

**SRI DHARMASTHALA MANJUNATHESHWARA COLLEGE  
(AUTONOMOUS)**

**Ujire – 574240**

**DEPARTMENT OF B.VOC  
(SOFTWARE AND APPLICATION DEVELOPMENT)**



**PROJECT REPORT ON  
“SMART RIDING HELMET BASED ON CLOUD AND INTERNET OF  
THINGS”**

**SUBMITTED BY:**

**ABHISHEK U S**

**211301**

**UNDER THE GUIDENCE OF**

**Mr Sammed Jain**

**Department of B.Voc**

**Software and App Development**

**SRI DHARMASTHALA MANJUNATHESHWARA COLLEGE  
(AUTONOMOUS)**

**Ujire – 574240**

**DEPARTMENT OF B.VOC  
(SOFTWARE AND APPLICATION DEVELOPMENT)**



**Certificate for the approval of project report**

This is to certify the following student of 3<sup>rd</sup> B.Voc has satisfactorily completed the project on  
**“SMART RIDING HELMET BASED ON CLOUD AND INTERNET OF THINGS”**  
for the Bachelor of Vocational Course Prescribed by the college during academic year 2023-2024.

**ABHISHEK U S  
211301**

(Mr. Sammed Jain)  
**Head of the Department**

(Mr. Sammed Jain)  
**Project Guide**

(Dr. Kumara Hegde)  
**Principal**

**Examiners:**

- 1.
- 2.

## **DECLARATION**

I hereby declare that this project work entitled “**SMART RIDING HELMET BASDE ON CLOUD AND INTERNET OF THINS**” has been prepared by us during the year 2023 – 2024 under the guidance of Assistant Professor **Mr. Sammed Jain** Department of software and app development, SDM College in the partial fulfilment of B. Voc degree prescribed by the college.

I also declare that this project is the outcome of our own efforts, that it has not been submitted anywhere else for the award of any degree.

**ABHISHEK U S**  
**211301**

## **ACKNOWLEDGEMENT**

I would like to thank our principal **Dr. Kumara Hegde**, for his support. I also thank HOD, Dept. of B.Voc in Software and Application Development, SDM College Ujire, for his valuable suggestions.

I would like to thank Assistant Professor **Mr. Sammed Jain** Department of B. Voc in Software and Application Development, SDM College Ujire, for their help and for providing guidance in developing the project.

**ABHISHEK U S**

**211301**

## INDEX

SL.NO	TITLE	PAGE NO.
1.	Project Synopsis	1-3
	1.1 Title of the project	
	1.2 Abstract	
	1.3 Introduction	
	1.4 Existing System	
	1.5 Proposed System	
	1.6 Hardware Requirement	
	1.7 Software Requirement	
2.	Introduction	4
	2.1 Scope	5
	2.2 Project Management	5-6
	2.2.1 Experimentation	
	2.2.2 Design	
	2.2.3 Development and testing	
	2.2.4 Real world testing	
	2.3 Overview and Benefits 2.3.1 Impact on community	
3.	Literature Review	7
	3.1 Abstract	
	3.2 Literature Survey	
4.	Software Requirement Specification	8
	4.1 Hardware Requirements	
	4.2 Software Requirements	9
	4.3 Hardware Component	10-19

	4.4 Software Requirements	20-28
	4.4.1 Arduino IDE	
	4.4.2 Installation of Arduino IDE	
	4.4.3 Launch Arduino IDE	
	4.4.4 IoT Cloud	
5.	Implementation	29-37
	5.1 IOT – Internet of things	
	5.1.1	
	5.1.2 IOT Architecture	
	5.1.3	
6.	5.1.4 IOT Features	38-43
	5.2 Block Diagram	
	System Design Specification	
	6.1 Design Approach	
	6.2 Overview	
	6.3 Data flow diagram	
	6.3.1 Symbols and notation used in DFD	
7.	6.3.2 Data flow diagram	44-47
	6.4 Circuit Diagram	
7.	7.1Screenshots	44-47
8	Testing	45-53
	8.1 Testing Approaches	
	8.2 Test cases	
	8.3Test Case Screenshots	
9	Source Code	54-62

10	Certificate	63
11	Conclusion	64
	10.1 Future Enhancement	65
12.	Bibliography	66

# PROJECT SYNOPSIS

## 1.1 Title of the project:

“SMART RIDING HELMET BASED ON CLOUD AND INTERNET OF THINGS”

## 1.2 Abstract

The main objective of this paper is to build a safety system which is integrated with the smart helmet and intelligent bike to reduce the probability of two-wheeler accidents and drunken drive cases. The push button checks if the person is wearing the helmet or not. Alcohol sensor detect the alcoholic content in riders' breath. If the rider is not wearing the helmet or if there is any alcohol content found in rider's breath, the bike remains off. The bike will start when the rider wears the helmet and if there is no alcoholic content present.

When the rider crashes, helmet hits the ground, accelerometer detects the motion and tilts of helmet and reports the occurrence of an accident.

In order to avoid the death and rash driving, drunk and drive, this project will be useful for the people.

## 1.3 Introduction

As the technology developed the rate of mishap is also increase. The riders avoid wearing helmet without any specific reason. Moreover, over speeding and drink and drive have become common issues. Due to the lack of experience or focus and violation of traffic rules, which leads to accidents So, with the help of technology we made sure that traffic rules are followed, problems mentioned above are avoided and their effects are minimized. The idea of developing this work comes from our social responsibility towards society. In many accidents that occur, there is a huge loss of life.

Many people die on roads every year that occur due to bike accidents. There are various reasons for accidents such as not having adequate ability to drive, defective two wheelers, rash driving, drink and drive, etc. But the main reason was the absence of helmet on the person which leads to immediate death due to brain damage. Therefore, it is important that there should be a facility to minimize the after effects of these accidents. However, the main goal of our work is to make it mandatory for the rider to wear a helmet during the ride, to prevent drink and drive scenario and over speeding or rash riding by motorcyclists and also provide proper medical attention when met with accident by alerting the concerned person which will provide solutions to other major issues for accidents



## **1.4 Existing System**

An important part of the accidents happens because the individual was either not wearing a helmet, or the accident was not revealed in time, or the person couldn't be safe in view of the late induction to an emergency clinic, or on the grounds that the person was riding while smashed. Sensors distributed, Wi-Fi empowered processor, and computing foundations are used for building the structure. The accident discovery is finished utilizing the accelerometer and the accident warning is finished utilizing the customer and server-based framework where the microcontroller is the customer and the server is an online administration. At the point when an accident happens, the related subtleties are sent to the crisis contacts by using a cloud-based administration.

The disadvantages of existing system are

- 1) Less exactness in the location of accidents, and
- 2) There is no framework to check if the rider is wearing the helmet or not.

## **1.5 Proposed System**

This paper describes the prototype of smart helmet using IOT, which ensures the safety and security of the bike rider. Here the system is responsible for the following functionalities.

- The system will not allow the rider to start the vehicle, if the rider is not wearing the helmet.
- It detects the consumption of alcohol, if the rider has consumed alcohol, the bike engine will not start.
- The system alerts the rider when the speed exceeds the limited value.
- When met with an accident it detects it and gives the notification to the registered contact with a location and picture information

## **1.6 Hardware Requirement**

- Arduino Uno
- Push Button
- RF Transmitter
- RF Receiver
- Alcohol Sensor
- Accelerometer
- ESP-01
- Ignition
- Relay
- GPS

## **1.7 Software Requirement**

- Arduino IDE
- IoT Cloud
- Embedded C Programming

## **1.8 DURATION OF THE PROJECT**

- Three Month

## **1.9 MEMBER OF THE PROJECT**

- Four Member

# INTRODUCTION

Internet of Things (IOT) is a concept where each device is assigned to an IP address and through that IP address anyone makes that device identifiable on internet. The mechanical and digital machines are provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction. The resulting network is called the “Internet of Things” (IoT). The recent developments in technology which permit the use of wireless controlling environments like, Bluetooth and Wi-Fi that have enabled different devices to have capabilities of connecting with each other. Using a WIFI shield to act as a Micro web server for the Arduino which eliminates the need for wired connections between the Arduino board and computer which reduces cost and enables it to work as a standalone device.

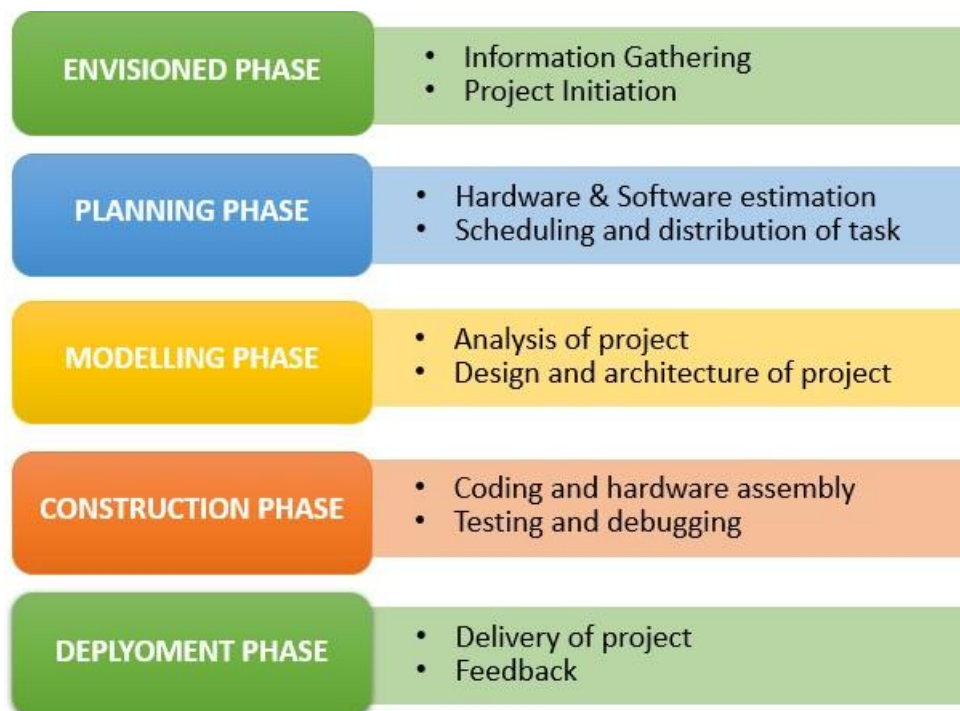
This project will monitor the indoor climate atmospheric and mechanical environments of commercial or public buildings, according to the World Health Organization (WHO). The temperature, humidity, air quality and lighting conditions of indoor space are highly related to human health, comfort perception, well-being, and work productivity. A plant wall system involves growing diverse types of green plants on a vertically supported system that is attached to an internal or external wall or is designed as a standalone product. A plant wall system consists of vegetation, growing medium and irrigation and drainage systems. In addition to the initial aesthetic decoration, green plants make significant contributions to indoor environments via evaporation, air purification and water retention, which improves indoor climates and reduce energy use.

## 2.1 Scope

To monitor the indoor climate atmospheric and mechanical environments of commercial or public buildings is the task of our project. So, by planting the plants vertical on the wall and using the Arduino code we are monitoring plants to grow healthy.

## 2.2 Project Management

Management of any project can be briefly disintegrated into several phases. Our project has been decomposed into following phases:



### 2.2.1 Experimentation

This phase involved discussions regarding necessary equipment for the project. The study of related already existing projects, gathering required theoretical learning. It also included figuring out the coding part, by developing simple algorithms and flowcharts to design the whole process.

### 2.2.2 Design

This phase involved the complete hardware assembly and installing the code to Arduino-uno and necessary features to be included.

### **2.2.3 Development and testing**

This phase had the development of the project. The microcontroller was connected to the Thingspeak cloud platform via wireless network (Wi-Fi) and the whole prototype was tested for identification and removal of bugs.

### **2.2.4 Real world testing**

The prototype was ready to be tested into the real world and integrated with various real time applications.

## **2.3 Overview and Benefits**

Active Wall plants are designed to perform certain tasks. An essential part is to maintain temperature and humidity of indoor climate which really effects on human health condition.

### **2.3.1 Impact on Community**

- Provide a good environment for the people.
- Provide facility to plant wall supplier's to make job easy.
- To explore the community about new technology.
- To create more better life community.
- Encourage everyone to innovate new technology.

# LITERATURE REVIEW

## 3.1 Abstract

The main objective of this paper is to build a safety system which is integrated with the smart helmet and intelligent bike to reduce the probability of two-wheeler accidents and drunken drive cases. The push button checks if the person is wearing the helmet or not. Alcohol sensor detect the alcoholic content in riders' breath. If the rider is not wearing the helmet or if there is any alcohol content found in rider's breath, the bike remains off. The bike will start when the rider wears the helmet and if there is no alcoholic content present.

When the rider crashes, helmet hits the ground, accelerometer detects the motion and tilts of helmet and reports the occurrence of an accident.

In order to avoid the death and rash driving, drunk and drive, this project will be useful for the people.

## 3.2 Literature Survey

Jennifer William et al. [1] proposed system in which the intelligent helmet ensures the safety of the biker by making it necessary to wear the helmet and assure that rider hasn't consume any alcohol while driving the vehicle. The system also helps in efficient handling of aftermath of accident by sending a SMS with the location of the biker to person's well-wisher's number to get proper and prompt medical attention, after meeting with an accident.

C. Prabha et al. [2] introduced system which can be used as a crash or rollover detector of the vehicle during and after a crash. When the vehicle meets with an accident the sensor in the system detects and alert system with SMS to the user defined mobile number. The GPS tracking and GSM alert-based algorithm is designed and implemented in embedded system domain. The proposed vehicle accident detection system traces the location of the rider automatically and sends an alert message regarding accident.

# SOFTWARE REQUIREMENT SPECIFICATION

Requirements specification is a specification of software requirements and hardware requirements required to do the project. Requirements analysis encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product or project, taking account of the possibly conflicting requirements of the various stakeholders, analyzing, documenting, validating, and managing software or system requirements.

## 4.1 Hardware Requirements

Sl. No	Hardware / Equipment	Specification
1	Push Button	Push button switch is a mechanical device that controls an electrical circuit
2	Alcohol Sensor	Gas sensor which detects the presence of alcohol content gas
3	RF Transmitter	RF transmitter is ideal for remote control applications
4	RF Receiver	RF Receiver is ideal for short-range remote-control applications
5	Accelerometer	Accelerometer is a device that measures the vibration
6	ESP-01	Wi-Fi Module.
7	Relay	Relay Module.
8	Arduino Uno	Arduino UNO is the best board to get started with electronics and coding
9	GPS	GPS is a space-based satellite navigation system
10	Ignition	Ignition key is a key used to start a vehicle's engine by turning the switch

## 4.2 Software Requirements

Sl. No	Software	Specification
1	Software	Arduino IDE
2	Language	C++
3	IoT Cloud	Thing Speak



## 4.3 Hardware Components

### 4.3.1 Push Button

A push button switch is a mechanical device that controls an electrical circuit by opening or closing it with the press of a button. Push buttons are often used for simple circuit changes, and can be either normally open (NO) or normally closed (NC). When activated, normally open switches complete the circuit, while normally closed switches break the circuit. Push buttons are also known as push switches.



Figure 4.3.1 Push button

#### Features:

- Latching: The switch remains on until you press it again to turn it off.
- Electrical specifications: The switch's voltage and current rating.
- Environmental resistance: The switch's resistance to water, dust, and temperature.
- Force required to activate: The amount of force needed to activate the switch.
- Button feel: The desired feel of the button.
- Aesthetic design: The button's appearance.

### 4.3.2 Alcohol sensor

It is a gas sensor which detects the presence of alcohol content gas concentration from 0.05 mg/L to 10 mg/L. It is highly sensitive to alcohol and a low cost semiconductor sensor which provides fast response and gives both digital and Analog output.



Figure 4.3.2 Alcohol sensor

#### Features:

- Sensitivity: Alcohol sensors can have high sensitivity to alcohol vapors
- Response time: Alcohol sensors can have a fast response time
- Output types: Alcohol sensors can have analog and digital outputs
- Output ranges: Alcohol sensors can have output ranges from 0V to 5V or 0V or 5V
- Sensing range: Alcohol sensors can have a sensing range from 0.04 mg/L to 4 mg/L
- On-board LED indicator: Alcohol sensors can have an on-board LED indicator
- Preheat duration: Alcohol sensors can have a preheat duration of 20 seconds

### 4.3.3 RF Transmitter

RF transmitter is ideal for remote control applications where low cost and longer range is required. The transmitter operates from a 1.5-12V supply, making it ideal for battery-powered applications. The transmitter employs a SAW-stabilized oscillator, ensuring accurate frequency control for best range performance. Output power and harmonic emissions are easy to control, making FCC and ETSI compliance easy.

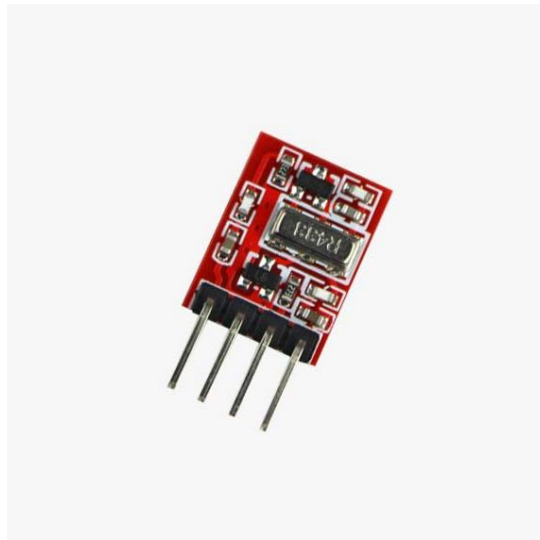


Figure 4.3.3 RF Transmitter

#### Features:

- Frequency range: The transmitter frequency range is typically 433.92 MHz.
- Supply voltage: The transmitter supply voltage is between 3 V and 6 V.
- Output power: The transmitter output power is between 4 Dbm and 12 Dbm.
- Low power consumption: RF transmitters have low power consumption.
- Crystal-stabilized oscillator: The transmitter module employs a crystal-stabilized oscillator to ensure accurate frequency control.
- Range: The transmitter has a range of about 100 meters in open space.

#### 4.3.4 RF Receiver

RF Receiver is ideal for short-range remote-control applications where cost is a primary concern. The receiver module requires no external RF components except for the antenna. It generates virtually no emissions, making FCC and ETSI approvals easy. The super-regenerative design exhibits exceptional sensitivity at a very low cost.

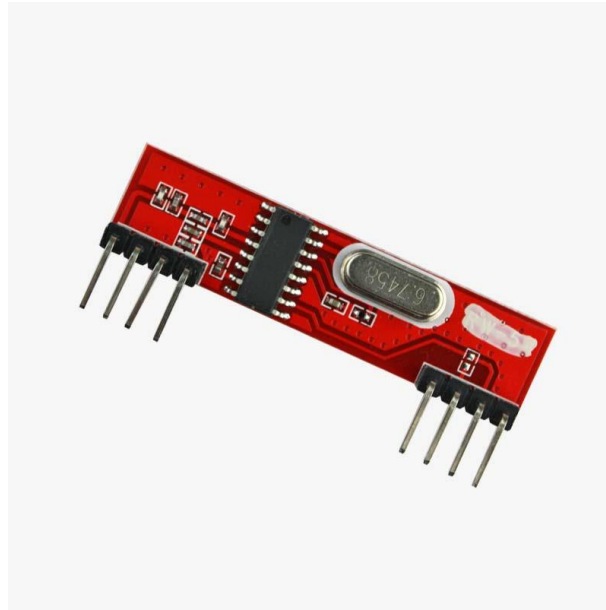


Figure 4.3.4 RF Receiver

#### Features:

- Supply voltage: The supply voltage of an RF receiver can be between 3 V and 6 V.
- Supply current: An RF receiver's supply current can be 3.5 mA.
- Operating voltage: An RF receiver's operating voltage can be 5 V.
- Transmission speed: An RF receiver can transmit data at a speed of 1–10 kbps.
- Frequency range: An RF receiver can have a frequency range of 433.92 MHz.
- Power consumption: An RF receiver can have low power consumption.

### 4.3.5 Accelerometer

An accelerometer is a device that measures the vibration, or acceleration of motion, of a structure. The force caused by vibration or a change in motion (acceleration) causes the mass to “squeeze” the piezoelectric material which produces an electrical charge that is proportional to the force exerted upon it

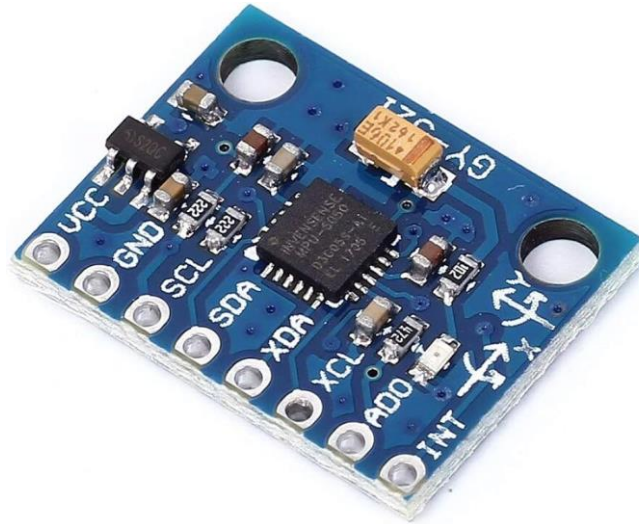


Figure 4.3.5 Accelerometer

#### Features:

- Accelerometer: 3 levels of sensitivity. Tilt angle.
- Data collection every 5, 15, 30, 60min. 6, 12, 24 hours
- Data transmission every 5, 15, 30, 60min. 6, 12, 24 hours
- Data collection by time or event (movement)
- Battery powered (up to 15.000 data packets)

### 4.3.6 ESP-01

It is a Wi-Fi module that allows microcontrollers access to a Wi-Fi network. This module is a self-contained SOC System On a Chip that doesn't necessarily need a microcontroller to manipulate inputs and outputs as you would normally do with an Arduino.

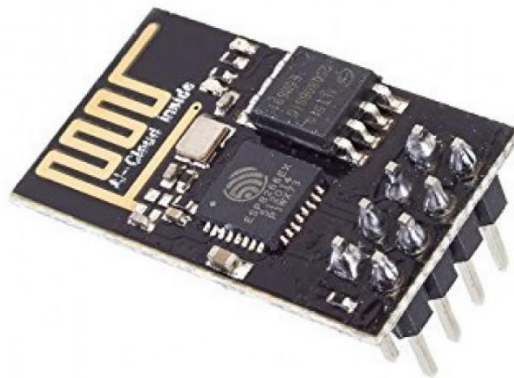


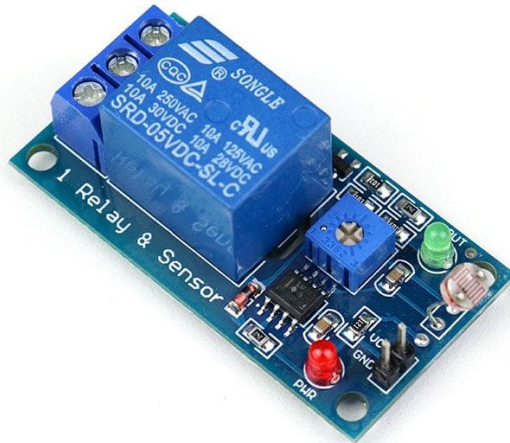
Figure 4.3.6 ESP-01

#### Features:

- It is a powerful Wi-Fi module available in a compact size at a very low price.
- It is based on the L106 RISC 32-bit microprocessor core and runs at 80 MHz
- It requires only 3.3 Volts power supply.
- The current consumption is 100 m Amps.
- The maximum Input/Output (I/O) voltage is 3.6 Volts.

### 4.3.7 Relay

A relay is an electrically operated switch. It consists of a set of input terminals for a single or multiple control signals, and a set of operating contact terminals. The switch may have any number of contacts in multiple contact forms, such as make contacts, break contacts, or combinations thereof.



• Figure 4.3.7 Relay

#### Features:

- Normal Voltage is 5V DC
- Normal Current is 70mA
- AC load current Max is 10A at 250VAC or 125V AC
- DC load current Max is 10A at 30V DC or 28V DC
- It includes 5-pins & designed with plastic material
- Operating time is 10msec
- Release time is 5msec
- Maximum switching is 300 operating per minute

### 4.3.8 Arduino UNO

The Arduino UNO is a standard board of Arduino. Arduino UNO is based on an ATmega328P microcontroller. It is easy to use compared to other boards, such as the Arduino Mega board, etc. The board consists of digital and analog Input/Output pins (I/O), shields, and other circuits. It is programmed based on IDE, which stands for Integrated Development Environment. It can run on both online and offline platforms.

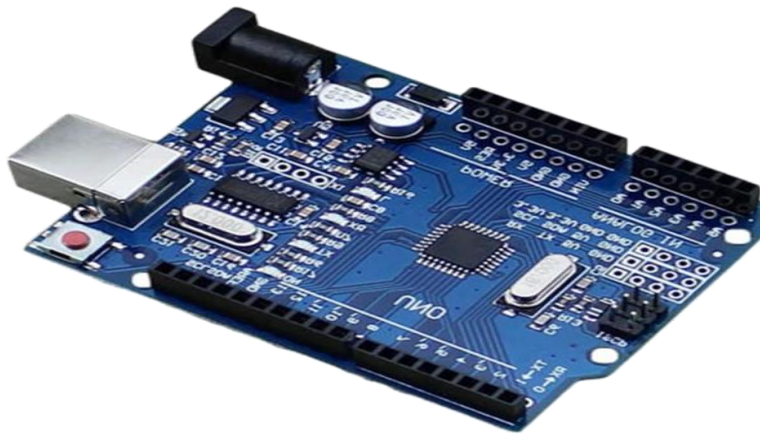


Figure 4.3.8 Arduino UNO

#### Features:

- **MEMORY:** The Arduino Uno has 32 KB memory. It comes with 2 KB of SRAM and also 1 KB of EEPROM.
- **CLOCK SPEED:** The Clock speed of the Arduino is 16 MHz so it can perform a particular task faster than the other processor or controller.
- **USB INTERFACE:** It means if we want to operate Arduino with PC, then we can do that and data communication between PC and Arduino become easy.
- **INPUT OUTPUT VOLTAGE:**  
The Arduino Uno can be powered via the USB connection or with an external power supply. If we are using external power then we can supply 6 to 20 volts. Arduino works on 5 volts.
- **INPUT OUTPUT PINS:** Each of the 14 digital pins on the Uno can be used as an input or output. 6 pins out of 14 can be used as PWM output. 6 pins can be used as analog pins.



### 4.3.9 GPS

The GPS is a space-based satellite navigation system that provides location and time information in all weather, anywhere on or near the Earth. The GPS program provides critical capabilities to military, civil and commercial users around the world. In addition, GPS is the backbone for modernizing the global air traffic system.



Figure 4.3.9 ESP8266 GPS

#### Features:

- Supports GPS, Galileo, GLONASS.
- Position accuracy 2.5 m CEP.
- Communication over I<sup>2</sup>C bus.
- Embedded antenna.
- Current consumption 26 mA in operation.
- Integrated power switch for low power operation.
- Cold start 26 s, Aided start 2 s.
- Operating voltage range: 2.7 V to 3.6 V.

### 4.3.10 Ignition

An ignition key is a key used to start a vehicle's engine by turning the switch that connects the battery to the ignition system and other electrical devices. When the key is turned on, a low voltage current flows through the ignition coil's primary windings, breaker points, and back to the battery, causing a magnetic field to form around the coil

c

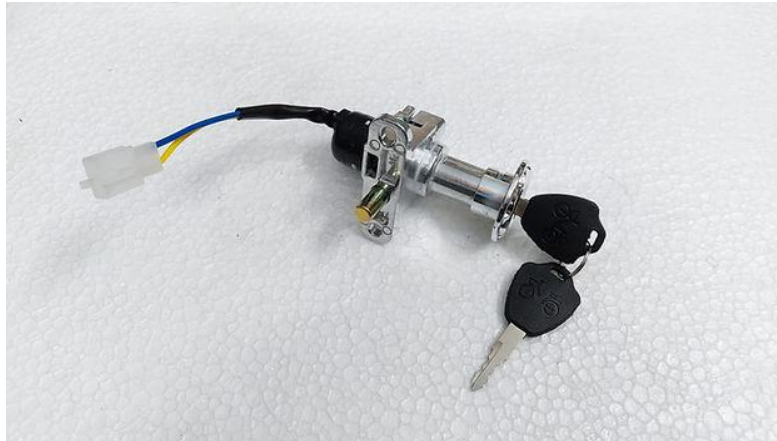


Figure 4.3.10 Ignition key

#### Features:

- **LOCK** The steering wheel locks to prevent theft and the key can only be removed in this position
- **ACCESSORY** The steering wheel unlocks and electrical accessories can be operated
- **ON** The warning lights can be checked before starting the engine and this is the normal running position after the engine is started
- **START** This position is only used to start the engine and the switch returns to the ON position when you let go of the ignition switch

## **4.1 Software Requirements**

### **4.1.1 Arduino IDE**

The Arduino integrated development environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in the programming language Java. It originated from the IDE for the languages Processing and Wiring. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple one-click mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus. The source code for the IDE is released under the GNU General Public License, version 2.

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub `main()` into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program `avrdude` to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

## 4.1.2 Installation of Arduino IDE

Get the latest version from the download page. You can choose between the Installer (.exe) and the Zip packages. We suggest you use the first one that installs directly everything you need to use the Arduino Software (IDE), including the drivers. With the Zip package you need to install the drivers manually. The Zip file is also useful if you want to create a portable installation.



### Arduino IDE 2.1.0

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the [Arduino IDE 2.0 documentation](#).

Nightly builds with the latest bugfixes are available through the section below.

SOURCE CODE

The Arduino IDE 2.0 is open source and its source code is hosted on [GitHub](#).

#### DOWNLOAD OPTIONS

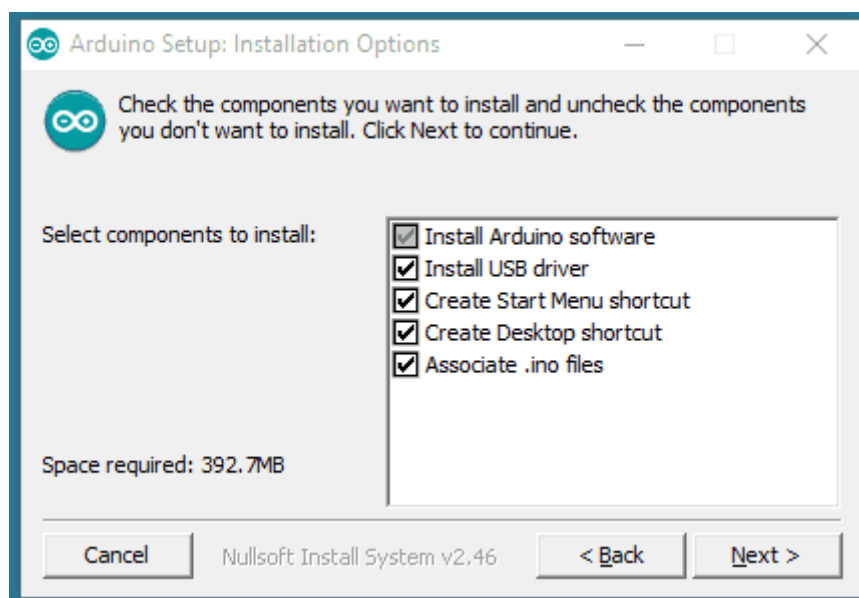
**Windows** Win 10 and newer, 64 bits  
**Windows** MSI installer  
**Windows** ZIP file

**Linux** AppImage 64 bits (X86-64)  
**Linux** ZIP file 64 bits (X86-64)

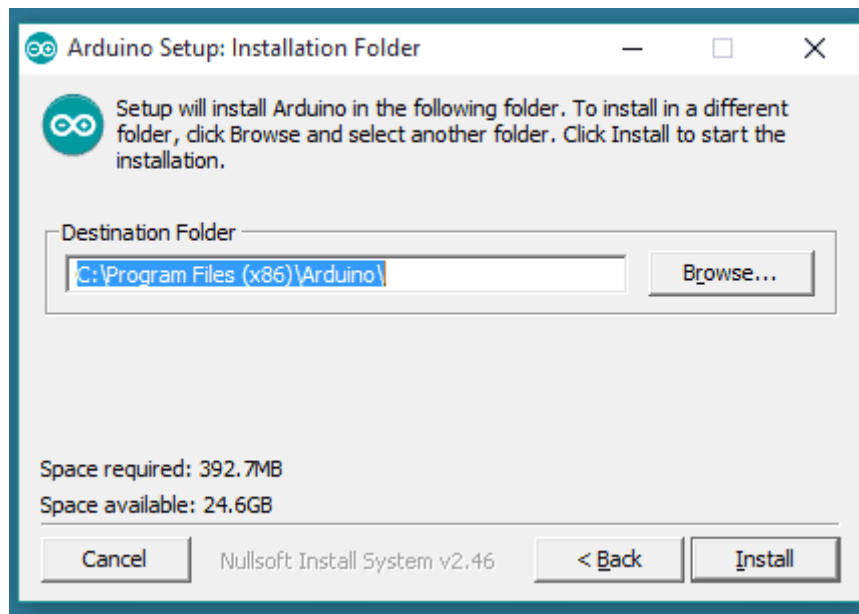
**macOS** Intel, 10.14: "Mojave" or newer, 64 bits  
**macOS** Apple Silicon, 11: "Big Sur" or newer, 64 bits

[Release Notes](#)

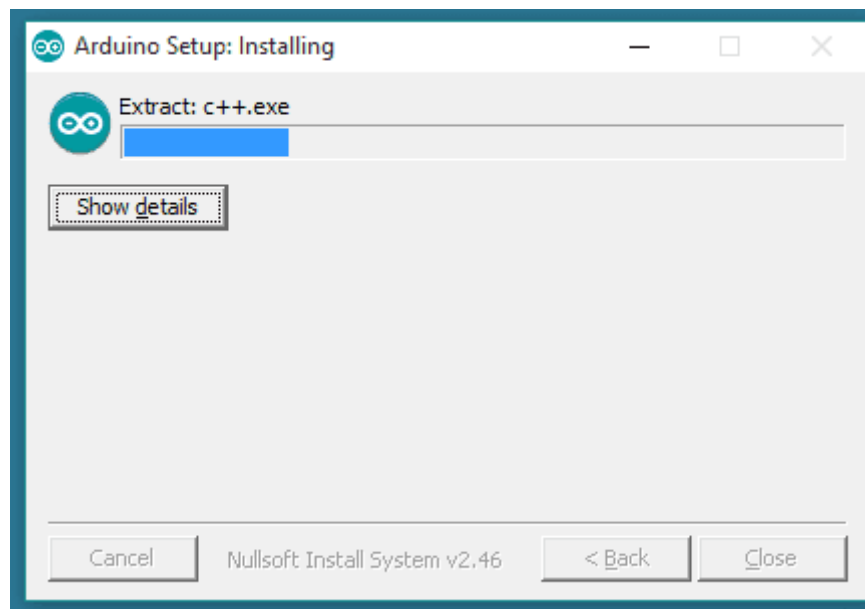
When the download finishes, proceed with the installation and please allow the driver installation process when you get a warning from the operating system.



Choose the components to install.



Choose the installation directory.

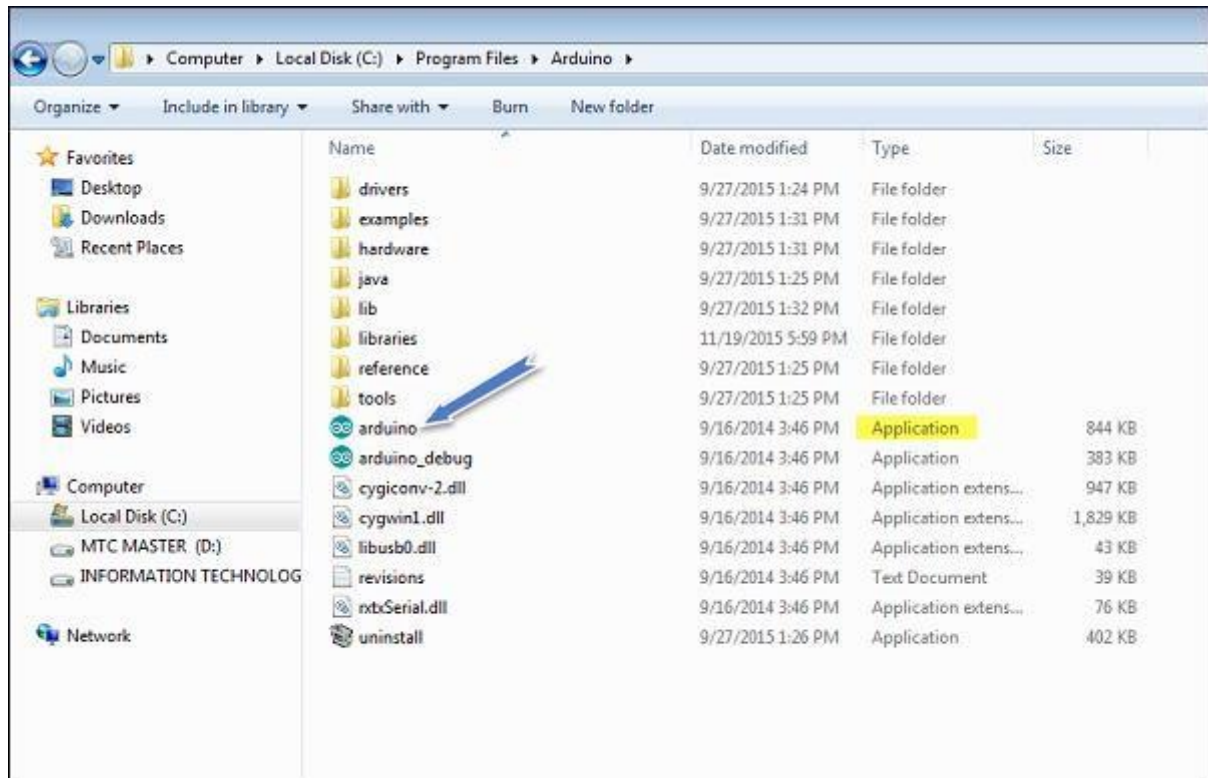


Installation in progress.

The process will extract and install all the required files to execute properly the Arduino Software (IDE).

### 4.1.3 Launch Arduino IDE

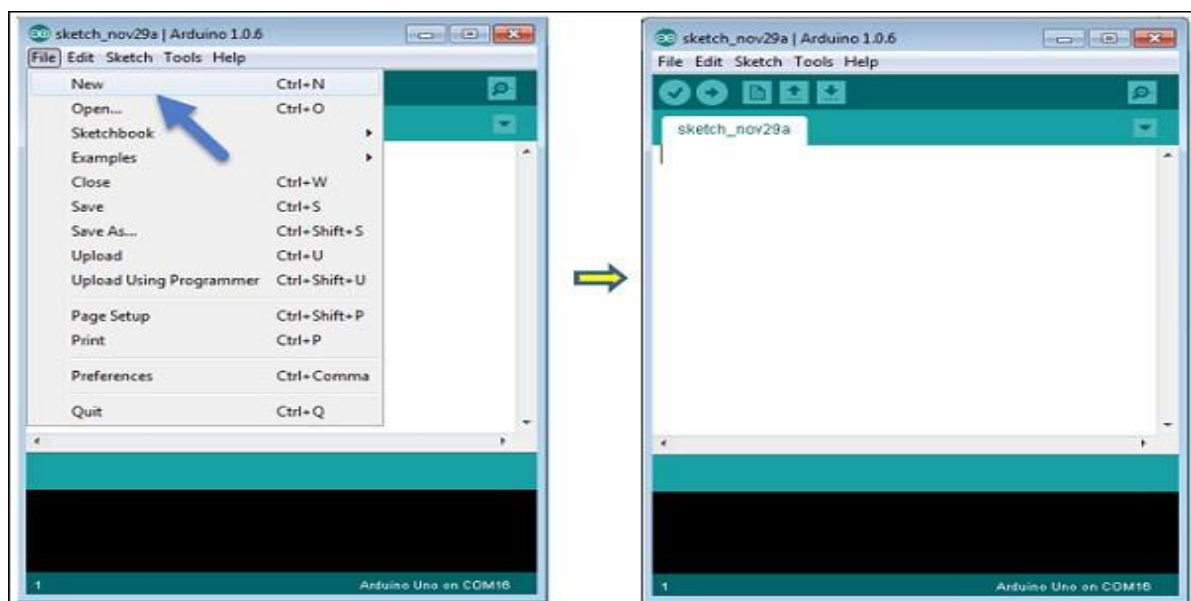
After your Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Double-click the icon to start the IDE.



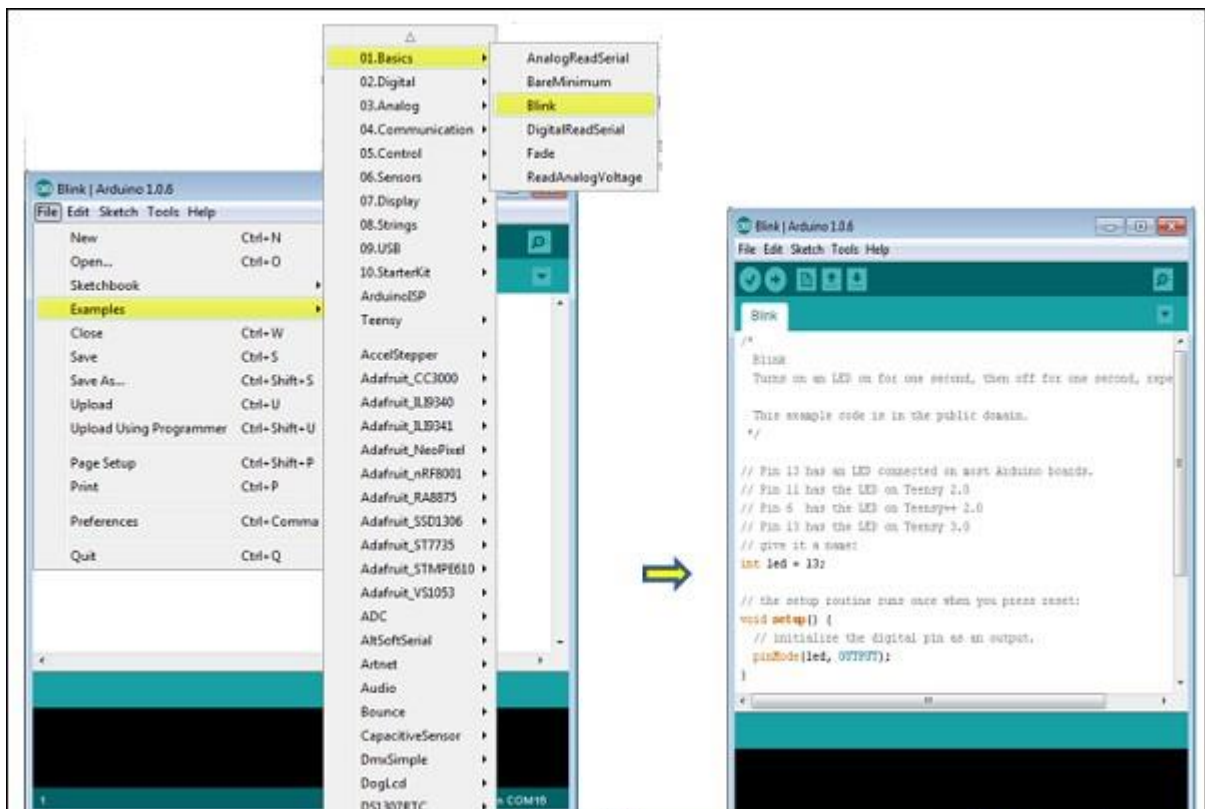
Once the software starts, you have two options –

- Create a new project.
- Open an existing project example.

To create a new project, select **File** → **New**.



To open an existing project example, select File → Example → Basics → Blink.



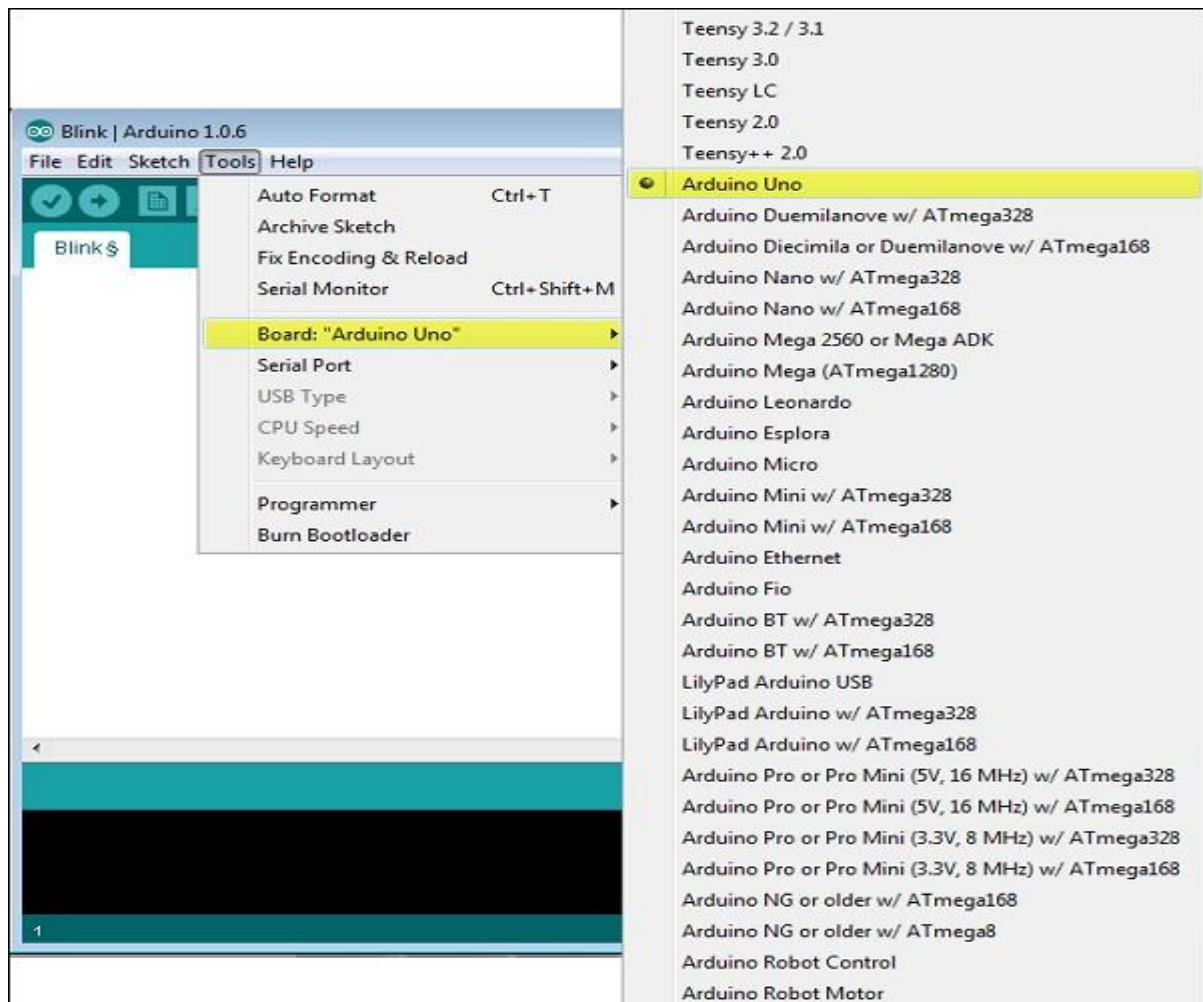
Here, we are selecting just one of the examples with the name Blink. It turns the LED on and off with some time delay. You can select any other example from the list.

### Select your Arduino board.

To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer.

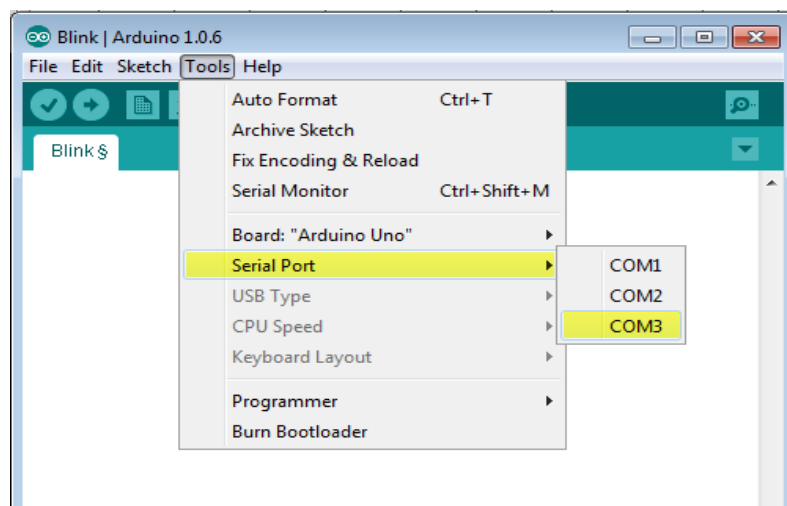
**Go to Tools** → Board and select your board.





### Select your serial port.

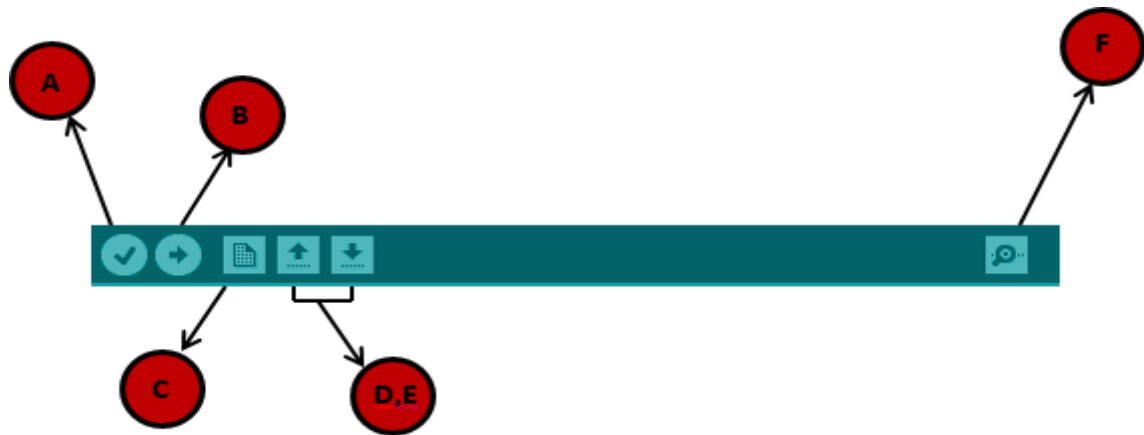
Select the serial device of the Arduino board. Go to Tools → Serial Port menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port.





## Upload the program to your board.

Before explaining how we can upload our program to the board, we must demonstrate the function of each symbol appearing in the Arduino IDE toolbar.



**A** – Used to check if there is any compilation error.

**B** – Used to upload a program to the Arduino board.

**C** – Shortcut used to create a new sketch.

**D** – Used to directly open one of the example sketch.

**E** – Used to save your sketch.

**F** – Serial monitor used to receive serial data from the board and send the serial data to the board.

Now, simply click the "Upload" button in the environment. Wait a few seconds; you will see the RX and TX LEDs on the board, flashing. If the upload is successful, the message "Done uploading" will appear in the status bar.

#### 4.1.4 IoT Cloud

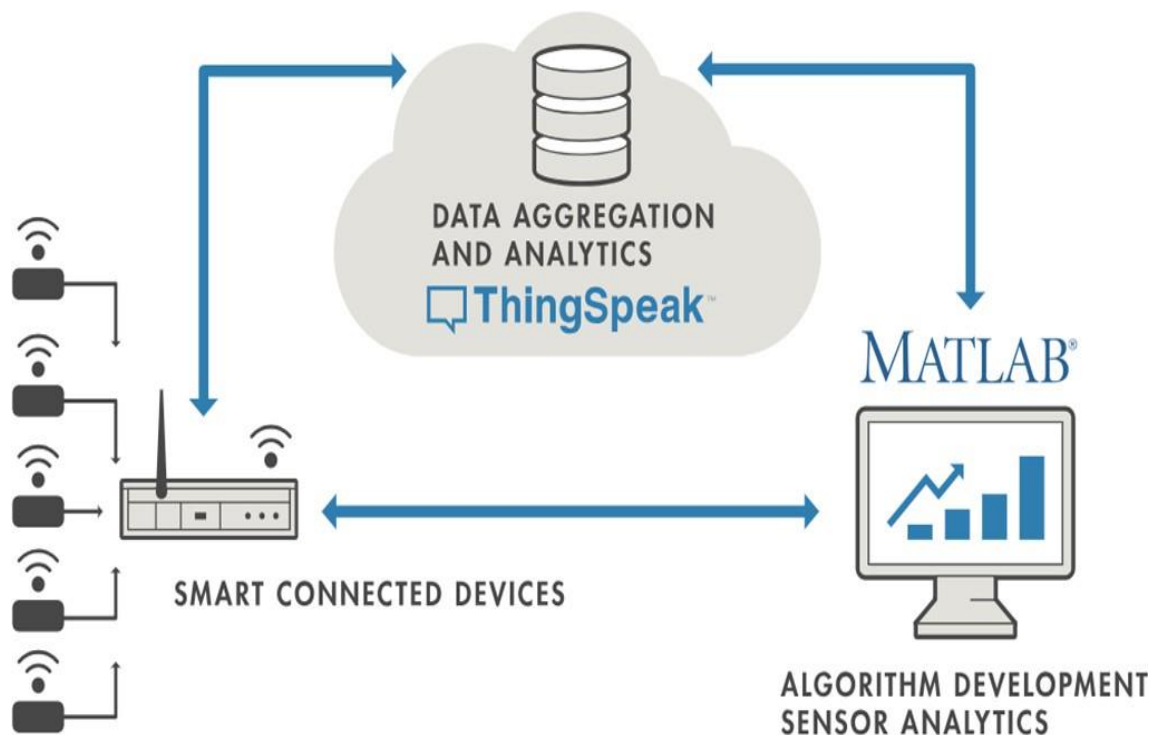
According to its developers, "ThingSpeak is an open-source Internet of Things (IoT) application and API to store and retrieve data from things using the HTTP protocol over the Internet or via a Local Area Network. ThingSpeak enables the creation of sensor logging applications, location tracking applications, and a social network of things with status updates".

ThingSpeak was originally launched by Io Bridge in 2010 as a service in support of IoT applications.

ThingSpeak has integrated support from the numerical computing software MATLAB from MathWorks, allowing ThingSpeak users to analyze and visualize uploaded data using MATLAB without requiring the purchase of a MATLAB license from MathWorks.

ThingSpeak has a close relationship with MathWorks, Inc. In fact, all of the ThingSpeak documentation is incorporated into the MathWorks' MATLAB documentation site and even enabling registered Mathworks user accounts as valid login credentials on the ThingSpeak website. The terms of service and privacy policy of ThingSpeak.com are between the agreeing user and Mathworks, Inc.

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.



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.

## ThingSpeak Key Features

- 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

To learn how you can collect, analyze and act on your IoT data with ThingSpeak, explore the topics below:



Send sensor data privately to the cloud.



Analyze and visualize your data with MATLAB.



Trigger a reaction.

## IMPLMENTATION

## 5.1 IOT (Internet of Things)

IoT refers to an Internet of Things (IoT). Connecting any device (including everything from cell phones, vehicles, home applications and other wearable embedded with sensors and actuators) with Internet so that objects can exchange data with each other on a network. It is interesting to note that there is a difference between IoT and the Internet, it is the absence of Human role. The IoT devices can create information about individual's behavior's, analyze it, and take action ( IoT is smarter than Internet).



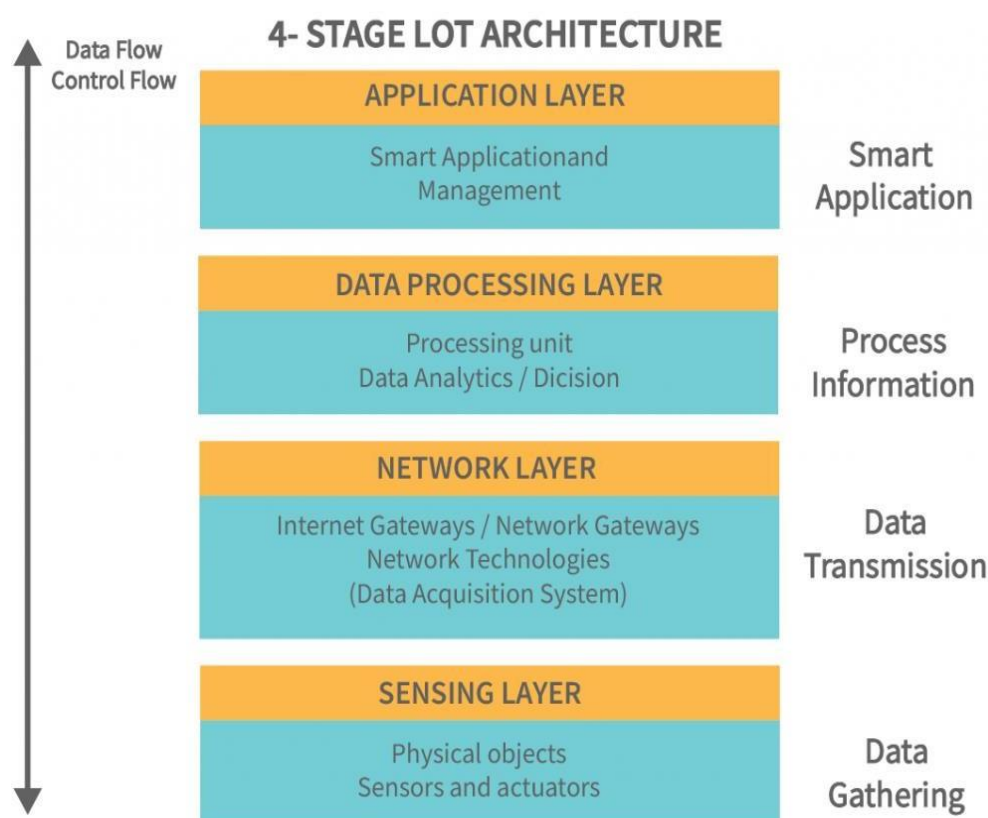
Figure 5.1 Internet of things

### 5.1.1 IoT Architecture

IoT architecture refers to the tangle of components such as sensors, actuators, cloud services, Protocols, and layers that make up IoT networking systems. In general, it is divided into layers that allow administrators to evaluate, monitor, and maintain the integrity of the system. The architecture of IoT is a four-step process through which data flows from devices connected to sensors, through a network, and then through the cloud for processing, analysis, and storage. With further development, the Internet of Things is poised to grow even further, providing users with new and improved experiences.

## Different Layers of IoT Architecture

In recent years, IoT technology has grown in popularity and it has a large variety of applications. IoT applications operate according to how they have been designed/developed based on the different application areas. However, there is no standard defined architecture of work that is strictly adhered to across the board. The complexity and number of architectural layers vary according to the specific business task at hand. A four-layer architecture is the standard and most widely accepted format.



As you can see from the above image, there are four layers present i.e., the Perception Layer, Network Layer, Processing Layer, and Application Layer.

### Perception/Sensing Layer

The first layer of any IoT system involves “things” or endpoint devices that serve as a conduit between the physical and the digital worlds. Perception refers to the physical layer, which includes sensors and actuators that are capable of collecting, accepting, and processing data over the network. Sensors and actuators can be connected either wirelessly or via wired connections. The architecture does not limit the scope of its components nor their location.

## **Network Layer**

Network layers provide an overview of how data is moved throughout the application. This layer contains Data Acquiring Systems (DAS) and Internet/Network gateways. A DAS performs data aggregation and conversion functions (collecting and aggregating data from sensors, then converting analog data to digital data, etc.). It is necessary to transmit and process the data collected by the sensor devices. That's what the network layer does. It allows these devices to connect and communicate with other servers, smart devices, and network devices. As well, it handles all data transmissions for the devices.

## **Processing Layer**

The processing layer is the brain of the IoT ecosystem. Typically, data is analyzed, pre-processed, and stored here before being sent to the data center, where it is accessed by software applications that both monitor and manage the data as well as prepare further actions. This is where Edge IT or edge analytics enters the picture.

## **Application Layer**

User interaction takes place at the application layer, which delivers application-specific services to the user. An example might be a smart home application where users can turn on a coffee maker by tapping a button in an app or a dashboard that shows the status of the devices in a system. There are many ways in which the Internet of Things can be deployed such as smart cities, smart homes, and smart health.

## **Stages of IoT Solutions Architecture**

There are several layers of IoT built upon the capability and performance of IoT elements that provides the optimal solution to the business enterprises and end-users. The IoT architecture is a fundamental way to design the various elements of IoT, so that it can deliver services over the networks and serve the needs for the future.

Following are the primary stages (layers) of IoT that provides the solution for IoT architecture.

### **Sensors/Actuators**

Sensors or Actuators are the devices that are able to emit, accept and process data over the network. These sensors or actuators may be connected either through wired or wireless. This contains GPS, Electrochemical, Gyroscope, RFID, etc. Most of the sensors need connectivity through sensors gateways. The connection of sensors or actuators can be through a Local Area Network (LAN) or Personal Area Network.

## Gateways and Data Acquisition

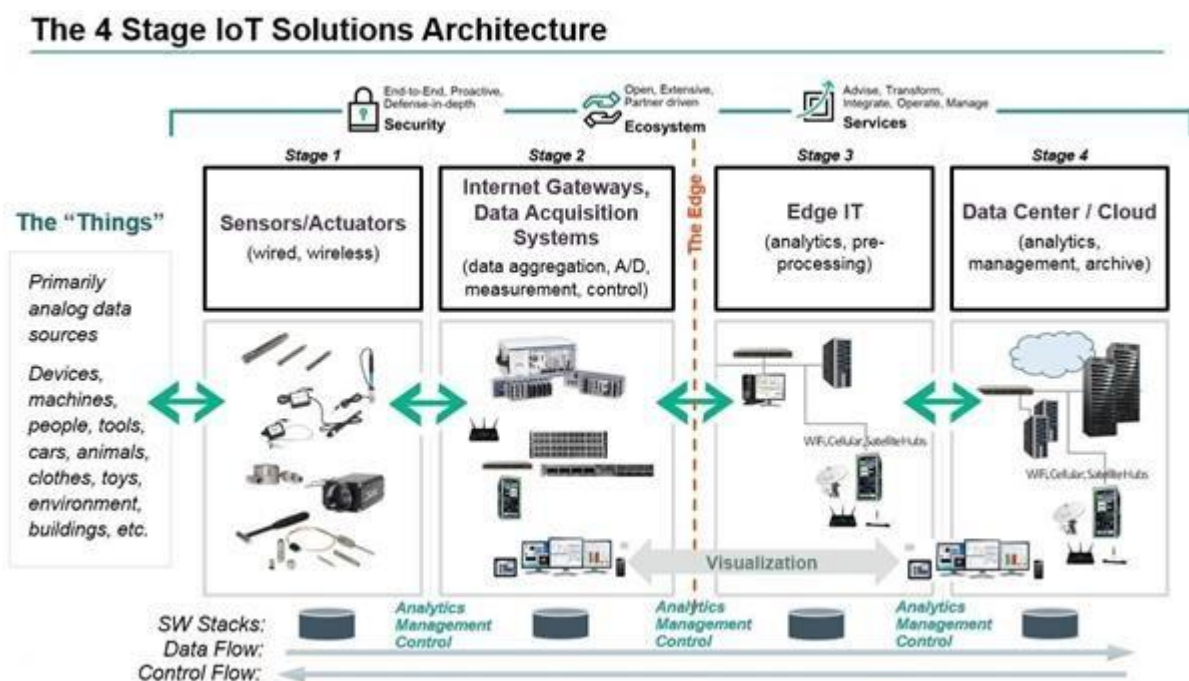
As the large numbers of data are produced by this sensors and actuators need the high-speed Gateways and Networks to transfer the data. This network can be of type Local Area Network (LAN such as WiFi, Ethernet, etc.), Wide Area Network (WAN such as GSM, 5G, etc.).

## Edge IT

Edge in the IoT Architecture is the hardware and software gateways that analyze and pre-process the data before transferring it to the cloud. If the data read from the sensors and gateways are not changed from its previous reading value then it does not transfer over the cloud, this saves the data used.

## Data center/ Cloud

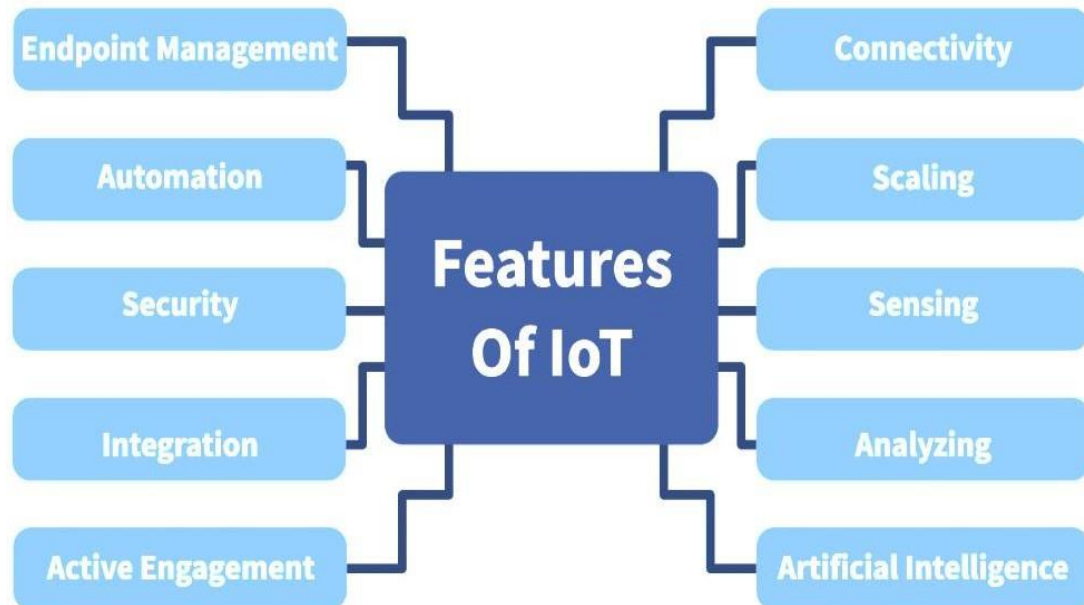
The Data Center or Cloud comes under the Management Services which process the information through analytics, management of device and security controls. Beside this security controls and device management the cloud transfer the data to the end users application such as Retail, Healthcare, Emergency, Environment, and Energy, etc.





### 5.1.2 Features of Internet of things

We have listed some of the features that make IoT what it is in the present digital scenario. IoT devices have several sets of features that are common. They are:



#### Connectivity

The heart and soul of IoT is its connectivity. Connectivity means the establishment of a connection between different devices (or nodes) so that they can communicate on their own. In IoT, various devices, sensors, computers, and data busses need to interact and communicate with each other. A fast, safe, and secure connection is a must for IoT to be of any business use. IoT also connects devices with cross-domain technology like cloud computing, artificial intelligence, and blockchain technology. We can connect them over radio waves, Wi-Fi, Bluetooth, or wires.

#### Scaling

IoT systems are designed in such a way that the number of devices, sensors, or computers can be scaled up and down according to the need. An IoT system should be elastic enough so that it can handle workload during peak demand hours and can resort back to the normal state when the demand is low.

#### Sensing

IoT devices gather information about their surroundings (such as temperature, light, sound, acceleration, pressure) and then, after analyzing the data, take a decision. Thus, sensors help in automation by gathering



information and taking actions that would otherwise, be done by humans. The raw data gathered, and the analyzed data, serve as the basis of the functioning of IoT. For example, in an automatic door, sensors would collect data through sensors such as radar sensors and optical sensors. If it detects a person coming, it will open the door automatically. Some sensors used in IoT are- Humidity sensor, temperature sensor, Accelerometer, Gyroscope, Motion sensor, image sensor, level sensor, and Proximity sensor.

## **Analyzing**

We know IoT gathers raw information through sensors, but why does IoT need data? What does IoT do with all that raw data? Data as such has no value of its own. It is meaningless and useless until it is purposefully processed to gain some meaningful insights from it. IoT gathers raw data to extract something meaningful out of it. Analyzing the raw data in terms of its structure, correlation, and usability is necessary because, if processed properly, it can be very useful. In the above-mentioned example of the automatic door for instance, after analyzing the data through sensors, it should be able to differentiate between a person and an animal.

## **Artificial Intelligence**

IoT becomes a lot more useful when combined with artificial intelligence. For instance, if you are out of groceries, your smart refrigerator can notify you to bring some on your way back home. Things like these have been made possible by the application of artificial intelligence. IoT devices collect raw data from their surroundings and convert them into something useful and insightful. The IoT devices and systems are also trained with various machine learning models so that they can better understand the changes in their surroundings and perform better.

## **Smaller Device**

Devices and equipment (like semiconductor chips, sensors) are getting smaller and smaller these days. In IoT, these small-built devices efficiently deliver precision and performance. It is fascinating to think that devices so small can deliver so much and enhance our quality of living (for example, small sensors can tell us the quality of air in that area, protecting us from pollution).

## **Dynamic Nature**

IoT systems should be dynamic in to change according to the changes in their environment to be of any business use. Let us understand this with an example. A smart air conditioner should be able to set the temperature of the room according to the prevalent weather conditions using the data gathered by the temperature sensor. It must also be able to set the perfect humidity level inside the room according to the changes in the humidity level of the surroundings.

## **Active Engagement**

IoT connects its devices and products with cross-domain technologies like cloud computing, artificial intelligence, blockchain technology, etc. An active engagement between these products and technologies is essential to gather and manipulate data to make it of business use. Raw data has huge potential and can improve business decisions considerably. Therefore, an active engagement between various IoT products and these technologies is the need of the hour.

## **Integration**

IoT integrates various cross-domain technologies like cloud computing, artificial intelligence, big data, and deep learning to provide users with a wonderful experience. Internet of things is no more internet of things, it has, in fact, become the internet of everything. An entire ecosystem of integrated devices performs with great efficiency to enhance our quality of living.

## **Automated**

Every technology comes with a certain degree of automation. In the case of IoT, the theme is all about automation. IoT was developed to make people's life and business easier with automation, i.e., IoT farming system automates irrigation and prevents wastage of water as well.

## **Security**

Security is one of the major concerns among the users of IoT. IoT systems carry and store a lot of sensitive information, so the security of the devices and the data flowing between them should be given foremost priority. Proper security and safety measures are implemented while designing an IoT system to prevent a possible breach of security. Resources and investment required to ensure a safe and conductive IoT system are huge, but its safety and security must be ensured. Failing to do so can lead to mistrust among its users and businesses and can reduce its demand.

## **Endpoint Management**

A well-trained and carefully implemented IoT is an asset for the business world. However, it is important to be the endpoint management of IoT systems, otherwise, the whole system might collapse. For example, let us say your smart fridge orders groceries to a retailer when you are out of it. But, if you are not at the home for some days, it might lead to the wastage of groceries and is a failure of IoT. Therefore, endpoint management is a necessary characteristic of the Internet of things.

## 5.2 Block Diagram.

### 5.2.1 Helmet Section

- Arduino Uno acts as a Controller of the system which is interfaced with the Gas, moisture, Humidity, Ultrasonic Sensor and LDR.
- LDR used to find out the intensity of light by using the intensity value we can apply PWM to the Light.
- The pump is controlled by Relay is switching through the IoT Depending upon the Moisture level in the Soil it will give alert message.
- All the sensor values can be monitor and control over the IoT.

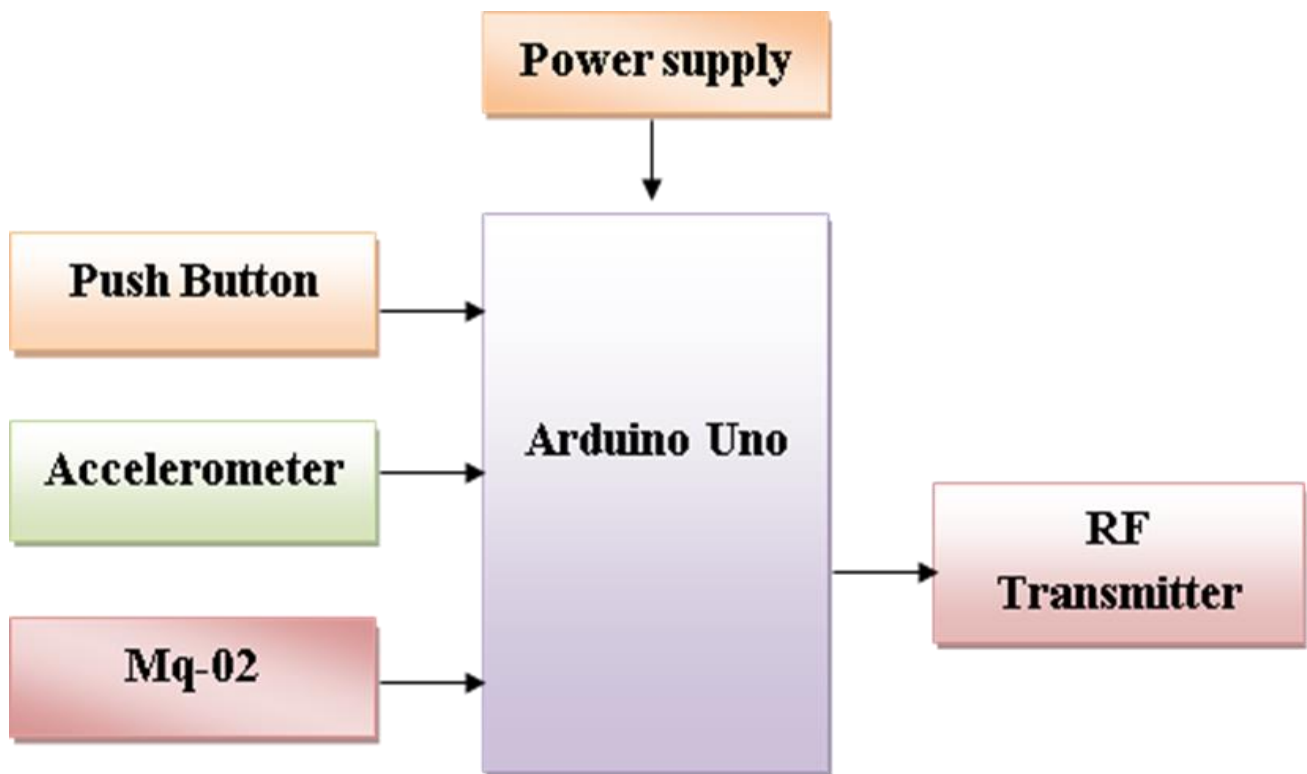


Figure 5.2.1 Block Diagram of Smart Helmet for safety and Accident Detection using IOT.

### 5.2.2 Bike Section

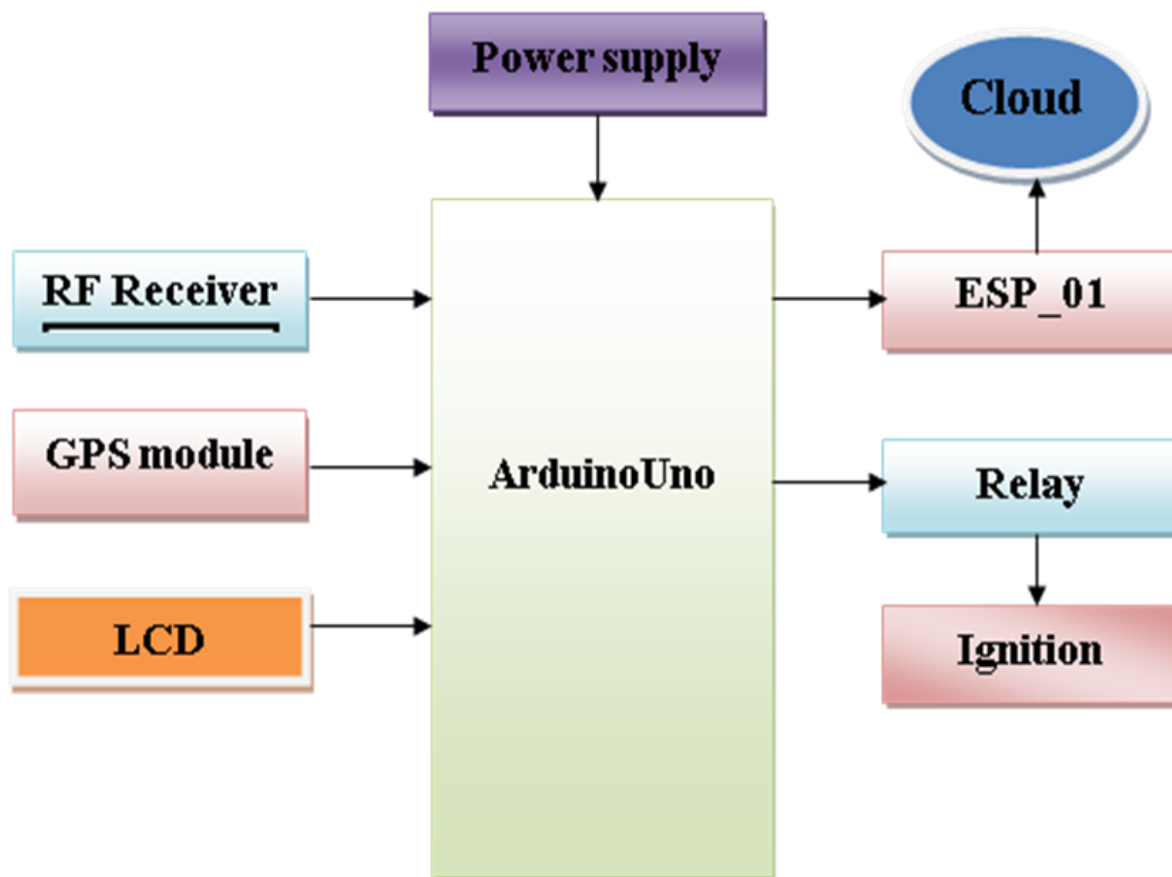


Figure 5.2.2 Block Diagram of Bike Section for safety and Accident Detection using IOT.

# SYSTEM DESIGN SPECIFICATION

## 6.1 Design Approach

System design is the process of defining the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system. It is meant to satisfy specific needs and requirements of a business or organization through the engineering of a coherent and well-running system.

## 6.2 Overview

This chapter overall explains about the system design for SMART RIDING HELMET BASED ON CLOUD AND INTERNET OF THINGS. This chapter also includes the Data Flow Diagram that follow the user requirement.

## 6.3 Data Flow Diagram

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually “say” things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. That’s why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems.

### 6.3.1 Symbols and Notations Used in DFDs

Three common systems of symbols are named after their creators:

- Yourdon and Coad
- Yourdon and DeMarco
- Gane and Sarson

One main difference in their symbols is that Yourdon-Coad and Yourdon-DeMarco use circles for processes, while Gane and Sarson use rectangles with rounded corners, sometimes called lozenges. There are other symbol variations in use as well, so the important thing to keep in mind is to be clear and consistent in the shapes and notations you use to communicate and collaborate with others.

**Using any convention's DFD rules or guidelines, the symbols depict the four components of data flow diagrams.**

### **External entity**

An outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system or a business system. They are also known as terminators, sources and sinks or actors. They are typically drawn on the edges of the diagram.

### **Process**

Any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules. A short label is used to describe the process, such as "Submit payment."

### **Data store**

Files or repositories that hold information for later use, such as a database table or a membership form. Each data store receives a simple label, such as "Orders."

### **Data flow**

The route that data takes between the external entities, processes and data stores. It portrays the interface between the other components and is shown with arrows, typically labeled with a short data name, like "Billing details."



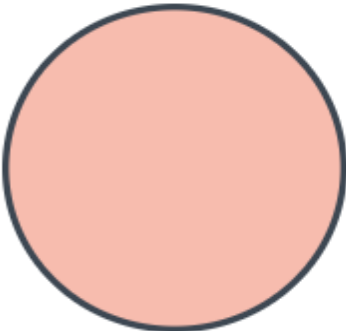
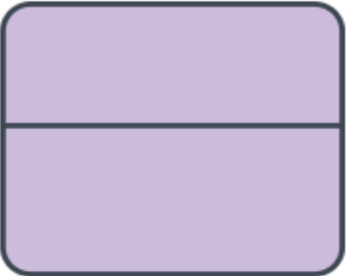




Notation	Yourdon and Coad	Gane and Sarson
External Entity		
Process		
Data Store		
Data Flow		

Figure 6.3.1 Symbols of Data flow diagram

### 6.3.2 Data Flow Diagram

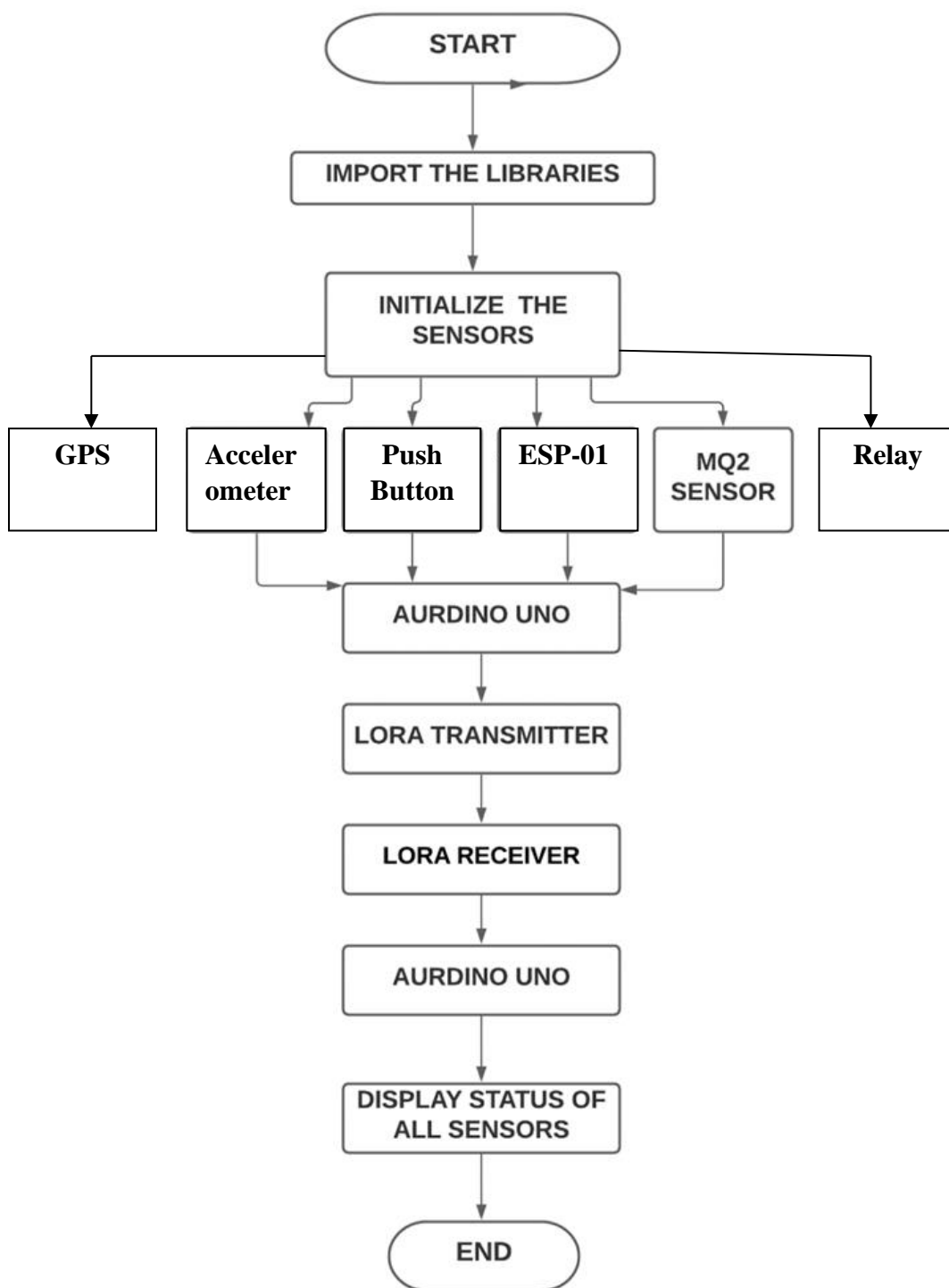


Figure 6.3.2 Data flow diagram of plant monitoring system



## 6.4 Circuit Diagram

### 6.4.1 Helmet Section

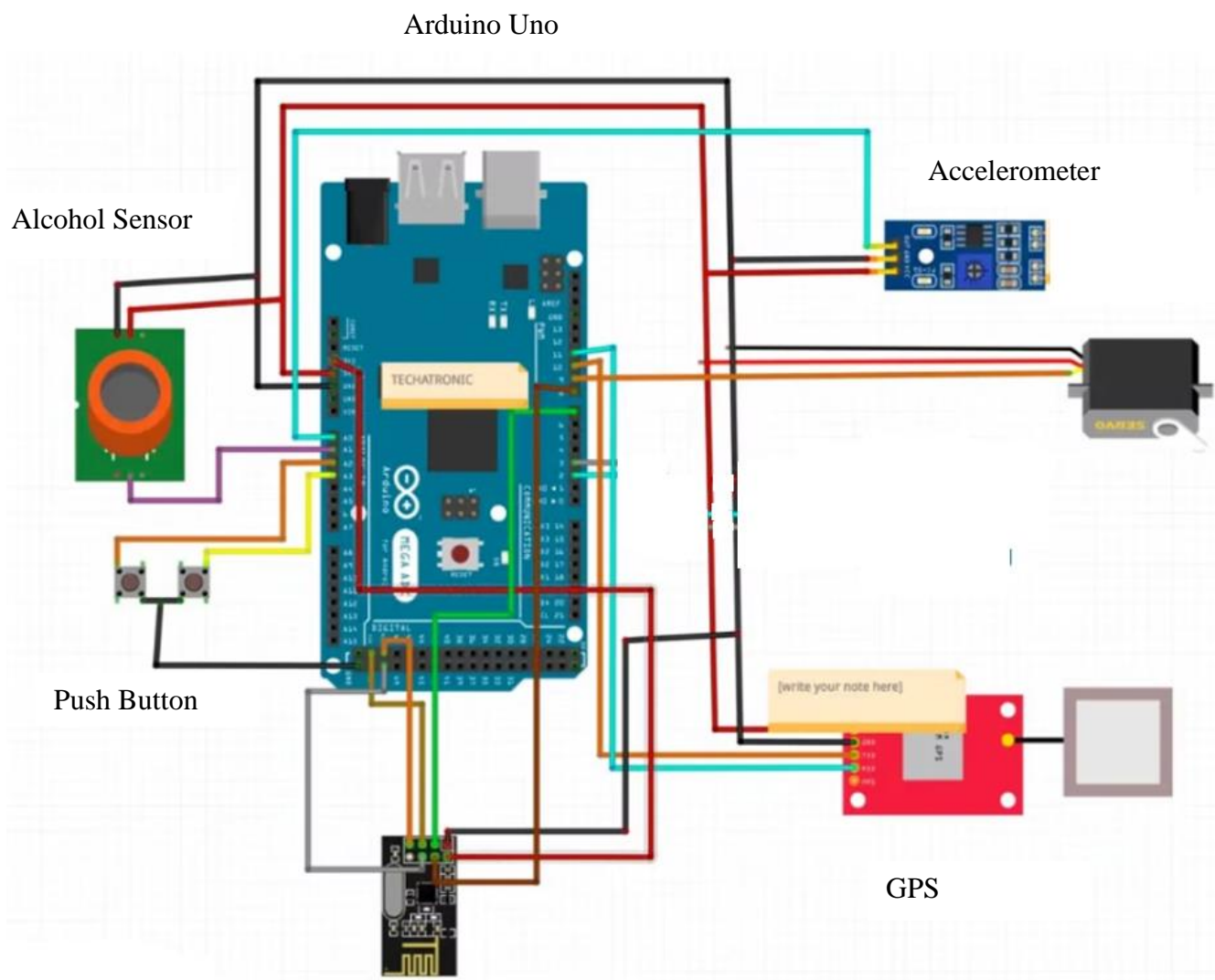


Figure 6.4.1 Helmet Section

### 6.4.2 Bike Section

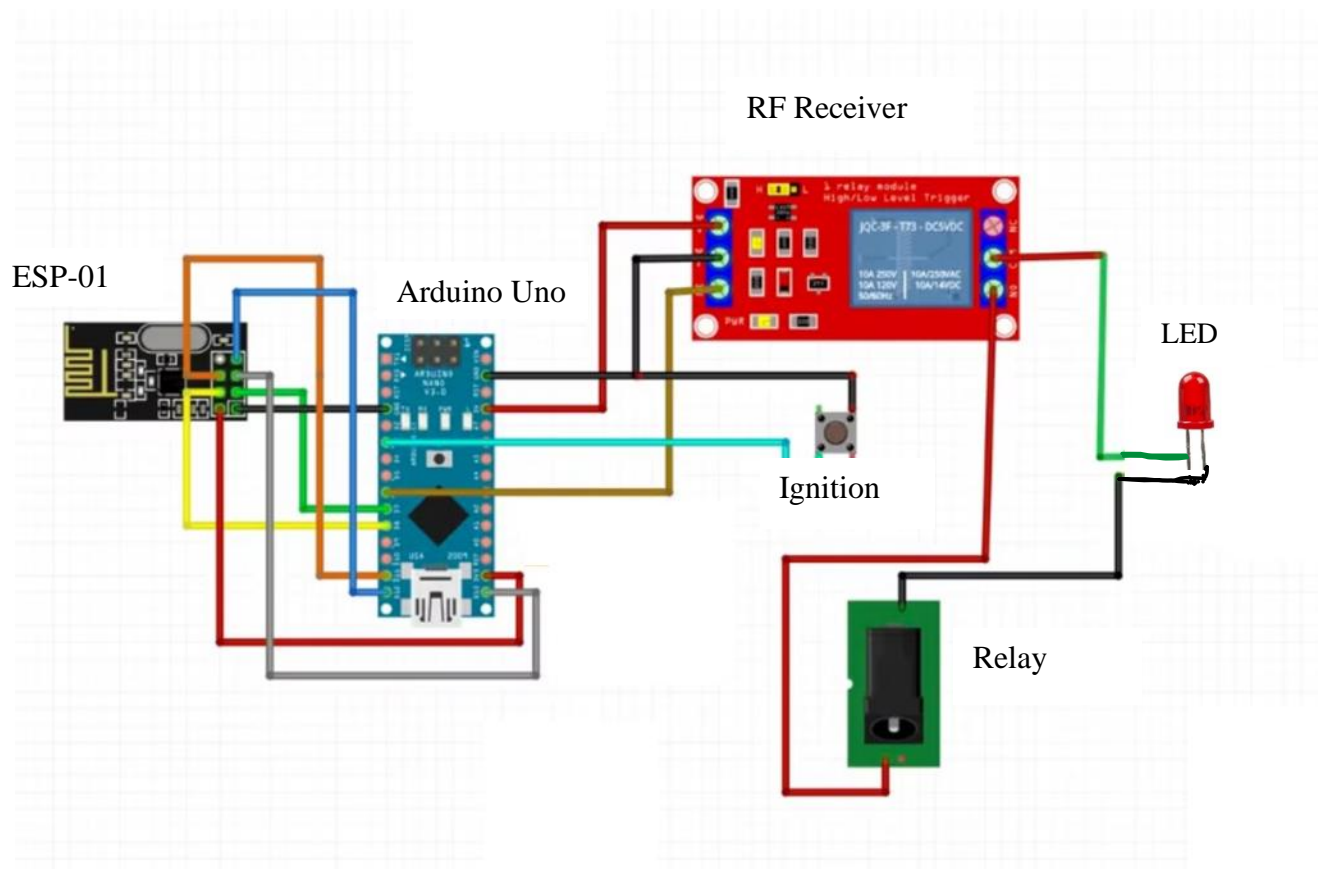


Figure 6.4.2 Bike Section

## SCREENSHOTS

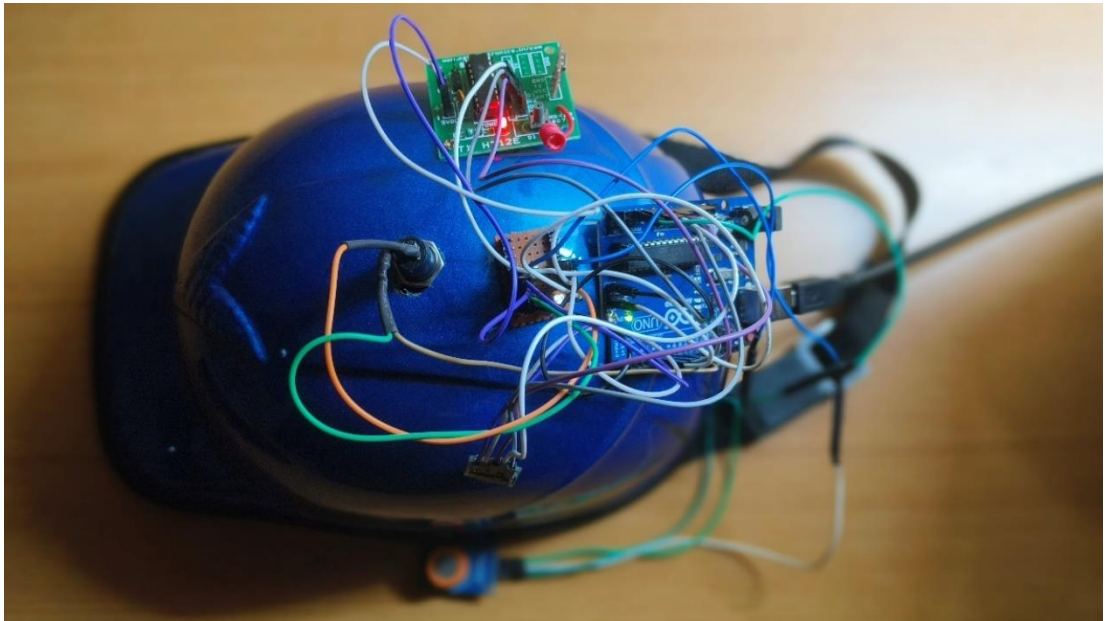


Figure 7.1 Bike Section

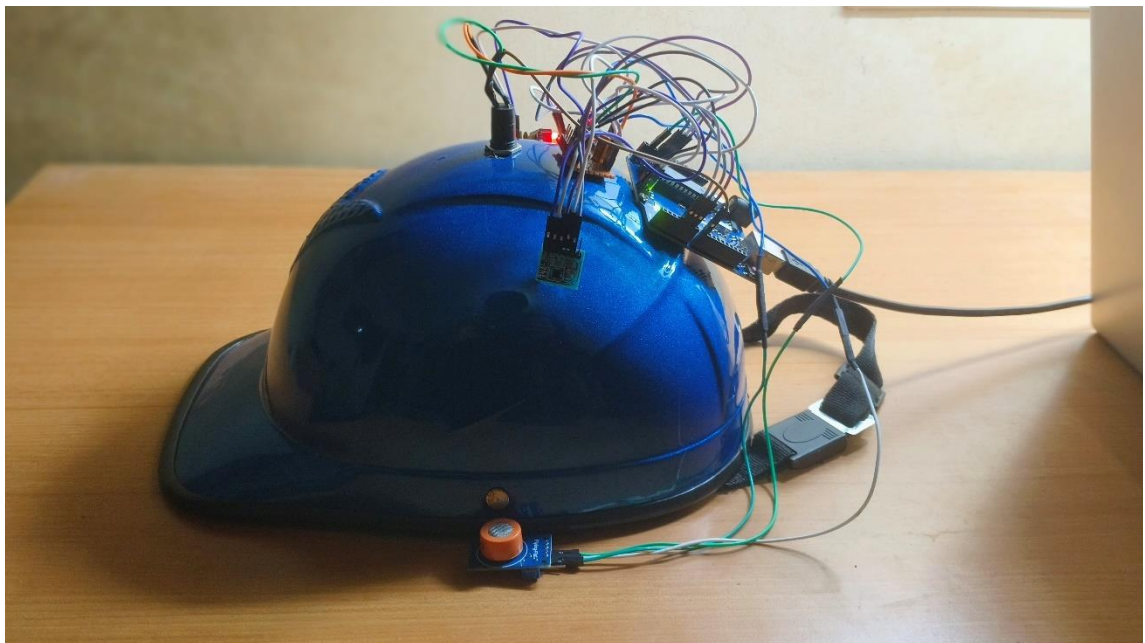


Figure 7.2 Bike Section



Figure 7.3 Bike Section

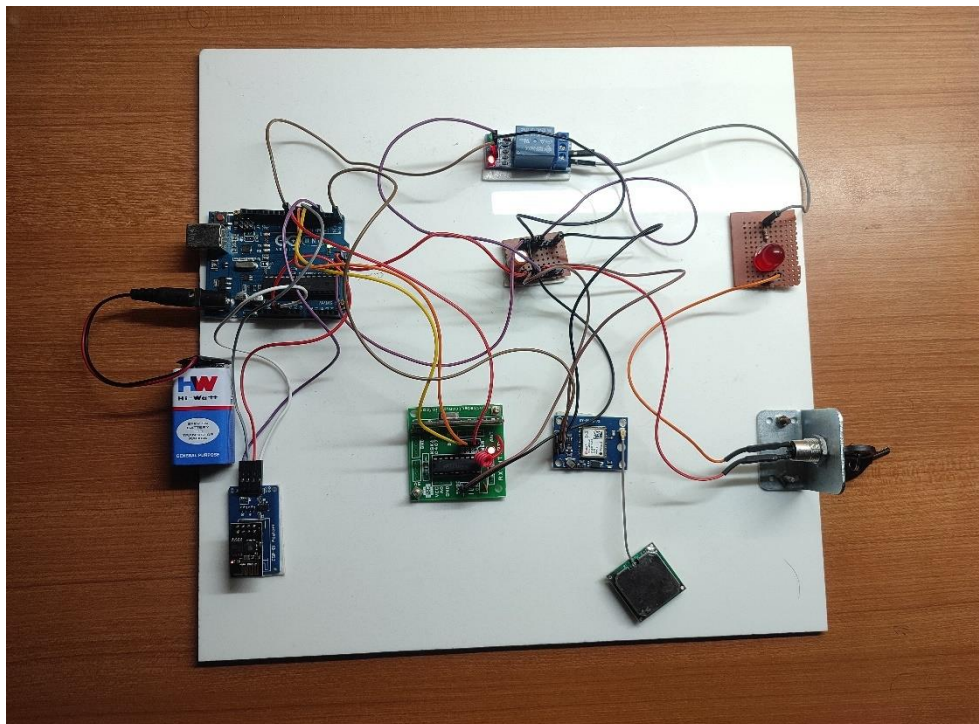


Figure 7.4 Bike Section



## 7.5 Bike Section

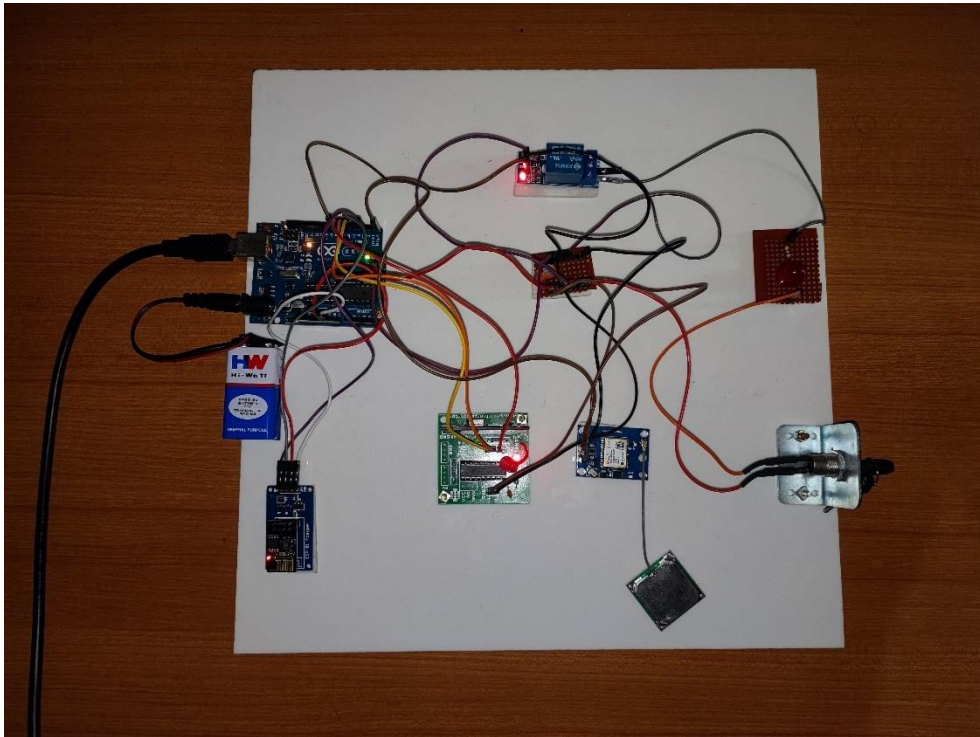
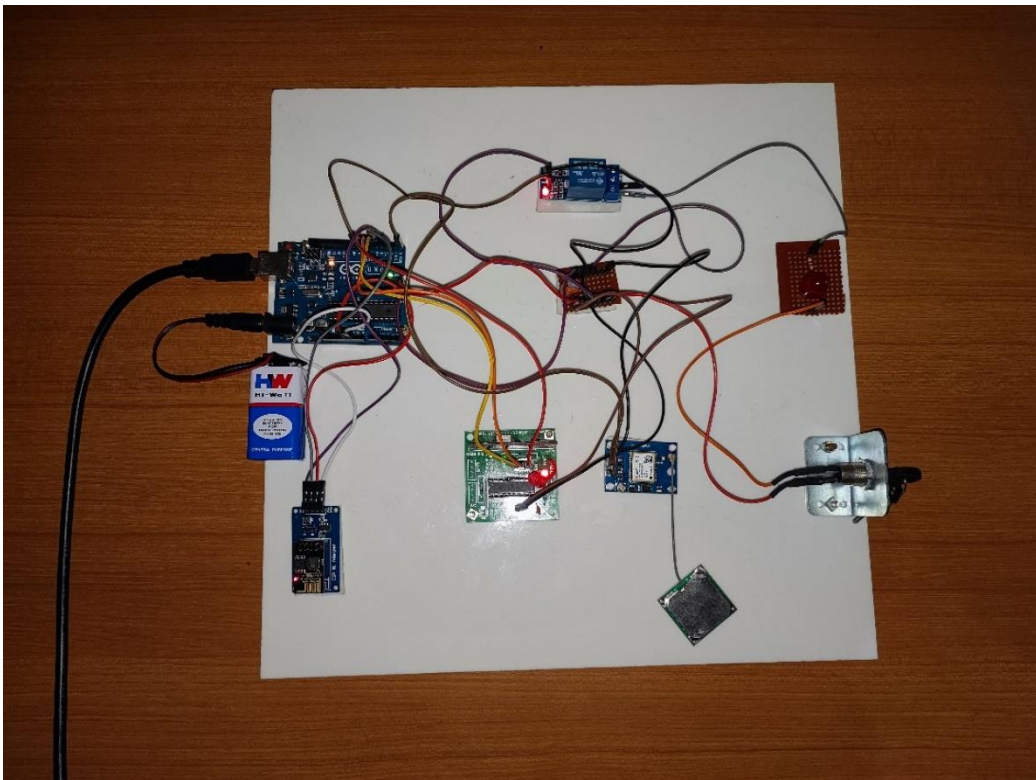


Figure 7.6 Bike Section



Channels Apps Devices Support
Commercial Use How to Buy AU

# Channel 2512656

Channel ID: 2512656  
Author: mwa0000028834730  
Access: Private

Private View Public View Channel Settings Sharing API Keys Data Import / Export

Add Visualizations Add Widgets Export recent data
MATLAB Analysis MATLAB Visualization

Channel Stats

Created: 13 days ago  
Last entry: about 15 hours ago  
Entries: 66

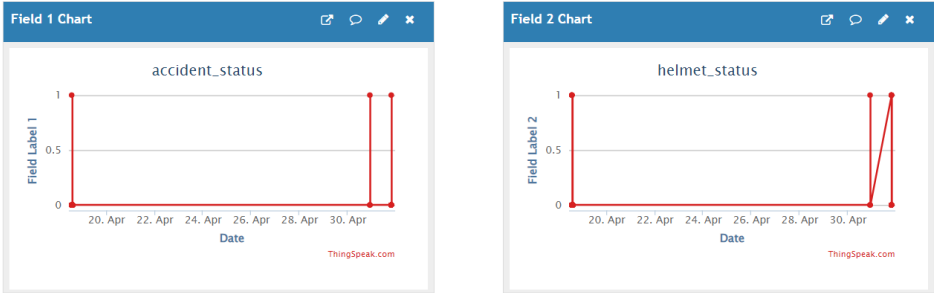


Figure 7.7 Data uploaded to thingspeak

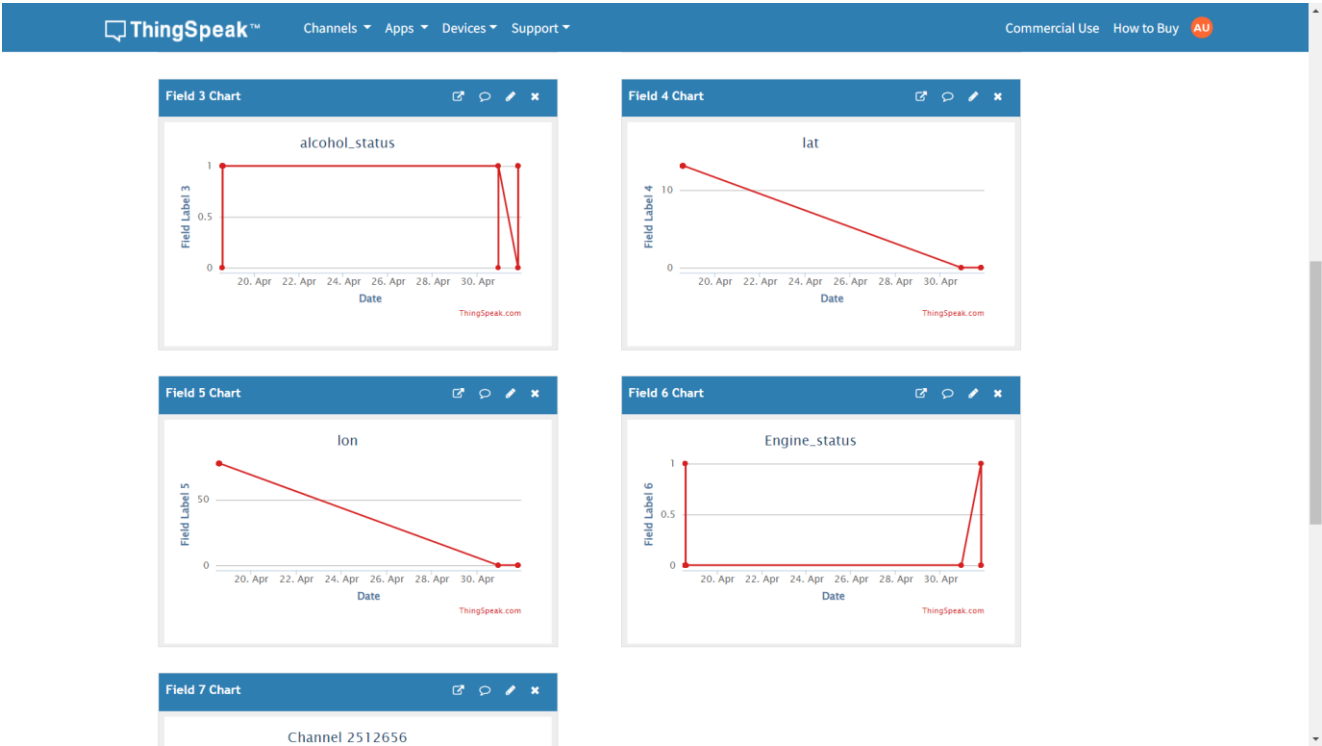


Figure 7.8 Data uploaded to thingspeak

# TESTING

IoT testing is a process that involves performing several tests on your IoT solution to ensure it is ready for real-life applications. The purpose of IoT testing is to find and fix vulnerabilities in your IoT solution so that you can be confident it will work as expected once it is deployed in the real world.

**IoT testing has two primary components:**

**Testing on a device:** This is how most people think about IoT testing. You connect a device to a mobile phone or computer, run some software, and do some checks. It's like running a VM on your device (which is what most people do).

## 9.1 IoT Testing Approaches

Before jumping into the types of IoT testing tools available, let's look at some of the most common types of testing:

### Functional

This approach focuses on verifying that the IoT device performs its intended function, such as sending data to a cloud-based server.

### Performance

This type of validation involves measuring how well a product performs its intended functions under conditions that might not be realistic or optimal. For example, performance testing might involve measuring how well a sensor can detect motion in an office building before it goes off and causes false alarms.

## Regression

This test is conducted to make sure that, even if changes are made after deploying the app, its functionality will remain unaffected.

## Integration

Integration testing ensures that all components in an IoT solution work together as intended.

## Security

This phase focuses on the extent to which the IoT device protects itself from hackers and other threats. It involves analyzing the system's security features, such as access control and encryption, to determine whether they are adequate.

## Privacy

This phase examines how an IoT device handles data privacy, including whether it can encrypt sensitive data and what information it transmits to third parties.

## Usability

This phase determines whether an IoT device is easy to use, intuitive, and works well across a range of devices and operating systems.

## 9.2 Test Cases

Test Case	Input	Expected o/p	Actual o/p	Result
1	Alcohol Sensor	If person drink alcohol it shows alcohol detect.	Alcohol Sensor Runs successfully	Passed
2	ESP-01	Wi-Fi module	Wi-Fi module ON	Passed
3	Relay	Relay is an electrically operated switch.	Relay ON	Passed
4	Accelerometer	Integrated circuit which is used to measure the acceleration	Accelerometer ON	Passed



5	All reading from the sensors	It should upload to the Thingspeak	Data uploaded	Passed
---	---------------------------------	---------------------------------------	---------------	--------

### 9.3 Test Cases Screenshots

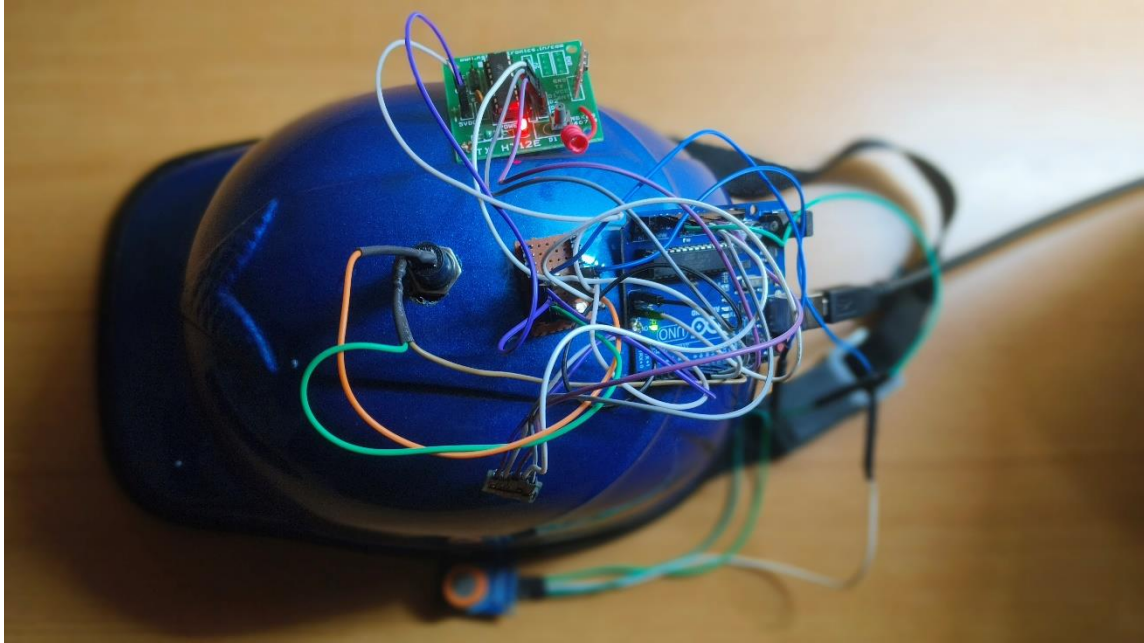


Figure 9.3.1 Helmet Section ON

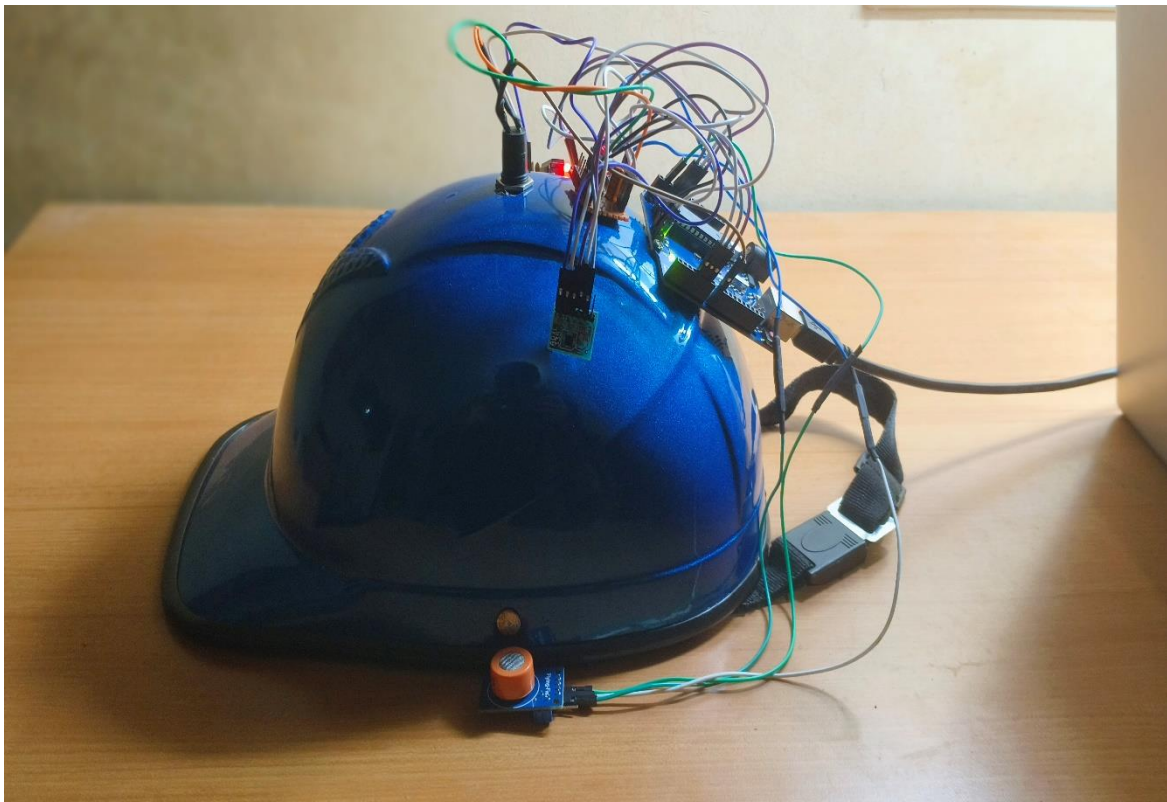


Figure 9.3.2 Helmet Section Accelerometer ON

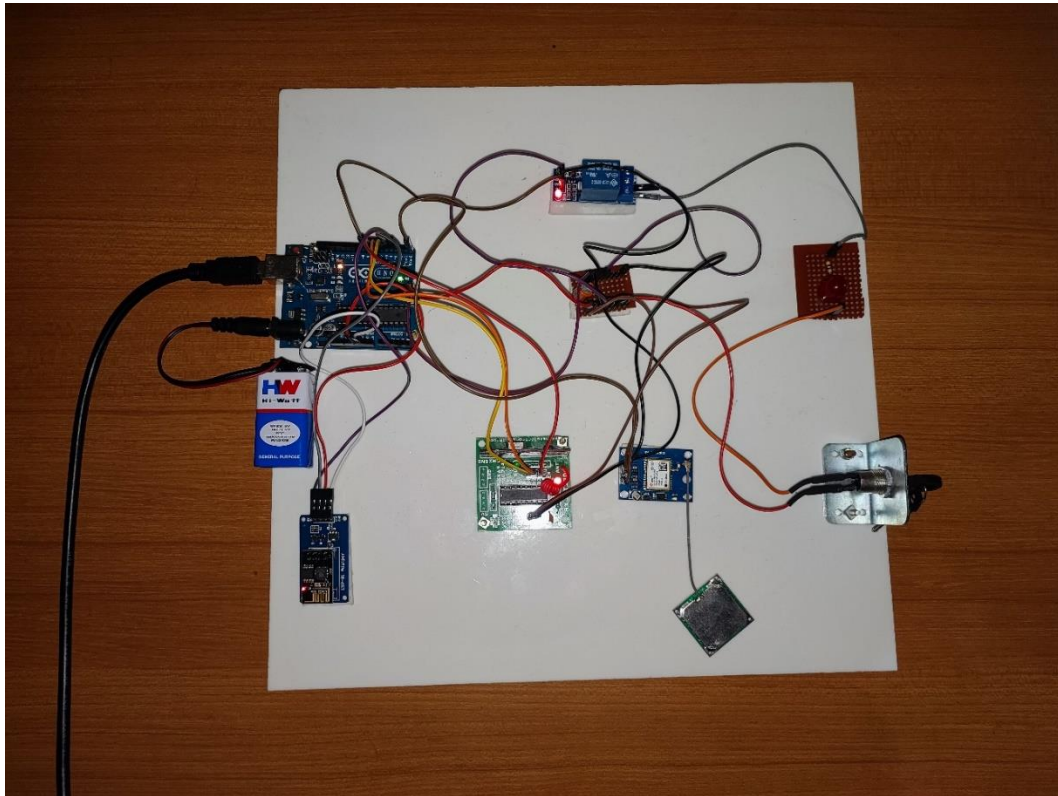
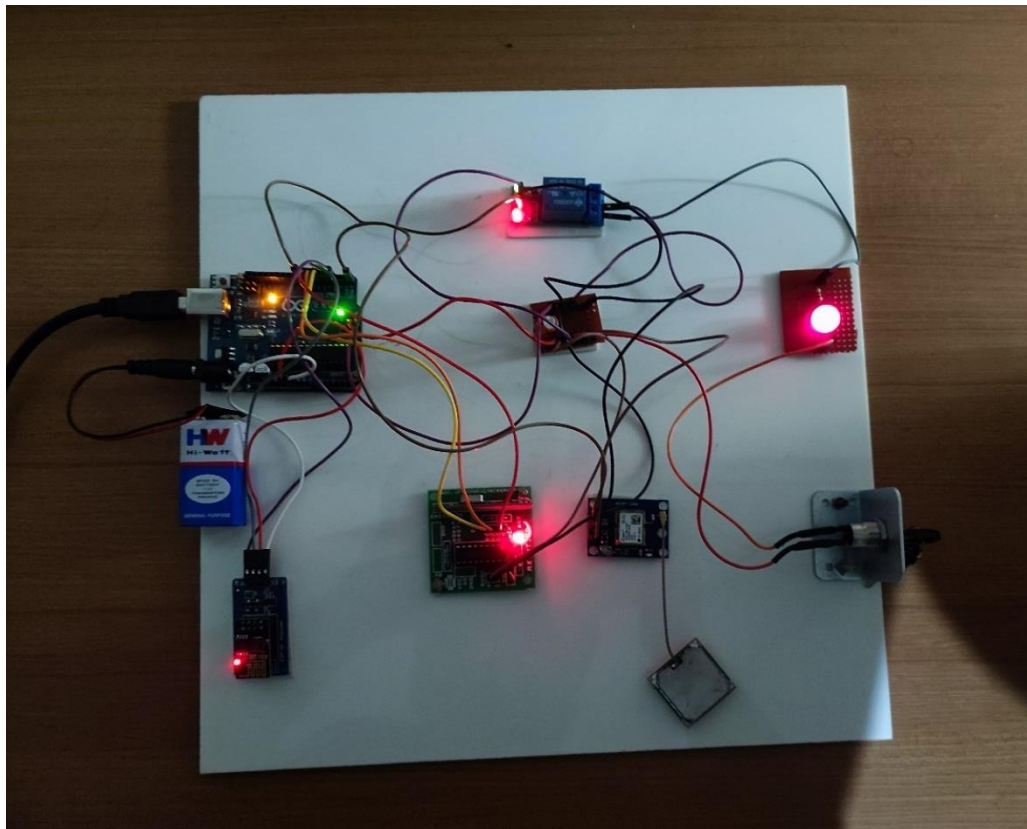


Figure 9.3.2 RF Transmitter



Channels
Apps
Devices
Support

Commercial Use
How to Buy

# Channel 2512656

Channel ID: **2512656**  
Author: **mwa000028834730**  
Access: Private

Private View
Public View
Channel Settings
Sharing
API Keys
Data Import / Export

Add Visualizations
Add Widgets
Export recent data

MATLAB Analysis
MATLAB Visualization

Channel Stats

Created: 13.days.ago  
Last entry: about.15.hours.ago  
Entries: 66

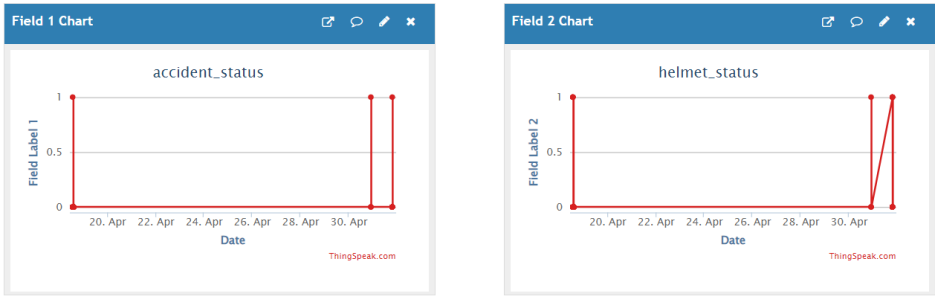


Figure 9.3.5 Data uploaded to thingspeak

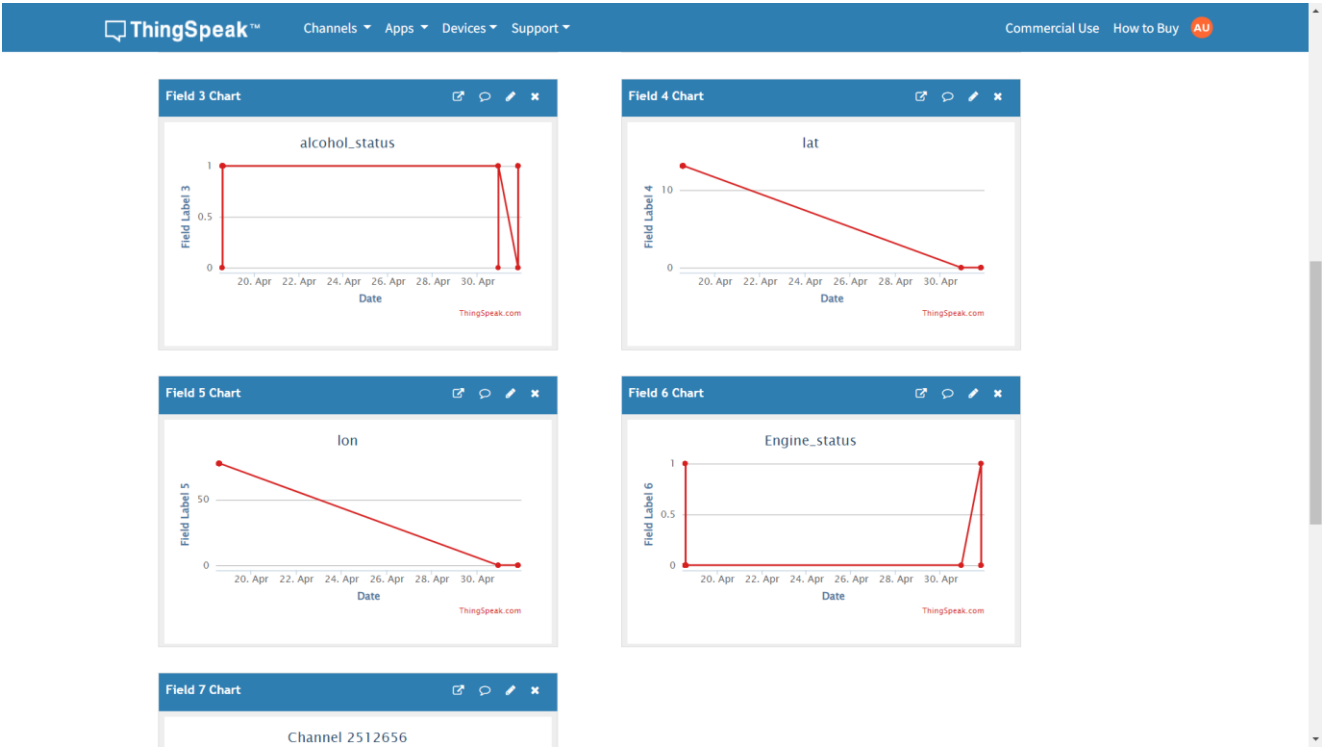


Figure 9.3.6 Data uploaded to thingspeak

## Source Code

### Bike Section:

#### helmet\_rx.ino

```
#define x A0          // ADXL sensor      //helmet part
#define y A1
#define z A2
#define mq_3 A4       // Alcoholic sensor
#define push 8        // push button
#define t1 4          // RF module
#define t2 5
#define t3 6

unsigned short level, X, Y, Z;

int ret_val;

int Alchol_thres = 60;

#define Accident_off digitalWrite(t1,0);
#define Accident_on  digitalWrite(t1,1);
#define Alcohol_off  digitalWrite(t2,0);
#define Alcohol_on   digitalWrite(t2,1);
#define helmet_off   digitalWrite(t3,0);
#define helmet_on    digitalWrite(t3,1);

int dir_print();

void setup()
{
    pinMode(x, INPUT);
    pinMode(y, INPUT);
    pinMode(z, INPUT);
    pinMode(mq_3, INPUT);
    pinMode(push, INPUT);
    pinMode(t1, OUTPUT);
    pinMode(t2, OUTPUT);
    pinMode(t3, OUTPUT);
    helmet_off;
    Accident_off;
    Alcohol_off;
    Serial.begin(9600);
```

```

}

void loop()
{
  int level=map(analogRead(mq_3),0,1023,0,100);
  Serial.println(level);

  int button=digitalRead(push);
  Serial.println(button);
  delay(1000);
  if (level < Alchol_thres && button == 1)
  {
    helmet_on;

    Serial.println ("heii:");
    while (1)
    {
      if (digitalRead(push) == 0 || level > Alchol_thres)
      {
        break;
      }
      //   char count = 5;
      ret_val = dir_print();
      int level=map(analogRead(mq_3),0,1023,0,100);
      Serial.println(level);
      if (ret_val == 0 && digitalRead(push) == 1 && level < Alchol_thres)
      {
        Serial.println("no accident");
        helmet_on;
        Alcohol_off;
        Accident_off;
        delay(100);
      }
      else if (ret_val == 0 && digitalRead(push) == 1 && level > Alchol_thres)
      {
        Alcohol_on;

```

```

    Serial.println("alcohol");
    delay(100);
}
else
{
    Accident_on;          // to stop engine
    Serial.println("accident");
}
//    delay(200);
}
}
else
{
    helmet_off;
    Accident_off;
    Alcohol_off;
}
delay(500);
}
int dir_print()
{
    X = analogRead(x);
    Y = analogRead(y);
    Z = analogRead(z);
    Serial.println(X);
    Serial.println(Y);
    Serial.println(Z);
    Serial.println(".....");
    delay(1000);
    // if (X > 400 || X < 270 || Y > 400 || Y < 270)
    if (Z < 430)
    {
        // int m=1;
        return 1;
    }
    else

```

```
{
//int m=0;
return 0;
}
}
```

## **esp02\_iot.h**

```
#include<TinyGPS.h>           // receiver part
#include"esp02_iot.h"
const int r1 = 5, r2 = 6, r3 = 7, relay_pin = 8;
float lat, lon;
TinyGPS gps;
int flag, helmet_status, alcohol_status, Engine_status, accident_status;
long Start1, Start;
void gps_read();
// void send_sms(String , String );
void setup()
{
  // Serial.begin(9600);
  pinMode(r1, INPUT);
  pinMode(r2, INPUT);
  pinMode(r3, INPUT);
  pinMode(relay_pin, OUTPUT);
  digitalWrite(relay_pin, 1);
  Start = millis();
  Start1 = millis();
  ini_iot();
  // digitalWrite(relay_pin, 0);
  // delay(5000);
  // digitalWrite(relay_pin, 1);
}
void loop()
{
  Serial.println(digitalRead(r1));
```



```

Serial.println(digitalRead(r2));
Serial.println(digitalRead(r3));
Serial.println(".....");
delay(500);
gps_read();
if (digitalRead(r1) == 0 && digitalRead(r2) == 0 && digitalRead(r3) == 1)
{
    digitalWrite(relay_pin, 0);
    alcohol_status = 0;
    Engine_status = 1;
    accident_status = 0;
    helmet_status = 1;
    flag = 0;
}
else if (digitalRead(r1) == 0 && digitalRead(r2) == 0 && digitalRead(r3) == 0 )
{
    digitalWrite(relay_pin, 1);
    Engine_status = 0;
    accident_status = 0;
    helmet_status = 0;
    alcohol_status = 1;
    flag = 0;
}
else if (digitalRead(r1) == 0 && digitalRead(r2) == 1 && digitalRead(r3) == 1)
{
    digitalWrite(relay_pin, 1);
    alcohol_status = 1;
    Engine_status = 0;
    accident_status = 0;
    helmet_status = 0;
    flag = 0;
}
else if (digitalRead(r1) == 1 && digitalRead(r2) == 0 && digitalRead(r3) == 1 )
{
    digitalWrite(relay_pin, 1);
    Engine_status = 0;

```

```

    helmet_status = 1;
    accident_status = 1;
}
String getData2 = "&field1=" + String(accident_status) + "&field2=" + String(helmet_status) +
"&field3=" + String(alcchol_status) + "&field4=" + String(lat) + "&field5=" + String(lon) +
"&field6=" + String(Engine_status);
if (millis() - Start > 40000)
{
    senddata(getData2);
    Start = millis();
}
}
void gps_read()
{
    Start1 = millis();
    while (millis() - Start1 < 2000)
    {
        while (Serial.available())
            if (gps.encode(Serial.read()))
            {
                gps.f_get_position(&lat, &lon);
            }
    }
}

```

## Helmet Section

### helmet tx.ino

```
#define x A0          // ADXL sensor      //helmet part
#define y A1
#define z A2
#define mq_3 A4       // Alcoholic sensor
#define push 8        // push button
#define t1 4          // RF module
#define t2 5
#define t3 6
unsigned short level, X, Y, Z;
int ret_val;
int Alchol_thres = 60;
#define Accident_off digitalWrite(t1,0);
#define Accident_on  digitalWrite(t1,1);
#define Alcohol_off  digitalWrite(t2,0);
#define Alcohol_on   digitalWrite(t2,1);
#define helmet_off   digitalWrite(t3,0);
#define helmet_on    digitalWrite(t3,1);
int dir_print();
void setup()
{
  pinMode(x, INPUT);
  pinMode(y, INPUT);
  pinMode(z, INPUT);
  pinMode(mq_3, INPUT);
  pinMode(push, INPUT);
  pinMode(t1, OUTPUT);
  pinMode(t2, OUTPUT);
  pinMode(t3, OUTPUT);
  helmet_off;
  Accident_off;
  Alcohol_off;
  Serial.begin(9600);}
```

```

void loop()
{
  int level=map(analogRead(mq_3),0,1023,0,100);
  Serial.println(level);
  int button=digitalRead(push);
  Serial.println(button);
  delay(1000);
  if (level < Alchol_thres && button == 1)
  {
    helmet_on;
    Serial.println ("heii:");
    while (1)
    {
      if (digitalRead(push) == 0 || level > Alchol_thres)
      {
        break;
      }
      //   char count = 5;
      ret_val = dir_print();
      int level=map(analogRead(mq_3),0,1023,0,100);
      Serial.println(level);
      if (ret_val == 0 && digitalRead(push) == 1 && level < Alchol_thres)
      {
        Serial.println("no accident");
        helmet_on;
        Alcohol_off;
        Accident_off;
        delay(100);
      }
    }
    else if(ret_val == 0 && digitalRead(push) == 1 && level >Alchol_thres)
    {
      Alcohol_on;
      Serial.println("alcohol");
      delay(100);
    }
  }
  else

```

```

    {
        Accident_on;          // to stop engine
        Serial.println("accident");
    }
//    delay(200);
    }}
else
{
    helmet_off;
    Accident_off;
    Alcohol_off;
}
delay(500);
}
int dir_print()
{
    X = analogRead(x);
    Y = analogRead(y);
    Z = analogRead(z);
    Serial.println(X);
    Serial.println(Y);
    Serial.println(Z);
    Serial.println(".....");
    delay(1000);
    // if (X > 400 || X < 270 || Y > 400 || Y < 270)
    if (Z < 430)
    {
        // int m=1;
        return 1;
    }
    else
    {
        //int m=0;
        return 0;
    }
}

```

## Certificate



**ಕರುನಾಡು ಟೆಕ್ನಾಲಜೀಸ್ ಪ್ರೈವೇಟ್ ಲಿಮಿಟೆಡ್**  
**KARUNADU TECHNOLOGIES PRIVATE LIMITED**

#17, A.T.K. Complex, 2nd & 4th Floor, Acharya College Main Road, Beside Karur Vysya Bank,  
Gutibasaveshwara Nagar, Chikkabanavara, Bengaluru - 560090

Ref No: KTPT/INT/2024/0006

Date: 18-April-2024

### To Whomsoever It May Concern

#### CERTIFICATE

This is to certify that **Mr/Ms. ABHISHEK U S** from **SDM College, Ujire** (Software and App Development) bearing **Reg.No-211301** has successfully completed internship programme at **Karunadu Technologies Private Limited** from 25-Jan-2024 to 18-April-2024 and found the candidate's performance is good.

During his/her Internship, he/she was exposed to the various activities in Internet of Things (IoT) and carried out a project on **"SMART RIDING HELMET BASED ON CLOUD AND INTERNET OF THINGS"**.

During the period of his/her Internship training with us he/she was found punctual, hardworking and inquisitive.

His/her association with us was very fruitful and we wish him/her all the best in his/her future endeavours.

  
**SUNIL KUMAR**  
**GUIDE**

Karunadu Technologies Pvt.Ltd.

  
**MAHESH DEGINAL**  
**Managing Director**  
Karunadu Technologies Pvt.Ltd.

#### Registered Office:

No. 59, 2nd Main, 1st Cross, Near Singapura Village Bus Stop, Vidyaranyapura, Bengaluru - 560064 ,  
**CIN:** U74999KA2018PTC114911, **Mob:** +91 9964823646 /+91 9902913646 **E-mail:** karunadutechnologies@gmail.com ,  
support@karunadutechnologies.com, **Website:** www.karunadutechnologies.com

## **CONCLUSION**

The system designed provides safety and reduces the after effects of the accident, notifying about the accident will provide timely care and treatments to the victim, reducing the severe impacts on the Raider. The fingerprint authorization prevents vehicle theft and provides security. The alcohol detection will prevent drink and drive scenario and the effects of drink and driving to public and the rider himself.

## **10.1 Future Enhancement:**

- Using this system, users can answer or make calls, listen to music and GPS navigation instructions.
- Control other smart applications via voice commands from a mobile phone without having to use the device.
- Easy access buttons allow users to control these functions directly from the helmet
- Create an android application to monitor Manually



## **Bibliography**

1. [https://www.tutorialspoint.com/internet\\_of\\_things/index.html](https://www.tutorialspoint.com/internet_of_things/index.html).
2. <https://www.javatpoint.com/iot-internet-of-things>.
3. <https://www.pantechsolutions.net/active-plant-wall-based-on-internet-of-things>.
4. <https://www.mathworks.com/products/thingspeak.html>.