AVR architecture
Grants easy access to the various sensors and actuators of the Esplora. For Arduino Esplora only. *More info*
> Examples

ETHERNET
ANY architecture
Enables network connection (local and Internet) using the Arduino Ethernet board or shield. For all Arduino boards. *More info*
> Examples

```
#include <Servo.h>
Servo batteria;
int tempi[6];
int contatore = 0;
int metro = 0;
int LedAscolto = 7;
int LedSuono = 8;
int metroOld = 0;
boolean su = false;

void setup(){
  Serial.begin(9600);
  batteria.attach(6);
  pinMode(LedAscolto, OUTPUT);
  pinMode(LedSuono, OUTPUT);
  metro = millis();
/* Comment */
}
void loop(){
  digitalWrite(LedAscolto, HIGH);
  digitalWrite(LedSuono, LOW);
    while(contatore<6){
    int toc = analogRead(A0);
    if(toc>60){
      Serial.print("Suono numero ");
      Serial.println(contatore);
      Serial.print(": ");
      Serial.println(toc);
         if(contatore==0){
            tempi[contatore] = 0;
            }
         else{
           tempi[contatore-1] = millis() - metro;
            Serial.print(" Intervallo: ");
```

Fig 3.13 Arduino online editor

CHAPTER 4

MICROCONTROLLER SPECIFICATIONS

4.1 Criteria For Choosing Microcontroller:

While choosing a microcontroller, make sure it meets the task at hand and that it is cost effective. We must see whether an 8-bit, 16-bit or 32-bit microcontroller can best handle the computing needs of a task. In addition, the following points should be kept in mind while choosing a microcontroller –

Speed – What is the highest speed the microcontroller can support?

Packaging – Is it 40-pin DIP (Dual-inline-package) or QFP (Quad flat package)? This is important in terms of space, assembling, and prototyping the end-product.

Power Consumption – This is an important criteria for battery-powered products.

Amount of RAM and ROM on the chip.

Count of I/O pins and Timers on the chip.

Cost per Unit – This is important in terms of final cost of the product in which the microcontroller is to be used.

4.2 Microcontroller used in Arduino- AT328:

This section discusses the AVR core architecture in general. The main function of the CPU core is to ensure correct program execution. The CPU must therefore be able to access memories, perform calculations, control peripherals, and handle interrupts.

In order to maximize performance and parallelism, the AVR uses a Harvard architecture – with separate memories and buses for program and data. Instructions in the program memory are executed with a single level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This concept enables instructions to be executed in every clock cycle. The program memory is In-System Reprogrammable Flash memory.

The fast-access Register File contains 32 x 8-bit general purpose working registers with a single clock cycle access time. This allows single-cycle Arithmetic Logic Unit (ALU) operation. In a typical ALU operation, two operands are output from the Register File, the operation is executed, and the result is stored back in the Register File – in one clock cycle.

Six of the 32 registers can be used as three 16-bit indirect address register pointers for Data Space addressing – enabling efficient address calculations. One of the these address pointers can also be used as an address pointer for look up tables in Flash program memory. These added function registers are the 16-bit X-, Y-, and Z-register, described later in this section.

4.3 Features Of ATmeg328 Microcontroller:

Operating Voltage: 5V.
Input Voltage: 7-12V.
Digital I/O Pins: 14 (of which 6 provide PWM output).
Analog Input Pins: 6.
DC Current: 40 mA.
DC Current: 50 mA.
Flash Memory: 32 KB.
SRAM: 2 KB.
EEPROM: 1 KB.
Clock Speed: 16 MHz.

4.4 Block Diagram Of AT328 CPU:

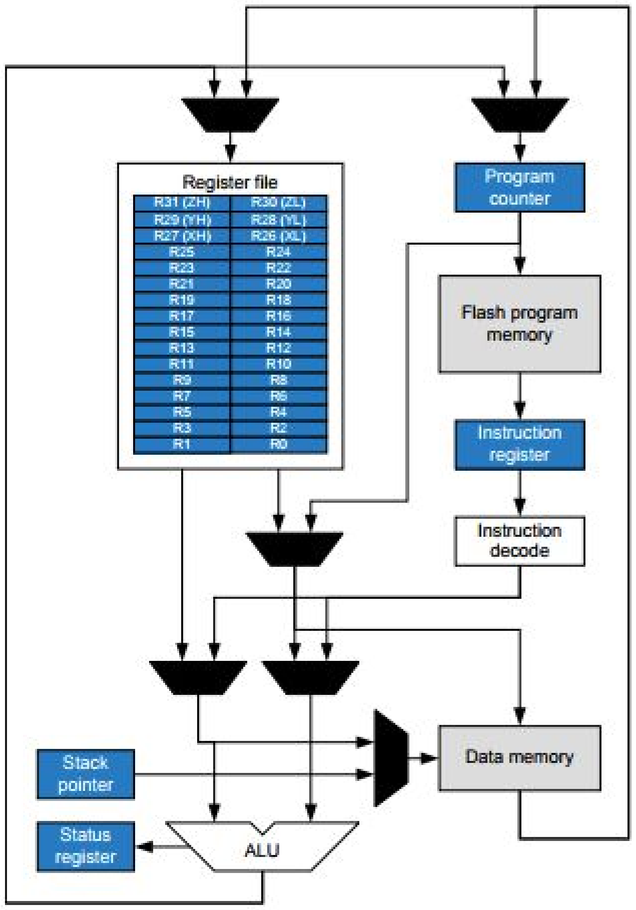


Figure 4.1:Block Diagram of 8051 Microcontroller

4.5 Pin Diagram Of AT89S51 Microcontroller:

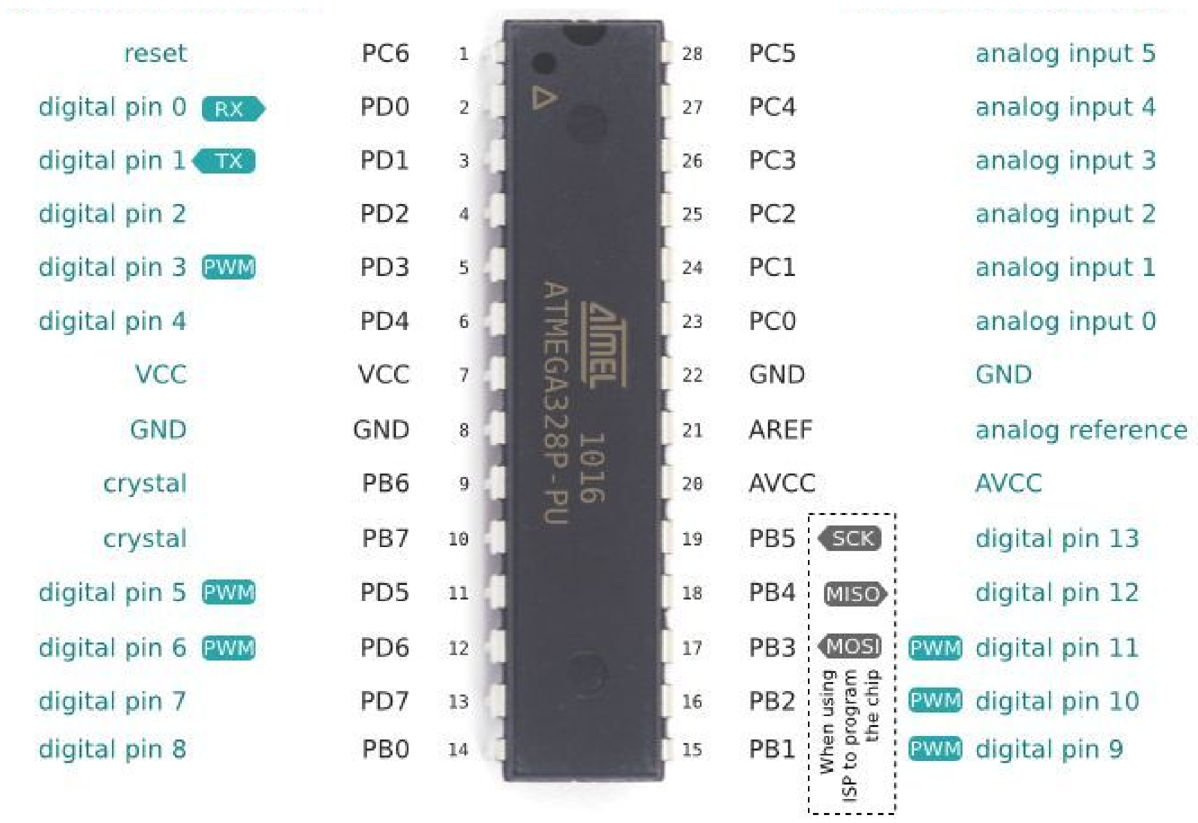


Figure 4.2: Pin diagram of AT328 Microcontroller

CHAPTER 5

WORKING OF PROJECT

Atmel offers a variety of high performance Flash 8051 microcontrollers featuring in-system programming (ISP). Their integrated Flash memory can be programmed either in parallel mode or in serial mode with the appropriate parallel/ISP software, respectively.
They feature a wide range of internal RAM configurations, plus rich features such as interrupt controllers and timer/counters. Select models feature an A/D converter, Boot Flash memory, and Programmable Counter Arrays. Their highly efficient design can help reduce system power consumption by bringing the clock frequency down to any value — even DC — without loss of data.
Software-selectable modes of reduced activity and an 8 bit clock prescaler can further control power consumption.
The rich functionality and efficiency of Atmel 8051 Flash ISP microcontrollers make them ideal for applications that need A/D conversion, pulse width modulation, high speed I/O and counting capabilities such as industrial control, consumer goods, alarms, motor control, and more.

5.1 Project Code:

Arduino Code:

```
#include <TinyGPS++.h>

#include <SoftwareSerial.h>

///* Create object named bt of the class SoftwareSerial */

SoftwareSerial GPS_SoftSerial(4,3);/* (Rx, Tx) */

/* Create an object named gps of the class TinyGPSPlus */

TinyGPSPlus gps;


volatile float minutes, seconds;

volatile int degree, secs, mins;


const int MQ3AOUTpin= 0;                    //the AOUT pin of the CO sensor goes into
analog pin A0 of the arduino

const int MQ3DOUTpin= 8;                    //the DOUT pin of the CO sensor goes into
analog pin A0 of the arduino
```

```
int value;

void setup()
{
  Serial.begin(57600); /* Define baud rate for serial communication */
  GPS_SoftSerial.begin(9600); /* Define baud rate for software serial communication */


pinMode(MQ3AOUTpin, INPUT);                      //sets the pin as an input to the arduino
pinMode(MQ3DOUTpin, INPUT);                       //sets the pin as an input to the arduino
}

void loop() {
        smartDelay(1000); /* Generate precise delay of 1ms */
        unsigned long start;
        double lat_val, lng_val, alt_m_val;
        uint8_t hr_val, min_val, sec_val;
        bool loc_valid, alt_valid, time_valid;
        lat_val = gps.location.lat(); /* Get latitude data */
        loc_valid = gps.location.isValid(); /* Check if valid location data is available */
        lng_val = gps.location.lng(); /* Get longtitude data */
        alt_m_val = gps.altitude.meters();  /* Get altitude data in meters */
        alt_valid = gps.altitude.isValid(); /* Check if valid altitude data is available */
        hr_val = gps.time.hour(); /* Get hour */
        min_val = gps.time.minute();  /* Get minutes */
        sec_val = gps.time.second();  /* Get seconds */
        time_valid = gps.time.isValid();  /* Check if valid time data is available */

        value= analogRead(MQ3AOUTpin);                  //reads the analaog value from the
sensor's AOUT pin
Serial.print("\nal value ");
Serial.print( value);                            //prints the sensor value of alcohol
```

```
        if (!loc_valid)
        {
          Serial.print("Latitude : ");
          Serial.println("*****");
          Serial.print("Longitude : ");
          Serial.println("*****");
        }
        else
        {
          DegMinSec(lat_val);

          Serial.print(" Latitude ");
          Serial.print(lat_val, 6);

          DegMinSec(lng_val); /* Convert the decimal degree value into degrees minutes seconds
form */

          Serial.print(" Longitude ");
          Serial.print(lng_val, 6);

        }


}

static void smartDelay(unsigned long ms)
{
  unsigned long start = millis();
  do
  {
    while (GPS_SoftSerial.available())  /* Encode data read from GPS while data is available
on serial port */
      gps.encode(GPS_SoftSerial.read());
/* Encode basically is used to parse the string received by the GPS and to store it in a
buffer so that information can be extracted from it */

  } while (millis() - start < ms);

}


void DegMinSec( double tot_val)   /* Convert data in decimal degrees into degrees minutes
seconds form */

{

  degree = (int)tot_val;

  minutes = tot_val - degree;

  seconds = 60 * minutes;

  minutes = (int)seconds;

  mins = (int)minutes;

  seconds = seconds - minutes;

  seconds = 60 * seconds;

  secs = (int)seconds;

}
```

NODE RED json code

Refer to the website

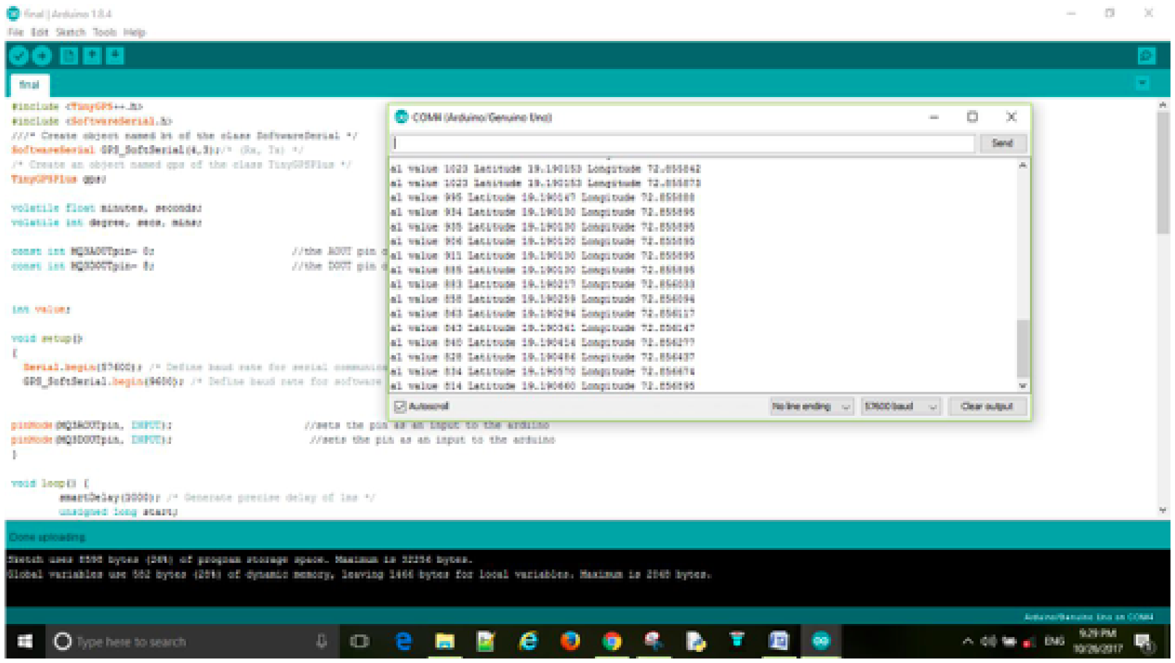
5.2 Working of project:

The project consists of two sections .One in which Arduino collects the data from MQ3 sensor and GPS module and transferring the required data to Node red for further processing. Second part consists of displaying data on server.

1) Sensing the alcohol from atmosphere

Arduino microcontroller will sense alcohol value through digital output pin of MQ3 sensor and also receives the Latitude and Longitude coordinates from GPS module serially at pin no.4 at each and every seconds. Then it will sends the alcohol value ,latitude value and longitude value in string format separated by space serially to node red.

Output on Serial Monitor of Arduino IDE :

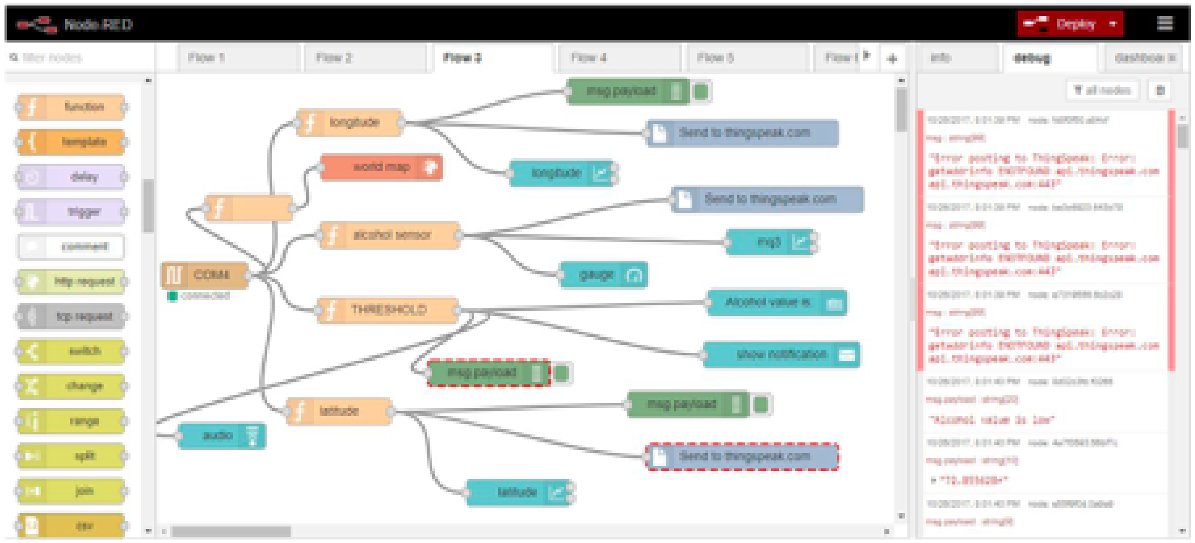


```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
///* Create object named bt of the class SoftwareSerial */
SoftwareSerial GPS_SoftSerial(4,3);/* (Rx, Tx) */
/* Create an object named gps of the class TinyGPSPlus */
TinyGPSPlus gps;

volatile float minutes, seconds;
volatile int degree, secs, mins;

const int MQ3AOUTpin= 0;                    //the AOUT pin o
const int MQ3DOUTpin= 8;                    //the DOUT pin o


int value;

void setup()
{
  Serial.begin(57600); /* Define baud rate for serial communica
  GPS_SoftSerial.begin(9600); /* Define baud rate for software


pinMode(MQ3AOUTpin, INPUT);                   //sets the pin as an input to the arduino
pinMode(MQ3DOUTpin, INPUT);                    //sets the pin as an input to the arduino
}

void loop() {
        smartDelay(1000); /* Generate precise delay of 1ms */
        unsigned long start;
```

```
al value 1023 Latitude 19.190153 Longitude 72.855842
al value 1023 Latitude 19.190153 Longitude 72.855873
al value 995 Latitude 19.190147 Longitude 72.855888
al value 934 Latitude 19.190130 Longitude 72.855895
al value 935 Latitude 19.190130 Longitude 72.855895
al value 956 Latitude 19.190130 Longitude 72.855895
al value 911 Latitude 19.190130 Longitude 72.855895
al value 885 Latitude 19.190130 Longitude 72.855895
al value 883 Latitude 19.190217 Longitude 72.856033
al value 858 Latitude 19.190259 Longitude 72.856094
al value 863 Latitude 19.190294 Longitude 72.856117
al value 843 Latitude 19.190341 Longitude 72.856147
al value 840 Latitude 19.190414 Longitude 72.856277
al value 828 Latitude 19.190486 Longitude 72.856437
al value 834 Latitude 19.190570 Longitude 72.856674
al value 814 Latitude 19.190660 Longitude 72.856795
```

```
Done uploading.
Sketch uses 5598 bytes (26%) of program storage space. Maximum is 32256 bytes.
Global variables use 582 bytes (28%) of dynamic memory, leaving 1466 bytes for local variables. Maximum is 2048 bytes.
```

Node Red : Flow Diagram



```
"Error posting to ThingSpeak: Error: getaddrinfo ENOTFOUND api.thingspeak.com api.thingspeak.com:443"
"Error posting to ThingSpeak: Error: getaddrinfo ENOTFOUND api.thingspeak.com api.thingspeak.com:443"
"Error posting to ThingSpeak: Error: getaddrinfo ENOTFOUND api.thingspeak.com api.thingspeak.com:443"
"Alcohol value is low"
"72.855628"
```

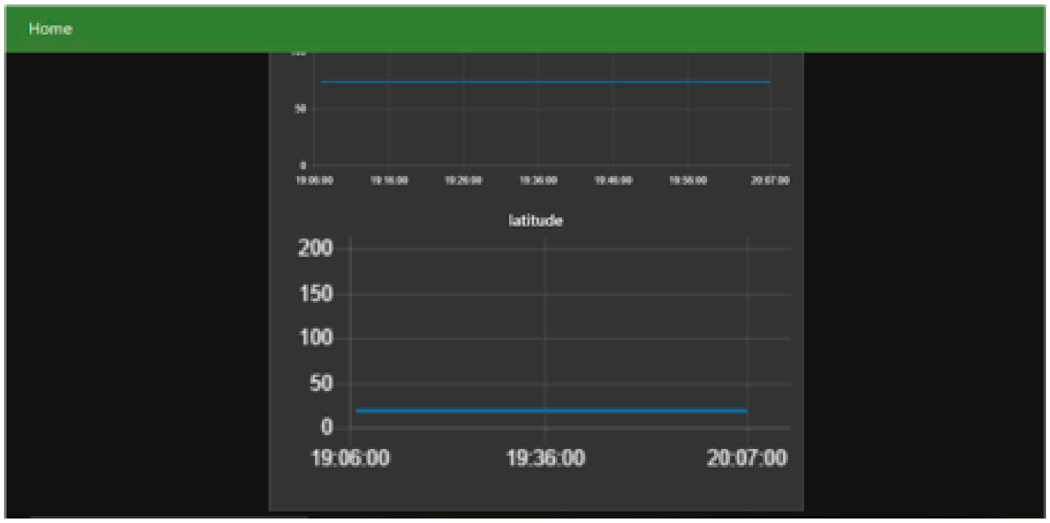
2) Processing and displaying of data

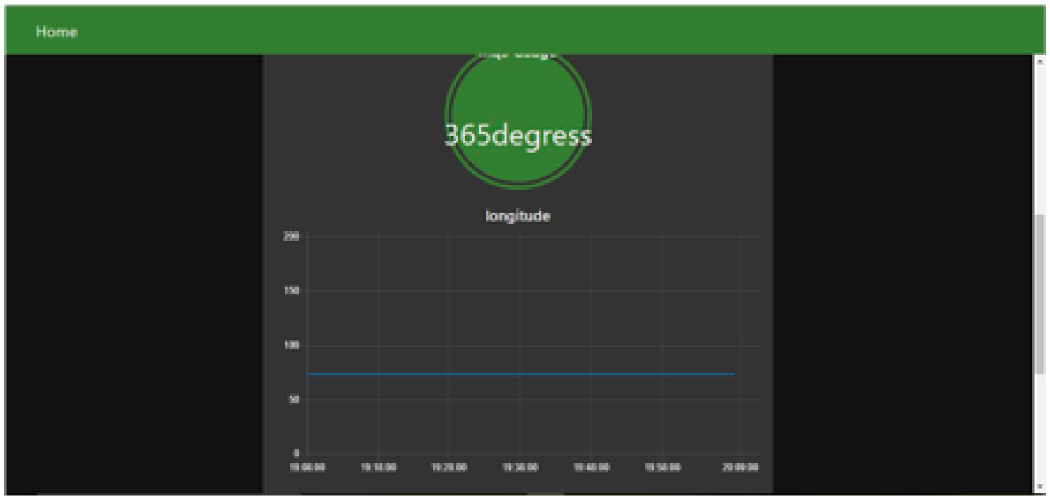
As soon as the data becomes available at serial port of local host serial node will proceeds the data i.e.String in payload form. Then it will given to three function node for fetching latitude value, longitude value and alcohol value using javascript language respectively. Also one another node is used to make decision on whether the alcohol has crossed the threshold or not.

Also at server side one can see the real time looging of data at local host server of node red dashboard and can trace the person on local google map.

Output on Node Red Dashboard: (http://localhost:1880/ui/#/1)

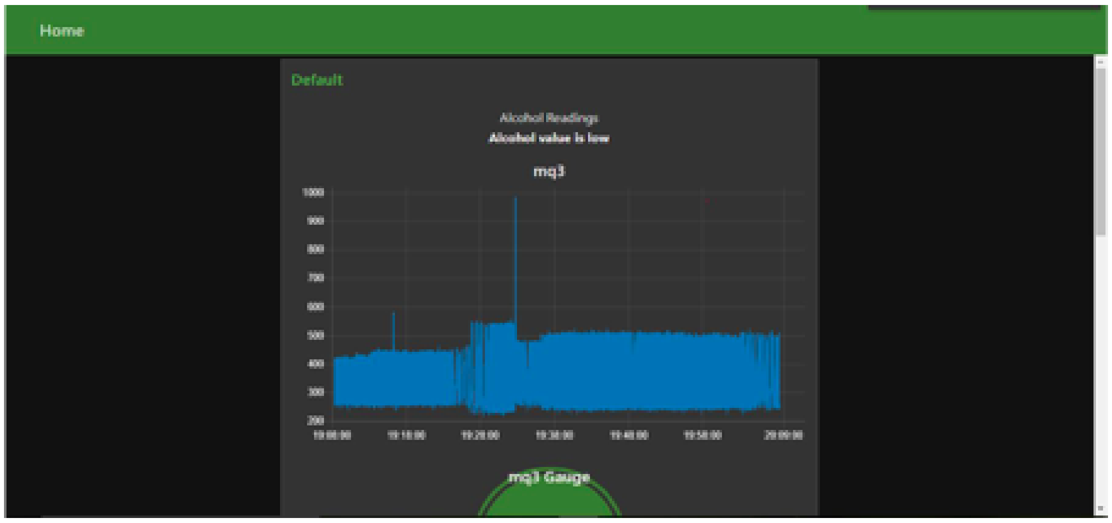
1)Latitude Value :



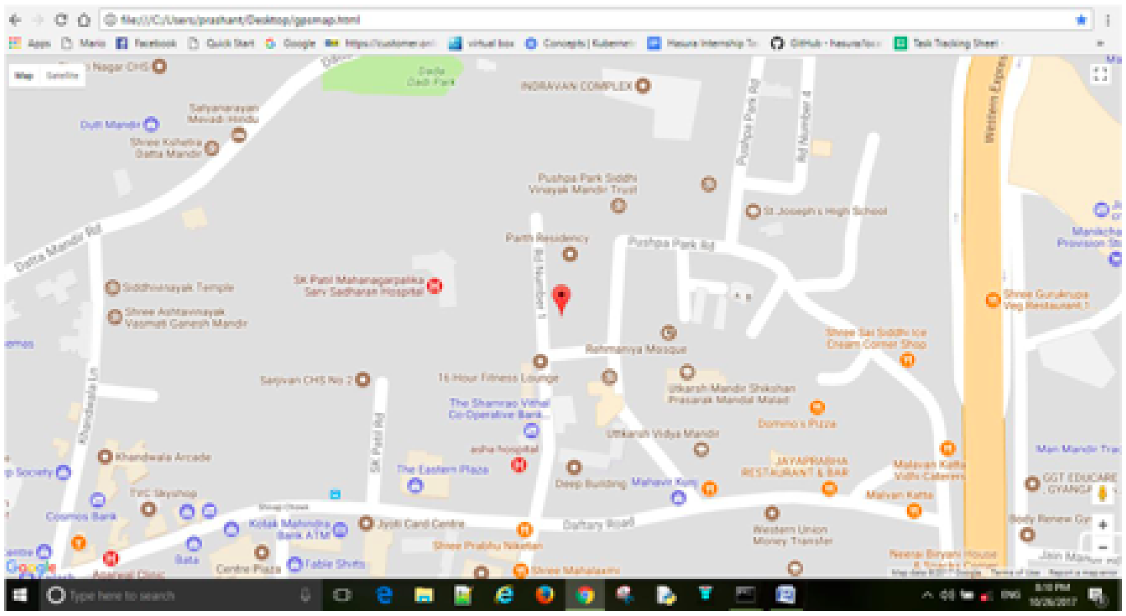


2)Longitude Value

3)Alcohol Value Of MQ3:



5)Output on Google Map Running on Localhost (file:///C:/Users/prashant/Desktop/gpsmap.html):



After processing the data it will sent to Thingspeak node to upload the data i.e.Latitude value ,Longitude value and MQ3 value on Thingspeak.com using GET request .Where using public or private view one can observe and analyse the data over internet

4)Output On ThingSpeak WEBSITE: (https://thingspeak.com/channels/******/private_show)

CHAPTER 6

ADVANTAGE AND DISADVANTAGE,LIMITATIONS AND FUTURE SCOPE

8.1 ADVANTAGES:

- We are using a node red which is a future of industrial iot.
- We can easily interface microcontroller with cloud and view on web page like freeboard or Thingspeak.

8.2 DISADVANTAGES:

- Needs to be connected wirelessly to the internet whereas now it's connected serially.
- Circuit component is very much costly for a single product,but for mass production soc will reduced the cost very effectively.

8.3 Limitation:

- GPS sensor needs enough open sky.

8.4 APPLICATION:

1. Get real time updates on every moment of our car on android phone or web page.
2. We can find out our car easily in giant parking space.
3. We will get instant alert updates of whether driver is drunk or not.

8.5 Future Scope:

➔ 1) We can add more sensor for analysis of temperature and humidity inside engine, speed of the car, level of petrol tank etc.

➔ 2) We can create a virtual boundary i.e. geo fence to protect our car from theft and will get an instant alerts.

CHAPTER 7
REFERENCES

Websites:

https://nodered.org/

https://thingspeak.com

https://arduino.cc