**1.INTRODUCTION**

**1.1 ARDUINO**

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

Over the years Arduino 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 that can be of great help to novices and experts alike.

Arduino 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 Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The software, too, is open-source, and it is growing through the contributions of users worldwide.

8

**1.1.1 Arduino as ADC**

Arduino reads the value from the specified analog pin. Arduino boards contain a multichannel, 10-bit analog to digital converter. This means that it will map input voltages between 0 and the operating voltage (5V or 3.3V) into integer values between 0 and 1023. On an Arduino UNO, for example, this yields a resolution between readings of: 5 volts / 1024 units or, 0.0049 volts (4.9 mV) per unit.

On ATmega based boards (UNO, Nano, Mini, Mega), it takes about 100 microseconds (0.0001 s) to read an analog input, so the maximum reading rate is about 10,000 times a second.

**Syntax - analogRead(pin)**

pin: the name of the analog input pin to read from (A0 to A5 on most boards, A0 to A6 on MKR boards, A0 to A7 on the Mini and Nano, A0 to A15 on the Mega).

**Returns**

The analog reading on the pin (int). Although it is limited to the resolution of the analog to digital converter (0-1023 for 10 bits or 0-4095 for 12 bits).

**Notes and Warnings**

If the analog input pin is not connected to anything, the value returned by analogRead() will fluctuate based on a number of factors (e.g. the values of the other analog inputs, how close your hand is to the board, etc.)

**1.2 Pulse Sensor**

A person’s heartbeat is the sound of the valves in his/her’s heart contracting or expanding as they force blood from one region to another. The number of times the heart beats per minute (BPM), is the heart beat rate and the beat of the heart that can be felt in any artery that lies close to the skin is the pulse.

**Two Ways to Measure a Heartbeat**

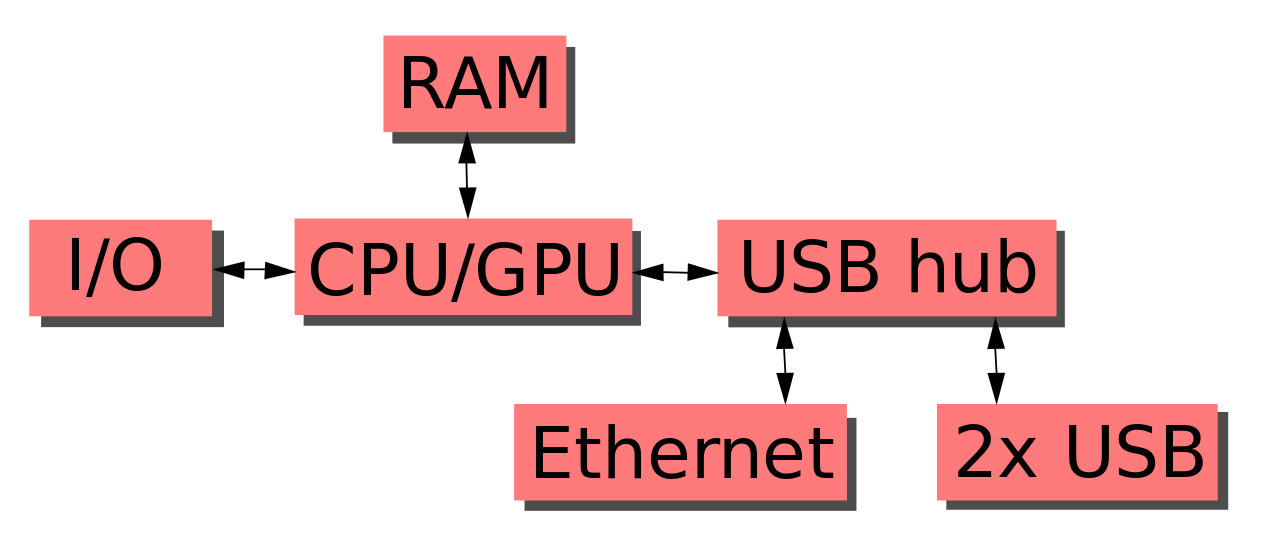
* **Manual Way**: Heart beat can be checked manually by checking one’s pulses at two locations- wrist (the **radial pulse**) and the neck (**carotid pulse**). The procedure is to place the two fingers (index and middle finger) on the wrist (or neck below the windpipe) and count the number of pulses for 30 seconds and then multiplying that number by 2 to get the heart beat rate. However pressure should be applied minimum and also fingers should be moved up and down till the pulse is felt.
* **Using a sensor**: Heart Beat can be measured based on optical power variation as light is scattered or absorbed during its path through the blood as the heart beat changes.

**Principle of Heartbeat Sensor**

The heartbeat sensor 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](https://www.edgefxkits.com/patient-health-monitoring-system-with-location-details-by-gps-over-gsm), 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.

**1.3 Raspberry Pi 3 model B**

The Raspberry Pi 3 Model B is a tiny credit card size computer. Just add a keyboard, mouse, display, ower supply, micro SD card with installed Linux Distribution and you'll have a fully-fledged computer that can run applications from word processors and spreadsheets to games.



**Functional block schematic of the Raspberry-Pi**

**1.3.1 Specification**

The headlining feature of the Pi 3 is the built-in WiFi and Bluetooth, but it doesn’t stop there. Here’s the complete specs for the Pi 3:

10

* SoC: Broadcom BCM2837 (roughly 50% faster than the Pi 2)
* CPU: 1.2 GHZ quad-core ARM Cortex A53 (ARMv8 Instruction Set)
* GPU: Broadcom VideoCore IV @ 400 MHz
* Memory: 1 GB LPDDR2-900 SDRAM
* USB ports: 4
* Network: 10/100 MBPS Ethernet, 802.11n Wireless LAN, Bluetooth 4.0.

**1.3.2** **Hardware in the Raspberry Pi**

**BCM2837**

This is the Broadcom chip used in the Raspberry Pi 3, and in later models of the Raspberry Pi 2. The underlying architecture of the BCM2837 is identical to the BCM2836. The only significant difference is the replacement of the ARMv7 quad core cluster with a quad-core ARM Cortex A53 (ARMv8) cluster.

The ARM cores run at 1.2GHz, making the device about 50% faster than the Raspberry Pi 2. The VideoCore IV runs at 400MHz.

**Power Supply**

The Raspberry Pi 3 is powered by a +5.1V micro USB supply. Exactly how much current (mA) the Raspberry Pi requires is dependent on what you connect to it. We have found that purchasing a 2.5A power supply from a reputable retailer will provide you with ample power to run your Raspberry Pi.

Typically, the model B uses between 700-1000mA depending on what peripherals are connected; the model A can use as little as 500mA with no peripherals attached. The maximum power the Raspberry Pi can use is 1 Amp. If you need to connect a USB device that will take the power requirements above 1 Amp, then you must connect it to an externally-powered USB hub.

The power requirements of the Raspberry Pi increase as you make use of the various interfaces on the Raspberry Pi. The GPIO pins can draw 50mA safely, distributed across all the pins; an individual GPIO pin can only safely draw 16mA. The HDMI port uses 50mA, the camera module requires 250mA, and keyboards and mice can take as little as 100mA or over 1000mA.

**Backpowering**

Backpowering occurs when USB hubs do not provide a diode to stop the hub from powering against the host computer. Other hubs will provide as much power as you want out each port. Please also be aware that some hubs will backfeed the Raspberry Pi. This means that the hubs will power the Raspberry Pi through its USB cable input cable, without the need for a separate micro-USB power cable, and bypass the voltage protection. If you are using a hub that backfeeds to the Raspberry Pi and the hub experiences a power surge, your Raspberry Pi could potentially be damaged.

**USB**

The Raspberry Pi Model B is equipped with two USB2.0 ports. These are connected to the LAN9512 combo hub/Ethernet chip IC3, which is itself a USB device connected to the single upstream USB port on BCM2835.On the Model A, the single USB2.0 port is directly wired to BCM2835.

The USB ports enable the attachment of peripherals such as keyboards, mice, webcams that provide the Pi with additional functionality.There are some differences between the USB hardware on the Raspberry Pi and the USB hardware on desktop computers or laptop/tablet devices.

The USB host port inside the Pi is an On-The-Go (OTG) host as the application processor powering the Pi, BCM2835, was originally intended to be used in the mobile market: i.e. as the single USB port on a phone for connection to a PC, or to a single device. In essence, the OTG hardware is simpler than the equivalent hardware on a PC.

OTG in general supports communication to all types of USB device, but to provide an adequate level of functionality for most of the USB devices that one might plug into a Pi, the system software has to do more work.

* **Supported devices**

In general, every device supported by Linux is possible to use with the Pi, subject to a few caveats detailed further down. Linux has probably the most comprehensive driver database for legacy hardware of any operating system (it can lag behind for modern device support as it requires open-source drivers for Linux to recognize the device by default).

If you have a device and wish to use it with a Pi, then plug it in. Chances are that it'll "just work". If you are running in a graphical interface (such as the LXDE desktop environment in Raspbian), then it's likely that an icon or similar will pop up announcing the new device.

If the device doesn't appear to work, then refer to the Troubleshooting section below.

* **General limitations**

The OTG hardware on Raspberry Pi has a simpler level of support for certain devices, which may present a higher software processing overhead. The Raspberry Pi also has only one root USB port: all traffic from all connected devices is funnelled down this bus, which operates at a maximum speed of 480mbps.

The USB specification defines three device speeds - Low, Full and High. Most mice and keyboards are Low-speed, most USB sound devices are Full-speed and most video devices (webcams or video capture) are High-speed. Generally, there are no issues with connecting multiple High-speed USB devices to a Pi.

The software overhead incurred when talking to Low- and Full-speed devices means that there are soft limitations on the number of simultaneously active Low- and Full-speed devices. Small numbers of these types of devices connected to a Pi will cause no issues.

* **Port Power Limits**

USB devices have defined power requirements, in units of 100mA from 100mA to 500mA. The device advertises its own power requirements to the USB host when it is first connected. In theory, the actual power consumed by the device should not exceed its stated requirement.

The USB ports on a Raspberry Pi have a design loading of 100mA each - sufficient to drive "low-power" devices such as mice and keyboards. Devices such as WiFi adapters, USB hard drives, USB pen drives all consume much more current and should be powered from an external hub with its own power supply. While it is possible to plug a 500mA device into a Pi and have it work with a sufficiently powerful supply, reliable operation is not guaranteed.

In addition, hot plugging high-power devices into the Pi's USB ports may cause a brownout which can cause the Pi to reset.

**GPIO**

General Purpose Input/output pins on the Raspberry Pi

GPIO pins can be configured as either general-purpose input, general-purpose output, or as one of up to six special alternate settings, the functions of which are pin-dependent.

There are three GPIO banks on BCM2835.

Each of the three banks has its own VDD input pin. On Raspberry Pi, all GPIO banks are supplied from 3.3V. Connection of a GPIO to a voltage higher than 3.3V will likely destroy the GPIO block within the SoC.

A selection of pins from Bank 0 is available on the P1 header on Raspberry Pi.

* GPIO pads

The GPIO connections on the BCM2835 package are sometimes referred to in the peripherals data sheet as "pads" — a semiconductor design term meaning 'chip connection to outside world'.

The pads are configurable CMOS push-pull output drivers/input buffers. Register-based control settings are available for:

* Internal pull-up / pull-down enable/disable
* Output drive strength
* Input Schmitt-trigger filtering
* Power-on states
* All GPIO pins revert to general-purpose inputs on power-on reset. The default pull states are also applied, which are detailed in the alternate function table in the ARM peripherals datasheet. Most GPIOs have a default pull applied.

**Interrupts**

Each GPIO pin, when configured as a general-purpose input, can be configured as an interrupt source to the ARM. Several interrupt generation sources are configurable:

* Level-sensitive (high/low)
* Rising/falling edge
* Asynchronous rising/falling edge
* Level interrupts maintain the interrupt status until the level has been cleared by system software (e.g. by servicing the attached peripheral generating the interrupt).

The normal rising/falling edge detection has a small amount of synchronization built into the detection. At the system clock frequency, the pin is sampled with the criteria for generation of an interrupt being a stable transition within a three-cycle window, i.e. a record of '1 0 0' or '0 1 1'. Asynchronous detection bypasses this synchronization to enable the detection of very narrow events.

**Alternative functions**

Almost all of the GPIO pins have alternative functions. Peripheral blocks internal to BCM2835 can be selected to appear on one or more of a set of GPIO pins, for example the I2C busses can be configured to at least 3 separate locations. Pad control, such as drive strength or Schmitt filtering, still applies when the pin is configured as an alternate function.

**SPI**

The Raspberry Pi is equipped with one SPI bus that has 2 chip selects.

The SPI master driver is disabled by default on Raspbian. To enable it, use raspi-config, or ensure the line dtparam=spi=on isn't commented out in /boot/config.txt, and reboot. If the SPI driver was loaded, you should see the device /dev/spidev0.0.

The SPI bus is available on the P1 Header:

MOSI P1-19

MISO P1-21

SCLK P1-23 P1-24 CE0

GND P1-25 P1-26 CE1

**1.4 LoRa(Long Range)**

LoRa (Long Range) is a patented digital wireless data communication technology developed by Cycleo of Grenoble, France, and acquired by Semtech in 2012. LoRa enables very-long-range transmissions (more than 10 km in rural areas) with low power consumption.[3] The technology is presented in two parts — LoRa, the physical layer and LoRaWAN (Long Range Wide Area Network), the upper layers.

LoRa PHY

The LoRa physical layer protocol is closed and proprietary; therefore, there is no freely available official documentation. However, several people have analyzed it and documented their findings.

LoRaWAN

LoRaWAN is the network on which LoRa operates, and can be used by IoT for remote and unconnected industries. LoRaWAN is a media access control (MAC) layer protocol but mainly is a network layer protocol for managing communication between LPWAN gateways and end-node devices as a routing protocol, maintained by the LoRa Alliance. Version 1.0 of the LoRaWAN specification was released in June 2015. In basic terms, one can consider LoRaWAN to be a new WiFi to connect new IoT devices across every industry.

LoRaWAN defines the communication protocol and system architecture for the network, while the LoRa physical layer enables the long-range communication link. LoRaWAN is also responsible for managing the communication frequencies, data rate, and power for all devices.Devices in the network are asynchronous and transmit when they have data available to send.

Data transmitted by an end-node device is received by multiple gateways, which forward the data packets to a centralized network server.The network server filters duplicate packets, performs security checks, and manages the network. Data is then forwarded to application servers. The technology shows high reliability for the moderate load, however, it has some performance issues related to sending acknowledgements.

**1.5 Softwares**

* **Arduino IDE**

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.

This software can be used with any Arduino board.

* **Introduction to the InfluxData Platform**

The InfluxData Platform is the leading modern time series platform designed from the ground up for metrics and events. It is comprised of four core components: Telegraf, InfluxDB, Chronograf, and Kapacitor (often referred to as the TICK stack). Each fulfills a specific role in managing your time-series data: data collection, data storage, data visualization, and data processing and alerting.

Enterprise versions of InfluxDB and Kapacitor provide clustering, access control, and incremental backup functionality for production infrastructures at scale.

The **TICK** stack - Open Source Components

**T**elegraf - Data collection

**I**nfluxDB - Data storage

**C**hronograf - Data visualization

**K**apacitor - Data processing and events

**Telegraf**

Data Collection

Telegraf is a data collection agent that captures data from a growing list of sources and translates it into Line Protocol data format for storage in InfluxDB. It’s “pluggable”, extensible architecture makes it easy to create plugins that both pull and push data from and to different sources and endpoints.

**InfluxDB**

Data Storage

InfluxDB stores data for any use case involving large amounts of timestamped data, including DevOps monitoring, log data, application metrics, IoT sensor data, and real-time analytics. It provides functionality that allows you to conserve space on your machine by keeping data for a defined length of time, then automatically downsampling or expiring and deleting unneeded data from the system.

* Key features

Here are some of the features that InfluxDB currently supports that make it a great choice for working with time series data.

\* Custom high performance datastore written specifically for time series data. The TSM engine allows for high ingest speed and data compression

\* Written entirely in Go. It compiles into a single binary with no external dependencies.

\* Simple, high performing write and query HTTP APIs.

\* Plugins support for other data ingestion protocols such as Graphite, collected, and OpenTSDB.

\* Expressive SQL-like query language tailored to easily query aggregated data.

\* Tags allow series to be indexed for fast and efficient queries.

\* Retention policies efficiently auto-expire stale data.

\* Continuous queries automatically compute aggregate data to make frequent queries more efficient.

**Chronograf**

Data Visuzalization

Chronograf is the user interface for the TICK stack that provides customizable dashboards, data visualizations, and data exploration. It also allows you to view and manage Kapacitor tasks.

* Key features

Infrastructure monitoring

\* View all hosts and their statuses in your infrastructure

\* View the configured applications on each host

\* Monitor your applications with Chronograf’s pre-created dashboards

Alert management

Chronograf offers a UI for Kapacitor, InfluxData’s data processing framework for creating alerts, running ETL jobs, and detecting anomalies in your data.

\* Generate threshold, relative, and deadman alerts on your data

\* Easily enable and disable existing alert rules

\* View all active alerts on an alert dashboard

\* Send alerts to the supported event handlers, including Slack, PagerDuty, HipChat, and more

Data visualization

\* Monitor your application data with Chronograf’s pre-created dashboards

\* Create your own customized dashboards complete with various graph types and template variables

\* Investigate your data with Chronograf’s data explorer and query templates

Database management

\* Create and delete databases and retention policies

\* View currently-running queries and stop inefficient queries from overloading your system

\* Create, delete, and assign permissions to users (Chronograf supports InfluxDB OSS and InfluxEnterprise user management)

Multi-organization and multi-user support

\* Create organizations and assign users to those organizations

\* Restrict access to administrative functions

\* Allow users to setup and maintain unique dashboards for their organizations

**Kapacitor**

Data Processing & Events

Kapacitor is a data processing framework that enables you to process and act on data as it is written to InfluxDB. This includes detecting anomalies, creating alerts based on user-defined logic, and running ETL jobs.

* Key features

Here are some of the features that Kapacitor currently supports that make it a great choice for data processing.

\* Process both streaming data and batch data.

\* Query data from InfluxDB on a schedule, and receive data via the line protocol and any other method InfluxDB supports.

\* Perform any transformation currently possible in InfluxQL.

\* Store transformed data back in InfluxDB.

\* Add custom user defined functions to detect anomalies.

\* Integrate with HipChat, OpsGenie, Alerta, Sensu, PagerDuty, Slack, and more.

* **Telegram**

Telegram Messenger is a messaging app that works over the internet, just like WhatsApp or Facebook Messenger. That means you can send messages for free by using a wi-fi connection or your mobile data allowance (providing you have enough data).

How does it differ from other messaging apps?

Telegram’s main selling point is security. It claims all its activities – including chats, groups and media – are encrypted, meaning even if they are intercepted, they won’t be visible without being deciphered first.

**2.Implementation**



**2.2 Hardware Used**

**1.Raspberry Pi:** The Raspberry Pi is a single-board, low-cost, high performance computer first developed in the United Kingdom by

the Raspberry Pi Foundation. The Raspberry Pi is slower than a modern

laptop or desktop but is still a complete Linux computer and can provide all the expected abilities that implies, at a low-power consumption level. The

Raspberry Pi is open hardware, with the exception of the primary chip on the Raspberry Pi, the Broadcom SoC (System on a Chip), which runs many of the main components of the board–CPU, graphics, memory, the

USB controller, etc.

**2.Arduino :** Arduino is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board.

The Arduino platform has become quite popular with people just starting out with electronics, and for good reason. Unlike most previous programmable circuit boards, the Arduino does not need a separate piece of hardware (called a programmer) in order to load new code onto the board – you can simply use a USB cable. Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program. Finally, Arduino provides a standard form factor that breaks out the functions of the micro-controller into a more accessible package.

**3. Pulse sensor :** Pulse Sensor is a well-designed plug-and-play heart-rate sensor for Arduino. Students, artists, athletes, makers, and game & mobile developers who want to easily incorporate live heart rate data into their projects can use it. The sensor clips onto a fingertip or earlobe and plugs right into Arduino with some jumper cables. It also includes an open-source monitoring app that graphs your pulse in real time.

The front of the sensor is the pretty side with the Heart logo. This side makes contact with the skin. On the front, you see a small round hole, which is where the LED shines through from the back, and there is a little square just under the LED. The square is an ambient light sensor, exactly as the one used in cellphones, tablets, and laptops, to adjust the screen brightness in different light conditions.

9

The LED shines light into the fingertip or earlobe, or other capillary tissue, and sensor reads the light that bounces back. The back of the sensor is where the rest of the parts are mounted. We put them there so they would not get in the way of the of the sensor on the front.

The cable is a 24 flat color-coded ribbon cable with 3 male header connectors.

RED wire = +3V to +5V

BLACK wire = GND

PURPLE wire = Signal

**3.Personal Computers:** For hospital end purpose we have used personal computer to show how code is going to run and visualize on dashboard.

**2.3 Softwares used:**

**2.3.1.NodeJS:**

• Perform asynchronous processing on single thread instead of classical multithread processing, minimize overhead & latency, maximize scalability

• Scale horizontally instead of vertically

• Ideal for applications that serve a lot of requests but don't use/need lots of computational power per request

• Not so ideal for heavy calculations, e.g. massive parallel computing

• Also: Less problems with concurrency

21

**2.3.2.Python:**

* Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.
* Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.
* Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance.
* Python supports modules and packages, which encourages program modularity and code reuse.
* The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

**2.3.2.Database:**

1. Collection of interrelated data

2. Set of programs to access the data

3. DBMS contains information about a particular enterprise

4. DBMS provides an environment that is both *convenient* and *efficient* to use.

5. Database Applications:

a. Banking: all transactions

b. Airlines: reservations, schedules c. Universities: registration, grades

d. Sales: customers, products, purchases

e. Manufacturing: production, inventory, orders, supply chain

f. Human resources: employee records, salaries, tax deductions

6. Databases touch all aspects of our lives

**TimeSeriesDataBase:**

A time series database (TSDB) is a software system that is optimized for handling time series data, arrays of numbers indexed by time (a datetime or a date time range). In some fields, these time series are called profiles, curves, or traces.

Ideally, repositories of time series are natively implemented using specialized database algorithms.[citation needed] However, it is possible to store time series as binary large objects (BLOBs) in a relational database or by using a VLDB approach coupled with a pure star schema.[citation needed] Efficiency is often improved if time is treated as a discrete quantity rather than as a continuous mathematical dimension.

**InfluxDB:**

Influx DB is an open-source time series database (TSDB) developed by Influx Data. It is written in Go and optimized for fast, high-availability storage and retrieval of time series data in fields such as operations monitoring, application metrics, Internet of Things sensor data, and real-time analytics. It also has support for processing data from Graphite.

**2.4.Operating System:**

• A modern computer consists of:

One or more processors

Main memory

Disks

Printers

Various input/output devices.

• Managing all these varied components requires a layer of software – the

Operating System (OS).

• An Operating System is a program that acts as an intermediary/interface between a user of a computer and the computer hardware.

• OS goals:

Control/execute user/application programs.

Make the computer system convenient to use.

Ease the solving of user problems.

Use the computer hardware in an efficient manner.

**2.5.Cloud:**

• The “no-need-to-know” in terms of the underlying details of infrastructure, applications interface with the infrastructure via the APIs.

• The “flexibility and elasticity” allows these systems to scale up and down at

will

– Utilising the resources of all kinds

• CPU, storage, server capacity, load balancing, and databases

The “pay as much as used and needed” type of utility computing and the “always on!, anywhere and any place” type of network-based computing.

• Cloud are transparent to users and applications, they can be built in multiple ways

– branded products, proprietary open source, hardware or software, or just off-the-shelf PCs.

In general, they are built on clusters of PC servers and off-the-shelf components plus Open Source software combined with in-house applications and/or system software

**2.6.MQTT**

The MQ Telemetry Transport (MQTT) protocol is a lightweight publish/subscribe protocol flowing over TCP/IP for remote sensors and control devices through low bandwidth, unreliable or intermittent communications**.**

**MQTT - Publish Subscribe Messaging aka One to Many:**

28

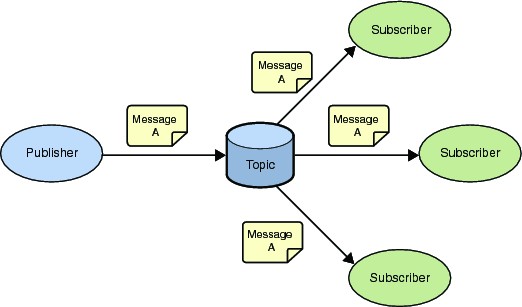
A Publish Subscribe messaging protocol allowing a message to be published once and multiple consumers (applications / devices) to receive the message providing decoupling between the producer and consumer(s)

A producer sends (publishes) a message (publication) on a topic (subject)

A consumer subscribes (makes a subscription) for messages on a topic (subject) A message server / broker matches publications to subscriptions

• If no matches the message is discarded

• If one or more matches the message is delivered to each matching subscriber/consumer



**2.7 Steps Involved**

1.Take pulse sensor data in Arduino.

2.After taking data from sensor , send it to Raspberry pi via USB.

3.Now from Raspberry pi, which is acting as a Gateway, sends data to InfluxDB and as well as to cloud.

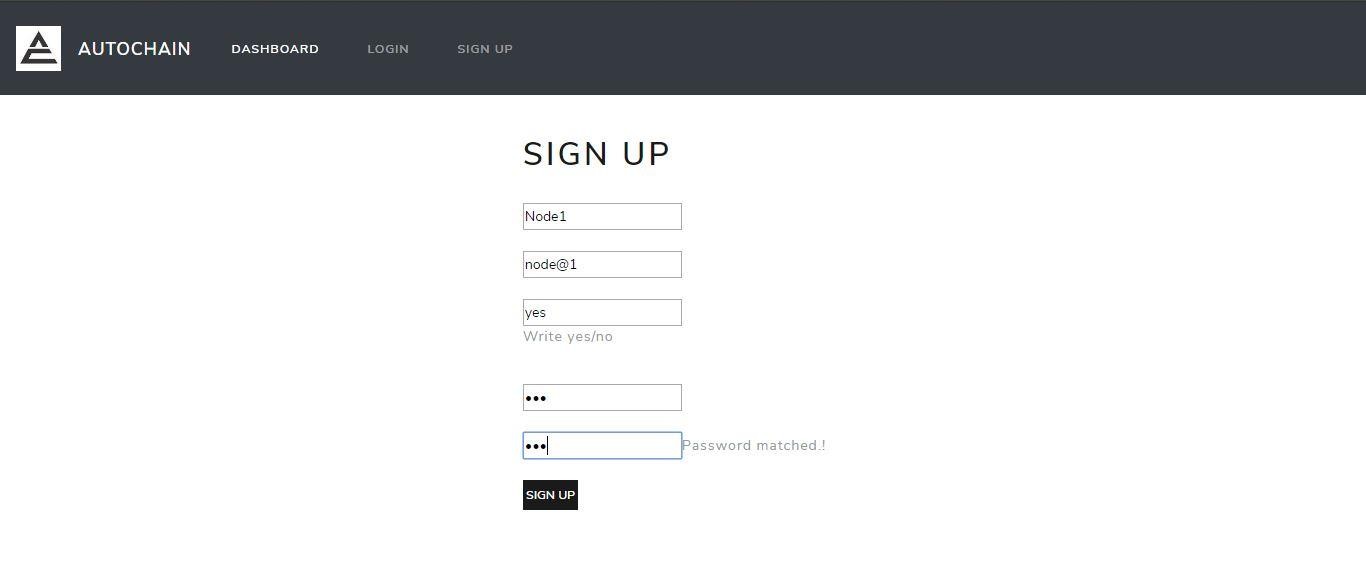
4.In Raspberry pi TICK script is generated using Chronograf which generate an alert on Telegram Bot and also shows graph of data.

5.Now Raspberry pi sends data to cloud which is Thingboard.

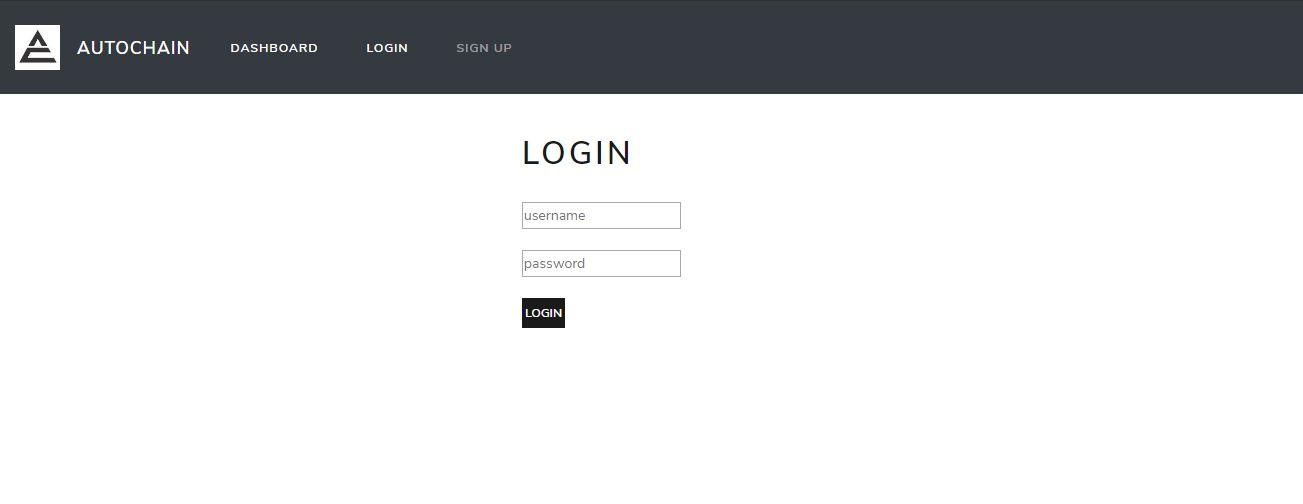
6.At the end for history purpose , there is map plotting.

**Results**

**3.1 Signup.**

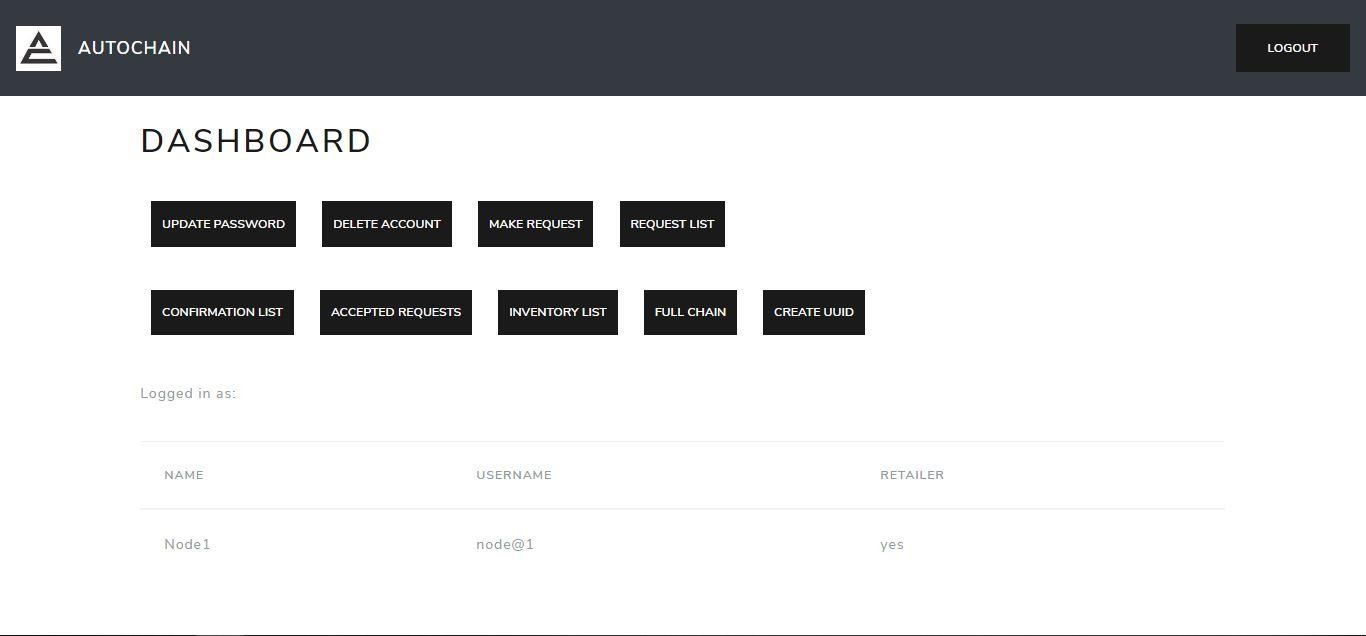


**3.2 Login**

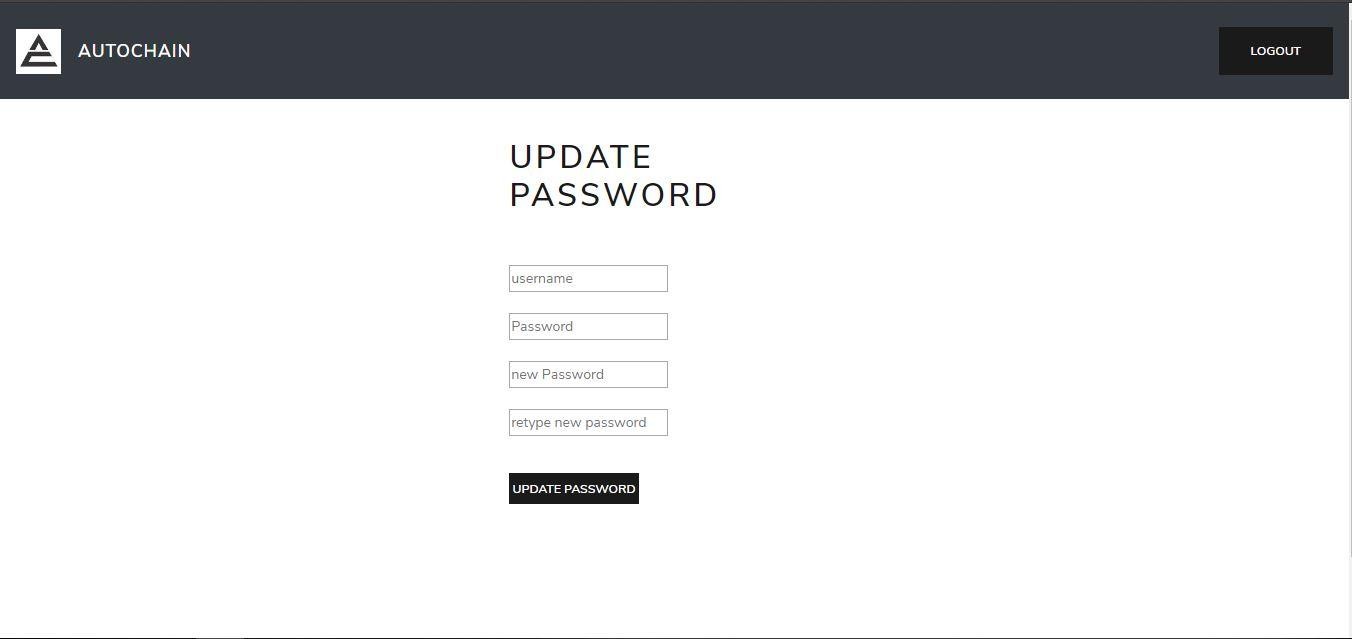


40

**3.3 Dashboard**

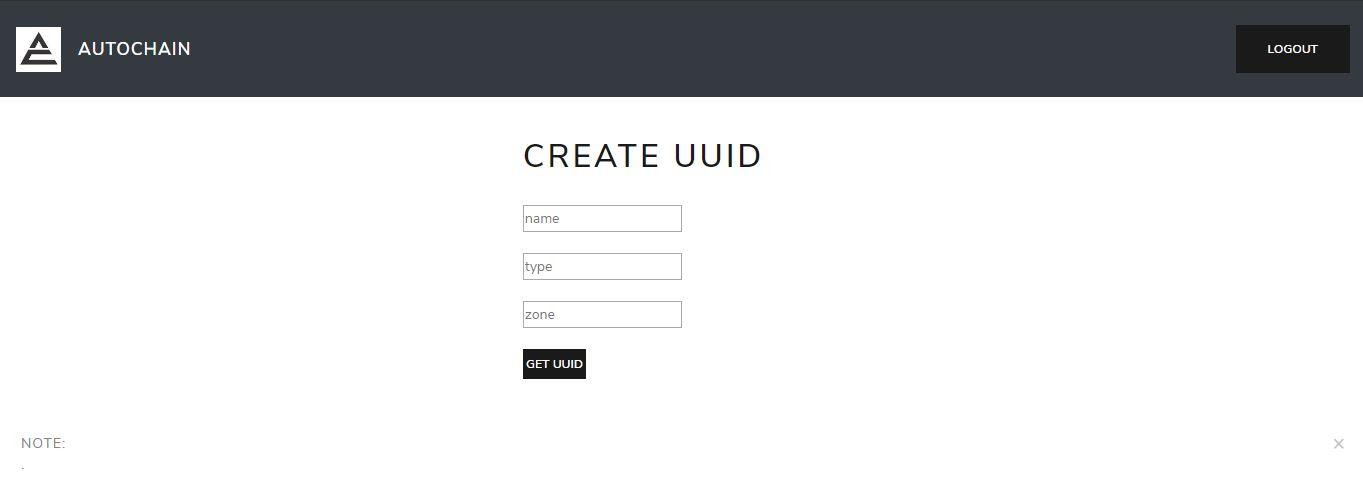


**3.4 Update Password**

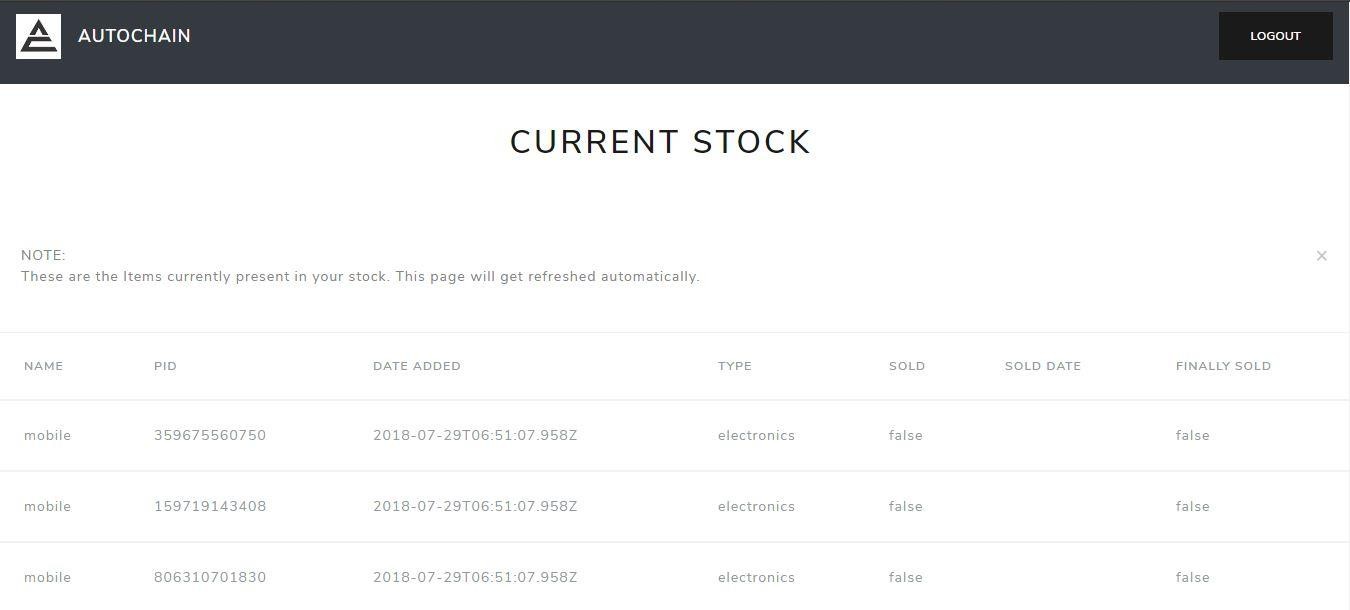


41

**3.5 Create UUID**

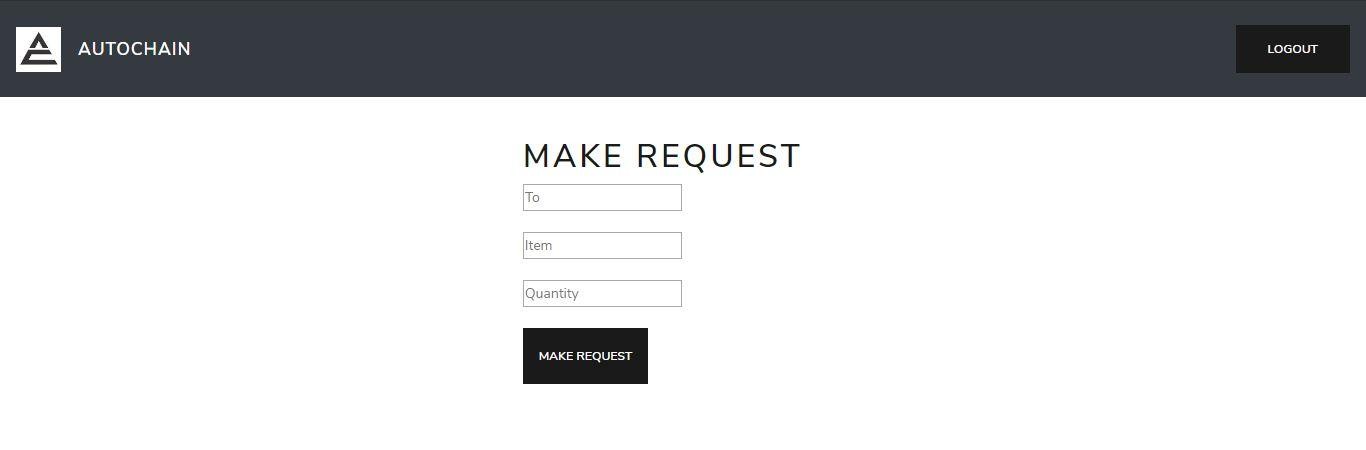


**3.6Inventory List**

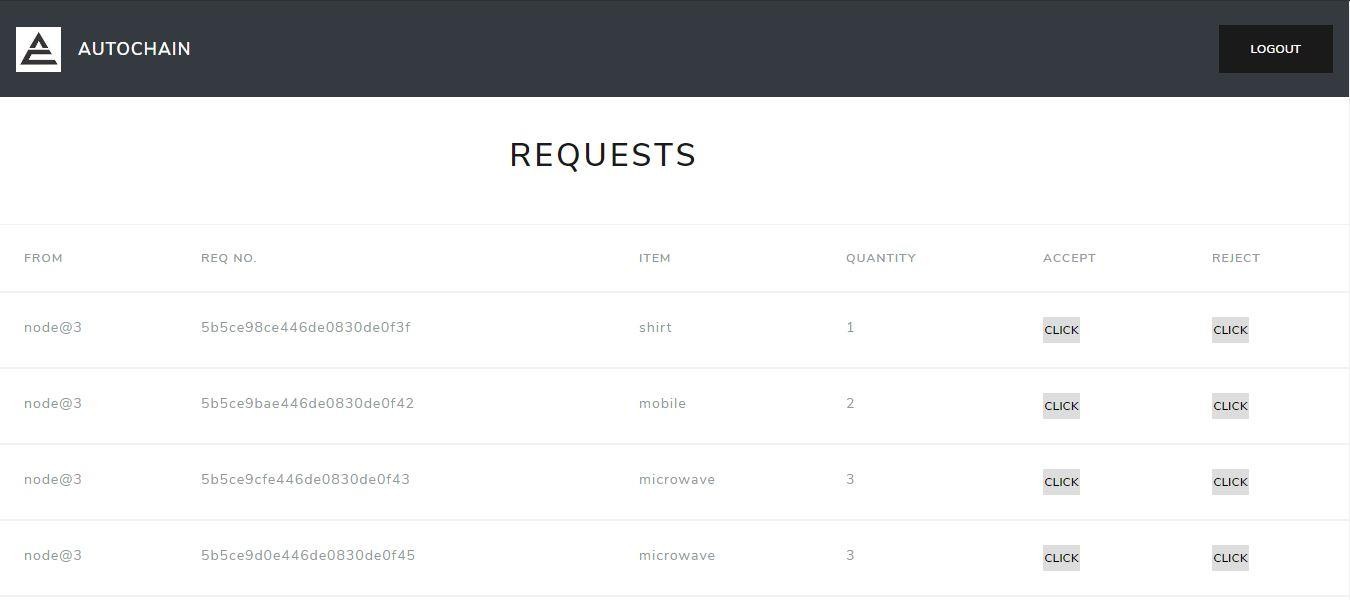


42

**3.7 Make Request**

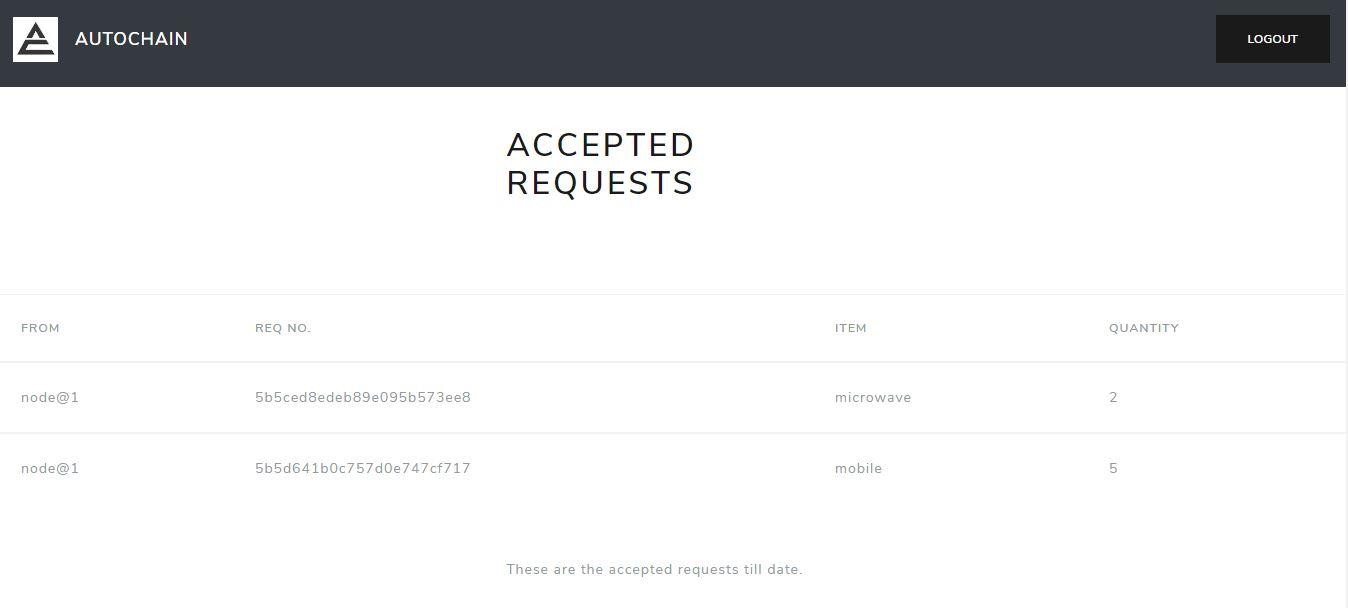


**3.8 Request List**

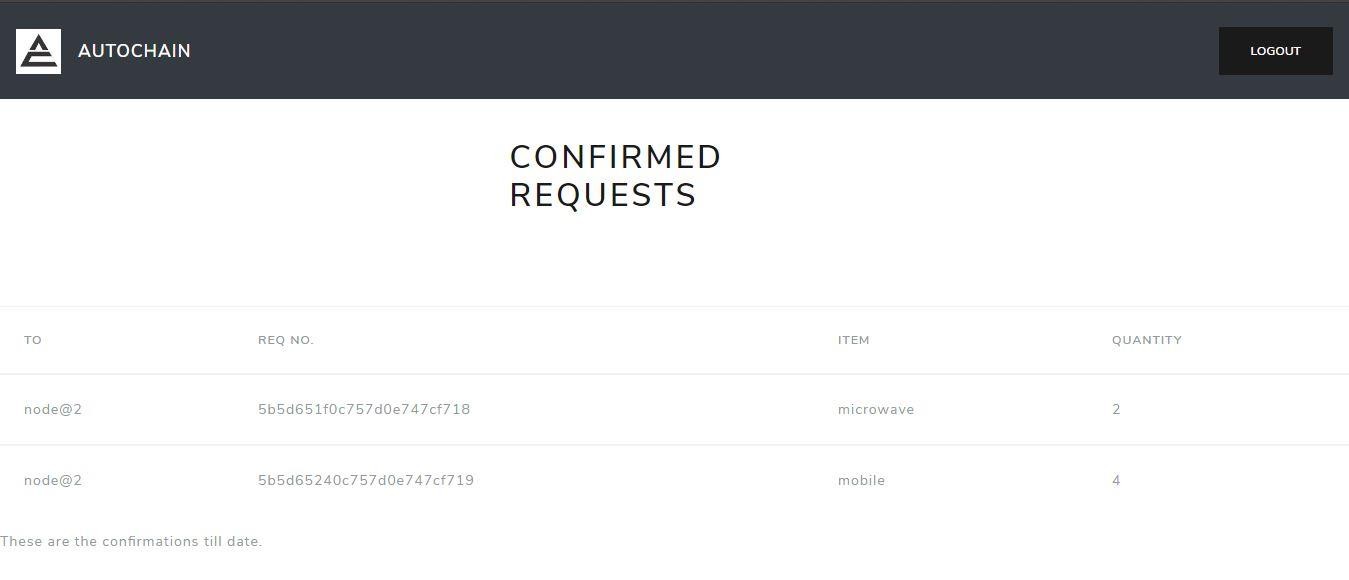


43

**3.9 Accepted Requests**

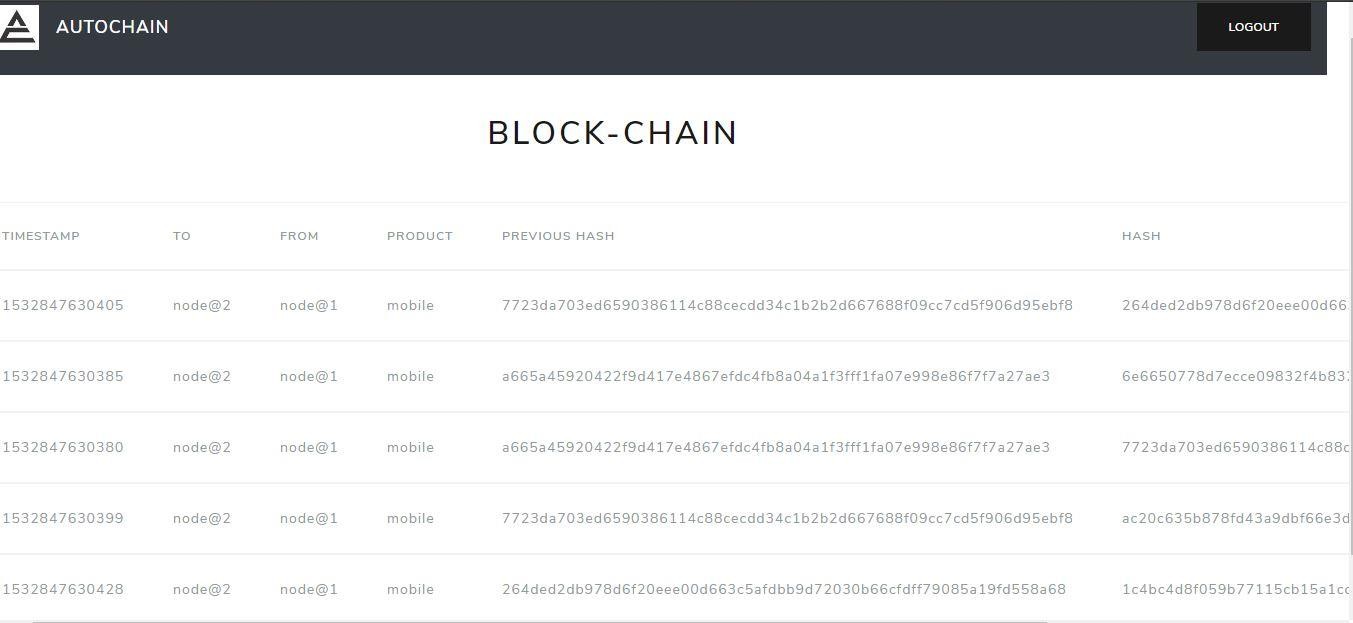


**4.0 Confirmed Request.**

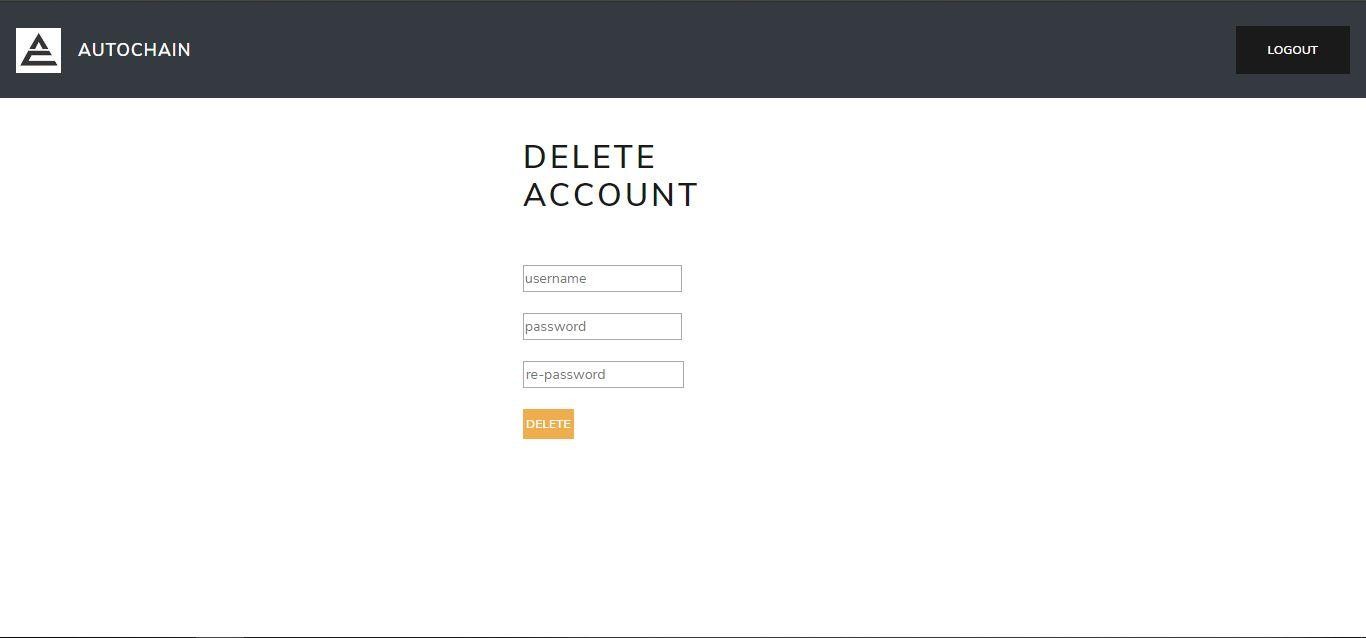


44

**4.1 Full Chain**



**4.2 Delete Account**



45

**4.Conclusions**

**4.1 Conclusion**

IoT changes the way the facilities are delivered to the healthcare industry. These technologies improve the product, causing a larger effect by bringing together minor changes.

**4.2 Challenges**

### Data security & privacy

One of the most significant threats that IoT poses is of data [security & privacy](https://www.peerbits.com/blog/10-iot-security-concerns-to-keep-in-mind-before-developing-apps.html). IoT devices capture and transmit data in real-time. However, most of the IoT devices lack data protocols and standards.



In addition to that, there is significant ambiguity regarding data ownership regulation. All these factors make the data highly susceptible to cybercriminals who can hack into the system and compromise Personal Health Information (PHI) of both patients as well as doctors.

Cybercriminals can misuse patient’s data to create fake IDs to buy drugs and medical equipment which they can sell later. Hackers can also file a fraudulent Insurance claim in patient’s name.

### Integration: multiple devices & protocols

Integration of multiple devices also causes hindrance in the implementation of IoT in the healthcare sector. The reason for this hindrance is that device manufacturers haven’t reached a consensus regarding communication protocols and standard.

So, even if the variety of devices are connected; the difference in their communication protocol complicates and hinders the process of data aggregation. This non-uniformity of the connected device’s protocols slows down the whole process and reduces the scope of scalability of IoT in healthcare.

### Data overload & accuracy

As discussed earlier, data aggregation is difficult due to the use of different communication protocols & standards. However, IoT devices still record a ton of data. The data collected by IoT devices are utilized to gain vital insights.

However, the amount of data is so tremendous that deriving insights from it are becoming extremely difficult for doctors which, ultimately affects the quality of decision-making. Moreover, this concern is rising as more devices are connected which record more and more data.

### Cost

Surprised to see Cost considerations in the challenge sections? I know most of you would be; but the bottom line is: IoT has not made the healthcare facilitates affordable to the common man yet. The boom in the Healthcare costs is a worrying sign for everybody especially the developed countries.

The situation is such that it gave rise to “Medical Tourism” in which patients with critical conditions access healthcare facilities of the developing nations which costs them as less as one-tenth. IoT in healthcare as a concept is a fascinating and promising idea.

However, it hasn’t solved the cost considerations as of now. To successfully implement IoT and to gain its total optimization the stakeholders must make it cost effective otherwise it will always remain out of everyone’s reach except the people from the high class.

**4.3 Future Scope**

The rise of IoT is exciting for everybody due to its different scope of use in various sectors. In Healthcare it has several applications. IoT in healthcare helps in:



* Reducing emergency room wait time
* Tracking patients, staff, and inventory
* Enhancing drug management
* Ensuring availability of critical hardware

IoT has also introduced several wearables & devices which has made lives of patients comfortable. These devices are as follows.

### Hearables

Hearables are new-age hearing aids which have completely transformed the way people who suffered hearing loss interact with the world. Nowadays, hearables are compatible with Bluetooth which syncs your smartphone with it.



It allows you to filter, equalize and add layered features to real-world sounds. Doppler Labs is the most suitable example of it.

### Ingestible sensors

Ingestible sensors are genuinely a modern-science marvel. These are pill-sized sensors which monitor the medication in our body and warns us if it detects any irregularities in our bodies.

These sensors can be a boon for a diabetic patient as it would help in curbing symptoms and provide with an early warning for diseases. Proteus Digital Health is one such example.

[](https://www.peerbits.com/blog/wearable-healthcare-apps-solutions-for-better-tomorrow.html)

### Moodables

Moodables are mood enhancing devices which help in improving our mood throughout the day. It may sound like science fiction, but it’s not far from reality.

Thync and Halo Neurosciences are already working on it and has made tremendous progress. Moodables are head-mounted wearables that send low-intensity current to the brain which elevates our mood.

### Computer vision technology

Computer vision technology along with AI has given rise to drone technology which aims to mimic visual perception and hence decision making based on it.



Drones like Skydio use computer vision technology to detect obstacles and to navigate around them. This technology can also be used for visually impaired people to navigate efficiently.

### Healthcare charting

IoT devices such as Audemix reduce much manual work which a doctor has to do during patient charting. It is powered by voice commands and captures the patient’s data. It makes the patient’s data readily accessible for review. It saves around doctors’ work by 15 hours per week.

48

**5.BIBLIOGRAPHY**

1. http://mqtt.org/documentation

2. https://www.python.org/

3. https://www.influxdata.com/

4 https://thingsboard.io/

5. https://www.raspberrypi.org/

6. https://www.arduino.cc/

7.https://www.peerbits.com/blog/internet-of-things-healthcare-applications-benefits-and-challenges.html

8. https://en.wikipedia.org/wiki/Time\_series

49