**INDIVIDUALIZE HEALTH CAUTIOUS AND COMMUNICATION**

|  |  |  |  |
| --- | --- | --- | --- |
| Dr.Satyasrikanth P  Associate Professor  Department of ECE  Atria Institute of Technology  Bangalore  satyarikanth123@yahoo.in | Nataraja N  Assistant Professor  Department of ECE  Atria Institute of Technology  Bangalore  nataraja.sp85@gmail.com | Dr. Keshavamurthy  Professor  Department of ECE  Atria Institute of Technology  Bangalore  keshavamurthy\_s@yahoo.com | Anne Gowda A B  Assistant Professor  Department of ECE  Sri Venkateshwara College of Engineering  Bangalore  annegowda.ece@gmail.com |

ABSTRACT

Health monitoring and security for humans in healthcare is an essential element that encompasses the continuous tracking and evaluation of diverse health indicators to identify and mitigate potential health issues. The field has witnessed significant progress with advancements in technology and medical research, leading to the development of advanced monitoring systems that enable real-time data collection and enhance overall health outcomes. The integration of wearables, sensors, and mobile health applications has empowered individuals to monitor their crucial health metrics, such as vital signs, physical activity levels, and sleep patterns, among others. Moreover, these devices possess the capability to notify healthcare providers about potential health concerns, facilitating prompt intervention and treatment.

**Keywords:** Healthcare, Wearable, sensors, Monitoring, Mobile Health

# INTRODUCTION

In recent years, the rapid growth of the internet has led to the emergence of the Internet of Things (IoT), where various devices are connected and communicate with each other. This interconnected network has paved the way for the development of smart technologies and generated a vast amount of data, ushering in a new era of information utilization. The applications of IoT span across different domains, including transport and logistics, healthcare, smart environment, and personal and social aspects. Governments worldwide are focusing on creating smart cities that leverage these emerging technologies to enhance competitiveness on the international stage. With the widespread use of smart devices, individuals are surrounded by technology that connects them to networks, social platforms, and other intelligent systems. The impact of IoT on daily life is substantial, influencing areas such as entertainment, work, and communication.

Individualized health caution and combination refer to the personalized approach of healthcare, where medical treatments and interventions are tailored to the specific needs of each individual. It involves considering various factors such as genetics, lifestyle, environment, and medical history to develop a comprehensive and customized healthcare plan.

In traditional healthcare practices, treatments are often generalized and applied uniformly to a broad population. However, individual differences play a significant role in how individuals respond to different treatments and interventions. This realization has led to a shift towards individualized healthcare, which aims to provide more effective and targeted care for each person.

The concept of individualized health caution encompasses several aspects. Firstly, it recognizes that each individual is unique, with distinct genetic makeup, physiological characteristics, and lifestyle factors. This understanding allows healthcare providers to develop personalized prevention strategies, diagnostic approaches, and treatment plans. By considering an individual's specific risk factors, vulnerabilities, and preferences, healthcare professionals can optimize outcomes and minimize potential side effects.

Combination therapy is an integral part of individualized health caution. It involves the use of multiple treatment modalities or interventions to address the complex nature of certain medical conditions. Rather than relying on a single treatment approach, combination therapy aims to target different aspects of a disease or condition simultaneously, improving efficacy and reducing the likelihood of treatment resistance.

The advancements in technology and the availability of vast amounts of health data have greatly facilitated the implementation of individualized health caution and combination therapy. Genetic testing, biomarker analysis, wearable devices, and electronic health records enable healthcare providers to gather comprehensive information about an individual's health status. This data, combined with sophisticated algorithms and artificial intelligence, can assist in the identification of personalized treatment options and predict optimal therapeutic approaches.

The benefits of individualized health caution and combination therapy are substantial. It allows for earlier detection and prevention of diseases, more precise diagnoses, and the development of targeted treatment plans. By tailoring healthcare interventions to each individual, healthcare providers can achieve better patient outcomes, minimize adverse effects, and improve overall healthcare efficiency. In individualized health caution and combination therapy represent a paradigm shift in healthcare towards personalized and targeted approaches. By considering individual variations and combining multiple treatment modalities, healthcare providers can optimize patient care, enhance treatment efficacy, and improve the overall quality of healthcare delivery.

Among the various domains benefiting from IoT, the field of medical and healthcare stands out as a key enabler. IoT devices play a crucial role in collecting, monitoring, evaluating, and notifying patients about their health information. The penetration of IoT devices in medical and healthcare encompasses health monitoring of medical parameters, diagnostics, medical equipment tracking, secure indoor environment access, smart hospital services, and even entertainment services. However, remote monitoring of patients by doctors remains a challenging task. Analyzing a patient's health condition requires the collection of various medical parameters and transmitting them to the doctor through a reliable networking channel, which poses its own set of challenges.

To address these challenges, several technologies come into play. The Internet of Things (IoT) refers to a system where interconnected computing devices, machines, objects, animals, or people are assigned unique identifiers and can transfer data over a network without human intervention. Cloud computing is another essential technology that allows the delivery of hosted services over the internet. It enables companies to consume computing resources as utilities rather than building and maintaining their own infrastructure. ESP-32, which stands for "Espressif Systems Product 32," is a widely used microcontroller module in building IoT projects. Additionally, the Blynk app provides a platform with iOS and Android applications for controlling devices like ESP-32 and Raspberry Pi over the internet, allowing users to build a graphical interface for their projects easily.

The implementation of health monitoring systems using IoT technology has numerous potential scenarios. For instance, in the case of a patient being introduced to a new drug with an unstable regulatory body system, continuous monitoring and data analysis can help assess the patient's response. Patients prone to heart attacks or those who have suffered one before can benefit from continuous monitoring of vital signs to predict and alert in advance of any indication of a potential cardiac event. Monitoring critical organ situations and detecting conditions that may lead to life-threatening situations are also vital, especially for individuals with advanced age and failing health conditions. Athletes during training can utilize health monitoring systems to determine which training regimens produce the best results.

However, there are challenges to the widespread adoption of health monitoring systems. Some existing systems require wireless detection and transmission of sensor information to a health server, often with subscription fees that may hinder accessibility, particularly in developing countries. Internet connectivity is also an issue, as real-time health connections require reliable and high-quality internet access, which may be lacking in certain regions. Moreover, many of these systems were initially developed in developed countries with well-established infrastructure, making their adaptation to developing countries more complex.

To overcome these challenges, it is crucial to approach health detection in developing countries with a ground-up approach that takes into account the basic minimal conditions presently available. Simplifying the design of patient monitoring systems based on the number of parameters they can detect can be a viable solution. For instance, a single parameter monitoring system focuses on detecting and analyzing a specific parameter, such as an electrocardiogram (ECG) reading, which can provide multiple readings based on the algorithm used. On the other hand, a multi-parameter monitoring system simultaneously tracks multiple parameters, often found in high-dependency units (HDUs), intensive care units (ICUs), or during surgery and post-surgery recovery in hospitals. These systems monitor parameters like ECG, blood pressure, and respiration rate, serving as indicators of a patient's health or recovery. The health monitoring systems enabled by IoT technology have the potential to revolutionize healthcare by extending it beyond traditional clinical settings. By leveraging wireless sensors, data analysis algorithms, and connectivity, these systems can provide valuable physiological information in home environments. However, challenges related to cost, internet connectivity, and infrastructure must be addressed to ensure the widespread accessibility and effectiveness of these systems, particularly in developing countries

The primary goal of this project is to develop a technology-oriented, low-cost, and easily accessible real-time health monitoring device. The device will leverage advancements in technology to provide continuous monitoring of an individual's health status. The focus of this health monitoring system is to cater to patients with chronic diseases. Chronic diseases are typically long-term conditions that require continuous monitoring to manage symptoms and prevent complications. As most chronic diseases are incurable, it becomes crucial to monitor the patient's health status while they are at home, allowing for early detection of any concerning changes. By implementing a more competent patient management system through real-time health monitoring, hospitals can utilize their resources more wisely and save money. Continuous monitoring of patients' health parameters can help in early detection of deteriorating conditions, preventing unnecessary hospitalizations, and enabling timely interventions.

The proposed system aims to provide convenience for both doctors and patients. Doctors can manage their patients from a single application, which streamlines the process of monitoring multiple patients efficiently. Simultaneously, patients can easily monitor their own health by wearing a lightweight device such as a bracelet, which collects and transmits real-time health data to the monitoring system. Real-time health monitoring systems, utilizing the Internet of Things (IoT), have the potential to significantly impact patient prioritization and urgent care. By continuously collecting and analyzing health data, doctors can identify patients who require immediate attention and provide them with timely care, potentially saving lives. The system can alert healthcare providers when a patient's health parameters deviate from normal ranges, enabling rapid response and intervention. Overall, the proposed real-time health monitoring system aims to improve patient outcomes, enhance resource management in hospitals, and provide a more convenient and efficient healthcare experience for both doctors and patients.

# LITERATURE REVIEW

A literature review is a critical analysis of existing literature on a particular topic or research question. It involves searching for and evaluating relevant sources, summarizing and synthesizing the information gathered, and presenting a comprehensive overview of the current state of knowledge on the topic. Development and Clinical Evaluation of a Home Healthcare System Measuring in Toilet, Bathtub and Bed without Attachment of Any Biological Sensors Daily monitoring of health condition at home is important for an effective scheme for early diagnosis, treatment, and prevention of lifestyle-related diseases such as adiposis, diabetes and cardiovascular diseases. While many commercially available devices for home health care monitoring are widely used, those are cumbersome in terms of self-attachment of biological sensors and self-operation of them. From this viewpoint, we have been developing a non-conscious physiological monitoring system without attachment of any sensors to the human body as well as any operations for the measurement. We developed some devices installed in a toilet, a bath, and a bed and showed their high measurement precision by comparison with simultaneous recordings of ordinary biological sensors directly attached to the body. To investigate that applicability to the health condition monitoring, we developed a monitoring system in combination with all the monitoring devices at hospital rooms and previously carried out the measurements of patients' health condition. Further, in this study, the health conditions were measured in 10 patients with cardiovascular disease or sleep disorder. From these results, the patients’ health conditions such as the body and excretion weight in the toilet, the ECG during taking the bath and the pulse and respiration rate during sleeping were successfully monitored in the hospital room, demonstrating its usefulness for monitoring the health condition of the subjects with cardiovascular disease or sleep disorder. 8 2.2 Intelligent wireless mobile patient monitoring system Nowadays

You are absolutely right that health is a significant concern, especially in times of advancements in technology. The recent outbreak of the coronavirus (COVID-19) serves as a prime example of how crucial healthcare is, as it has had a significant impact on the global economy. In areas affected by epidemics, it is essential to employ remote health monitoring technology to monitor patients effectively. Currently, one of the leading solutions for this purpose is an Internet of Things (IoT)-based health monitoring system [1].

The IoT-based health monitoring system leverages the power of interconnected devices to remotely monitor the health status of patients. This system involves the use of wearable devices equipped with sensors that collect various health-related data, such as vital signs and symptoms. These devices, which may include smartwatches, fitness trackers, or specialized medical devices, continuously transmit the collected data to a centralized platform for analysis and monitoring.

By utilizing IoT technology, healthcare professionals can remotely monitor patients' health conditions in real-time, even from a distance. This allows for early detection of any health deterioration or warning signs, enabling timely intervention and treatment. Moreover, IoT-based health monitoring systems facilitate continuous monitoring, allowing healthcare providers to gather comprehensive data over extended periods, which can aid in accurate diagnoses and personalized treatment plans.

In the context of epidemics or contagious diseases, remote health monitoring becomes even more critical. It helps minimize direct physical contact between healthcare providers and patients, reducing the risk of disease transmission. Additionally, IoT-based systems can support contact tracing efforts by tracking and analyzing the movement and health data of individuals, aiding in the containment of the spread of infectious diseases.

Furthermore, IoT-based health monitoring systems improve healthcare efficiency and resource allocation. By remotely monitoring patients, healthcare professionals can prioritize resources and focus on those in immediate need of medical attention. This optimizes the utilization of healthcare facilities, reduces overcrowding, and ensures that resources are allocated appropriately. An IoT-based health monitoring system [1] is an effective solution for monitoring patients, especially in areas affected by epidemics or widespread diseases. It enables remote monitoring, early detection of health issues, efficient resource allocation, and contributes to the overall improvement of healthcare outcomes.

Health data wirelessly using various communication technologies such as Wi-Fi, Bluetooth, and GSM/GPRS. This system aims to overcome the limitations of uninterrupted communication in developing countries like Bangladesh. By utilizing IoT technology, the proposed system enables real-time monitoring of individuals' health conditions. It collects data from various sensors attached to the individuals, such as heart rate monitors, temperature sensors, and activity trackers. The collected data is then transmitted wirelessly to a central server for further analysis and storage. One of the key features of this system is its ability to function both online and offline. In areas with limited or no internet connectivity, the system can store the data locally and transmit it once a connection becomes available. This ensures that health data is not lost even in the absence of a continuous communication system. The proposed IoT-based health monitoring system in [2] has the potential to improve healthcare services in remote areas by providing real-time health information to healthcare professionals. They can remotely monitor patients' vital signs, detect abnormalities, and take timely actions based on the transmitted data. This system can also facilitate early diagnosis and intervention in critical situations, leading to improved healthcare outcomes. This work presents an IoT-based real-time health monitoring system designed to overcome communication challenges in developing countries like Bangladesh. By utilizing wireless communication technologies and offline data storage capabilities, the system enables continuous monitoring and reporting of individuals' health conditions. Implementing such a system could have significant implications for healthcare delivery and contribute to better health outcomes in resource-constrained settings. In [2] author aims to investigate health monitoring architectures for both individual and group scenarios within the context of the Internet of Medical Things (IoMT). The IoMT refers to the interconnected network of medical devices and wearable devices that collect and transmit health-related data to improve patient care and enable remote monitoring.

Given the increasing popularity of wearable devices and their wide range of applications in medical and disaster rescue efforts, it is crucial to address the challenges posed by the dynamic nature of the IoMT topology. The frequent posture alterations and mobility of users can create difficulties in resource allocation and routing strategies for health monitoring.

To begin, let's discuss the health monitoring architecture for individuals in the IoMT. Typically, an individual-centric architecture focuses on monitoring the health status and vital signs of a single user. This involves wearable devices, such as smartwatches or fitness trackers, that collect various physiological parameters such as heart rate, blood pressure, body temperature, and activity levels. These devices communicate with a central hub or gateway, which aggregates and processes the data. The processed information can be stored locally or transmitted to a remote server for further analysis by healthcare professionals or an artificial intelligence system. This architecture enables real-time monitoring, early detection of health issues, and personalized healthcare interventions for individuals.

Indeed, IoT (Internet of Things) has the potential to significantly reduce healthcare costs, save time, and streamline diagnostic testing procedures. By integrating IoT technologies into healthcare systems [3], various intelligent applications can be developed to improve efficiency and outcomes. Some of the areas where IoT can be applied include smart health, home, city, parking, agriculture, and industry.

In the context of healthcare, IoT enables the development of modern patient healthcare management systems that utilize sensors and networks [4-9]. These systems allow for continuous health monitoring by using connected components on various devices. The sensors collect relevant health data and transmit it to smartphones or other processing modules such as Arduino Uno and Nodemcu.

By leveraging IoT in healthcare, several benefits can be realized. Firstly, it offers a straightforward and user-friendly approach to tracking and managing healthcare problems. Patients can easily monitor their health conditions using connected devices and receive real-time feedback or alerts. Healthcare providers can also access this data remotely, allowing for timely interventions and proactive care.

Energy efficiency is another advantage of IoT in healthcare. By utilizing low-power sensors and optimized data transmission protocols, IoT devices can operate for extended periods without requiring frequent battery changes or recharges. This aspect is particularly important for wearable devices and remote monitoring systems.

Scalability and interoperability are inherent features of IoT. Healthcare systems can be expanded to accommodate a large number of devices and users without significant infrastructure changes. Additionally, IoT devices and platforms can be designed to integrate with existing healthcare systems, enabling seamless data exchange and interoperability between different components.

Cost-effectiveness is a crucial factor in healthcare, and IoT can contribute to reducing expenses. By enabling remote patient monitoring and early detection of health issues, IoT can help prevent costly hospitalizations and emergency interventions. Furthermore, IoT-based solutions can optimize healthcare resource allocation, reduce manual labor, and improve overall operational efficiency.

Finally, IoT-based healthcare systems save time for both patients and healthcare providers. With real-time data collection and analysis, diagnosis and treatment decisions can be made more efficiently. Remote monitoring eliminates the need for frequent in-person visits, reducing travel time and waiting periods. This can be particularly beneficial for patients in rural or underserved areas.

In summary, IoT offers a promising approach to revolutionize healthcare by reducing costs, saving time, and improving diagnostic procedures. Through the integration of sensors, networks, and intelligent applications, IoT can enhance patient care, enable remote monitoring, and optimize healthcare operations.

Wearable devices [10] have got increasing popularity in wide applications in medical and disaster rescue efforts to ensure the health and safety of users, which facilitates the development of the Internet of Medical Things (IoMT). Due to the posture alteration and mobility of users, the topology of the IoMT changes frequently, which increase the difficulty for resource allocation and routing strategy, in the article [10], authors respectively probe into the health monitoring architectures of the IoMT for both individual and group, allowing the monitored users to move at will. Furtherly, combined with the diversity of disaster rescuers, we build an IoMT-based disaster rescuer health monitoring system with searchers, doctors and porters. Authors in [11] identified the technological advances made so far, analyzing the challenges to be overcome and provides an approach of future trends. Through selected works, it is possible to notice that further studies are important to improve current techniques and that novel concept and technologies of IoT are needed to overcome the identiﬁed challenges. Authors in [12-14] proposed a new ECG quality IoT-assisted signal analysis framework for applications of cardiac health surveillance is introduced. This paper provides an ECG-SSA methodology for the automated evaluation of the quality of ECG signals obtained in the sense of patient and physical activity. Results from experiments show that the suggested ECGSQA is equivalent to other existing methods on the basis of morphological and RR interval and machine learning strategies. The analysis shows that the ECG signals are severely corrupted during increased physical activities.

# METHODOLOGY

Remote patient monitoring utilizing IoT technologies has emerged as a transformative approach in healthcare, enabling the monitoring of patients' vital parameters from a distance. This method is particularly valuable when the patient is located far away from the caregiver or healthcare facility, facilitating timely interventions and improved care management. In this essay, we will delve into the significance of monitoring patients' vital parameters, including body temperature and pulse rate, to gain insights into their condition and discuss the role of IoT in achieving this goal shown in Figure 1.

GSM MODULE

MODULE

ESP-32 CONTROLLER

TEMPERATURE SENSOR

IR SENSOR

GLUCOSE SENSOR

SENSOR

PULSE SENSOR

SENSOR

WIFI

RELAY

ECG SENSOR

SENSOR

POWER SUPPLY

SUPPLY

BLYNK

APP

BLYNK SERVER

**Figure1: Proposed Block diagram**

**Monitoring Vital Parameters:**

The primary objective of remote patient monitoring is to gain comprehensive knowledge of a patient's condition, irrespective of their physical location. To accomplish this, monitoring multiple parameters becomes crucial. Body temperature is one of the fundamental indicators of a patient's health. By obtaining the patient's temperature, initial insights can be gained regarding their condition. A low temperature may suggest excessive sweating or hypothermia, while an elevated temperature indicates the presence of a fever.

In addition to temperature, monitoring the patient's blood circulation is vital for assessing their overall health and well-being. This can be achieved by measuring their pulse rate. Utilizing pulse rate-detecting sensors, changes in the patient's cardiovascular system can be tracked. Abnormal pulse rates may indicate irregular heartbeats or other cardiovascular issues, thereby alerting healthcare providers to potential problems.

**IoT in Remote Patient Monitoring:**

The Internet of Things plays a crucial role in enabling efficient remote patient monitoring. IoT devices, equipped with sensors and connectivity capabilities, collect and transmit data from the patient to a central system or caregiver's device. In this context, IoT devices are utilized to measure and transmit temperature and pulse rate data.

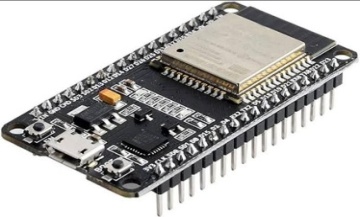
Once the data is collected, it can be analyzed using algorithms and software designed for health monitoring. By comparing the measured values against established ranges and predefined thresholds, healthcare providers can evaluate the patient's condition. If any abnormalities or concerns are identified, relevant messages or alerts can be sent to a designated mobile device. The Blynk app, for example, can be used to receive these notifications, displaying essential information about the patient's condition in real-time.

Remote patient monitoring utilizing IoT technologies provides an innovative and effective means of monitoring patients' vital parameters, regardless of their location. By monitoring parameters such as body temperature and pulse rate, healthcare providers can gain valuable insights into a patient's condition and promptly identify potential issues. The integration of IoT devices and sensors allows for seamless data collection, analysis, and communication, enabling timely interventions and enhanced care management. As technology continues to advance, the potential for IoT in remote patient monitoring is vast, paving the way for improved healthcare outcomes and patient well-being

# IMPLEMENTATION

**ESP 32 Controllers**

The ESP-32 controller is a versatile microcontroller board that is commonly used in IoT projects due to its built-in Wi-Fi and Bluetooth capabilities. It provides a convenient platform for integrating various sensors and modules to enable advanced functionalities. Here's an overview of the components you mentioned and how they can be used with the ESP-32 controller as shown in Fig 2.



**Figure 2: ESP 32 Controllers**

**Temperature Sensor:**

A temperature sensor, such as the popular DHT11 or DS18B20, can be connected to the ESP-32 controller to measure ambient or body temperature. The ESP-32 can read the sensor data and transmit it wirelessly to a remote server or display it locally shown in Fig 3. **Contactless temperatur** is a precision IC **temperature sensor** with its output proportional to the temperature (in oC). The sensor circuitry is sealed and therefore it is not subjected to oxidation and other processes. With **MLX90614**, temperature can be measured more accurately than with a thermistor. It also possess low self heating and does not cause more than 0.1oC temperature rise in still air.     The operating temperature range is from -55°C to 150°C. The output voltage varies by 10mV in response to every oC rise/fall in ambient temperature, i.e., its scale factor is 0.01V/oC.

.

**Figure 3: Temperature Sensor**

**HEART BEAT SENSOR**

The heartbeat sensor shown in Fig 4 is based on the principle of photo phlethysmography. It measures the change in volume of blood through any organ of the body which causes a change in the light intensity through that organ (a vascular region). In case of applications where heart pulse rate is to be monitored, the timing of the pulses is more important. The flow of blood volume is decided by the rate of heart pulses and since light is absorbed by blood, the signal pulses are equivalent to the heart beat pulses.

There are two types of photoplethysmography:

**Transmission**: Light emitted from the light emitting device is transmitted through any vascular region of the body like earlobe and received by the detector.

**Reflection**: Light emitted from the light emitting device is reflected by the regions for a product or service. These systems allow greater transparency, control, and performance when applied to any industry or system.



**Fig 4 : Heart Beat Sensor**

IoT systems have applications across industries through their unique flexibility and ability to be suitable in any environment. They enhance data collection, automation, operations, and much more through smart devices and powerful enabling technology.

IoT systems allow users to achieve deeper automation, analysis, and integration within a system. They improve the reach of these areas and their accuracy. IoT utilizes existing and emerging technology for sensing, networking, and robotics.

IoT exploits recent advances in software, falling hardware prices, and modern attitudes towards technology. Its new and advanced elements bring major changes in the delivery of products, goods, and services; and the social, economic, and political impact of those changes.

**IR Sensor:**

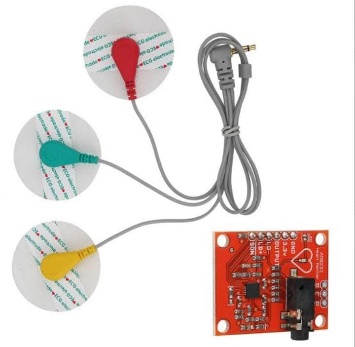
An infrared (IR) sensor shown in Fig 5, like the commonly used IR receiver module, can be used with the ESP-32 to detect infrared signals. This can be utilized for applications such as remote control systems, home automation, or proximity detection.



**Figure 5: IR Sensor**

**ECG Sensor:**

The ESP-32 can interface with an ECG (Electrocardiogram) sensor shown in Fig 6 to measure and monitor the electrical activity of the heart. ECG sensors, such as the AD8232 module, can be connected to the ESP-32 to capture heart rate and waveform data for healthcare or fitness applications.



**Figure 6: ECG Sensor**

**Glucose Sensor:**

A glucose sensor as in Fig 7, such as the Freestyle Libre or Dexcom G6, is designed to measure blood glucose levels. While the ESP-32 itself does not have a built-in glucose sensor, it can interface with compatible glucose sensors to retrieve data. The collected data can be used for diabetes management or continuous glucose monitoring systems.



**Figure 7: Glucose Sensor**

**Pulse Sensor:**

A pulse sensor as in Fig 8, such as the MAX30102 or the Pulse Sensor Amped, can be connected to the ESP-32 to measure heart rate and blood oxygen saturation levels. This data can be utilized for health and fitness tracking applications.



**Figure 8: Pulse Sensor**

**GSM Module:**

A GSM (Global System for Mobile Communications) module as shown in Fig 9 is a compact electronic device that enables mobile communication capabilities for other devices, such as microcontrollers or computers. It uses a SIM (Subscriber Identity Module) card to connect to a cellular network and transmit data through voice or text messages. GSM modules can be used in various applications such as security systems, health monitoring systems, and vending machines, among others. They enable health communication and control of devices over long distances and can operate on different frequency bands depending on the region they are deployed in. GSM modules can support different communication standards such as 2G, 3G, or 4G/LTE, with higher generations offering faster data transfer rates and better network coverage. Some modules also come equipped with GPS (Global Positioning System) or GNSS (Global Navigation Satellite System) capabilities, enabling location-based services. GSM modules can be programmed and controlled using standard AT commands, making them easy to integrate with other devices and software applications. Some modules also offer additional features such as built-in TCP/IP protocols, which enable direct communication with servers and cloud-based applications.



**Figure 9: GSM Sensor**

**Relay:**

A relay as in Fig 10 is an electromechanical switch that can be controlled by the ESP-32. By connecting a relay module to the ESP-32, you can remotely control high-power devices such as lights, motors, or appliances using digital signals.



**Figure 10: Relay**

With the ESP-32 controller and these various sensors and modules, you can create a wide range of IoT applications, including temperature monitoring systems, home automation projects, health monitoring devices, and remote control systems. The ESP-32's flexibility and connectivity options make it a popular choice for prototyping and implementing IoT solutions

ESP-32 is an open-source electronics platform based on easy-to-use hardware and software. [ESP-32 boards](https://www.arduino.cc/en/Main/Products) 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 [ESP-32 programming language](https://www.arduino.cc/en/Reference/HomePage) (based on [Wiring](http://wiring.org.co/)), and [the ESP-32 Software (IDE)](https://www.arduino.cc/en/Main/Software), based on [Processing](https://processing.org/).

Over the years ESP-32 has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of [accessible knowledge](http://forum.arduino.cc/) that can be of great help to novices and experts alike.

ESP-32 was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the ESP-32 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 ESP-32 boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The [software](https://www.arduino.cc/en/Main/Software), too, is open-source, and it is growing through the contributions of users worldwide.

ESP-32 is a computer hardware and software company, project, and user community that designs and manufactures [microcontroller](https://en.wikipedia.org/wiki/Microcontroller) kits for building digital devices and interactive objects that can sense and control objects in the physical world. The project's products are distributed as [open-source hardware](https://en.wikipedia.org/wiki/Open-source_hardware) and [software](https://en.wikipedia.org/wiki/Open-source_software), which are licensed under the [GNU Lesser General Public License](https://en.wikipedia.org/wiki/GNU_Lesser_General_Public_License) (LGPL) or the [GNU General Public License](https://en.wikipedia.org/wiki/GNU_General_Public_License) (GPL),[[1]](https://en.wikipedia.org/wiki/Arduino#cite_note-1) permitting the manufacture of ESP-32 boards and software distribution by anyone. ESP-32 boards are available commercially in preassembled form, or as [do-it-yourself](https://en.wikipedia.org/wiki/Do-it-yourself) kits.

The project's board designs use a variety of microprocessors and controllers. These systems provide sets of digital and analog [input/output](https://en.wikipedia.org/wiki/Input/output) (I/O) pins that may be interfaced to various expansion boards ("shields") and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus ([USB](https://en.wikipedia.org/wiki/USB)) on some models, for loading programs from personal computers. The microcontrollers are mainly programmed using a dialect of features from the programming languages [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++](https://en.wikipedia.org/wiki/C%2B%2B). In addition to using traditional compiler toolchains, the ESP-32 project provides an [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) based on the [Processing](https://en.wikipedia.org/wiki/Processing_(programming_language)) language project.

The ESP-32 project started in 2005 as a program for students at the [Interaction Design Institute Ivrea](https://en.wikipedia.org/wiki/Interaction_Design_Institute_Ivrea) in [Ivrea](https://en.wikipedia.org/wiki/Ivrea), Italy,[[2]](https://en.wikipedia.org/wiki/Arduino#cite_note-kushner-2) aiming to provide a low-cost and easy way for novices and professionals to create devices that interact with their environment using [sensors](https://en.wikipedia.org/wiki/Sensor) and [actuators](https://en.wikipedia.org/wiki/Actuator). Common examples of such devices intended for beginner hobbyists include simple [robots](https://en.wikipedia.org/wiki/Robot), [thermostats](https://en.wikipedia.org/wiki/Thermostat), and [motion detectors](https://en.wikipedia.org/wiki/Motion_detector).

ESP-32/Genuino Uno is a microcontroller board based on the ATmega328P ([datasheet](http://www.atmel.com/Images/doc8161.pdf)). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.. You can tinker with your UNO without worring too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.

"Uno" means one in Italian and was chosen to mark the release of ESP-32 Software (IDE) 1.0. The Uno board and version 1.0 of ESP-32 Software (IDE) were the reference versions of ESP-32, now evolved to newer releases. The Uno board is the first in a series of USB ESP-32 boards, and the reference model for the ESP-32 platform; for an extensive list of current, past or outdated boards see the ESP-32 index of boards.

### **Why ESP-32?**

Thanks to its simple and accessible user experience, ESP-32 has been used in thousands of different projects and applications. The ESP-32 software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux. Teachers and students use it to build low cost scientific instruments, to prove chemistry and physics principles, or to get started with programming and robotics. Designers and architects build interactive prototypes, musicians and artists use it for installations and to experiment with new musical instruments. Makers, of course, use it to build many of the projects exhibited at the Maker Faire, for example. ESP-32 is a key tool to learn new things. Anyone - children, hobbyists, artists, programmers - can start tinkering just following the step by step instructions of a kit, or sharing ideas online with other members of the ESP-32 community.

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Net media’s BX-24, Phi gets, MIT's Handy board, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. ESP-32 also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

* **Inexpensive** - ESP-32 boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the ESP-32 module can be assembled by hand, and even the pre-assembled ESP-32 modules cost less than $50
* **Cross-platform** - The ESP-32 Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.
* **Simple, clear programming environment** - The ESP-32 Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the ESP-32 IDE works.
* **Open source and extensible software** - The ESP-32 software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from ESP-32 to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your ESP-32 programs if you want to.
* **Open source and extensible hardware** - The plans of the ESP-32 boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the [breadboard version of the module](https://www.arduino.cc/en/Main/Standalone) in order to understand how it works and save money.

The ESP32 is a powerful and versatile microcontroller system-on-chip (SoC) developed by Espressif Systems. It serves as a successor to the widely popular ESP8266, offering improved performance, increased functionality, and enhanced capabilities. In this three-page document, we will explore the top view of the ESP32, providing a detailed description of its key components, features, and functionalities.

**System-on-Chip (SoC):**

The ESP32 integrates a high-performance 32-bit Xtensa LX6 dual-core processor, operating at up to 240MHz. This SoC not only provides processing power but also includes a wide range of peripherals, such as digital interfaces, analog interfaces, GPIOs, SPI, UART, I2C, and more. The presence of a dual-core architecture enables efficient multitasking, allowing the execution of multiple tasks simultaneously.

**Wi-Fi and Bluetooth Connectivity:**

One of the standout features of the ESP32 is its built-in support for both Wi-Fi (802.11 b/g/n) and Bluetooth (v4.2 and BLE). This combination makes it an ideal choice for a variety of IoT applications that require wireless connectivity. The Wi-Fi module supports station mode, soft access point (AP) mode, and simultaneous AP and station mode. Bluetooth support includes classic Bluetooth and BLE, enabling seamless communication with other devices.

**Memory and Storage:**

The ESP32 is equipped with a generous amount of memory to accommodate diverse applications. It features up to 520KB of SRAM for data storage and manipulation, as well as up to 4MB of flash memory for program storage. Additionally, it supports external SPI flash up to 16MB, providing ample space for data logging, web server hosting, and other storage-intensive tasks.

**Power Management:**

Efficient power management is crucial for IoT devices, and the ESP32 incorporates various features to optimize energy consumption. It supports different power modes, including active mode, sleep mode, deep-sleep mode, and hibernation mode. These modes allow developers to strike a balance between performance and power consumption, extending battery life and enhancing overall system efficiency.

**Security:**

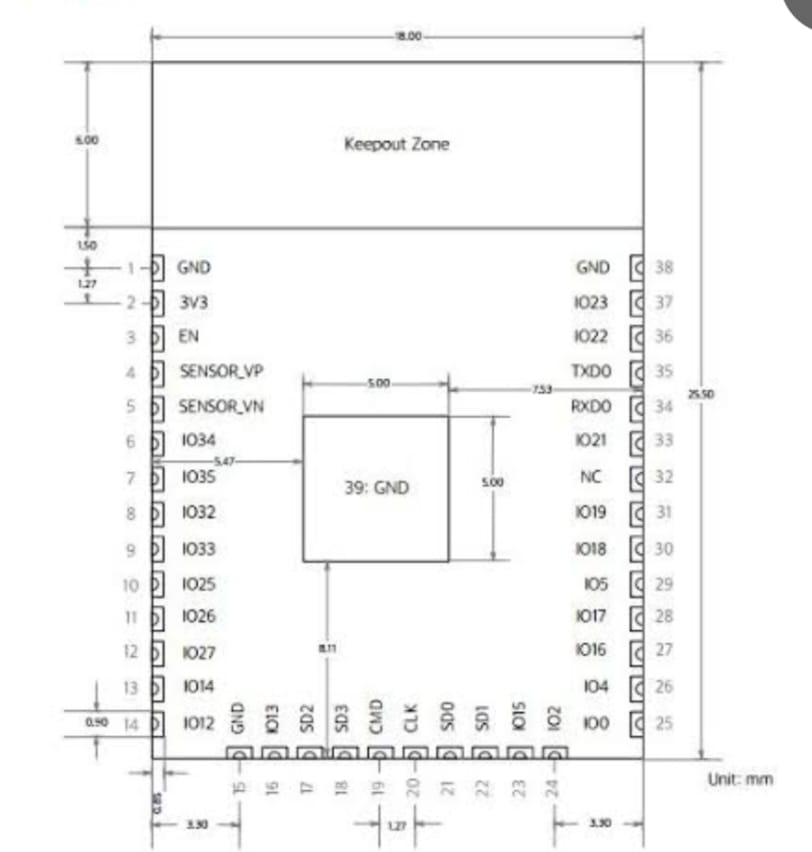
Security is a paramount concern in the IoT landscape, and the ESP32 incorporates robust security features to protect data and ensure device integrity. It supports various cryptographic algorithms, including RSA, AES, SHA, and ECC, enabling secure communication and data encryption. Additionally, it provides secure boot, flash encryption, and secure storage mechanisms to safeguard sensitive information.

**Peripherals and Interfaces:**

The ESP32 offers a wide range of peripherals and interfaces to facilitate seamless integration with external components. It includes multiple UART, SPI, and I2C interfaces for connecting sensors, actuators, and other devices. Moreover, it supports capacitive touch sensors, ADCs, DACs, PWM, and SDIO for additional functionality. These features make the ESP32 highly adaptable to various project requirements. The Atmel®picoPower® ATmega328/P is a low-power CMOS 8-bit microcontroller based on the AVR® enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega328/P achieves throughputs close to 1MIPS per MHz. This empowers system designer to optimize the device for power consumption versus processing speed.FreaturesHigh Performance, Low Power Atmel®AVR® 8-Bit Microcontroller Family Advanced RISC Architecture and the pin configuration is Fig 11

The ESP-32 is a powerful, low-cost, low-power system-on-a-chip (SoC) microcontroller with built-in Wi-Fi and Bluetooth capabilities. It is designed and produced by Espressif Systems and is a successor to the ESP8266 microcontroller.

The ESP-32 has two Xtensa 32-bit LX6 microprocessors with clock speeds up to 240 MHz, and includes a variety of peripherals such as 34 GPIO pins, multiple analog-to-digital converters, SPI, I2C, UART, and PWM interfaces, and a 12-bit SAR ADC with up to 18 channels. It also includes a range of security features such as hardware-accelerated AES, SHA-2, and RSA encryption, as well as hardware-based secure boot and flash encryption



**FIG 11: Pin Configuration of ESP 32**

.

One of the key features of the ESP-32 is its built-in Wi-Fi and Bluetooth capabilities, which allow it to connect to a variety of wireless networks and devices. It supports both 2.4 GHz and 5 GHz Wi-Fi bands, as well as Bluetooth Classic and BLE (Bluetooth Low Energy) protocols.

The ESP-32 is widely used in a variety of IoT (Internet of Things) applications, such as smart home automation, industrial automation, and robotics, due to its low power consumption, high performance, and rich feature set. It can be programmed using a variety of programming languages, including C, C++, Python, and Arduino.

The top view of the ESP-32 chip shows a small rectangular shape with a size of approximately 7mm x 7mm. The top of the chip is covered with a black epoxy coating, which protects the internal components of the chip.

In the center of the chip, there are two microprocessor cores, which are connected by an internal interconnect. These cores are surrounded by a number of peripheral interfaces, such as GPIO pins, SPI, I2C, UART, and PWM.

Around the edges of the chip, there are a number of bonding pads, which are used to connect the chip to an external printed circuit board (PCB). These bonding pads are arranged in a grid pattern and are used for power and ground connections, as well as for connecting the peripheral interfaces to external devices.

On the top surface of the chip, there are also a number of markings, which indicate the part number, manufacturer, and other relevant information. These markings are typically printed in white ink and are located near the corners of the chip.

Overall, the top view of the ESP-32 chip shows a compact and highly integrated microcontroller with a rich set of features and interfaces, designed for a wide range of IoT applications.

**Software development**

**Programming**

The ESP-32/Genuino Uno can be programmed with the ([ESP-32 Software](https://www.arduino.cc/en/Main/Software) (IDE)). Select "ESP-32/Genuino Uno from the Tools > Board menu (according to the microcontroller on your board). For details, see the [reference](https://www.arduino.cc/en/Reference/HomePage) and [tutorials](https://www.arduino.cc/en/Tutorial/HomePage).

The ATmega328 on the ESP-32/Genuino Uno comes preprogrammed with a [bootloader](https://www.arduino.cc/en/Hacking/Bootloader?from=Tutorial.Bootloader) that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol ([reference](http://www.atmel.com/Images/doc2525.pdf), [C header files](http://www.atmel.com/dyn/resources/prod_documents/avr061.zip)).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header using [ESP-32 ISP](https://www.arduino.cc/en/Main/ArduinoISP) or similar; see [these instructions](https://www.arduino.cc/en/Hacking/Programmer) for details.

The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available in the ESP-32 repository. The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by:

* On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then rese ing the 8U2.
* On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode.

You can then use [Atmel's FLIP software](http://www.atmel.com/products/microcontrollers/default.aspx) (Windows) or the [DFU programmer](http://dfu-programmer.github.io/) (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See [this user-contributed tutorial](http://forum.arduino.cc/index.php/topic,111.0.html) for more information.

**Warnings**

The ESP-32/Genuino Uno has a resettable poly1wfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

**Differences with other boards**

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

**Power**

The ESP-32/Genuino Uno board can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the GND and Vin pin headers of the POWER connector.

The board can operate on an external supply from 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may become unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

* Vin. The input voltage to the ESP-32/Genuino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
* 5V.This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.
* 3V3. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
* GND. Ground pins.
* IOREF. This pin on the ESP-32/Genuino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs to work with the 5V or 3.3V.

**Memory**

The ATmega328 has 32 KB (with 0.5 KB occupied by the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the [EEPROM library](https://www.arduino.cc/en/Reference/EEPROM)).

**Input and Output**

Each of the 14 digital pins on the Uno can be used as an input or output, using [pinMode()](https://www.arduino.cc/en/Reference/PinMode), [digitalWrite()](https://www.arduino.cc/en/Reference/DigitalWrite), and [digitalRead()](https://www.arduino.cc/en/Reference/DigitalRead) functions. They operate at 5 volts. Each pin can provide or receive 20 mA as recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50k ohm. A maximum of 40mA is the value that must not be exceeded on any I/O pin to avoid permanent damage to the microcontroller.

In addition, some pins have specialized functions:

* Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
* External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
* PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.
* SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.
* LED: 13. There is a built-in LED driven by digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
* TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.

**Communication**

**-**ESP-32/Genuino Uno has a number of facilities for communicating with a computer, another ESP-32/Genuino board, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The 16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, [on Windows, a .inf file is required](https://www.arduino.cc/en/Guide/Windows#toc4). The ESP-32 Software (IDE) includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A [Software Serial library](https://www.arduino.cc/en/Reference/SoftwareSerial) allows serial communication on any of the Uno's digital pins.

The ATmega328 also supports I2C (TWI) and SPI communication. The ESP-32 Software (IDE) includes a Wire library to simplify use of the I2C bus; see the [documentation](https://www.arduino.cc/en/Reference/Wire) for details. For SPI communication, use the [SPI library](https://www.arduino.cc/en/Reference/SPI).

Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the ESP-32/Genuino Uno board is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The ESP-32 Software (IDE) uses this capability to allow you to upload code by simply pressing the upload button in the interface toolbar. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Uno board contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see [this forum thread](http://forum.arduino.cc/index.php/topic,22974.0.html) for details.

ESP-32 programs may be written in any [programming language](https://en.wikipedia.org/wiki/Programming_language) with a compiler that produces binary machine code. Atmel provides a development environment for their microcontrollers, AVR Studio and the newer Atmel Studio, which can be used for programming ESP-32.

The ESP-32 project provides the ESP-32 [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE), which is a [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) application written in the programming language [Java](https://en.wikipedia.org/wiki/Java_(programming_language)). It originated from the IDE for the languages [*Processing*](https://en.wikipedia.org/wiki/Processing_(programming_language)) and [*Wiring*](https://en.wikipedia.org/wiki/Wiring_(development_platform)). It was created for people with no profound knowledge of electronics. It includes a code editor with features such as [syntax highlighting](https://en.wikipedia.org/wiki/Syntax_highlighting), [brace matching](https://en.wikipedia.org/wiki/Brace_matching), cutting-pasting and searching-replacing text, and automatic indenting, and provides simple one-click mechanism to compile and upload programs to an ESP-32 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 ESP-32 is called a "sketch".[[40]](https://en.wikipedia.org/wiki/Arduino#cite_note-43) Sketches are saved on the development computer as files with the file extension .ino. ESP-32 Software (IDE) pre-1.0 saved sketches with the extension .pde.

The ESP-32 IDE supports the languages [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++](https://en.wikipedia.org/wiki/C%2B%2B) using special rules to organize code. The ESP-32 IDE supplies a [software library](https://en.wikipedia.org/wiki/Software_library) from the [Wiring](https://en.wikipedia.org/wiki/Wiring_(development_platform)) project, which provides many common input and output procedures. User-written code only requires two functions, for starting the sketch and the main programs loop, that are compiled and linked with a program stub *main()* into an executable [cyclic executive](https://en.wikipedia.org/wiki/Cyclic_executive) program with the [GNU toolchain](https://en.wikipedia.org/wiki/GNU_toolchain), also included with the IDE distribution. The ESP-32 IDE employs the program *avrdude* to convert the executable code into a text file in hexadecimal coding that is loaded into the ESP-32 board by a loader program in the board's firmware

A minimal ESP-32 C/C++ sketch, as seen by the ESP-32 IDE programmer, consist of only two functions:

* *setup()*: This function is called once when a sketch starts after power-up or reset. It is used to initialize variables, input and output pin modes, and other libraries needed in the sketch.
* *loop()*: After setup() is called, this function is called repeatedly by a program loop in the main program. It controls the board until it is powered off or is reset.

Most ESP-32 boards contain a [light-emitting diode](https://en.wikipedia.org/wiki/Light-emitting_diode) (LED) and a load resistor connected between pin 13 and ground, which is a convenient feature for many tests and program functions.[[44]](https://en.wikipedia.org/wiki/Arduino#cite_note-Blink_Tutorial-47) A typical program for a beginning ESP-32 programmer blinks an LED repeatedly. This program is usually loaded in the ESP-32 by the manufacturer. In the ESP-32 environment, a user might write such a program as shown

#define LED\_PIN 13 // Pin number attached to LED.

void setup() {

pinMode(LED\_PIN, OUTPUT); // Configure pin 13 to be a digital output.

}

void loop() {

digitalWrite(LED\_PIN, HIGH); // Turn on the LED.

delay(1000); // Wait 1 second (1000 milliseconds).

digitalWrite(LED\_PIN, LOW); // Turn off the LED.

delay(1000); // Wait 1 second.

}

This program uses the functions pinMode(), digitalWrite(), and delay(), which are provided by the internal libraries included in the IDE environment.

**IOT FEATURES**

The most important features of IoT include artificial intelligence, connectivity, sensors, active engagement, and small device use. A brief review of these features is given below −

* **AI** − IoT essentially makes virtually anything “smart”, meaning it enhances every aspect of life with the power of data collection, artificial intelligence algorithms, and networks. This can mean something as simple as enhancing your refrigerator and cabinets to detect when milk and your favorite cereal run low, and to then place an order with your preferred grocer.
* **Connectivity** − New enabling technologies for networking, and specifically IoT networking, mean networks are no longer exclusively tied to major providers. Networks can exist on a much smaller and cheaper scale while still being practical. IoT creates these small networks between its system devices.
* **Sensors** − IoT loses its distinction without sensors. They act as defining instruments which transform IoT from a standard passive network of devices into an active system capable of real-world integration.
* **Active Engagement** − Much of today's interaction with connected technology happens through passive engagement. IoT introduces a new paradigm for active content, product, or service engagement.
* **Small Devices** − Devices, as predicted, have become smaller, cheaper, and more powerful over time. IoT exploits purpose-built small devices to deliver its precision, scalability, and versatility.

**IOT ADVANTAGES**

The advantages of IoT span across every area of lifestyle and business. Here is a list of some of the advantages that IoT has to offer −

* **Improved Customer Engagement** − Current analytics suffer from blind-spots and significant flaws in accuracy; and as noted, engagement remains passive. IoT completely transforms this to achieve richer and more effective engagement with audiences.
* **Technology Optimization** − The same technologies and data which improve the customer experience also improve device use, and aid in more potent improvements to technology. IoT unlocks a world of critical functional and field data.
* **Reduced Waste** − IoT makes areas of improvement clear. Current analytics give us superficial insight, but IoT provides real-world information leading to more effective management of resources.
* **Enhanced Data Collection** − Modern data collection suffers from its limitations and its design for passive use. IoT breaks it out of those spaces, and places it exactly where humans really want to go to analyze our world. It allows an accurate picture of everything.

**IOT SOFTWARE**

IoT software addresses its key areas of networking and action through platforms, embedded systems, partner systems, and middleware. These individual and master applications are responsible for data collection, device integration, real-time analytics, and application and process extension within the IoT network. They exploit integration with critical business systems (e.g., ordering systems, robotics, scheduling, and more) in the execution of related tasks.

## Data Collection

This software manages sensing, measurements, light data filtering, light data security, and aggregation of data. It uses certain protocols to aid sensors in connecting with real-time, machine-to-machine networks. Then it collects data from multiple devices and distributes it in accordance with settings. It also works in reverse by distributing data over devices. The system eventually transmits all collected data to a central server.

## Device Integration

Software supporting integration binds (dependent relationships) all system devices to create the body of the IoT system. It ensures the necessary cooperation and stable networking between devices. These applications are the defining software technology of the IoT network because without them, it is not an IoT system. They manage the various applications, protocols, and limitations of each device to allow communication.

## Real-Time Analytics

These applications take data or input from various devices and convert it into viable actions or clear patterns for human analysis. They analyze information based on various settings and designs in order to perform automation-related tasks or provide the data required by industry.

## Application and Process Extension

These applications extend the reach of existing systems and software to allow a wider, more effective system. They integrate predefined devices for specific purposes such as allowing certain mobile devices or engineering instruments access. It supports improved productivity and more accurate data collection.

**IOT TECHNOLOGY AND PROTOCOLS**

IoT primarily exploits standard protocols and networking technologies. However, the major enabling technologies and protocols of IoT are RFID, NFC, low-energy Bluetooth, low-energy wireless, low-energy radio protocols, LTE-A, and WiFi-Direct. These technologies support the specific networking functionality needed in an IoT system in contrast to a standard uniform network of common systems.

## NFC and RFID

RFID (radio-frequency identification) and NFC (near-field communication) provide simple, lowenergy, and versatile options for identity and access tokens, connection bootstrapping, and payments.

* RFID technology employs 2-way radio transmitter-receivers to identify and track tags associated with objects.
* NFC consists of communication protocols for electronic devices, typically a mobile device and a standard device.

## Low-Energy Bluetooth

This technology supports the low-power, long-use need of IoT function while exploiting a standard technology with native support across systems.

## Low-Energy Wireless

This technology replaces the most power hungry aspect of an IoT system. Though sensors and other elements can power down over long periods, communication links (i.e., wireless) must remain in listening mode. Low-energy wireless not only reduces consumption, but also extends the life of the device through less use.

## Radio Protocols

ZigBee, Z-Wave, and Thread are radio protocols for creating low-rate private area networks. These technologies are low-power, but offer high throughput unlike many similar options. This increases the power of small local device networks without the typical costs.

## LTE-A

LTE-A, or LTE Advanced, delivers an important upgrade to LTE technology by increasing not only its coverage, but also reducing its latency and raising its throughput. It gives IoT a tremendous power through expanding its range, with its most significant applications being vehicle, UAV, and similar communication.

## WiFi-Direct

WiFi-Direct eliminates the need for an access point. It allows P2P (peer-to-peer) connections with the speed of WiFi, but with lower latency. WiFi-Direct eliminates an element of a network that often bogs it down, and it does not compromise on speed or throughput.

**BLYNK App**

Blynk is a popular mobile application platform designed for IoT (Internet of Things) projects. It provides a user-friendly interface that allows you to control and monitor your IoT devices and sensors remotely. With the Blynk app, you can create custom dashboards, control widgets, and visualize data in real-time.

Here are some key features and functionalities of the Blynk app:

**Dashboard Creation:** Blynk allows you to design custom dashboards by adding various widgets, such as buttons, sliders, gauges, graphs, and displays. These widgets can be arranged and customized to suit your specific project requirements.

**Widget Control:** Each widget in the Blynk app is associated with a specific functionality. For example, a button widget can be used to turn on or off a connected device, a slider widget can control the intensity or speed of a device, and a gauge widget can display sensor data. You can interact with these widgets in real-time to control your IoT devices remotely.

**Data Visualization:** Blynk provides real-time data visualization capabilities, allowing you to monitor sensor data, device status, or any other relevant information. You can display data in the form of graphs, gauges, or simple value displays, making it easy to track and analyze the performance of your IoT devices.

**Event and Notification Management:** Blynk supports event-driven programming, allowing you to define triggers and actions based on specific conditions. You can set up notifications to receive alerts or push messages when certain events occur. For example, you can receive an alert when a sensor detects a specific threshold or when a device is turned on/off.

**Cloud Connectivity:** Blynk offers cloud connectivity, which means you can remotely access and control your IoT devices from anywhere with an internet connection. This allows you to monitor and manage your devices even when you are away from the physical location.

**Integration with IoT Platforms:** Blynk provides seamless integration with popular IoT platforms and hardware, including Arduino, Raspberry Pi, ESP8266, ESP32, and more. It offers a wide range of libraries and examples to help you quickly get started with your IoT projects.

Overall, the Blynk app simplifies the process of creating mobile interfaces for controlling and monitoring IoT devices. It provides an intuitive and user-friendly platform for building IoT applications without the need for extensive mobile app development knowledge.

Blynk is a toolset for all makers, badass inventors, designers, teachers, nerds and geeks who would love to use their smartphones to control electronics like ESP-32, RaspberryPi and similar ones. We’ve done all the hard work of establishing internet connection, building an app and writing hardware code.

 With Blynk, as shown in Fig 12 you simply snap together an amazing interface from various widgets we provide, upload the example code to your hardware and enjoy seeing first results in under 5 minutes! It works perfectly for newbie makers and saves tons of time for evil geniuses.

Blynk will work with all popular boards and shields. We wanted to give you full freedom when deciding how to plug Blynk into your existing or new project.You will also enjoy the convenience of Blynk Cloud. Which is, by the way is free and open-source.

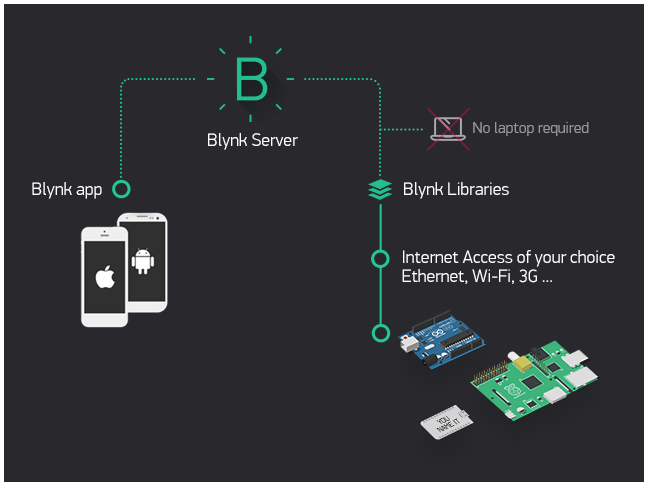
Imagine a prototyping board on your smartphone where you drag and drop buttons, sliders, displays, graphs and other functional widgets. And in a matter of minutes these widgets can control ESP-32 and get data from it.

Blynk is not an app that works only with a particular shield. Instead, it's been designed to support the boards and shields you are already using. And it works on iOs and Android.

**UPD:** Blynk also works over USB. This means you can tinker with the app by connecting it to your laptop or desktop while waiting for some internet shield to arrive.

**Blynk works over the Internet.**So the one and only requirement is that your hardware can talk to the Internet.

No matter what type of connection you choose - Ethernet, Wi-Fi or maybe this new ESP8266 everyone is talking about – Blynk libraries and example sketches will get you online, connect to Blynk Server and pair up with your smartphone.

****

**Figure 12 : Blynk architecture**

Currently, Blynk libraries work with:

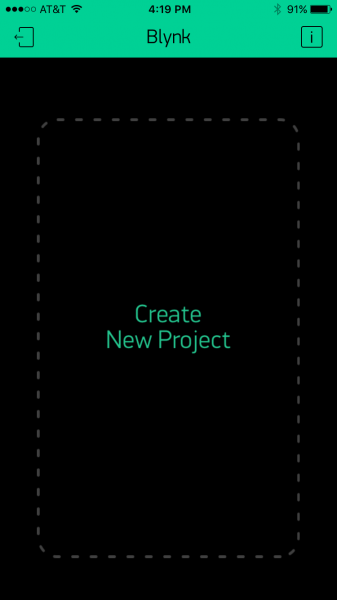
* USB
* Ethernet shield
* WiFi shield
* ESP-32 with Ethernet
* ESP-32 YÚN (testing in progress)
* ESP8266
* Raspberry Pi (Blynk will communicate with Pi's GPIOs)
* more ESP-32 compatible shields and boards (this list will be updated as we test the compatibility)

It's not that easy to take ESP-32 out of your home network, so we've built a Blynk server. It handles all the authentication and communication, and also keeps an eye on your board while the smartphone is offline. Blynk server runs on Java and is open-source. You will be able to run it locally if you really need to. Messaging between mobile apps , Blynk Server and ESP-32 is based on a simple, lightweight and fast binary protocol over TCP/IP sockets.

**CREATING A PROJECT IN BLYNK APP**

After downloading the app, create an account and log in. Welcome to Blynk!

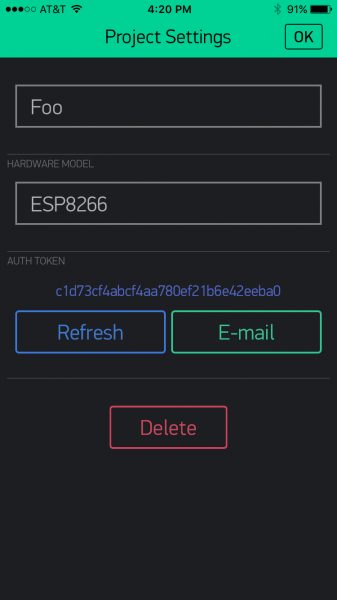
You’ll also need to install the **Blynk ESP-32 Library**, which helps generate the firmware running on your ESP8266. Download the latest release from Blynk’sGitHub repo, and follow along with the directions there to install the required libraries as sgown in Fig 13

[](https://cdn.sparkfun.com/assets/learn_tutorials/4/4/5/blynk-blank.PNG)

**Figure 13: CREATING A BLYNK APP**

### **Create a Blynk Project**

Next, click the “Create New Project” in the app to create a new Blynk app. Give it any name you please, just make sure the “Hardware Model” is set to **ESP8266** as shown in Fig 14.

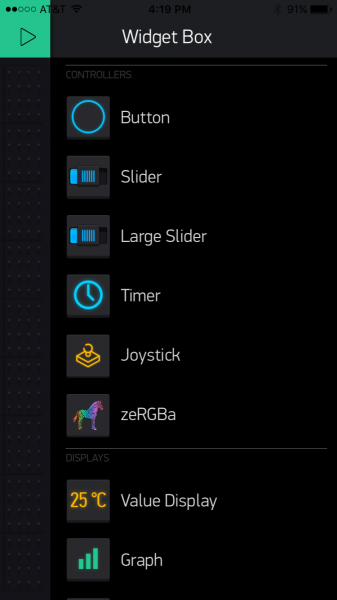
[](https://cdn.sparkfun.com/assets/learn_tutorials/4/4/5/Blynk-new.PNG)

**Figure 14 : ESP-32 HARDWARE SETTING**

The **Auth Token** is very important – you’ll need to stick it into your ESP8266’s firmware. For now, copy it down or use the “E-mail” button to send it to yourself.

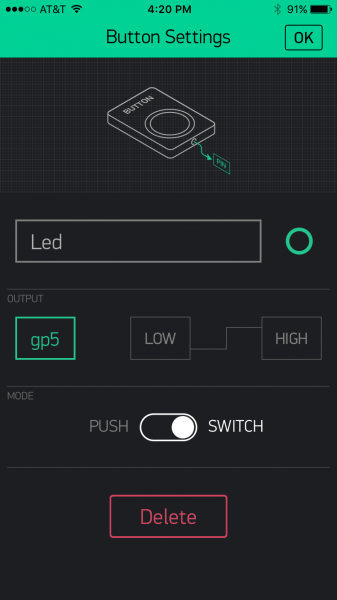
### **Add Widgets to the Project**

Then you’ll be presented with a blank new project. To open the widget box, click in the project window to open as shown Fig 15.

[](https://cdn.sparkfun.com/assets/learn_tutorials/4/4/5/Blynk-widgetBox.PNG)

**Figure 15: BLYNK APP WIDGET BOX**

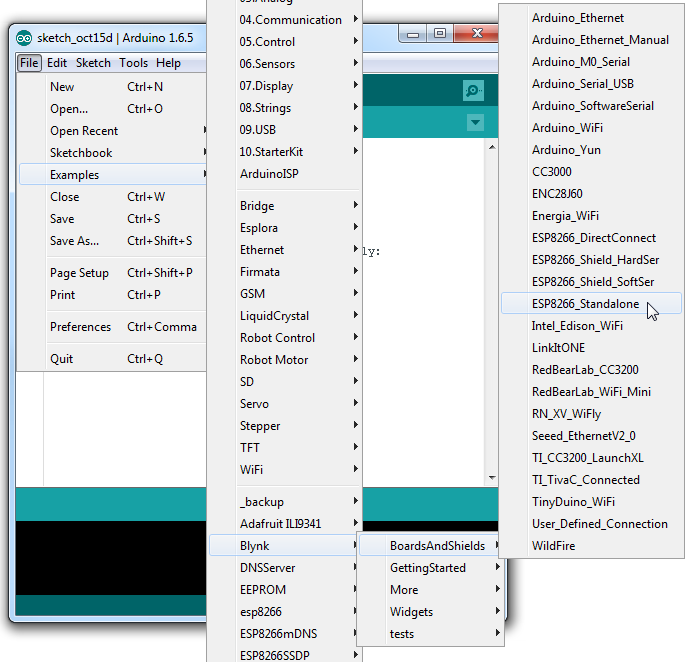
Add a **Button**, then click on it to change its settings. Buttons can toggle outputs on the ESP8266. Set the button’s output to **gp5**, which is tied to an LED on the Thing Dev Board. You may also want to change the action to “Switch” as shown in Fig 16

[](https://cdn.sparkfun.com/assets/learn_tutorials/4/4/5/Blynk-Button.PNG)

**Figure 16 BLYNK APP BUTTON SETTING**

### **Upload the Blynk Firmware**

Now that your Blynk project is set up, open ESP-32 and navigate to the **ESP8266\_Standalone as shown in Fig 17** example in the **File**>**Examples**>**Blynk**>**BoardsAndShields** menu.

[](https://cdn.sparkfun.com/assets/learn_tutorials/4/4/5/Blynk-example.png)

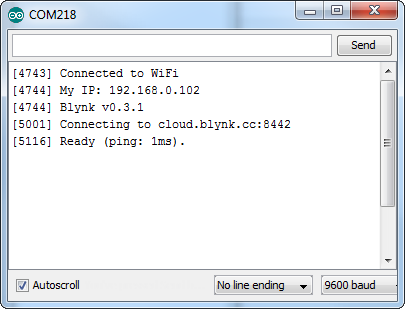
**Figure 17: BLYNK APP SET UP**

Before uploading, make sure to paste your **authorization token** into the auth[] variable. Also make sure to **load your WiFi network settings into the Blynkbegin(auth, "ssid", "pass")** function as shown in Fig 17.

Then upload!

### **Run the Project**

After the app has uploaded, open the serial monitor, setting the baud rate to 9600. Wait for the “Ready (ping: xms).” Message as shown in Fig 18.

[](https://cdn.sparkfun.com/assets/learn_tutorials/4/4/5/Blynk-Serial.png)

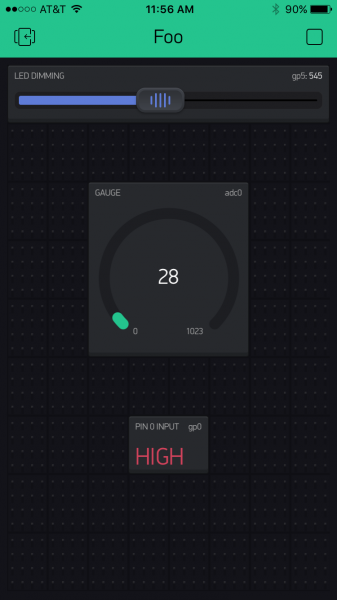
**Figure 18: RUNNING THE PROJECT**

Then click the “Run” button in the top right corner of the Blynk app. Press the button and watch the LED!

[](https://cdn.sparkfun.com/assets/learn_tutorials/4/4/5/esp8266-blynk.jpg)

**Figure 19: CONNECTING THE DEVICE TO BLYNK APP**

Then add more widgets to the project. They should immediately work on the ESP8266 without uploading any new firmware. As shown in Fig 19

**[](https://cdn.sparkfun.com/assets/learn_tutorials/4/4/5/Blynk-fullExample.PNG)**

**Figure 29: OUTPUT**

You can add analog output sliders, digital input monitors, analog input gauges.and the output is shown in Fig20

**Performance Requirements**

The sprawling nature of IoT requires comprehensive management of the entire network, wired and wireless, right to the edge as devices – smart and not so smart – seek access and data transfer to core network components. This is why the switch is key. All the connected devices and sensors are transmitting data on the network, but sending data from devices straight to the data centre can be inefficient, cause bottlenecks on the network, and impact performance.

An intelligent network needs to extend functionality right to the edge so data can be analysed and processed on the way to the core, or from device to device. To manage the increased flow of IoT traffic, switches at the edge of the network will need to offer enhanced security and integrated analytics.

Not every device is smart. Poorly secured 'smart' devices such as smart watches and activity trackers pose a threat to essential network security – as do traditional 'dumb' devices such as door locks.

Simply monitoring and controlling the flow of packets to and from IoT devices is not enough to guarantee security. All devices right out to the network edge must be made smarter by the network management and the switches on the network.

**Logical database requirements**

Cloud Storage is a service that allows to save data on offsite storage system managed by third-party and is made accessible by a **web services API.**

## Storage Devices

Storage devices can be broadly classified into two categories:

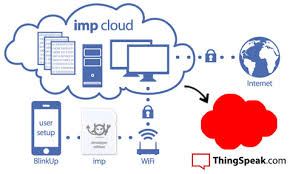
* Block Storage Devices
* File Storage Devices

### Block Storage Devices

The **block storage devices** offer raw storage to the clients. These raw storage are partitioned to create volumes.

**ThingSpeak:**

ThingSpeak as shown ib Fig 21 is an IoT platform and cloud service that enables users to collect, analyze, and visualize data from their IoT devices. It provides a convenient way to store, retrieve, and analyze sensor data in real-time. Here are some key features and functionalities of ThingSpeak:



**FIG 21: THINGSPEAK ARCHITECTURE**

**Data Collection:** ThingSpeak allows you to send data from your IoT devices to its cloud server. You can send data using various protocols such as HTTP, MQTT, and TCP/IP. ThingSpeak provides APIs and libraries for popular programming languages like MATLAB, Python, and Arduino to facilitate easy integration with your IoT devices.

**Data Storage:** Once the data is sent to ThingSpeak, it is stored in channels. A channel is a collection of data associated with a specific IoT device or sensor. ThingSpeak provides a RESTful API to create and manage channels. Each channel can have multiple fields to store different types of data.

**Data Visualization:** ThingSpeak offers built-in visualization tools that allow you to create custom dashboards and plots to visualize your sensor data. You can create line plots, bar graphs, histograms, or custom visualizations using MATLAB functions. ThingSpeak also supports real-time updates, allowing you to monitor and visualize data as it arrives.

**Data Analytics:** ThingSpeak provides powerful analytics features that enable you to analyze your sensor data. You can apply mathematical operations, perform data filtering, and apply algorithms to extract insights from your data. Additionally, you can use MATLAB code to perform complex data analysis and create predictive models.

**IoT Integrations:** ThingSpeak seamlessly integrates with other IoT platforms and services, such as MATLAB Analytics, IFTTT, and ThingHTTP. This allows you to leverage additional capabilities and connect your IoT applications with various external services and platforms.

**Webhooks and Alerts:** ThingSpeak supports webhooks and alerts, which allow you to trigger actions based on specific conditions in your data. You can set up alerts to notify you via email or SMS when certain thresholds are met or when anomalies are detected in your sensor data.

**Collaboration and Sharing:** ThingSpeak enables collaboration by allowing you to share your channels with others. You can control the permissions for shared channels, allowing others to view, edit, or analyze the data.

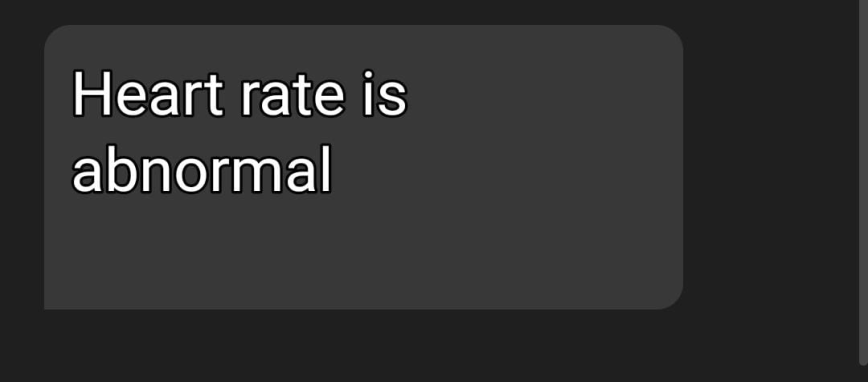
ThingSpeak provides a comprehensive IoT platform for collecting, storing, analyzing, and visualizing sensor data. Its integration with MATLAB and extensive API support makes it a versatile tool for developing IoT applications and performing advanced data analytics.

# RESULTS & DISCUSSIONS

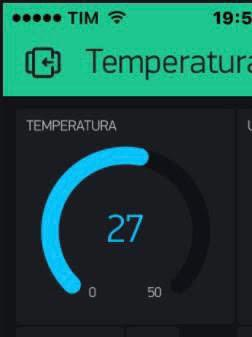
The system can accurately measure and monitor various vital signs, including temperature, pulse rate, glucose level, and ECG, it also can send alerts to a designated phone number if the pulse rate is abnormal, enabling timely medical attention. And the system can control a water pump, providing additional functionality for health and hygiene applications.

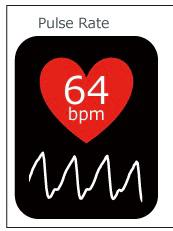
The proposed system can be a valuable asset in hospitals, clinics, and homes where timely health monitoring and security are critical and can potentially improve the quality of life for individuals with health conditions, by providing early detection of health issues and enabling timely medical attention the Temperature output of the patient and its pulse rate are shown in Fig 22 and 23.

The system may raise privacy concerns, and proper measures should be taken to ensure the security of the data collected and transmitted. The system can be expanded with additional sensors or functionality to enhance its capabilities, depending on specific requirements and use cases.If the heart rate is below 60 or above 100 ,it will notify that heart rate is abnormal as shown in Fig 21

.

**Figure 22: HEART RATE OUTPUT**

****

**Figure 23 TEMPERATURE OUTPUT**

**Fig: 24: PULSE RATE OUTPUT**

**APPLICATIONS AND ADVANTAGES:**

**Diabetes:**Diabetes management requires control of multiple parameters: blood pressure, weight, and blood glucose. The real-time delivery of blood glucose and blood pressure readings enables immediate alerts for patient and healthcare providers to intervene when needed. There is evidence to show that daily diabetes management involving RPM is just as effective as usual clinic visit every 3 months.

**Congestive heart failure**: A systematic review of the literature on home monitoring for heart failure patients indicates that RPM improves quality of life, improves patient-provider relationships, shortens duration of stay in hospitals, decreases mortality rate, and reduces costs to the healthcare system.

**Veterans Health Administration**: The Veterans Health Administration (VHA), United States’ largest integrated healthcare system, is an early adopter which became highly involved in the implementation and evaluation of RPM technologies. It has expanded use of RPM beyond common chronic disease applications, to post-traumatic stress disorder, cancer and palliative care. VHA’s findings indicate improvements in a wide range of metrics, including decrease in emergency department visits, hospitalizations, and nursing home admissions. Findings from the VHA Care Coordination/Home Telehealth program show that RPM deployment resulted in significant savings to the organization.

**Telehealth Response Watch**: For patients with dementia that are at risk for falls, RPM technology promotes safety and prevents harm through continuous surveillance .RPM sensors can be affixed to the individual or their assistive mobility devices such as canes and walkers. The sensors monitor an individual’s location, gait, linear acceleration and angular velocity, and utilize a mathematical algorithm to predict the likelihood for falls, detect movement changes, and alert caregivers if the individual has fallen.Furthermore, tracking capabilities via Wi-Fi, global positioning system (GPS) or radio frequency enables caregivers to locate wandering elders.

# CONCLUSION

Absolutely! A health monitoring and security system that leverages advanced technologies can indeed be a valuable tool for tracking and reporting on an individual's health status. By incorporating wearables, mobile devices, and cloud computing, such a system can provide real-time data on vital signs, activity levels, and other key health metrics. This enables healthcare providers to monitor patients remotely and offer timely advice and interventions. In addition to collecting and updating patient information, a comprehensive health monitoring system can offer several additional features. For example, it can integrate with ambulance services, allowing for quick response and coordination in emergencies. By providing access to a list of leading doctors and their specialties, patients can easily connect with the appropriate healthcare professional for further consultation or treatment. Including information about hospitals and their special facilities enables patients to make informed decisions about where to seek care. Moreover, a health monitoring and security system plays a vital role in maintaining the well-being and safety of individuals in various settings, particularly healthcare facilities. By incorporating sensors, data storage and analysis tools, and alarms or notifications, the system can promptly identify potential health issues or security breaches. This ensures that healthcare providers and patients are alerted to any critical situations requiring immediate attention. To guarantee the privacy and confidentiality of patients' health information, it is crucial for the system to adhere to industry standards and regulations, such as the Health Insurance Portability and Accountability Act (HIPAA) and the General Data Protection Regulation (GDPR). Compliance with these standards helps protect sensitive data and ensures that patient information is handled securely. A well-designed and properly implemented health monitoring and security system can enhance patient outcomes, improve remote monitoring capabilities, and increase security in healthcare settings. By combining advanced technologies, adhering to privacy regulations, and offering additional features, such a system can provide comprehensive support for healthcare providers and patients alike.

# REFERENCES

1. S.H. Almotiri, M. A. Khan, and M. A. Alghamdi. Mobile health (m- health) system in the context of iot. In 2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), pages 39–42, Aug 2016
2. MOHAMMAD NURUZZAMAN BHUIYAN 1, MD MASUM BILLAH 2, FARZANA BHUIYAN3,MD MAHBUBUR RAHMAN6, (Member, IEEE), MD SIPON MIAH6,7, (Member, IEEE) Based on Design and Implementation of a Feasible Model for the IoT Based Ubiquitous Healthcare Monitoring System for Rural and Urban Areas,date of publication 29 August 2022, date of current version 7 September 2022
3. Banka, Shubham and Madan, Isha and Saranya, SS "Smart healthcare monitoring using IoT" In International Journal of Applied Engineering Research, pp. 11948–11989, volume 13, number 15, 2018
4. Rahaman, Ashikur and Islam, Md Milon and Islam, Md Rashedul and Sadi, Muhammad Sheikh and Nooruddin, Sheikh "Developing IoT Based Smart Health Monitoring Systems: A Review" In Rev. d’Intelligence Artif, pp. 435–440, volume 13, number 6, 2019
5. Al-Ali, Abdul-Rahman and Zualkernan, Imran A and Rashid, Mohammed and Gupta, Ragini and AliKarar, Mazin "A smart home energy management system using IoT and big data analytics approach" In IEEE Transactions on Consumer Electronics, pp. 426–434, volume 63, number 4, 2017
6. Zanella, Andrea and Bui, Nicola and Castellani, Angelo and Vangelista, Lorenzo and Zorzi, Michele "Internet of things for smart cities" In IEEE Internet of Things journal, pp. 22-32, volume 1, number 1, 2014
7. Lin, Trista and Rivano, Hervé and Le Mouël, Fré déric "A survey of smart parking solutions" In IEEE Transactions on Intelligent Transportation Systems, pp. 22-32, volume 18, number 12, 2017
8. Condoluci, Massimo and Araniti, Giuseppe and Mahmoodi, Toktam and Dohler, Mischa "Enabling the IoT machine age with 5G: Machine-type multicast services for innovative real-time applications" In IEEE Access, volume 4, pp. 5555–5569, 2016
9. Condoluci, Massimo and Araniti, Giuseppe and Mahmoodi, Toktam and Dohler, Mischa "Enabling the IoT machine age with 5G: Machine-type multicast services for innovative real-time applications" In IEEE Access, volume 4, pp. 5555–5569, 2016
10. K. Wei, L. Zhang, Y. Guo and X. Jiang, "Health Monitoring Based on Internet of Medical Things: Architecture, Enabling Technologies, and Applications," in *IEEE Access*, vol. 8, pp. 27468-27478, 2020, doi: 10.1109/ACCESS.2020.2971654.
11. JOEL J. P. C. RODRIGUES, (Senior Member, IEEE), DANTE BORGES DE REZENDE SEGUNDO1,VICTOR HUGO C. DE ALBUQUERQUE 4, (Member, IEEE).Based on Enabling Technologies for the Internet of Health Things, date of publication January 4, 2018, date of current version March 19, 2018.
12. K. Zhang and W. Ling, "Health Monitoring of Human Multiple Physiological Parameters Based on Wireless Remote Medical System," in *IEEE Access*, vol. 8, pp. 71146-71159, 2020, doi: 10.1109/ACCESS.2020.2987058.
13. G. Cai, Y. Fang, J. Wen, G. Han and X. Yang, "QoS-Aware Buffer-Aided Relaying Implant WBAN for Healthcare IoT: Opportunities and Challenges," in *IEEE Network*, vol. 33, no. 4, pp. 96-103, July/August 2019, doi: 10.1109/MNET.2019.1800405.
14. Devi CU, TV P, Madhavi M. IoT Based Health Care Monitoring System using Wireless sensor Networks. Indian Journal of Public Health Research & Development. vol. 9, no. 12, pp. 1579–1583, 2018.

.