CHAPTER 1: INTRODUCTION

The Internet of things (IoT) is the network of physical devices, vehicles, home appliances, and other items embedded with electronics, software, sensors, actuators, and connectivity which enables these things to connect, collect and exchange data IoT involves extending Internet connectivity beyond standard devices, such as desktops, laptops, smartphones and tablets, to any range of traditionally dumb or non-internet-enabled physical devices and everyday objects. Embedded with technology, these devices can communicate and interact over the Internet, and they can be remotely monitored and controlled. With the arrival of driverless vehicles, a branch of IoT, i.e. the Internet of Vehicle starts to gain more attention. The definition of the Internet of things and other technologies has evolved due to convergence of multiple technologies, real-time analytics, machine learning, commodity sensors, and embedded systems. Traditional fields of embedded systems, wireless sensor networks.

The control systems, automation (including home and building automation), and others all contribute to enabling the Internet of things. IoT devices are a part of the larger concept of home automation, which can include lighting, heating and air conditioning, media and security systems. Long term benefits could include energy savings by automatically ensuring lights and electronics are turned off.

A smart home or automated home could be based on a platform or hubs that control smart devices and appliances. For instance, using Apple's HomeKit, manufacturers can get their home products and accessories be controlled by an application in iOS devices such as the iPhone and the Apple Watch. This could be a dedicated app or iOS native applications such as Siri. This can be demonstrated in the case of Lenovo's Smart Home Essentials, which is a line of smart home devices that are controlled through Apple's Home app or Siri without the need for a Wi-Fi bridge. There are also dedicated smart home hubs that are offered as standalone platforms to connect different smart home products and these include the Amazon Echo, Apple's HomePod, and Samsung's Smart Things Hub. The Internet of Medical Things (also called the internet of health things) is an application of the IoT for medical and health related purposes, data collection and analysis for research, and monitoring. This 'Smart Healthcare', as it can also be called, led to the creation of a digitized healthcare system, connecting available medical resources and healthcare services.

IoT devices can be used to enable remote health monitoring and emergency notification systems. These health monitoring devices can range from blood pressure and heart rate monitors to advanced devices capable of monitoring specialized implants, such as pacemakers,

Fitbit electronic wristbands, or advanced hearing aids. Some hospitals have begun implementing "smart beds" that can detect when they are occupied and when a patient is attempting to get up. It can also adjust itself to ensure appropriate pressure and support is applied to the patient without the manual interaction of nurses. A 2015 Goldman Sachs report indicated that healthcare IoT devices "can save the United States more than $300 billion in annual healthcare expenditures by increasing revenue and decreasing cost. Moreover, the use of mobile devices to support medical follow-up led to the creation of 'm-health', used "to analyse, capture, transmit and store health statistics from multiple resources, including sensors and other biomedical acquisition systems". Specialized sensors can also be equipped within living spaces to monitor the health and general well-being of senior citizens, while also ensuring that proper treatment is being administered and assisting people regain lost mobility via therapy as well. These sensors create a network of intelligent sensors that are able to collect, process, transfer and analyse valuable information in different environments, such as connecting in-home monitoring devices to hospital-based systems. Other consumer devices to encourage healthy living, such as connected scales or wearable heart monitors, are also a possibility with the IoT.] End-to-end health monitoring IoT platforms are also available for antenatal and chronic patients, helping one manage health vitals and recurring medication requirements.

As of 2018 IoMT was not only being applied in the clinical laboratory/industry, but also in the healthcare and health insurance industries. IoMT in the healthcare industry is now permitting doctors, patients and others involved (i.e. guardians of patients, nurses, families, etc.) to be part of a system, where patient records are saved in a database, allowing doctors and the rest of the medical staff to have access to the patient's information. Moreover, IoT-based systems are patient-centered, which involves being flexible to the patient's medical conditions. IoMT in the insurance industry provides access to better and new types of dynamic information. This includes sensor-based solutions such as biosensors, wearable's, connected health devices and mobile apps to track customer behaviour. This can lead to more accurate underwriting and new pricing models.

The IoT can assist in the integration of communications, control, and information processing across various transportation systems. Application of the IoT extends to all aspects of transportation systems (i.e. the vehicle. the infrastructure, and the driver or user). Dynamic interaction between these components of a transport system enables inter and intra vehicular communication. Smart traffic control, smart city, smart parking, electronic toll collection

systems, logistic and fleet management, vehicle control, and safety and road assistance. In Logistics and Fleet Management for example, The IoT platform can continuously monitor the location and conditions of cargo and assets via wireless sensors and send specific alerts when management exceptions occur (delays, damages, thefts, etc.). If combined with Machine Learning then it also helps in reducing traffic accidents by introducing drowsiness alerts to drivers and providing self driven cars too. The IoT can realize the seamless integration of various manufacturing devices equipped with sensing, identification, processing, communication, actuation, and networking capabilities.

Based on such a highly integrated smart cyberphysical space, it opens the door to create whole new business and market opportunities for manufacturing. Network control and management of manufacturing equipment, asset and situation management, or manufacturing process control bring the IoT within the realm of industrial applications and smart manufacturing as well. The IoT intelligent systems enable rapid manufacturing of new products, dynamic response to product demands, and real-time optimization of manufacturing production and supply chain networks, by networking machinery, sensors and control systems together. Digital control systems to automate process controls, operator tools and service information systems to optimize plant safety and security are within the purview of the IoT. But it also extends itself to asset management via predictive maintenance, statistical evaluation, and measurements to maximize reliability. Smart industrial management systems can also be integrated with the Smart Grid, thereby enabling real-time energy optimization. Measurements, automated controls, plant optimization, health and safety management, and other functions are provided by a large number of networked sensors.

The term industrial Internet of things (IIoT) is often encountered in the manufacturing industries, referring to the industrial subset of the IoT. IIoT in manufacturing could generate so much business value that it will eventually lead to the fourth industrial revolution, so the so-called Industry 4.0. It is estimated that in the future, successful companies will be able to increase their revenue through Internet of things by creating new business models and improve productivity, exploit analytics for innovation, and transform workforce. The potential of growth by implementing IIoT may generate $12 trillion of global GDP by 2030. While connectivity and data acquisition are imperative for IIoT, they should not be the purpose, rather the foundation and path to something bigger. Among all the technologies, predictive maintenance is probably a relatively "easier win" since it is applicable to existing assets and management systems. The objective of intelligent maintenance systems is to reduce unexpected

downtime and increase productivity. And to realize that alone would generate around up to 30% over the total maintenance costs. Industrial big data analytics will play a vital role in manufacturing asset predictive maintenance, although that is not the only capability of industrial big data. Cyber-physical systems (CPS) is the core technology of industrial big data and it will be an interface between human and the cyber world. Cyber-physical systems can be designed by following the 5C (connection, conversion, cyber, cognition, configuration) architecture, and it will transform the collected data into actionable information, and eventually interfere with the physical assets to optimize processes.

An IoT-enabled intelligent system of such cases was proposed in 2001 and later demonstrated in 2014 by the National Science Foundation Industry/University Collaborative Research Center for Intelligent Maintenance Systems (IMS) at the University of Cincinnati on a bandsawmachine in IMTS 2014 in Chicago. Bandsaw machines are not necessarily expensive, but the bandsaw belt expenses are enormous since they degrade much faster. However, without sensing and intelligent analytics, it can be only determined by experience when the band saw belt will actually break. The developed prognostics system will be able to recognize and monitor the degradation of band saw belts even if the condition is changing, advising users when is the best time to replace the belt. This will significantly improve user experience and operator safety and ultimately save on costs. Monitoring and controlling operations of sustainable urban and rural infrastructures like bridges, railway tracks and on- and offshore wind-farms is a key application of the IoT.

The IoT infrastructure can be used for monitoring any events or changes in structural conditions that can compromise safety and increase risk. IoT can benefit the construction industry by cost saving, time reduction, better quality workday, paperless workflow and increase in productivity. It can help in taking faster decisions and save money with Real-Time Data Analytics. It can also be used for scheduling repair and maintenance activities in an efficient manner, by coordinating tasks between different service providers and users of these facilities. IoT devices can also be used to control critical infrastructure like bridges to provide access to ships. Usage of IoT devices for monitoring and operating infrastructure is likely to improve incident management and emergency response coordination, and quality of service, up-times and reduce costs of operation in all infrastructure related areas. Even areas such as waste management can benefit from automation and optimization that could be brought in by the IoT. There are several planned or ongoing large-scale deployments of the IoT, to enable better management of cities and systems. For example, Songdo, South Korea, the first

of its kind fully equipped and wired smart city, is gradually being built, with approximately 70 percent of the business district completed as of June 2018. Much of the city is planned to be wired and automated, with little or no human intervention.

Another application is a currently undergoing project in Santander, Spain. For this deployment, two approaches have been adopted. This city of 180,000 inhabitants has already seen 18,000 downloads of its city smartphone app. The app is connected to 10,000 sensors that enable services like parking search, environmental monitoring, digital city agenda, and more. City context information is used in this deployment so as to benefit merchants through a spark deals mechanism based on city behaviour that aims at maximizing the impact of each notification. Other examples of large-scale deployments underway include the Sino-Singapore Guangzhou Knowledge City work on improving air and water quality, reducing noise pollution, and increasing transportation efficiency in San Jose, California and smart traffic management in western Singapore. French based company in the France , Sigfox, commenced building an ultra-narrowband wireless data network in the San Francisco Bay Area in 2014, the first business to achieve such a deployment in the U.S. It subsequently announced it would set up a total of 4000 base stations to cover a total of 30 cities in the U.S. by the end of 2016, making it the largest IoT network coverage provider in the country thus far. Another example of a large deployment is the one completed by New York Waterways in New York City to connect all the city's vessels and be able to monitor them live 24/7. The network was designed and engineered by Fluid mesh Networks, a Chicago-based company developing wireless networks for critical applications. The NYWW network is currently providing coverage on the Hudson River, East River, and Upper New York Bay. With the wireless network in place, NY Waterway is able to take control of its fleet and passengers in a way that was not previously possible. New applications can include security, energy and fleet management, digital signage, public Wi-Fi, paperless ticketing and others. Significant numbers of energy-consuming devices (e.g. switches, power outlets, bulbs, televisions, etc.) already integrate Internet connectivity, which can allow them to communicate with utilities to balance power generation and energy usage and optimize energy consumption as a whole. These devices allow for remote control by users, or central management via a cloud-based interface, and enable functions like scheduling (e.g., remotely powering on or off heating systems, controlling ovens, changing lighting conditions etc.). The smart grid is a utility-side IoT application; systems gather and act on energy and power-related information to improve the efficiency of the production and distribution of electricity. Using advanced metering infrastructure (AMI) Internet-connected

devices, electric utilities not only collect data from end-users, but also manage distribution automation devices like transformers. Environmental monitoring applications of the IoT typically use sensors to assist in environmental protection by monitoring air or water quality, atmospheric or soil conditions. and can even include areas like monitoring the movements of wildlife and their habitats. Development of resource-constrained devices connected to the Internet also means that other applications like earthquake or tsunami early-warning systems can also be used by emergency services to provide more effective aid. IoT devices in this application typically span a large geographic area and can also be mobile. It has been argued that the standardization IoT brings to wireless sensing will revolutionize this area.

In a home the kitchen plays a major role in keeping the family alive and together, as the kitchen serves as the place where meals are prepared and also the meeting point where members of the family meet to eat. Amongst all the appliances or devices that are found in a kitchen, refrigerator is of great importance. This is due to the fact that the refrigerator house or keep intact food items that are needed in the home. Traditional refrigerators have been performing great task in preserving food items for a period of time, but there is need for more efficient ways of preserving and managing food items. With the innovations in technology came the Internet of Things (IoT) where divers appliance are connected together courtesy the internet, home appliance inclusive. Internet Refrigerator, which is a typical IoT also got innovated with the expectations to make life more convenient and comfortable by managing the kitchen more efficiently. It is expected to manage items or resources kept in it, save unnecessary cost, save food wastage, plan an organized menu, as well as organized shopping list. This research work looks at Internet Refrigerator as a typical Internet of things, its components, benefits and challenges as well as its acceptability.

The “Smart Refrigerator” is purely the project based on IOT, this project is made with the help of Raspberry Pi. As in today’s time all people are busy and need new and fast technologies so that their time would be saved. In this project refrigerator consists of compartment like for vegetables, for water or soft drinks, for milk etc. and the information of the items inside the refrigerator will be stored in server. We will access the server using the mobile application, and can check the details that how many items are there in the fridge. We can also check the validity of the items that after how many days it is going to expire. We can also control our fridge if we are not in our house we can on/off it if required. If we forget to close the door of the fridge then we will get notification in the mobile. One can regulate the temperature of freezer and refrigerator. There is also a function through which we can check the level of water bottle. The

Smart Refrigerator module is designed to convert any existing refrigerator into an intelligent cost effective appliance using sensors .The smart refrigerator is capable of sensing and monitoring its contents. The smart refrigerator is also able to remotely notify the user about scarce products via SMS (Short Message Service) and email. It also facilitates the purchase of scarce items by providing a link of the online vendor of that particular item. Additional functionality includes the acknowledgement of a placed order in order to avoid the purchase of the same item by different users of the same smart refrigerator.

CHAPTER 2: COMPONENTS

2.1 Arduino

Arduino is open source computer hardware and software company, project and user community that designs and manufactures microcontroller-based kits for building digital devices and interactive objects that can sense and control objects in the physical world. These systems provide sets of digital and analog I/O pins that can be interfaced to various expansion boards ("shields") and other circuits. The boards feature serial communications interfaces, including USB on some models, for loading programs from personal computers. For programming the microcontrollers, the Arduino platform provides an integrated development environment (IDE) based on the Processing project, which includes support for the C, C++ and Java programming languages. The first Arduino was introduced in 2005, aiming to provide an inexpensive and easy way for novices and professionals to create devices that interact with their environment using sensors and actuators. Common examples of such devices intended for beginner hobbyists include simple robots, thermostats, and motion detectors. Arduino boards are available commercially in preassembled form, or as doit-yourself kits. The hardware design specifications are openly available, allowing the Arduino boards to be manufactured by anyone. Adafruit Industries estimated in mid-2011 that over 300,000 official Arduino had been commercially produced, and in 2013 that 700,000 official boards were in users' hands.

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

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

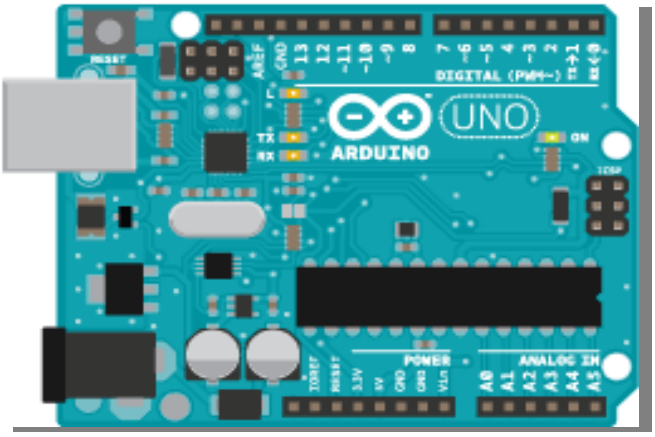


FIG 2.1 Arduino

2.2 Ultrasonic Sensor

Ultrasonic transducers or ultrasonic sensors are a type of acoustic sensor divided into three broad categories: transmitters, receivers and transceivers. Transmitters convert electrical signals into ultrasound, receivers convert ultrasound into electrical signals, and transceivers can both transmit and receive ultrasound. In a similar way to radar and sonar, ultrasonic transducers are used in systems which evaluate targets by interpreting the reflected signals. For example, by measuring the time between sending a signal and receiving an echo the distance of an object can be calculated. Passive ultrasonic sensors are basically microphones that detect ultrasonic noise that is present under certain conditions. Ultrasound can be used for measuring wind speed and direction (anemometer), tank or channel fluid level, and speed through air or water. For measuring speed or direction, a device uses multiple detectors and calculates the speed from the relative distances to particulates in the air or water. To measure tank or channel liquid level, and also sea level (tide gauge), the sensor measures the distance (ranging) to the surface of the fluid. Further applications include: humidifiers, sonar, medical ultrasonography, burglar alarms, non-destructive testing and wireless charging.

Systems typically use a transducer which generates sound waves in the ultrasonic range, above 18 kHz, by turning electrical energy into sound, then upon receiving the echo turn the sound waves into electrical energy which can be measured and displayed. Ultrasound can also be used to make point-to-point distance measurements by transmitting and receiving discrete bursts of ultrasound between transducers. This technique is known as Sonomicrometry where the transit-time of the ultrasound signal is measured electronically (i.e. digitally) and converted

mathematically to the distance between transducers assuming the speed of sound of the medium between the transducers is known. This method can be very precise in terms of temporal and spatial resolution because the time-of-flight measurement can be derived from tracking the same incident (received) waveform either by reference level or zero crossing. This enables the measurement resolution to far exceed the wavelength of the sound frequency generated by the transducers. Ultrasonic transducers convert AC into ultrasound, as well as the reverse. Ultrasonics, typically refers to piezoelectric transducers or capacitive transducers. Piezoelectric crystals change size and shape when a voltage is applied; AC voltage makes them oscillate at the same frequency and produce ultrasonic sound. Capacitive transducers use electrostatic fields between a conductive diaphragm and a backing plate. The beam pattern of a transducer can be determined by the active transducer area and shape, the ultrasound wavelength, and the sound velocity of the propagation medium. The diagrams show the sound fields of an unfocused and a focusing ultrasonic transducer in water, plainly at differing energy levels. Since piezoelectric materials generate a voltage when force is applied to them, they can also work as ultrasonic detectors. Some systems use separate transmitters and receivers, while others combine both functions into a single piezoelectric transceiver.

FIG 2.2 Ultrasonic sensor

2.3 Node MCU

NodeMCU is an open source IOT platform. It includes firmware which runs on the ESP8266 Wi-Fi SOC form Espressif Systems, and hardware which is based on the ESP-12 module. The term “NodeMCU” by default refers to the firmware rather than the development kit. The firmware uses the Lua scripting language. It is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. It uses many open source projects such as luacjson and spiffs.

NodeMCU was created shortly after the ESP8266 came out. On December 30, 2013, Espressif Systems began production of the ESP8266. The ESP8266 is a Wi-Fi SoC integrated with a Tensilica Xtensa LX106 core, widely used in IoT applications. NodeMCU started on 13 Oct

2014, when Hong committed the gerber file of an ESP8266 board, named devkit v0.9. Later that month, Tuan PM ported MQTT client library from Contiki to the ESP8266 SoC platform, and committed to NodeMCU project, then NodeMCU to easily drive LCD, Screen, OLED, even VGA displays. As Arduino.cc began developing new MCU boards based on non-AVR processors like the ARM/SAM MCU and used in the Arduino Due, they needed to modify the Arduino IDE so that it would be relatively easy to change the IDE to support alternate tool chain to allow Arduino C/C++ to be compiled down to these new processors. They did this with the introduction of the Board Manager and the SAM Core. A "core" is the collection of software components required by the Board Manager and the Arduino IDE to compile an Arduino C/C++ source file down to target MCU's machine language. NodeMCU is a Wi-Fi SOC (system on a chip) produced by Espressif Systems.

It is based ESP8266 -12E Wi-Fi module. It is a highly integrated chip designed to provide full internet connectivity in a small package. It can be programmed directly through USB port using LUA programming or Arduino IDE. ESP8266 module in tune has a micro controller with Wi-Fi. You can program ESP8266using Arduino, NodeMCU IDE or ESP8266 SDK. No, NodeMCU is a firmware for LUA script. NodeMCU devkit is a wifi controller board of ESP8266 SoC. The ESP8266 is a low-cost Wi-Fi microchip with full TCP/IP stack and microcontroller capability produced by Shanghai-based Chinese manufacturer Espressif Systems. The chip first came to the attention of western makers in August 2014 with the ESP-01 module, made by a third-party manufacturer Ai-Thinker. Usable pins. The ESP8266 has 17 GPIO pins (0-16), however, you can only use 11 of them, because 6 pins (GPIO 6 - 11) are used to connect the flash memory chip. This is the small 8-legged chip right next to the ESP8266. If you try to use one of these pins, you might crash your program.

FIG 2.3 NodeMCU

2.4 Jumper Wires

A jump wire (also known as jumper, jumper wire, jumper cable, DuPont wire, or DuPont cable – named for one manufacturer of them) is an electrical wire, or group of them in a cable, with a connector or pin at each end (or sometimes without them – simply "tinned"), which is normally used to interconnect the components of a breadboard or other prototype or test circuit, internally or with other equipment or components, without soldering. Individual jump wires are fitted by inserting their "end connectors" into the slots provided in a breadboard, the header connector of a circuit board, or a piece of test equipment. Jumper wires are simply wires that have connector pins at each end, allowing them to be used to connect two points to each other without soldering. Jumper wires are typically used with breadboard and other prototyping tools in order to make it easy to change a circuit as needed. Fairly simple. In fact, it doesn't get much more basic than jumper wires. Though jumper wires come in a variety of colors, the colors don't actually mean anything. This means that a red jumper wire is technically the same as a black one. But the colors can be used to your advantage in order to differentiate between types of connections, such as ground or power. Jumper wires typically come in three versions: male-to-male, male-to-female and female-to-female. The difference between each is in the end point of the wire. Male ends have a pin protruding and can plug into things, while female ends do not and are used to plug things into. Male-to-male jumper wires are the most common and what you likely will use most often. When connecting two ports on a breadboard, a male-to-male wire is what you'll need.

There are different types of jumper wires. Some have the same type of electrical connector at both ends, while others have different connectors. Some common connectors are:

- Solid tips – are used to connect on/with a breadboard or female header connector. The arrangement of the elements and ease of insertion on a breadboard allows increasing the mounting density of both components and jump wires without fear of short-circuits. The jump wires vary in size and colour to distinguish the different working signals.
- Crocodile clips – are used, among other applications, to temporarily bridge sensors, buttons and other elements of prototypes with components or equipment that have arbitrary connectors, wires, screw terminals etc.
- Banana connectors – are commonly used on test equipment for DC and low-frequency AC signals.

- Registered jack (RJnn) – are commonly used in telephone (RJ11) and computer networking (RJ45).
- RCA connectors – are often used for audio, low-resolution composite video signals, or other low-frequency applications requiring a shielded cable.
- RF connectors – are used to carry radio frequency signals between circuits, test equipment, and antennas.

FIG 2.4 Jumper Wires

2.5 Push Button

A push-button (also spelled pushbutton) or simply button is a simple switch mechanism for controlling some aspect of a machine or a process. Buttons are typically made out of hard material, usually plastic or metal. The surface is usually flat or shaped to accommodate the human finger or hand, so as to be easily depressed or pushed. Buttons are most often biased switches, although many un-biased buttons (due to their physical nature) still require a spring to return to their un-pushed state. Terms for the pushing a button include pressing, depressing, mashing, slapping, hitting and punching.

The "push-button" has been utilized in calculators, push-button telephones, kitchen appliances, and various other mechanical and electronic devices, home and commercial. In industrial and commercial applications, push buttons can be connected together by a mechanical linkage so that the act of pushing one button causes the other button to be released. In this way, a stop button can "force" a start button to be released. This method of linkage is used in simple manual operations in which the machine or process has no electrical circuits for control. Red pushbuttons can also have large heads (called mushroom heads) for easy operation and to facilitate the stopping of a machine. These pushbuttons are called emergency stop buttons and for increased safety are mandated by the electrical code in many jurisdictions. This large mushroom shape can also be found in buttons for use with operators who need to wear gloves for their work and could not actuate a regular flush-mounted push button.

As an aid for operators and users in industrial or commercial applications, a pilot light is commonly added to draw the attention of the user and to provide feedback if the button is pushed. Typically this light is included into the center of the pushbutton and a lens replaces the pushbutton hard center disk. The source of the energy to illuminate the light is not directly tied to the contacts on the back of the pushbutton but to the action the pushbutton controls. In this way a start button when pushed will cause the process or machine operation to be started and a secondary contact designed into the operation or process will close to turn on the pilot light and signify the action of pushing the button caused the resultant process or action to start. To avoid an operator from pushing the wrong button in error, pushbuttons are often color-coded to associate them with their function. Commonly used colors are red for stopping the machine or process and green for starting the machine or process. In popular culture, the phrase "the button" (sometimes capitalized) refers to a (usually fictional) button that a military or government leader could press to launch nuclear weapons. A push button switch is a small, sealed mechanism that completes an electric circuit when you press on it. When it's on, a small metal spring inside makes contact with two wires, allowing electricity to flow.

FIG 2.5 Push Button

2.6 LCD Display

LCD (liquid crystal display) is the technology used for displays in notebook and other smaller computers. Like light-emitting diode (LED) and gas-plasma technologies, LCDs allow displays to be much thinner than cathode ray tube (CRT) technology. LCDs consume much less power than LED and gas-display displays because they work on the principle of blocking light rather than emitting it. LCD (Liquid Crystal Display) screen is an electronic display module and find a wide range of applications. A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits. These modules are preferred over seven segments and other multi segment LEDs. The reasons being: LCDs are economical; easily programmable; have no limitation of displaying special & even custom characters (unlike in seven

segments), animations and so on. A 16x2 LCD means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. This LCD has two registers, namely, Command and Data. The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD. Click to learn more about internal structure of a LCD. An LCD is made with either a passive matrix or an active matrix display , display grid. The active matrix LCD is also known as a thin film transistor (TFT) display. The passive matrix LCD has a grid of conductors with pixels located at each intersection in the grid. A current is sent across two conductors on the grid to control the light for any pixel. An active matrix has a transistor located at each pixel intersection, requiring less current to control the luminance of a pixel. Some passive matrix LCD's have dual scanning, meaning that they scan the grid twice with current in the same time that it took for one scan in the original technology. However, active matrix is still a superior technology. Small LCD screens are common in portable consumer devices such as digital cameras, watches, calculators, and mobile telephones, including smartphones. LCD screens are also used on consumer electronics products such as DVD players, video game devices and clocks.

Each pixel of an LCD typically consists of a layer of molecules aligned between two transparent electrodes, and two polarizing filters(parallel and perpendicular), the axes of transmission of which are (in most of the cases) perpendicular to each other. Without the liquid crystal between the polarizing filters, light passing through the first filter would be blocked by the second (crossed) polarizer. Before an electric field is applied, the orientation of the liquid-crystal molecules is determined by the alignment at the surfaces of electrodes. In a twisted nematic (TN) device, the surface alignment directions at the two electrodes are perpendicular to each other, and so the molecules arrange themselves in a helical structure, or twist.

FIG 2.6 LCD Display

2.7 DHT11

This DHT11 Temperature and Humidity Sensor features a calibrated digital signal output with the temperature and humidity sensor capability. It is integrated with a high-performance 8-bit microcontroller. Its technology ensures the high reliability and excellent long-term stability. This sensor includes a resistive element and a sensor for wet NTC temperature measuring devices. It has excellent quality, fast response, anti-interference ability and high performance. Each DHT11 sensors features extremely accurate calibration of humidity calibration chamber. The calibration coefficients stored in the OTP program memory, internal sensors detect signals in the process, we should call these calibration coefficients. The single-wire serial interface system is integrated to become quick and easy. Small size, low power, signal transmission distance up to 20 meters, enabling a variety of applications and even the most demanding ones. The product is 4-pin single row pin package. Convenient connection, special packages can be provided according to users need. The digital temperature and humidity sensor DHT11 is a composite sensor that contains a calibrated digital signal output of temperature and humidity. The technology of a dedicated digital modules collection and the temperature and humidity sensing technology are applied to ensure that the product has high reliability and excellent long-term stability. The sensor includes a resistive sense of wet component and an NTC temperature measurement device, and is connected with a high-performance 8-bit microcontroller. Each DHT11 sensors features extremely accurate calibration of humidity calibration chamber. The calibration coefficients stored in the OTP program memory, internal sensors detect signals in the process, we should call these calibration coefficients.

Supply Voltage: +5 V
Temperature range: 0-50 °C
Error of ± 2 °C
Humidity: 20-90% RH ± 5% RH error
Interface: Digital

FIG 2.7 DHT11

CHAPTER 3: FIREBASE

Firebase is Google's mobile platform that helps you quickly develop high-quality apps and grow your business. Firebase gives you the tools to develop high-quality apps, grow your user base, and earn more money.

3.1 History

Firebase evolved from Envolve, a prior startup founded by James Tamplin and Andrew Lee in 2011. Envolve provided developers an API that enables the integration of online chat functionality into their websites. After releasing the chat service, Tamplin and Lee found that it was being used to pass application data that weren't chat messages. Developers were using Envolve to sync application data such as game state in real time across their users. Tamplin and Lee decided to separate the chat system and the real-time architecture that powered it. They founded Firebase as a separate company in April 2012.

Firebase Inc. raised seed funding in May 2012. The company further raised Series A funding in June 2013. In October 2014, Firebase was acquired by Google. In October 2015, Google acquired Divshot to merge it with the Firebase team. Since the acquisition, Firebase has grown inside Google and expanded their services to become a unified platform for mobile developers. Firebase now integrates with various other Google services to offer broader products and scale for developers. In January 2017, Google acquired Fabric and Crashlytics from Twitter to join those services to the Firebase team. Firebase launched Cloud Firestore, a Document Database, in October 2017.

3.2 Add Firebase to your app

To add Firebase to your app, you'll need a Firebase project and a short snippet of initialization code that has a few details about your project.

1. Create a Firebase project in the Firebase console.

•If you don't have an existing Firebase project, click Add project, then enter either an existing Google Cloud Platform project name or a new project name.

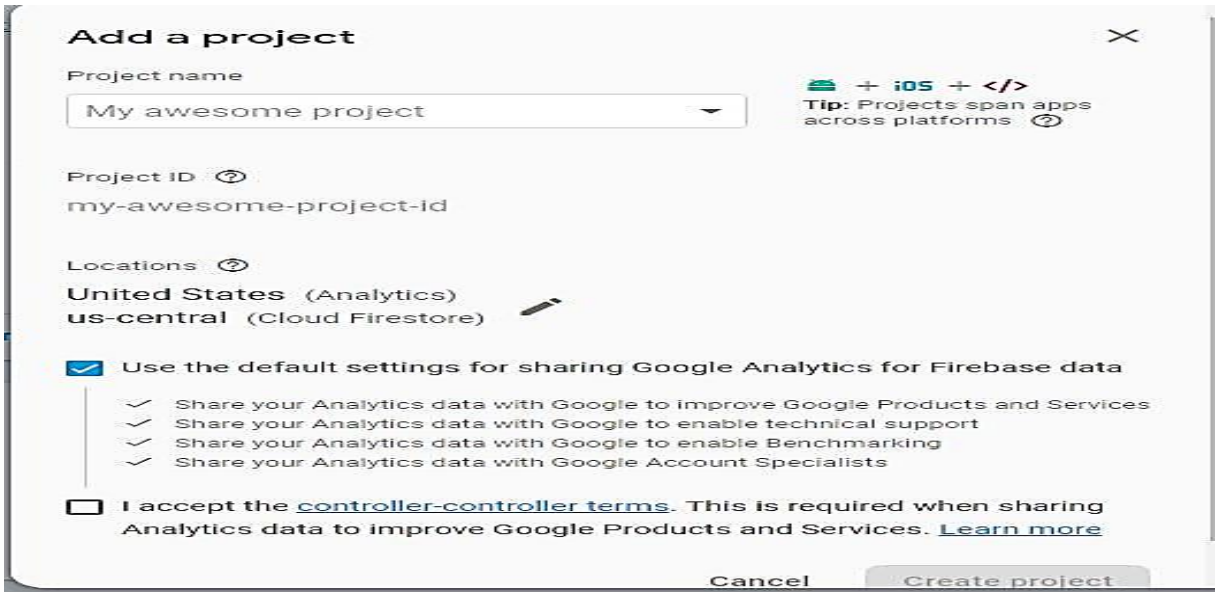


FIG 3.1 Add Project

•If you have an existing Firebase project that you'd like to use, select that project from the console.

2. From the project overview page in the Firebase console, click Add Firebase to your web app. If your project already has an app, select Add App from the project overview page.

3. Copy and paste your project's customized code snippet in the <head> tag of your page, before other script tags.

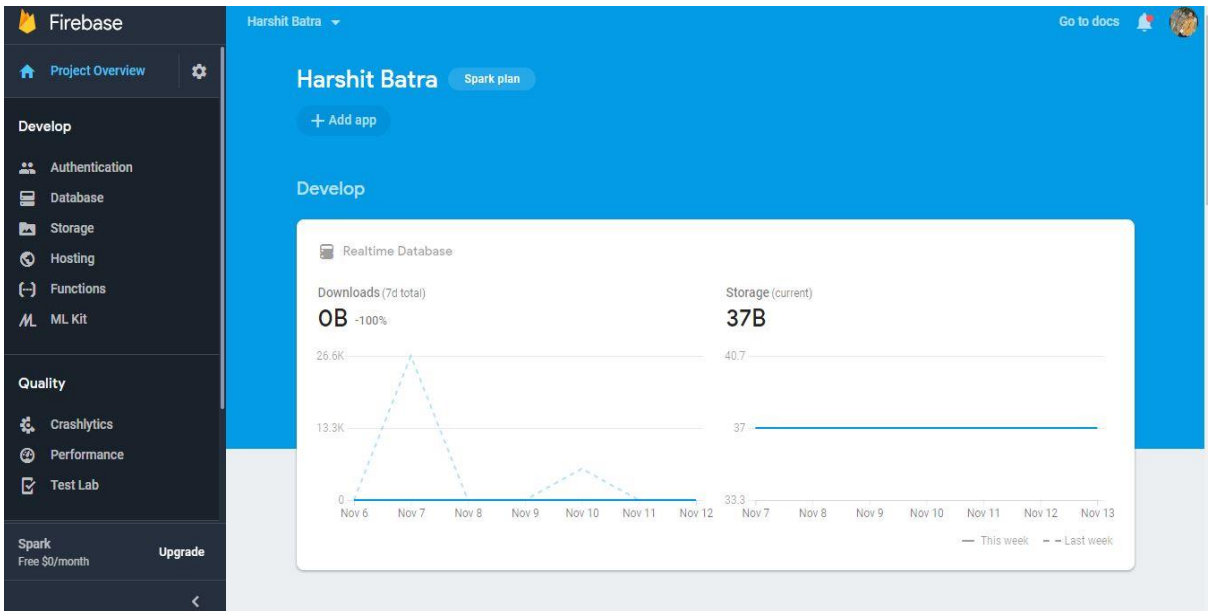


FIG 3.2 Firebase Console

Below is an example initialization code snippet:

```
<script src="https://www.gstatic.com/firebasejs/5.5.8/firebase.js"></script>
<script>
 // Initialize Firebase
 // TODO: Replace with your project's customized code snippet
 var config = {
   apiKey: "<API_KEY>",
   authDomain: "<PROJECT_ID>.firebaseapp.com",
   databaseURL: "https://<DATABASE_NAME>.firebaseio.com",
   projectId: "<PROJECT_ID>",
   storageBucket: "<BUCKET>.appspot.com",
   messagingSenderId: "<SENDER_ID>",
 };
 firebase.initializeApp(config);
</script>
```

FIG 3.3 Code Snippet

The snippet contains initialization information to configure the Firebase JavaScript SDK to use Authentication, Cloud Storage, the Realtime Database, and Cloud Firestore. You can reduce the amount of code that your app uses by only including the features that you need. The individually installable components are:

firebase-app — The core firebase client (required)

firebase-auth — Firebase Authentication (optional)

firebase-database — Firebase Realtime Database (optional)

firebase-firestore — Cloud Firestore (optional)

firebase-storage — Cloud Storage (optional)

firebase-messaging — Firebase Cloud Messaging (optional)

firebase-functions — Cloud Functions for Firebase (optional)

3.3 Host your web app using Firebase Hosting

If you're building a web app and your web app is entirely static content, you can deploy it using Firebase Hosting.

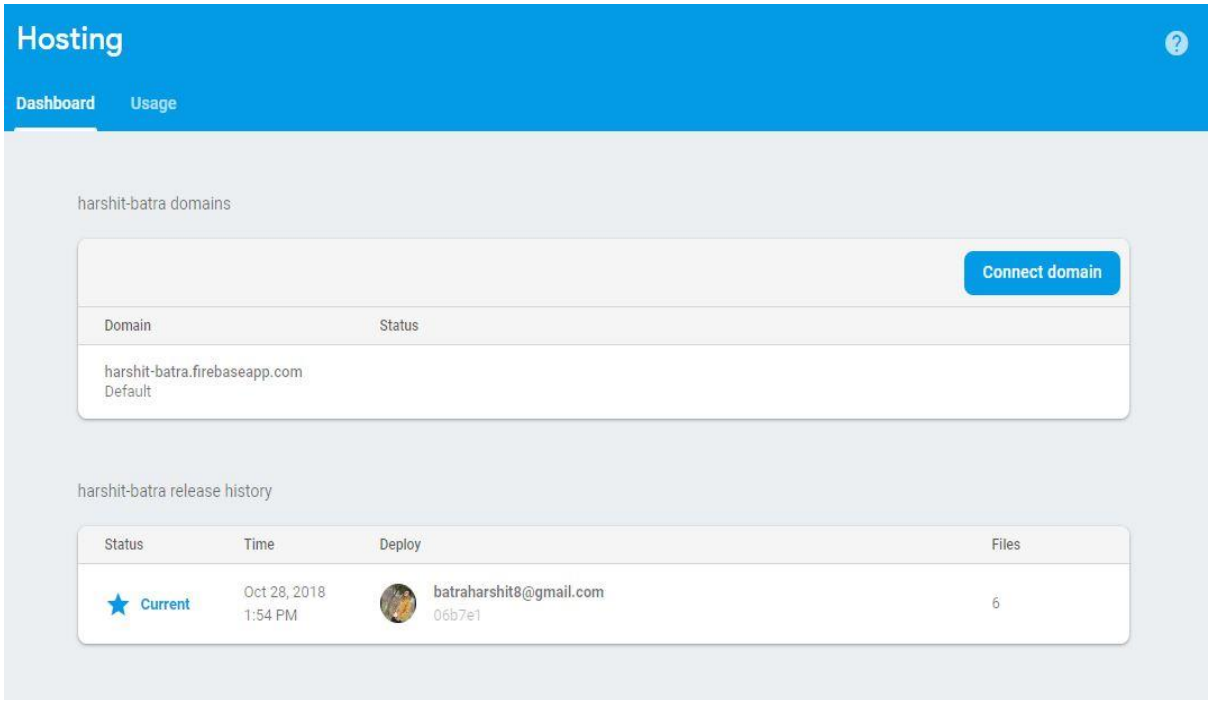


FIG 3.4 Firebase Hosting

Firebase Hosting provides fast and secure hosting for your web app and your static and dynamic content. Firebase Hosting is production-grade web content hosting for developers. With a single command, you can quickly and easily deploy web apps and serve both static and dynamic content to a global content delivery network (CDN).

3.4 Key capabilities

Served over a secure connection	The modern web is secure. Zero-configuration SSL is built into Firebase Hosting, so content is always delivered securely.
Fast content delivery	Each file that you upload is cached on SSDs at CDN edges around the world. No matter where your users are, the content is delivered fast.
Rapid deployment	Using the Firebase CLI, you can get your app up and running in seconds. Command line tools make it easy to add deployment targets into your build process.

One-click rollbacks	Quick deployments are great, but being able to undo mistakes is even better. Firebase Hosting provides full versioning and release management with one-click rollbacks.

3.5 How does it work?

Firebase Hosting is built for the modern web developer. Websites and apps are more powerful than ever with the rise of front-end JavaScript frameworks like Angular and static generator tools like Jekyll. Whether you are deploying a simple app landing page or a complex Progressive Web App (PWA), Hosting gives you the infrastructure, features, and tooling tailored to deploying and managing websites and apps. Hosting gives your project a subdomain on the firebaseapp.com domain. Using the Firebase CLI, you can deploy files from local directories on your computer to your Hosting server. Beyond serving static content, you can use Cloud Functions for Firebase to serve dynamic content on your site. All content is served over an SSL connection from the closest edge server on our global CDN.

In addition to content hosting, Firebase Hosting offers lightweight hosting configuration options for you to build sophisticated PWAs. You can easily rewrite URLs for client-side routing or set up custom headers. Once you're ready to take a site to production, you can connect your own domain name to Firebase Hosting. We automatically provision an SSL certificate for your domain so that all your content is served securely.

3.6 Implementation path

Firebase Hosting gives you a fast, secure, and reliable way to host your app's static assets (like HTML, CSS, JavaScript, and media files) and serve your dynamic content. Our production-grade hosting is backed by a global CDN. Hosting serves your content over SSL, by default, and can be used with your own custom domain or on a subdomain of firebaseapp.com.

To get started with Firebase Hosting, go to your Firebase project's Hosting page, then click Get Started. Next, install the Firebase CLI (a command line tool) so that you can initialize and deploy your site.

Install the Firebase CLI

- The Firebase CLI (Command Line Interface) requires Node.js and npm, which can both be installed by following the instructions on https://nodejs.org/. Installing Node.js also

installs npm. Once you have Node.js and npm installed, you can install the Firebase CLI via npm:

npm install -g firebase-tools

- This installs the globally available firebase command. To update to the latest version, simply re-run the same command.

Access your Firebase projects

- To connect your local machine to your Firebase account and obtain access to your Firebase projects, run the following command:

firebase login

Initialize your site

- From the root of your project directory, run the following command:

firebase init

- Running the firebase init command creates a firebase.json configuration file in the root of your project directory. You can learn more about this file in the customize hosting behavior section of this guide.

Select a public root directory

- When you initialize your site, you are prompted for a directory to use as the public root (default is "public"). If you don't already have a valid index.html file in your public root directory, one is created for you.

Deploy your site

- To deploy your site, run the following command from your project's root directory:

firebase deploy

- Your content will be deployed to your Firebase project's default Hosting site, your-firebase-project-id.firebaseapp.com.

Manage and rollback deploys

- From your Firebase project's Hosting page, you can see a full history of your deploys. If you have multiple Hosting sites, click **View** for the desired site to see its deployment history. To roll back to a previous deploy, hover over its entry in the list, click the overflow menu icon, then click **Rollback**.

3.7 Authenticate with Firebase using Password-Based Accounts using Javascript

You can use Firebase Authentication to let your users authenticate with Firebase using their email addresses and passwords, and to manage your app's password-based accounts.

Before you begin

Add Firebase to your JavaScript project. If you haven't yet connected your app to your Firebase project, do so from the Firebase console.

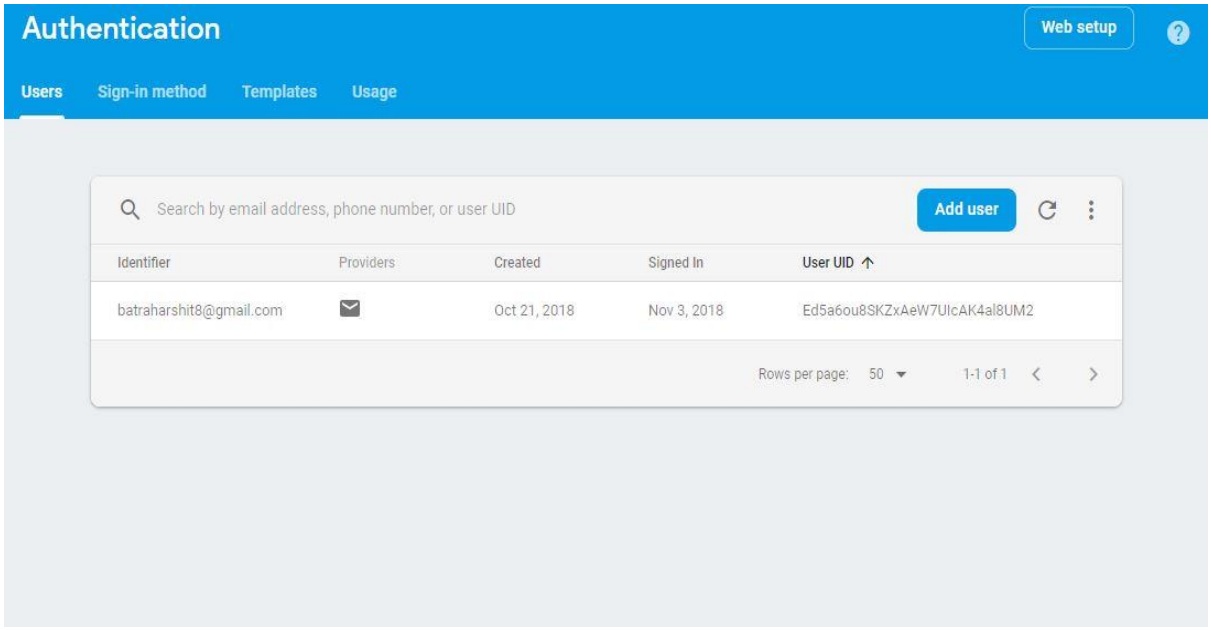


FIG 3.5 Authentication

Enable Email/Password sign-in: In the Firebase console, open the Auth section.

On the Sign in method tab, enable the Email/password sign-in method and click Save.

Create a password-based account

To create a new user account with a password, complete the following steps in your app's sign-up page:

When a new user signs up using your app's sign-up form, complete any new account validation steps that your app requires, such as verifying that the new account's password was correctly typed and meets your complexity requirements.

Create a new account by passing the new user's email address and password to createUserWithEmailAndPassword:

```
firebase.auth().createUserWithEmailAndPassword(email, password).catch(function(error) {
  // Handle Errors here.
  var errorCode = error.code;
  var errorMessage = error.message;
  // ...
});
```

If the new account was created, the user is signed in automatically. Have a look at the Next steps section below to get the signed in user details. Sign in a user with an email address and

password. The steps for signing in a user with a password are similar to the steps for creating a new account. In your app's sign-in page, do the following:

When a user signs in to your app, pass the user's email address and password to signInWithEmailAndPassword:

```
firebase.auth().signInWithEmailAndPassword(email, password).catch(function(error) {
  // Handle Errors here.
  var errorCode = error.code;
  var errorMessage = error.message;
  // ...
});
```

Have a look at the Next steps section below to get the signed in user details. This is also where you can catch and handle errors. For a list of error codes have a look at the Auth Reference Docs.

Next steps

After a user signs in for the first time, a new user account is created and linked to the credentials—that is, the user name and password, phone number, or auth provider information—the user signed in with. This new account is stored as part of your Firebase project, and can be used to identify a user across every app in your project, regardless of how the user signs in. In your apps, the recommended way to know the auth status of your user is to set an observer on the Authobject. You can then get the user's basic profile information from the User object. See Manage Users.

In your Firebase Realtime Database and Cloud Storage Security Rules, you can get the signed-in user's unique user ID from the auth variable, and use it to control what data a user can access.

You can allow users to sign in to your app using multiple authentication providers by linking auth provider credentials to an existing user account.

To sign out a user, call signOut:

```
firebase.auth().signOut().then(function() {
  // Sign-out successful.
}).catch(function(error) {
  // An error happened.
});
```

3.8 Firebase Realtime Database

Store and sync data with our NoSQL cloud database. Data is synced across all clients in realtime, and remains available when your app goes offline. The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in realtime to every connected client. When you build cross-platform apps with our iOS, Android, and JavaScript SDKs, all of your clients share one Realtime Database instance and automatically receive updates with the newest data.

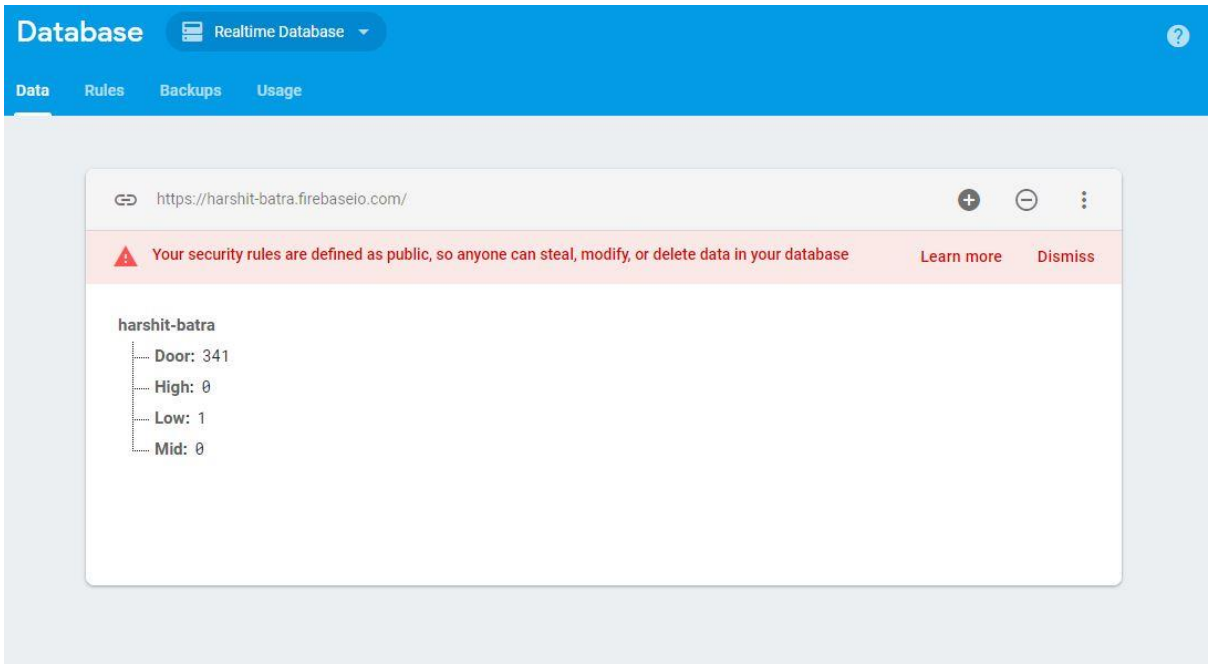


FIG 3.6 Firebase Database

3.9 How does it work?

The Firebase Realtime Database lets you build rich, collaborative applications by allowing secure access to the database directly from client-side code. Data is persisted locally, and even while offline, realtime events continue to fire, giving the end user a responsive experience. When the device regains connection, the Realtime Database synchronizes the local data changes with the remote updates that occurred while the client was offline, merging any conflicts automatically. The Realtime Database provides a flexible, expression-based rules language, called Firebase Realtime Database Security Rules, to define how your data should be structured and when data can be read from or written to. When integrated with Firebase Authentication, developers can define who has access to what data, and how they can access it.

The Realtime Database is a NoSQL database and as such has different optimizations and functionality compared to a relational database. The Realtime Database API is designed to only allow operations that can be executed quickly. This enables you to build a great realtime experience that can serve millions of users without compromising on responsiveness. Because of this, it is important to think about how users need to access your data and then structure it accordingly.

3.10 Key capabilities

Realtime	Instead of typical HTTP requests, the Firebase Realtime Database uses data synchronization—every time data changes, any connected device receives that update within milliseconds. Provide collaborative and immersive experiences without thinking about networking code.
Offline	Firebase apps remain responsive even when offline because the Firebase Realtime Database SDK persists your data to disk. Once connectivity is reestablished, the client device receives any changes it missed, synchronizing it with the current server state.
Accessible from Client Devices	The Firebase Realtime Database can be accessed directly from a mobile device or web browser; there's no need for an application server. Security and data validation are available through the Firebase Realtime Database Security Rules, expression-based rules that are executed when data is read or written.
Scale across multiple databases	With Firebase Realtime Database on the Blaze pricing plan, you can support your app's data needs at scale by splitting your data across multiple database instances in the same Firebase project. Streamline authentication with Firebase Authentication on your project and authenticate users across your database instances. Control access to the data in each database with custom Firebase Realtime Database Rules for each database instance.

3.11 Implementation Path

Integrate the Firebase Realtime Database SDKs	Quickly include clients via Gradle, CocoaPods, or a script include.

Create Realtime Database References	Reference your JSON data, such as "users/user:1234/phone_number" to set data or subscribe to data changes.
Set Data and Listen for Changes	Use these references to write data or subscribe to changes.
Enable Offline Persistence	Allow data to be written to the device's local disk so it can be available while offline.
Secure your data	Use Firebase Realtime Database Security Rules to secure your data.

3.12 Installation & Setup in JavaScript

The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in realtime to every connected client. When you build cross-platform apps with our Android, iOS, and JavaScript SDKs, all of your clients share one Realtime Database instance and automatically receive updates with the newest data.

Prerequisites

- Add and configure the Firebase JavaScript client SDK into your app.
- Configure Firebase Database Rules

  The Realtime Database provides a declarative rules language that allows you to define how your data should be structured, how it should be indexed, and when your data can be read from and written to. By default, read and write access to your database is restricted so only authenticated users can read or write data. To get started without setting up Authentication, you can configure your rules for public access. This does make your database open to anyone, even people not using your app, so be sure to restrict your database again when you set up authentication.
- Initialize the Realtime Database JavaScript SDK
- You must specify your Realtime Database URL when initializing your JavaScript SDK.
- You can find your Realtime Database URL in the Database tab in the Firebase console. It will be in the form of https://<databaseName>.firebaseio.com.
- Initialize your SDK using the following code snippet:

  ```
  // Set the configuration for your app
    // TODO: Replace with your project's config object
    var config = {
      apiKey: "apiKey",
  ```

```
  authDomain: "projectId.firebaseapp.com",
  databaseURL: "https://databaseName.firebaseio.com",
  storageBucket: "bucket.appspot.com"
};
firebase.initializeApp(config);

// Get a reference to the database service
var database = firebase.database();
```

You're ready to start using the Firebase Realtime Database!

3.13 Read and Write Data on the Web

Get a database reference

To read or write data from the database, you need an instance of firebase.database.Reference:

```
// Get a reference to the database service
var database = firebase.database();
```

3.13.1 Reading and writing data

This document covers the basics of retrieving data and how to order and filter Firebase data. Firebase data is retrieved by attaching an asynchronous listener to a firebase.database.Reference. The listener is triggered once for the initial state of the data and again anytime the data changes.

Note: By default, read and write access to your database is restricted so only authenticated users can read or write data. To get started without setting up Authentication, you can configure your rules for public access. This does make your database open to anyone, even people not using your app, so be sure to restrict your database again when you set up authentication.

3.13.2 Basic write operations

For basic write operations, you can use set() to save data to a specified reference, replacing any existing data at that path. For example a social blogging application might add a user with set() as follows:

```
function writeUserData(userId, name, email, imageUrl) {
  firebase.database().ref('users/' + userId).set({
    username: name,
    email: email,
    profile_picture : imageUrl
  });
}
```

Using set() overwrites data at the specified location, including any child nodes.

3.13.3 Listen for value events

To read data at a path and listen for changes, use the on() or once() methods of firebase.database.Reference to observe events. You can use the value event to read a static snapshot of the contents at a given path, as they existed at the time of the event. This method is triggered once when the listener is attached and again every time the data, including children, changes. The event callback is passed a snapshot containing all data at that location, including child data. If there is no data, the snapshot will return false when you call exists() and null when you call val() on it. The value event is called every time data is changed at the specified database reference, including changes to children. To limit the size of your snapshots, attach only at the lowest level needed for watching changes. For example, attaching a listener to the root of your database is not recommended.

The following example demonstrates a social blogging application retrieving the star count of a post from the database:

```
var starCountRef = firebase.database().ref('posts/' + postId + '/starCount');
starCountRef.on('value', function(snapshot) {
  updateStarCount(postElement, snapshot.val());
});
```

The listener receives a snapshot that contains the data at the specified location in the database at the time of the event. You can retrieve the data in the snapshot with the val() method.

3.13.4 Read data once

In some cases you may want a snapshot of your data without listening for changes, such as when initializing a UI element that you don't expect to change. You can use the once() method to simplify this scenario: it triggers once and then does not trigger again. This is useful for data that only needs to be loaded once and isn't expected to change frequently or require active listening. For instance, the blogging app in the previous examples uses this method to load a user's profile when they begin authoring a new post:

```
var userId = firebase.auth().currentUser.uid;
return firebase.database().ref('/users/' + userId).once('value').then(function(snapshot) {
  var username = (snapshot.val() && snapshot.val().username) || 'Anonymous';
  // ...
});
```

CHAPTER 4: IMPLEMENTATION AND WORKING

This section deals with the detailed hardware and software description and implementation of the various sensors and components.

4.1 Hardware and Software Implementation

This section provides description of the hardware and software used for implementation of the Smart Refrigeration module.The hardware used for Smart Refrigeration module are: Arduino uno board, Ultrasonic sensor, Push Button, Servo Motor. Figure shows the block diagram of Smart Refrigerator.

4.1.1 Development Tools

Programming for Arduino may be written in any programming language with a compiler that produces binary machine code like C and C++ programming. AVR Studio and the newer Atmel Studio, which can be used for programming Arduino microcontroller. Atmel provides a development environment for their microcontrollers.

The Arduino project provides the Arduino integrated development environment (IDE), which is a cross-platform application written in the programming code language Java. It originated from the IDE for the languages Processing and Wiring. It was created for people with no profound knowledge of electronics. It provides features such as syntax highlighting, cutting-pasting, and searching-replacing text, and automatic indenting, and provides simple one-click mechanism a code editor with 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 series of menus.

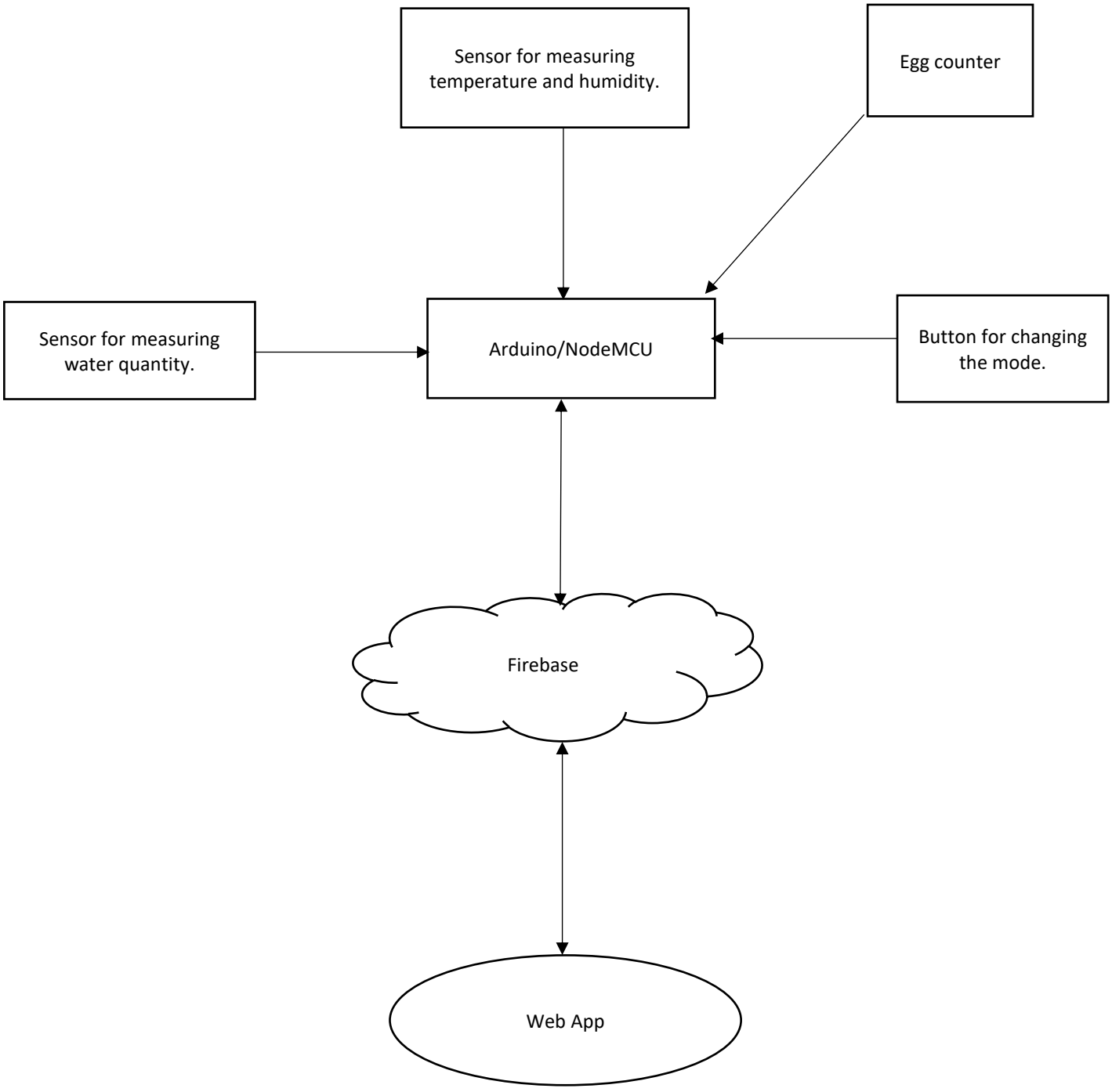
A program written with the IDE for Arduino is called as a "sketch." Sketches are saved on the development computer as the file extension with'. no.' Arduino Software (IDE) pre-1.0 saved sketches with the extension .pde.

The Arduino IDE supports the programming languages C and C++ using special rules to organize code. The Arduino IDE also supplies a software library from the Wiring project that gives many common input and output procedures. The user-written code only requires two functions, for starting the sketch and the main programs loop, that is compiled and then linked with a program stub *main()* into an executable cyclic executive program with the GNU tool chain, also included with the IDE distribution. The Arduino IDE employs the program *air dude*

to convert from executable code into a text file in hexadecimal coding that is loaded into the Arduino board by a loader program in the board's firmware.

This section explains the working and functionality of various components used in Smart Refrigerator Module.

4.1.2 Block Diagram



4.2 Working

1. User is asked to login on the web page so that he can get the details of his refrigerator.

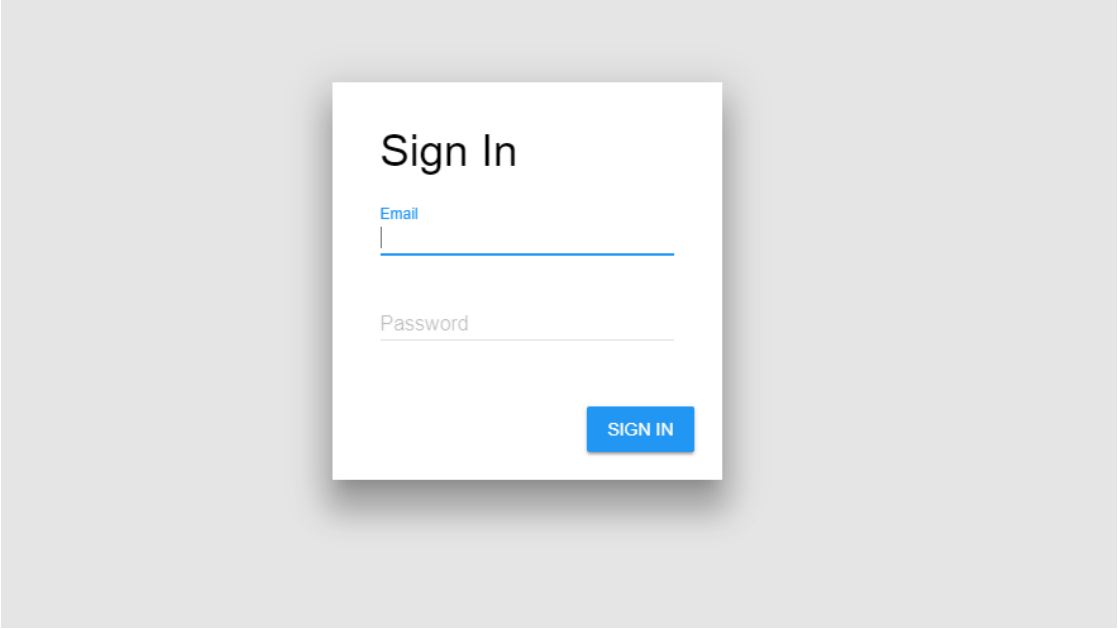


FIG 4.1 Login Page

2. When user will login he can see various components on dashboard such as number of eggs in the refrigerator, amount of water level, current temperature of the refrigerator, humidity inside the refrigerator, mode on which fridge is working, and the door timer.

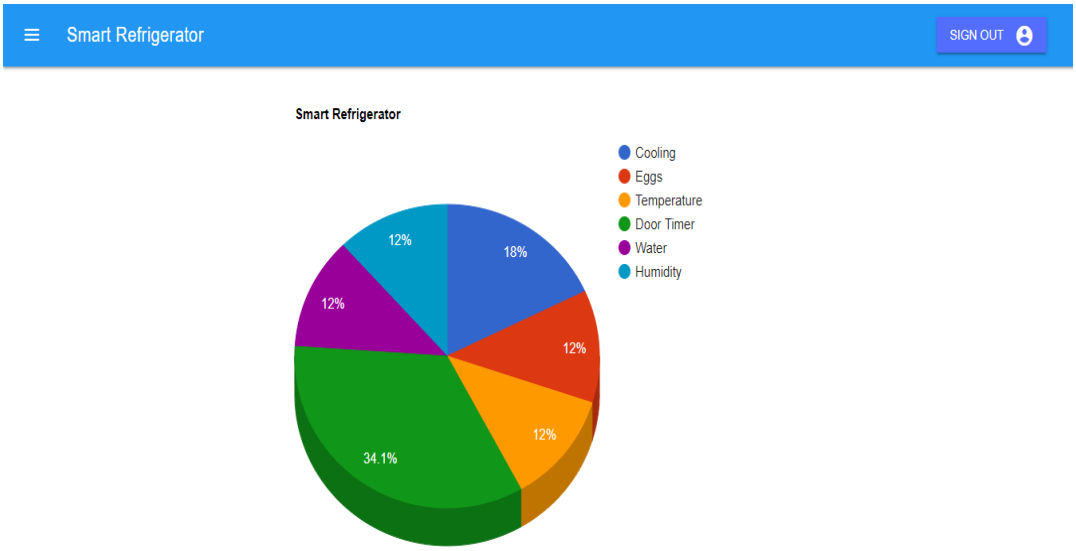


FIG 4.2 Refrigerator Data

3. The moment user opens the door, door timer will start which the user can see on the web page also. If user forget to close the door the user be notified after interval of time. The notification will be displayed both onweb page and mobile.

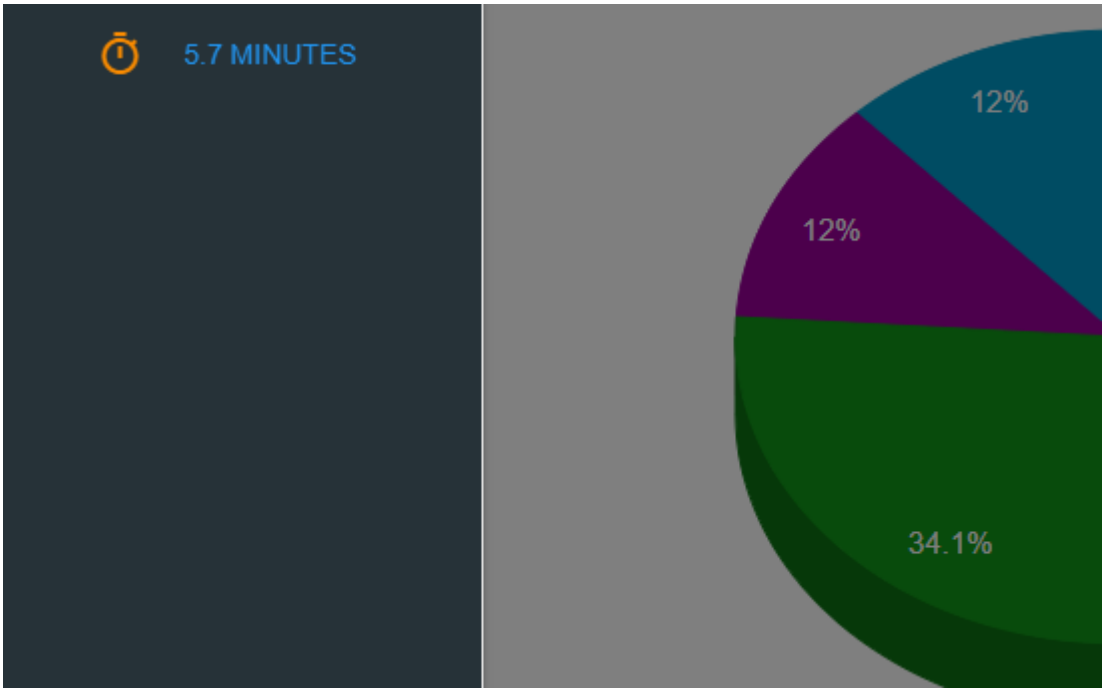


FIG 4.3 Door Timer

4. If user does not close the door manually, user can close the door through mobile also. The door will be closed by the servo motor attached to it.

FIG 4.4 Door Closing by Servo

5. Closing of the door through app gives the functionality to close it from anywhere. If the user is not around he can still close the door.

6. The user can set the mode of the fridge. There are three modes in the fridge:

- Low: In this mode the fridge will work at extreme low temperature. This mode is used when user wants extra cooling. This mode can be activated by clicking on the low button on the web page. A pop up will also display the mode on web page.
- Medium: In this mode the fridge will work at normal temperature. This mode is standard mode. Refrigerator will usually be in medium mode. This mode can also be activated by clicking on the medium button on the web page. A pop up will also display the mode on web page.
- High: In this mode the fridge will work high temperature. This mode is used when user does not want much cooling. This mode s generally used when user is not at home and there is nothing much in the fridge and user wants to save electricity .This mode can be activated by clicking on the high button on the web page. A pop up will also display the mode on web page.

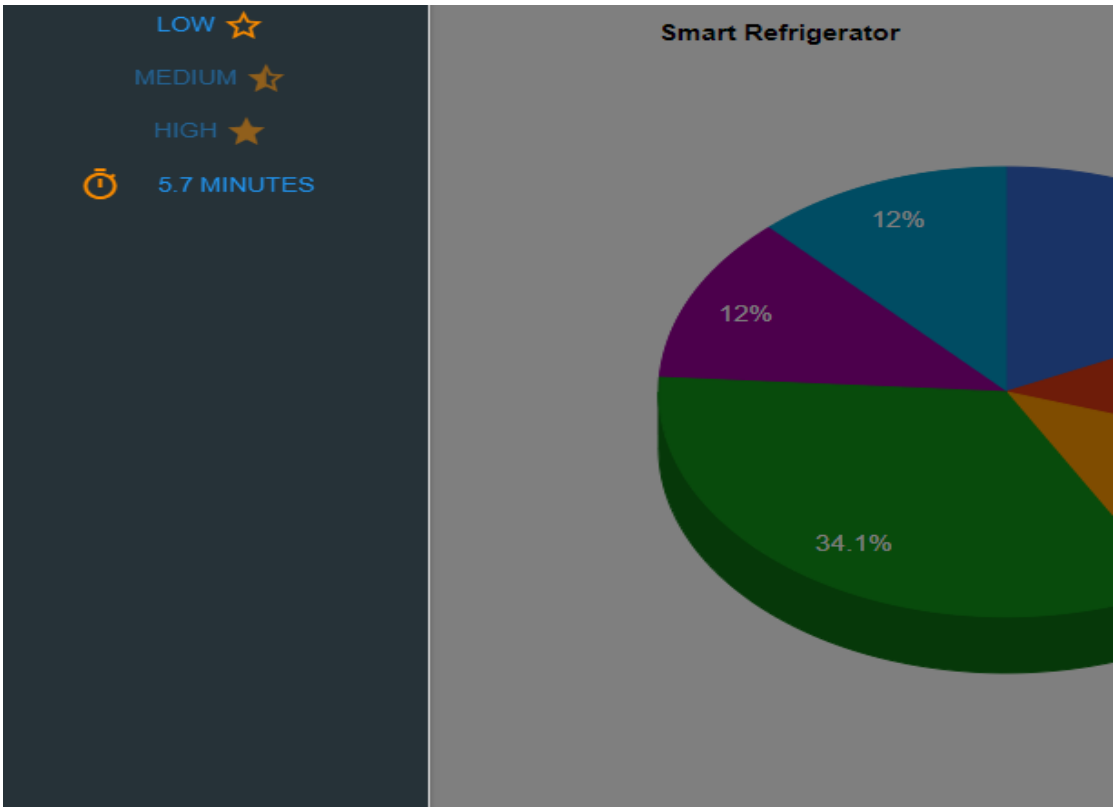


FIG 4.5 Modes of Refrigerator

7. There are three LED installed on the door of the fridge which indicates the current mode on which refrigerator is working. If Low led is on than medium and high led will be off and if medium led is on then low and high led will be off. By this user will get to know which mode is activated.

FIG 4.6 LEDs

8. There is push button installed on the door. This button is used to change the mode of the refrigerator.

If user press it once then refrigerator will switch to LOW mode. If user press it twice than refrigerator will switch to MEDIUM mode. If user press it thrice the refrigerator will switch to HIGH mode. Accordingly led will change placed on the door.

FIG 4.7 Push Button

9. User can change the mode through button or through web page. Both are connected to firebase and are in sync with each other.

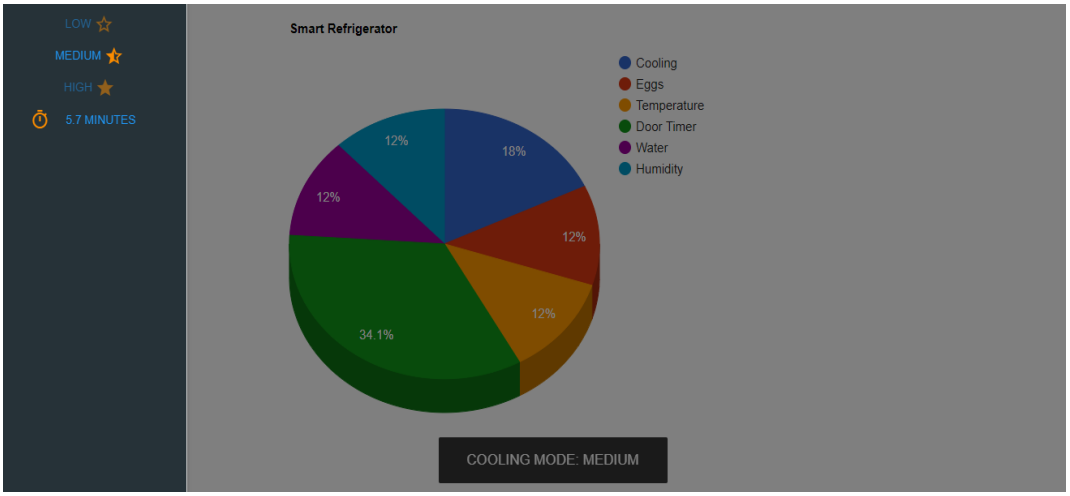


FIG 4.8 Pi Chart

10. There is a jar placed inside the fridge which is used to hold the water. An ultrasonic sensor is installed on it which monitors the water level. If the water level gets low below certain level user will be notified to fill the jar. It also send the distance value on the firebase server.

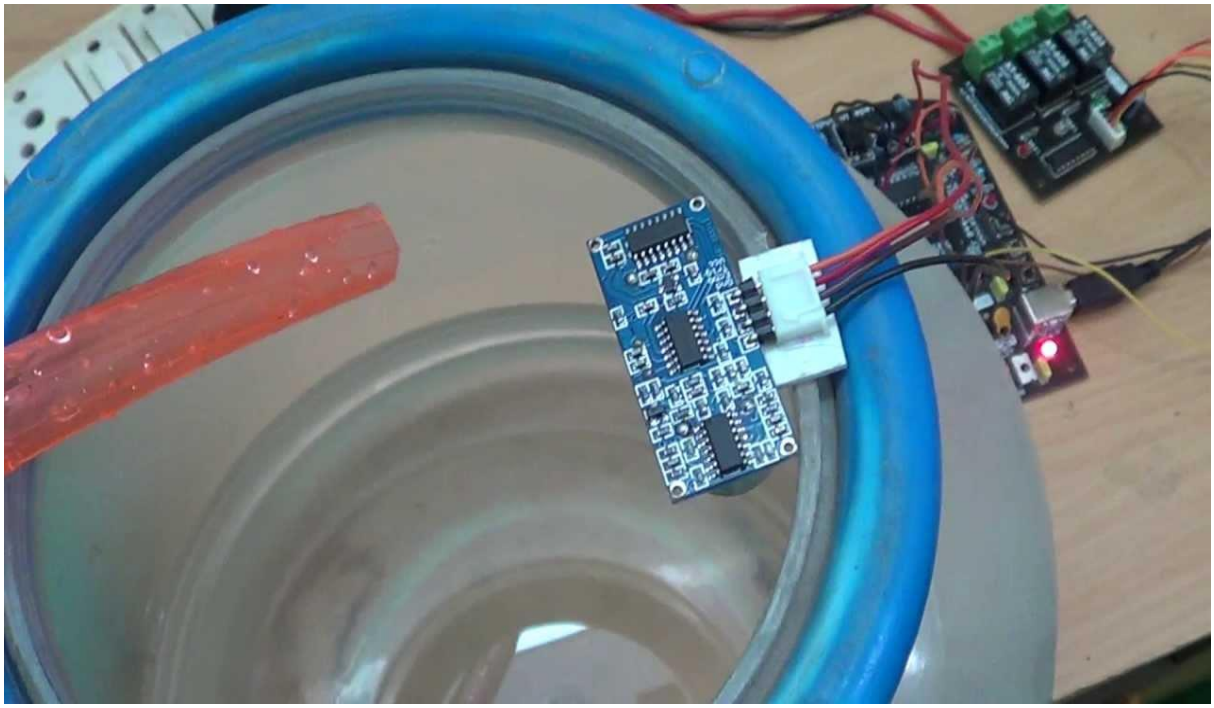


FIG 4.9 Water level detection by Ultrasonic sensor

11. There is a DHT sensor installed in the refrigerator which monitors the temperature and humidity inside the refrigerator. On the dashboard of web page user can see the temperature and humidity value of real time.

FIG 4.10 DHT11

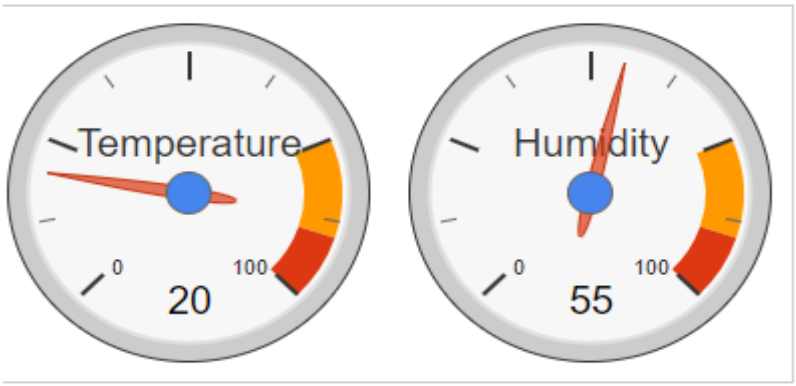


FIG 4.11 Temperature and humidity meter

12. There is egg tray in the fridge, under the egg tray push button is installed. This is used to count the number of eggs present in the fridge. User can see amount of eggs present in the fridge anytime. If count of eggs is less than 2 than user will be notifies to bring more eggs. The moment user picks or place the egg on tray the value will be changed in pie chart of dashboard .

FIG 4.12 Eggs Tray

4.3 Coding

4.3.1 Index.html

```
<!Doctype html>
<html>
<head>
   <title>Smart Fridge</title>

   <link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
   <link rel="stylesheet" href="https://code.getmdl.io/1.3.0/material.blue-indigo.min.css" />
   <script defer src="https://code.getmdl.io/1.3.0/material.min.js"></script>
   <link rel="stylesheet" href="style.css" />

<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
   </head>
<body>

   <dialog id="loginDialog" class="mdl-dialog">
      <h4 class="mdl-dialog__title">Sign In</h4>
      <div class="mdl-dialog__content">
         <div id=" loginError" class="mdl-textfield mdl-js-textfield mdl-textfield--floating-
label">
            <input class="mdl-textfield__input" type="text" id="loginEmail">
            <label class="mdl-textfield__label" for="sample3">Email</label>
          </div>
          <div class="mdl-textfield mdl-js-textfield mdl-textfield--floating-label">
            <input class="mdl-textfield__input" type="text" id="loginPassword">
            <label class="mdl-textfield__label" for="sample3">Password</label>
          </div>
      </div>
      <div class="mdl-dialog__actions">
        <!-- Colored raised button -->
<button id="loginBtn" class="mdl-button mdl-js-button mdl-button--raised mdl-js-ripple-
effect mdl-button--colored">
   Sign In
 </button>
 <div id="loginProgress" class="page-loader mdl-spinner mdl-js-spinner is-active"></div>
      </div>
    </dialog>

   <div class="login-cover">
     <div class="page-loader mdl-spinner mdl-js-spinner is-active"></div>
      </div>

      <div class="mdl-layout mdl-js-layout">
            <header class="mdl-layout__header">
              <div class="mdl-layout-icon"></div>
              <div class="mdl-layout__header-row">
                <span class="mdl-layout__title">Smart Refrigerator</span>
```

```
            <div class="mdl-layout-spacer"></div>
            <nav class="mdl-navigation">
            <button id="signOutBtn" class="mdl-button mdl-js-button mdl-button--raised
mdl-button--accent">
          Sign Out <i class="button-right-icon material-icons">
              account_circle</i>
          </button>

            </nav>
           </div>
          </header>
          <div class="mdl-layout__drawer mdl-color--blue-grey-900">
            <nav class="mdl-navigation">
              <label class="mdl-button mdl-js-button mdl-button--primary"> <i
class="material-icons orange600">
                  whatshot
                  </i>
                  <label id="tmp" class="mdl-button mdl-js-button mdl-button--primary"
>Temperature</label>
                </label>
                <label class="mdl-button mdl-js-button mdl-button--primary"> <i
class="material-icons orange600">
                    ac_unit
                    </i>
                    <label id="hmd" class="mdl-button mdl-js-button mdl-button--
primary" >HUMIDITY</label>
                  </label>
              <button id="lo" onclick="low()" class="mdl-button mdl-js-button mdl-button--
primary">
                Low
                 <i class="material-icons orange600">
                 star_border
                 </i>
                </button>
              <button id="me" onclick="mid()" class="mdl-button mdl-js-button mdl-button--
primary">
                Medium
                 <i class="material-icons orange600">
                 star_half
                 </i>
                </button>
              <button id="hi" onclick="high()" class="mdl-button mdl-js-button mdl-button--
primary">
                High
                 <i class="material-icons orange600">
                 star
                 </i>
                </button>
              <label class="mdl-button mdl-js-button mdl-button--primary"> <i
class="material-icons orange600">
```

```
                    timer
                    </i>
                    <label id="time" class="mdl-button mdl-js-button mdl-button--primary"
>Timer</label>
                  </label>
                </nav>
              </div>
            </div>

        <script src="https://www.gstatic.com/firebasejs/5.5.2/firebase.js"></script>
        <script>
          // Initialize Firebase
          var config = {
            apiKey: "AIzaSyColVVJ8iwGb0Ueoo29R8wD2loAg5sCLMI",
            authDomain: "harshit-batra.firebaseapp.com",
            databaseURL: "https://harshit-batra.firebaseio.com",
            projectId: "harshit-batra",
            storageBucket: "harshit-batra.appspot.com",
            messagingSenderId: "446187830986"
          };
          firebase.initializeApp(config);
        </script>
    <script src="https://code.jquery.com/jquery-3.1.0.js"></script>
    <script src="index1.js"></script>
    <div id="piechart_3d" style="width: 1000px; height:750px; margin: 0 auto;"></div>
    <div id="snackbar">Some text some message..</div>
</body>
</html>
```

4.3..2 Index.js

```
firebase.auth().onAuthStateChanged(function(user){
if(user){

    $(".login-cover").hide();

    var dialog = document.querySelector('#loginDialog');
    if (! dialog.showModal) {
        dialogPolyfill.registerDialog(dialog);
      }
      dialog.close();
}
else
{
    $(".login-cover").show();
    var dialog = document.querySelector('#loginDialog');
    if (! dialog.showModal) {
        dialogPolyfill.registerDialog(dialog);
      }
      dialog.showModal();
```

```
}
});

$("#loginBtn").click(
function()
{
   var email = $("#loginEmail").val();
   var password = $("#loginPassword").val();

   if(email != "" && password != "")
   {
      $("#loginProgress").show();
      $("#loginBtn").hide();
      firebase.auth().signInWithEmailAndPassword(email,password).catch(function (error){
         $("#loginError").show().text(error.message);

         $("#loginProgress").hide();
         $("#loginBtn").show();
      });
   }
}
);

$("#signOutBtn").click(
function()
{
   firebase.auth().signOut().then(function() {
      // Sign-out successful.
    }).catch(function(error) {
      alert(error.message);
    });
}
);


// Buttons
var fire1 = firebase.database().ref().child("Low");
var fire2 = firebase.database().ref().child("Mid");
var fire3 = firebase.database().ref().child("High");
var col = document.getElementById("lo");
var col1 = document.getElementById("me");
var col2 = document.getElementById("hi");
function low()
{
   var h;
   fire1.on('value',function(datasnapshot){
         h = datasnapshot.val();
      });
      if(h==0)
       {
```

```
            col.style.opacity="1";
            col1.style.opacity="0.5";
            col2.style.opacity="0.5";
            fire1.set(1);
            fire2.set(0);
            fire3.set(0);
          }

}

function mid()
{
   var h;
   fire2.on('value',function(datasnapshot){
            h = datasnapshot.val();
        });
        if(h==0)
         {
            col.style.opacity="0.5";
            col1.style.opacity="1";
            col2.style.opacity="0.5";
            fire1.set(0);
            fire2.set(1);
            fire3.set(0);
         }
 }

function high()
{
   var h;
   fire3.on('value',function(datasnapshot){
            h = datasnapshot.val();
        });
        if(h==0)
         {
            col.style.opacity="0.5";
            col1.style.opacity="0.5";
            col2.style.opacity="1";
            fire1.set(0);
            fire2.set(0);
            fire3.set(1);
         }
   }


   fire2.on('value',function(datasnapshot){
        if(datasnapshot.val()==1)
        {
            var x = document.getElementById("snackbar");
            x.innerText="COOLING MODE: MEDIUM";
```

```
            x.className = "show";
            setTimeout(function(){ x.className = x.className.replace("show", ""); }, 3000);
            col1.style.opacity="1";
        }
        else
        {
            col1.style.opacity="0.5";
        }
    });


    fire1.on('value',function(datasnapshot){
     if(datasnapshot.val()==1)
     {
        var x = document.getElementById("snackbar");
        x.innerText="COOLING MODE: LOW";
        x.className = "show";
        setTimeout(function(){ x.className = x.className.replace("show", ""); }, 3000);
        col.style.opacity="1";
     }
     else
     {
        col.style.opacity="0.5";
     }
});


fire3.on('value',function(datasnapshot){
    if(datasnapshot.val()==1)
    {
        var x = document.getElementById("snackbar");
        x.innerText="COOLING MODE: HIGH";
        x.className = "show";
        setTimeout(function(){ x.className = x.className.replace("show", ""); }, 3000);
        col2.style.opacity="1";
    }
    else
    {
        col2.style.opacity="0.5";
    }
});



//Timer


var hed =document.getElementById("time");
```

```
var fi = firebase.database().ref().child("Door");

fi.on('value',function(datasnapshot){
   hed.innerText=(datasnapshot.val()/60).toFixed(1)+" "+"minutes";
   if(datasnapshot.val()%60==0)
   {
      var x = document.getElementById("snackbar");
      x.innerText="Hey,Door is opened for so long.Please close it !";
      x.className = "show";
      setTimeout(function(){ x.className = x.className.replace("show", ""); }, 3000);
   }
});


//Pie chart
var t = firebase.database().ref().child("Door");
var x;
t.on('value',function(datasnapshot){
   x=(datasnapshot.val()/60);
     google.charts.load("current", {packages:["corechart"]});
     google.charts.setOnLoadCallback(drawChart);
     function drawChart() {
       var data = google.visualization.arrayToDataTable([
         ['Task', 'Hours per Day'],
         ['Cooling',   3],
         ['Eggs',      2],
         ['Temperature',  2],
         ['Door Timer', x],
         ['Water',    2],
         ['Humidity', 2]
       ]);

       var options = {
         title: 'Smart Refrigerator',
         is3D: true,
       };

       var chart = new
google.visualization.PieChart(document.getElementById('piechart_3d'));
       chart.draw(data, options);
     }
   });
```

4.3.3 Style.css

```
.button-right-icon{

   margin-left: 10px;
}

.login-cover{
   background:#fff;
   position: fixed;
   top: 0;
   left: 0;
   width: 100%;
   height: 100vh;
   z-index: 800;
   display: flex;
   justify-content: center;
   align-items: center;
}

#loginProgress{
   display: none;
}

#loginError{
   display: none;
}


.material-icons.orange600 { color: #FB8C00; }

#snackbar {
   visibility: hidden;
   min-width: 250px;
   margin-left: -125px;
   background-color: #333;
   color: #fff;
   text-align: center;
   border-radius: 2px;
   padding: 16px;
   position: fixed;
   z-index: 1;
   left: 50%;
   bottom: 30px;
   font-size: 17px;
}

#snackbar.show {
   visibility: visible;
   -webkit-animation: fadein 0.5s, fadeout 0.5s 2.5s;
```

```
    animation: fadein 0.5s, fadeout 0.5s 2.5s;
}

@-webkit-keyframes fadein {
    from {bottom: 0; opacity: 0;}
    to {bottom: 30px; opacity: 1;}
}

@keyframes fadein {
    from {bottom: 0; opacity: 0;}
    to {bottom: 30px; opacity: 1;}
}

@-webkit-keyframes fadeout {
    from {bottom: 30px; opacity: 1;}
    to {bottom: 0; opacity: 0;}
}

@keyframes fadeout {
    from {bottom: 30px; opacity: 1;}
    to {bottom: 0; opacity: 0;}
}
```

4.3.4 Node MCU Program

```
#include <ESP8266WiFi.h>
#include <FirebaseArduino.h>
int but=D0;
int val=0;
int c=0;
// Set these to run example.
#define FIREBASE_HOST "harshit-batra.firebaseio.com"
#define FIREBASE_AUTH "3ojq4qJYwBa5HnUXk11bNtwKnBQXOHtX2YRfvtvk"
#define WIFI_SSID "Harshit"
#define WIFI_PASSWORD "Batra1998"
void setup() {
  Serial.begin(9600);
  pinMode(but,INPUT);
  // connect to wifi.
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("connecting");
```

```
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println();
  Serial.print("connected: ");
  Serial.println(WiFi.localIP());
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);// put your setup code here, to run
once:

}

void check()
{
FirebaseObject object=Firebase.get("/");
int h = object.getInt("High");
int m = object.getInt("Mid");
int l = object.getInt("Low");
//Serial.println(h);
//Serial.println(m);
//Serial.println(l);
if(h==1)
  {
    c=3;
  }
  else if(m==1)
  {
    c=2;
  }
  else if(l==1)
  {
    c=1;
  }
}
```

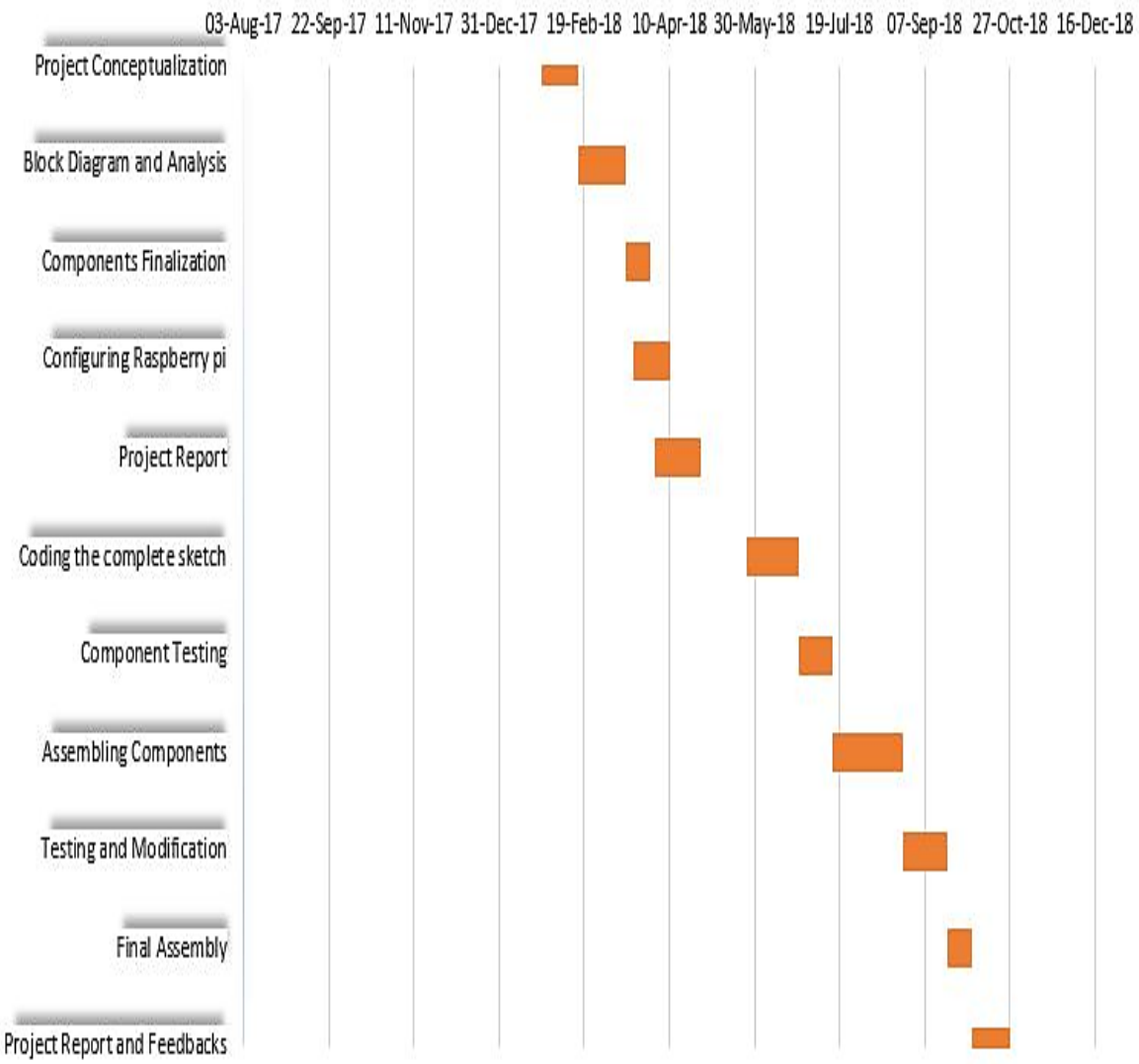
```
void lo()
{
  Firebase.setInt("Low",1);
  Firebase.setInt("Mid",0);
  Firebase.setInt("High",0);
}

void me()
{
  Firebase.setInt("Low",0);
  Firebase.setInt("Mid",1);
  Firebase.setInt("High",0);
}


void hi()
{
  Firebase.setInt("Low",0);
  Firebase.setInt("Mid",0);
  Firebase.setInt("High",1);
}
int a=0;
void loop() {
  a=a+1;
val =digitalRead(but);
if(val==0 && c<=3)
{
  check();
  if(c==3)
  {
    c=0;
  }
  c++;
  if(c==1)
```

```
  {
    lo();
  }
  else if(c==2)
  {
    me();
  }
  else
  {
    hi();
  }
}
Firebase.setInt("Door",a);
delay(500);
}
```

4.3.5 GANTT CHART



4.3.6 FUTURE SCOPE

The concept of smart refrigerator is far more reaching than notifying the user about the contents of the refrigerator. It should give importance on maintaining a healthier lifestyle by providing the nutritional value of the contents. The future smart refrigerator can use MQ4 sensor for detecting methane gas released from the stuff inside the refrigerator. We can also implement camera for checking the refrigerator live from anywhere in the world. Camera will help to visualize the items inside the refrigerator and to monitor the replacement of food items kept inside it. The smart refrigerator will also perform other functions such as dietary control, eating routine analysis etc. Other thing we can add is, smart refrigerators provide both cold and hot water. You select a temperature and amount of water you want heated and your smart refrigerator sends a notification to your smart phone when your heated water is ready. A few even come with a Keurig single-cup coffee maker built in, saving counter space and making your morning routine just a bit simpler.

We can also implement:

- Alert you when the water filter needs to be changed
- Turn the ice maker on or off from your smartphone
- Set expiration dates and receive notifications to use food while its fresh
- Upload photos for display
- Create individual profiles for each family member to send them personal notes and to-do lists
- Use a whiteboard option to leave messages for your family
- Transparent touchscreens allow you to look inside the fridge without opening the door

4.3.7 PROS OF SMART FRIDGE

Refrigerators are becoming smarter every year with the latest models even assisting you with grocery shopping and staying in touch with friends. The Samsung Family Hub is the latest one that does all of this and more, and other brands are trying to catch up. These smart fridges come with touchscreens that are more like tablets. In fact, they allow you to do pretty much everything you can do with a tablet and more, like controlling your refrigerator's temperature

and humidity settings. On the display you can showcase your family calendar, type in notes and reminders, and display your favourite pictures. Some brands also include speakers and cameras, allowing the refrigerator to play your favourite music and take pictures every time the door closes. These pictures allow you to see on your phone what you may have forgotten on your shopping list while at the store, or to let your spouse know just where the ketchup is without having to get up and find it for them. With some, you can even order groceries from a local store. A smart refrigerator can make grocery shopping easier. A camera allows you to peek inside the fridge even if you aren't home, and you can keep a running list of what you need. You can also control the temperature and humidity to keep your food fresh as long as possible.

4.3.8 CONS OF SMART FRIDGE

The first con is that these smart refrigerators are considerably expensive than traditional ones—and so are the repairs. With Wi-Fi capability, your smart refrigerator can be hacked or become infected with a virus—meaning that anything you gave it access to, such as your email or social media, could be at risk. The other issue with these extra features is that after only a year or two many brands discontinue any security updates as they develop the newest refrigerator and turn their focus to it. This not only makes it more vulnerable to an attack, but it can also eliminate the access to any services and features that were being offered at the time you purchased it.

REFERENCES

1. https://en.wikipedia.org/wiki/Internet_refrigerator

2. https://www.arduino.cc/en/Guide/Introduction

3. https://firebase.google.com/

4. https://www.elprocus.com/ultrasonic-detection-basics-application/

5. https://www.instructables.com/id/Quick-Start-to-Nodemcu-ESP8266-on-Arduino-IDE/

6. https://www.instructables.com/

7. https://create.arduino.cc/projecthub

8. https://www.arduino.cc/en/Main/ArduinoGSMShield