**1. Introduction**

**1.1 PROJECT OVERVIEW**

This is an IoT based system which sends the real-time data of patient to the doctor’s screen via cloud. It sends the patient’s data (Blood Pressure, Pulse Rate, Body Temperature, etc.) from the sensors which are attached to human body and interfaced with ESP microcontroller which eventually sends it to Raspberry Pi and then to AWS cloud where the data is stored and is visualized with the help of ElasticSearch and Kibana tools.

**1.2 SCOPE**

Current projects on smart ambulances mainly focus on traffic control problems. This project focuses on the medical condition perspective of the person which we feel is very important, so even during heavy traffic conditions, the doctor can keep track of and maintain the health condition of the patient with the help of new-age technologies such as IoT and Cloud.

**1.3 OBJECTIVE**

When a person is to be taken to a hospital during emergency situations, the doctors in the emergency ward do not get the current condition of the patient until the person reaches the hospital. This system will give the doctors, much needed preparation time by sending them the real time condition of the patient, so that the person can be treated as quickly as possible as time is a very critical factor. Moreover, the doctor can also send instruction to the ambulance staff while they are on their way to the hospital.

**2. System Analysis**

**2.1 TOOLS AND TECHNOLOGIES**

The following tools and technologies have been used to build this project:

1. NodeMCU (ESP8266)
2. MQTT Protocol
3. Raspberry Pi 3B+
4. AWS IoT Core
5. AWS ElasticSearch Service
6. Kibana
7. Sensors

**NodeMCU (ESP8266):**

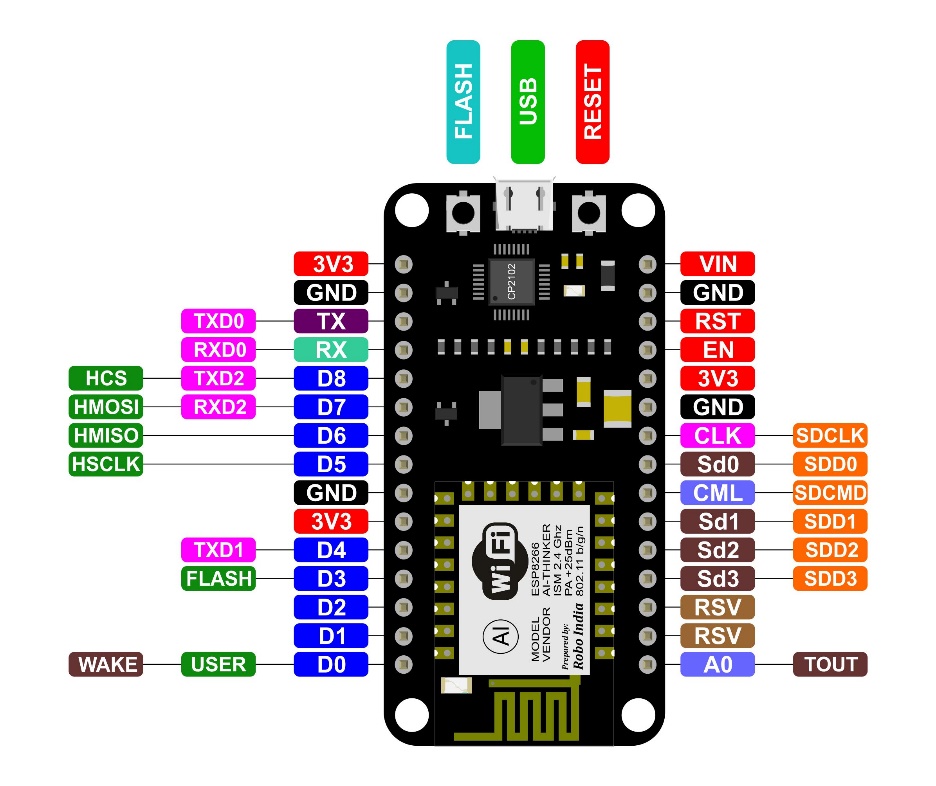


Fig.1

ESP8266 is Wi-Fi enabled system on chip (SoC) module developed by Espressif system. It is mostly used for development of IoT (Internet of Things) embedded applications.

It employs a 32-bit RISC CPU based on the Tensilica Xtensa L106 running at 80 MHz (or overclocked to 160 MHz). It has a 64 KB boot ROM, 64 KB instruction RAM and 96 KB data RAM. External flash memory can be accessed through SPI.

ESP8266 module is low cost standalone wireless transceiver that can be used for end-point IoT developments.

ESP8266 comes with capabilities of

* 2.4 GHz Wi-Fi (802.11 b/g/n, supporting WPA/WPA2),
* general-purpose input/output (16 GPIO),
* Inter-Integrated Circuit (I²C) serial communication protocol,
* analog-to-digital conversion (10-bit ADC)
* Serial Peripheral Interface (SPI) serial communication protocol,
* I²S (Inter-IC Sound) interfaces with DMA(Direct Memory Access) (sharing pins with GPIO),
* UART (on dedicated pins, plus a transmit-only UART can be enabled on GPIO2), and
* pulse-width modulation (PWM).

**MQTT (MQ Telemetry Transport) Protocol :**

MQTT is a lightweight**publish/subscribe** messaging protocol designed for M2M (machine to machine) telemetry in low bandwidth environments.

It was designed by Andy Stanford-Clark (IBM) and Arlen Nipper in 1999 for connecting Oil Pipeline telemetry systems over satellite.

Although it started as a proprietary protocol it was released Royalty free in 2010 and became an OASIS standard in 2014.

**MQTT** is fast becoming one of the main protocols for**IOT** (internet of things) deployments.

**Raspberry Pi 3B+ :**



Fig.2

A Raspberry Pi is a credit card-sized computer originally designed for education, inspired by the 1981 BBC Micro. But due to its small size and accessible price, it was quickly adopted by tinkerers, makers, and electronics enthusiasts for projects that require more than a basic microcontroller (such as Arduino devices).

The Raspberry Pi is open hardware, with the exception of the primary chip on the Raspberry Pi, the Broadcomm SoC, which runs many of the main components of the board–CPU, graphics, memory, the USB controller, etc.

The Raspberry Pi was designed for the Linux operating system, and many Linux distributions now have a version optimized for the Raspberry Pi. Two of the most popular options are Raspbian, which is based on the Debian operating system, and Pidora, which is based on the Fedora operating system.

**AWS IoT Core:**

AWS IoT provides secure, bi-directional communication between Internet-connected devices such as sensors, actuators, embedded micro-controllers, or smart appliances and the AWS Cloud. This enables you to collect telemetry data from multiple devices, and store and analyze the data. You can also create applications that enable your users to control these devices from their phones or tablets.

Components:

* Device gateway
* Message broker
* Rules engine
* Security and Identity service
* Registry
* Group registry
* Device shadow
* Device Provisioning Service
* Custom Authentication Service
* Jobs Service

**AWS ElasticSearch Service :**

Amazon Elasticsearch Service, is a fully managed service that makes it easy for you to deploy, secure, operate, and scale Elasticsearch to search, analyze, and visualize data in real-time. With Amazon Elasticsearch Service you get easy-to-use APIs and real-time analytics capabilities to power use-cases such as log analytics, full-text search, application monitoring, and clickstream analytics, with enterprise-grade availability, scalability, and security. The service offers integrations with open-source tools like Kibana for data ingestion and visualization.

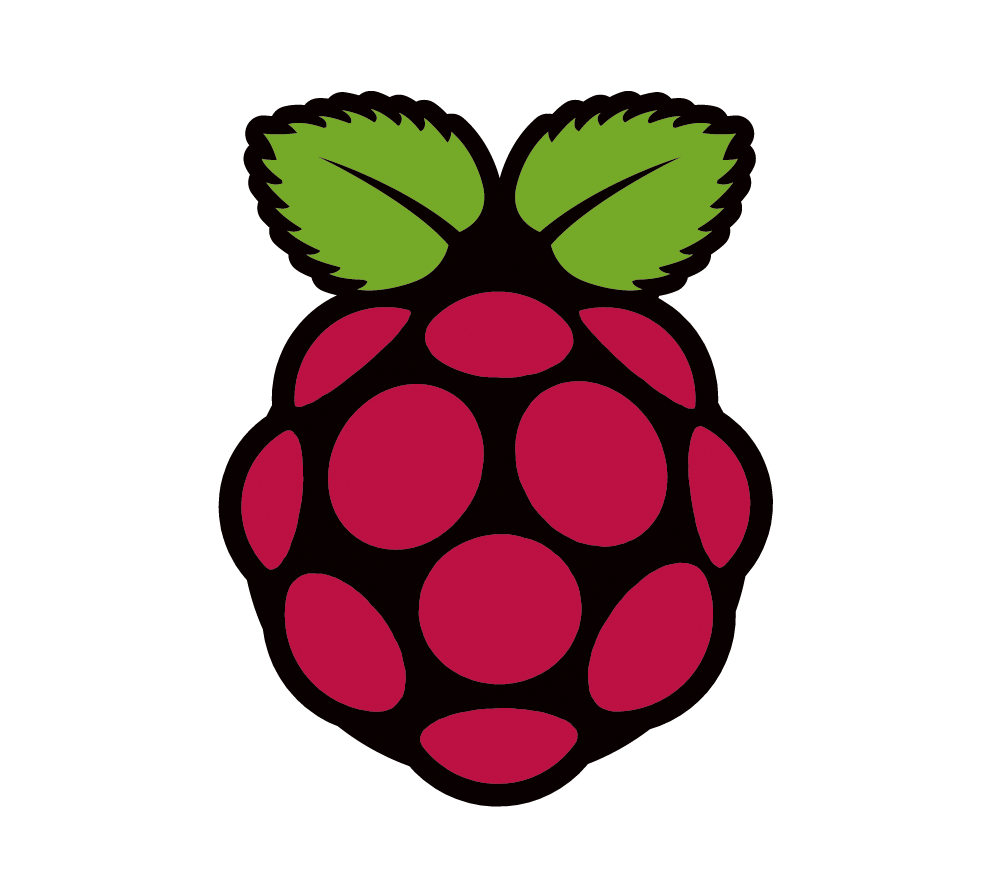
**Kibana:**

Kibana is an open source data visualization plugin for Elasticsearch. It provides visualization capabilities on top of the content indexed on an Elasticsearch cluster. Users can create bar, line and scatter plots, or pie charts and maps on top of large volumes of data.

The combination of Elasticsearch, Logstash, and Kibana, referred to as the "Elastic Stack" (formerly the "ELK stack"), is available as a product or service. Logstash provides an input stream to Elastic for storage and search, and Kibana accesses the data for visualizations such as dashboards.

**3. System Design**

**3.1 DATA FLOW OF THE SYSTEM:**



Sensor

Sensor

Sensor

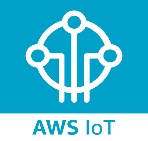
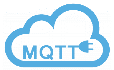


Fig 3

**3.2 FUNCTIONING OF THE SYSTEM:**

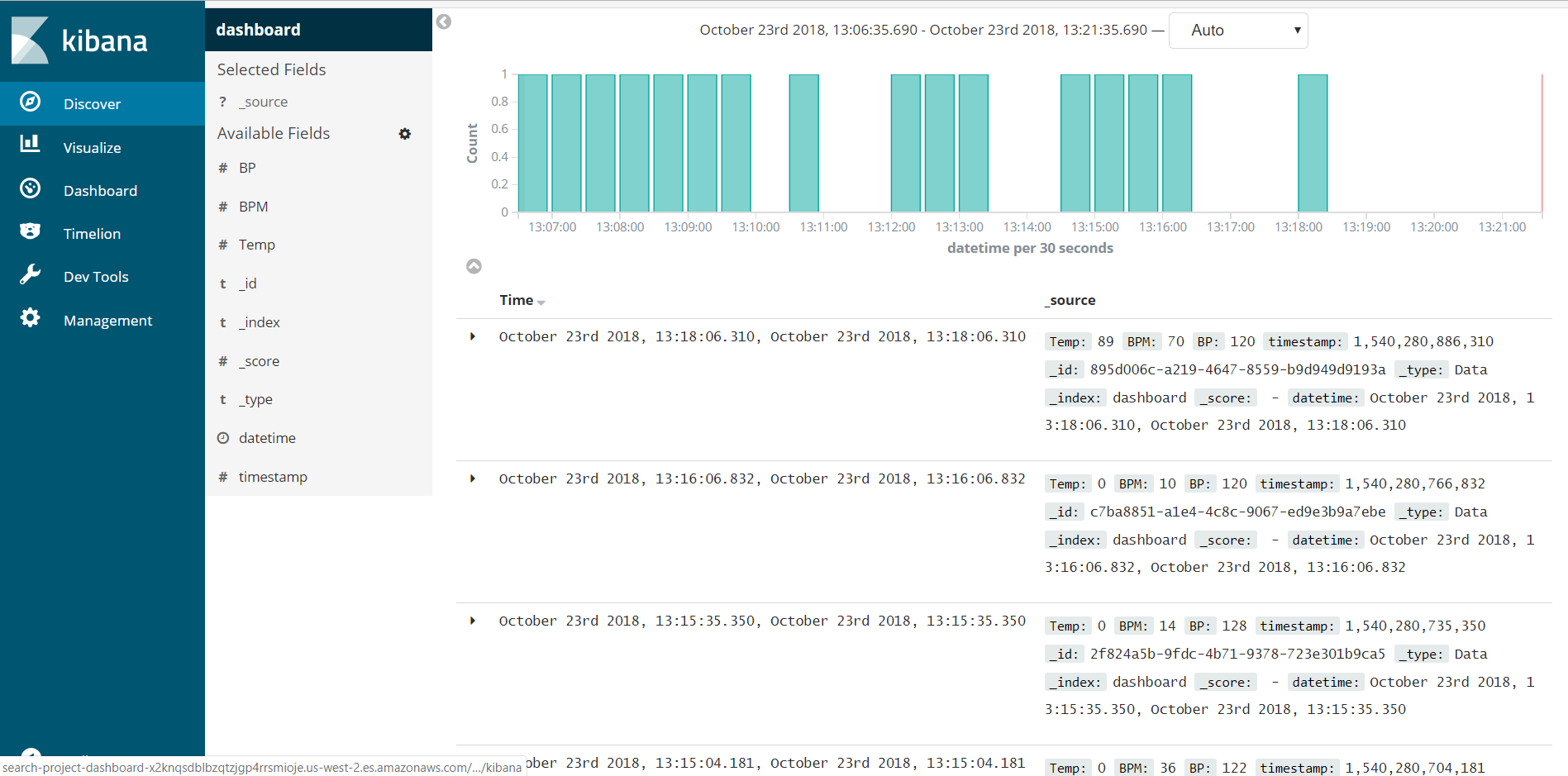
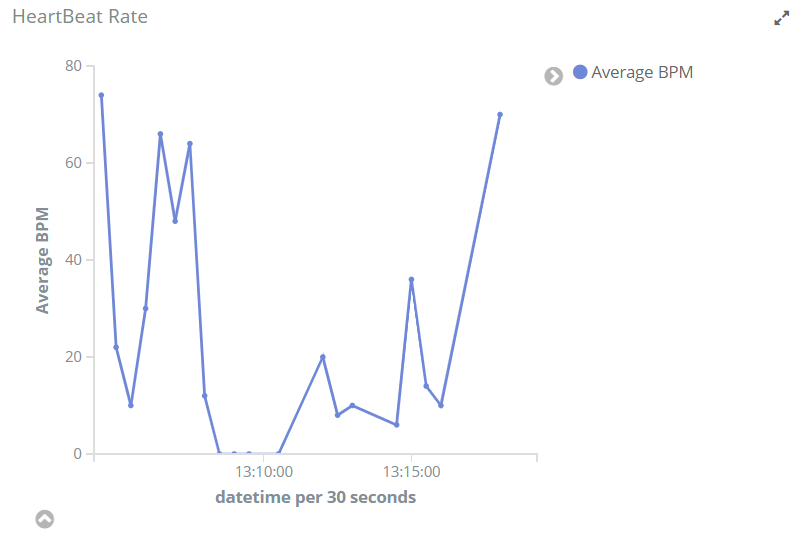
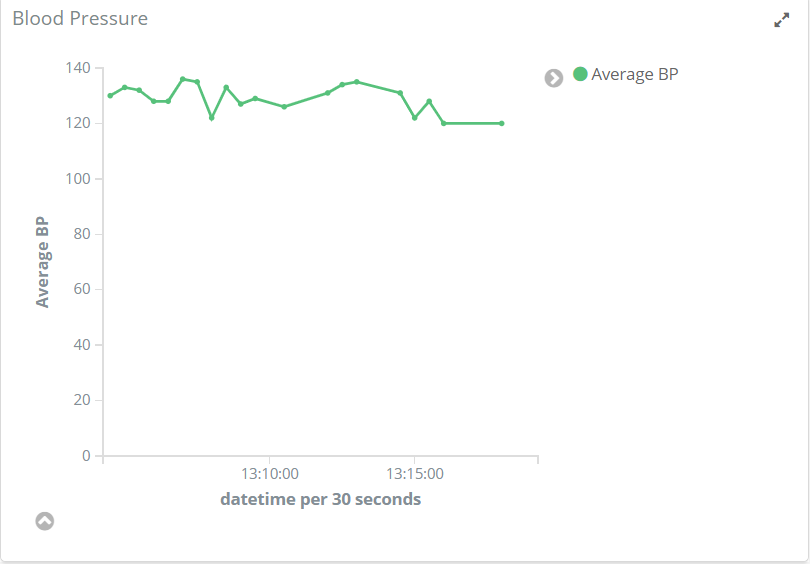
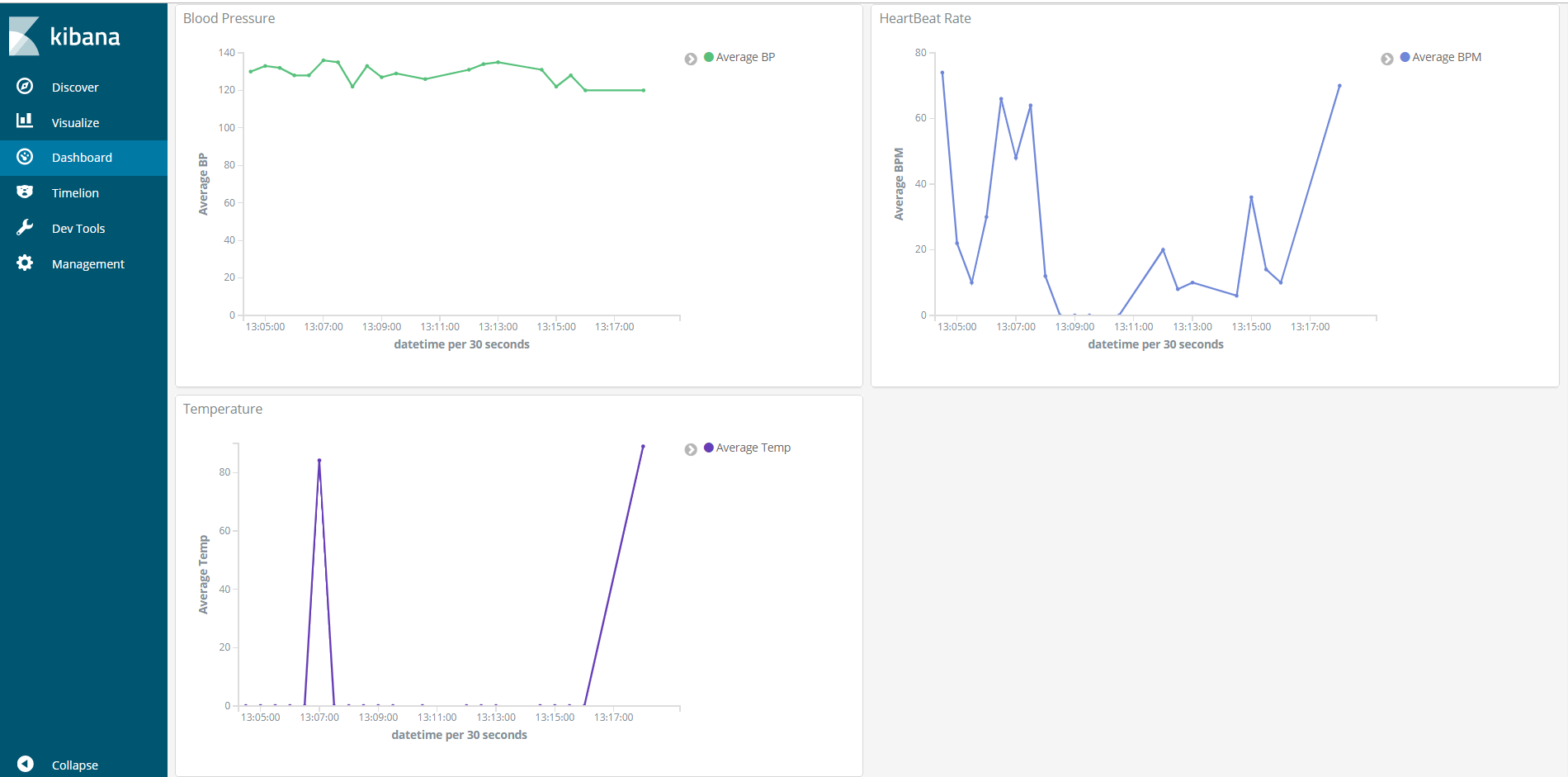
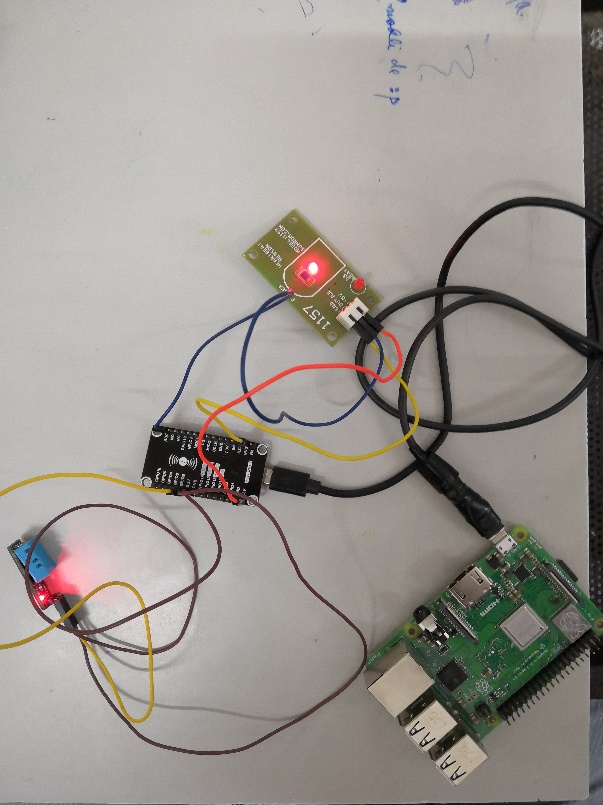
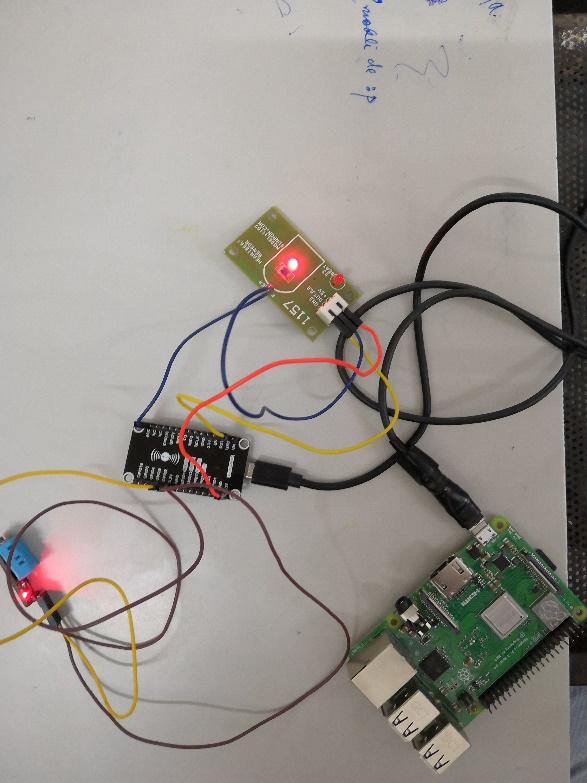
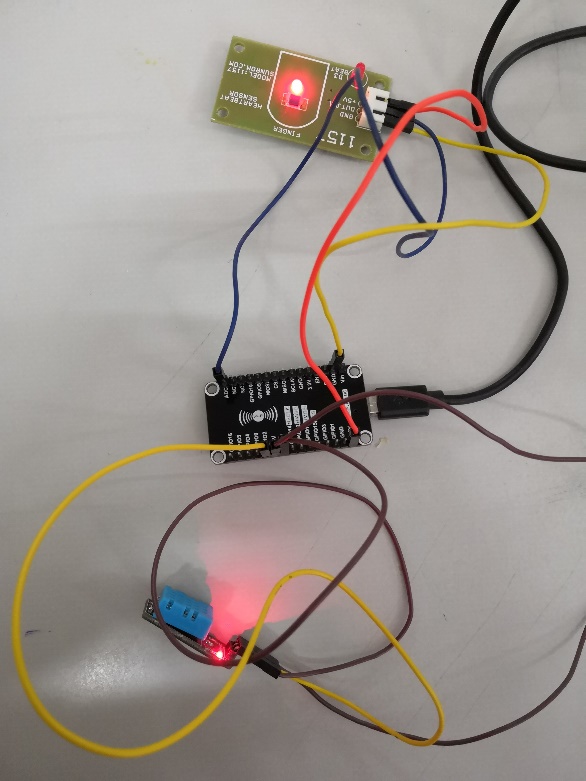
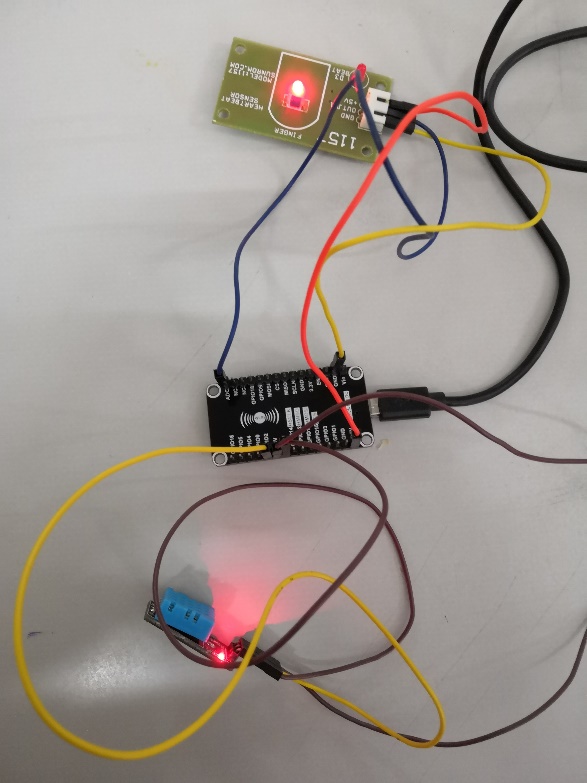
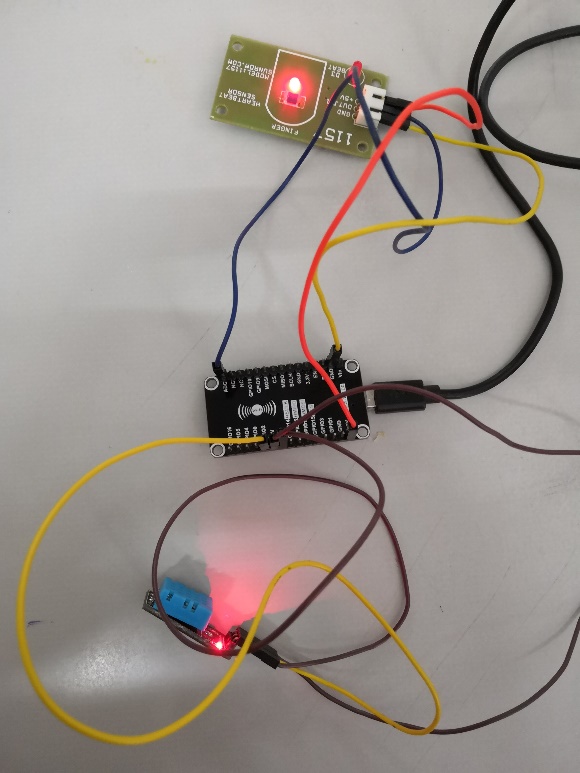
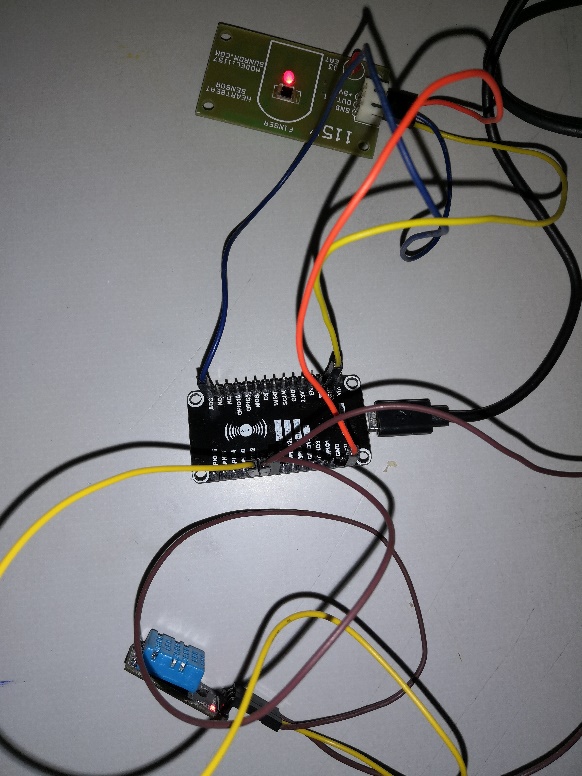
* This project can be used in the ambulances where the patient’s health can be monitored by the doctor present at the hospital.
* This can be done by the measuring the health related data of the patients using the various sensors that are attached with the ESP8266.
* ESP8266 then sends the data to the Raspberry Pi with the help of MQTT broker.
* The data received by the Pi is sent over to AWS IoT where it is stored and is made ready for visualization.
* For visualizing the data, the AWS services that are being used are ElasticSearch and Kibana.
* Once the data reaches Kibana, it is presented in the form of graphs and charts which can be inferred and analysed by the doctor easily.

**4. Implementation**

**4.1 CODING STANDARDS**

|  |
| --- |
| def on\_connect(client, userdata, flags, rc):  #print ("Connected with rc: " + str(rc))  clientconnected\_flag=True  client.subscribe("data/temp")    def on\_message(client, userdata, msg):  data=msg.payload    data= data.decode('UTF-8')  #j=json.loads(data)  #temp=j["Temp"]  #bpm=j["BPM"]  #bp=random.randint(120,139)  if True:  #payload = '{ "temperature": ' + str(temp) + ',"bpm": '+ str(bpm) + ',"bp":' + str(bp) +' }'  #print (payload)  print (data)  print(myMQTTClient.publish("$aws/things/raspberryPi/shadow/update", data, 0))  #sleep(4)    else:  print (".")  sleep(1) |

**4.2 SNAPSHOTS OF PROJECT**

****

**5. Future Enhancement**

While this project offers a lot of functionality, but to deploy it in the real-world environment, it still lacks some features such as GPS monitoring for current location of the ambulance, selection of the nearest hospital using maps and various machine learning algorithms, and analytics of the data collected so that we can infer learning outcomes from it and can be used to predict the diseases from the symptoms of the person. We would like to add the above mentioned features to our project and deploy the prototype in an ambulance and make it available for people.

**6. Conclusion**

The learning outcomes of this project have been fruitful. Starting this project with basic knowledge of Python, Embedded C, Raspberry Pi, we learnt these technologies in depth and got to learn new technologies such as various AWS services and deploying an IoT project into real-world environment. Besides that, we enhanced our experience in working under different learning and working conditions. Working towards a goal which started from a simple idea to building it to final stage and executing it made us realize the value a team brings to the table and responsibilities an individual possesses in a team.

This project also helped us realize how technology can be used to uplift the facilities that are already built by the most advanced species on the earth aka humans. Now we have basic idea about how we can implement various technologies to solve actual first world problems that were not possible before this digital age.

**7. References**

1. <https://aws.amazon.com/elasticsearch-service/>
2. <https://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html>
3. <https://opensource.com/resources/raspberry-pi>
4. <https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/>
5. https://www.quora.com/What-is-esp8266-wifi-module