

الجمهورية الجزائرية الديمقراطية الشعبية

وزارة التعليم العالي والبحث العلمي

UNIVERSITÉ BADJI MOKHTAR - ANNABA  
BADJI MOKHTAR – ANNABA UNIVERSITY



جامعة باجي مختار – عنابة

Faculté : Sciences de l'Ingénierat

Département : Electronique

Domaine : Sciences et Technologie

Filière : Télécommunications

Spécialité : Réseaux et Télécommunications

Mémoire

Présenté en vue de l'obtention du Diplôme de Master

**Thème:**

**Wearable IoT object using Raspberry Pi**

Présenté par : *LARBI Abdel Djilil et BENTERKI Brahim*

Encadrant : *FEZARI Mohamed Professeur Université Badji Mokhtar - Annaba*

**Jury de Soutenance :**

TAIBI Mahmoud	Professeur	U. B. M. Annaba	Président
FEZARI Mohamed	Professeur	U. B. M. Annaba	Encadrant
BOUTERAA Nadia	M. C.	U. B. M. Annaba	Examineur

Année Universitaire : 2019/2020

## Remerciements:

➤ *A Mes chers Parents, aucune dédicace ne saurait être assez éloquente pour exprimer ce que vous méritez, pour tous les sacrifices que vous n'avez cessé de me donner depuis ma naissance, durant mon enfance et même à l'âge adulte. Je vous dédie ce travail en témoignage de mon profond amour. Puisse Dieu, le tout puissant, vous préserver et vous accorder santé, longue vie et bonheur.*

➤ *A notre encadreur Mr. FEZARI Mohamed pour sa disponibilité et ses précieux conseils.*

➤ *Aux membres du jury qui nous ont fait l'honneur d'accepter de juger notre modeste travail.*

➤ *A mon binôme BENTIERKI Brahim et mes amis(es), vous avez toujours été présents pour les bons conseils.*

*Votre affection et votre soutien m'ont été d'un grand secours au Long de Mon parcours.*

➤ *A mes frères, ma sœur et toute ma grande famille, les mots ne suffisent guère pour exprimer l'attachement, l'amour et l'affection que je porte pour vous.*

➤ *A tous ceux qui, de près ou de loin, nous ont aidés à la réalisation de ce travail.*

*LARBI Abdeldjalil*

- *Mes parents*
- *Ma mère, qui a œuvré pour ma réussite, de par son amour, son soutien, tous les sacrifices consentis et ses précieux conseils, pour toute son assistance et sa présence dans ma vie, reçois à travers ce travail aussi modeste soit-il, l'expression de mes sentiments et de mon éternelle gratitude.*
- *Mon père, qui peut être fier et trouver ici le résultat de longues années de sacrifices et de privations pour m'aider à avancer dans la vie. Puisse Dieu faire en sorte que ce travail porte son fruit; Merci pour les valeurs nobles, l'éducation et le soutien permanent venu de toi.*
- *A mon frère Fakhri, ma petite sœur Salma je vous aime très fort.*
- *Aux membres du jury qui nous ont fait l'honneur d'accepter de juger notre modeste travail.*
- *A LARBI Abdeljalil qui a été mon binôme durant tout notre cursus.*
- *Je voudrais exprimer mes gratitudes à tous mes amis (es) qui m'ont apporté leur support moral.*

***BENTERKI Ibrahim***

## ملخص :

تعتبر انترنت الأشياء جيلا جديدا من الانترنتالذي يتيح التفاهم بين الأجهزة المترابطة مع بعضها (عبر بروتوكول الإنترنت) . وتشمل هذه الأجهزة الأدوات والمستشعرات والحساسات وأدوات الاصطناعي المختلفة وغيرها.

في هذا المشروع نحاول صنع خوذة ذكية بكفاءات جيدة و ربطها بشبكة انترنت محلية لتمكن المحارب داخل ساحة المعركة من الاتصال عن بعد المحطة القاعدية للجيش , هذه الأخيرة بإمكانها استقبال المعلومات و القيام بإرسال أوامر للجندي عبر هذه الخوذة في حالات الخطر .

## Résumé :

L'internet des objets est la nouvelle génération d'internet qui permet l'interconnexion des objets entre eux (par internet protocole), ces derniers comprennent des outils, des capteurs et divers outils d'intelligence artificielle, entre autres.

Dans ce projet, nous essayons de construire un casque intelligent et le reliant via un réseau internet local pour permettre au soldat à l'intérieur du champ de bataille de communiquer à distance avec la station de base de l'armée, cette dernière peut recevoir des informations et d'envoyer des ordres au soldat via ce casque en cas de danger.

## Abstract:

The Internet of Things is the new generation of the Internet that allows objects to be interconnected with each other (via internet protocol), the latter including tools, sensors and various artificial intelligence tools, among others.

In this project, we are trying to construct a smart helmet and connect it through a local internet network to allow the soldier inside the battlefield to communicate remotely with the army base station; the latter can receive information and send orders to the soldier via this helmet in case of danger.

### Chapter I:

Figure	Page	Description
1	4	different IoT systems
2	5	smart helmet of riders with camera
3	6	smart IoT system for old people
4	6	system of health monitoring and assisted living
5	7	wearable glove smart system

### Chapter II:

Figure	Page	Description
1	9	The first electronic schema of different hardware components shows how system
2	10	The final electronic schema of different hardware components
3	10	hardware synoptic of soldier's Wearable device
4	11	different ports of the Raspberry pi
5	12	comparison between raspberry pi and Arduino
6	14	construction of the smart helmet

### Chapter III:

Figure	Page	Description
1	20	installing VNC server on raspberry pi
2	21	configuration interface of the raspberry pi
3	21	updating Raspbian and installing Python3
4	22	enabling i2c protocol and Camera module
5	23	result of flask application
6	25	result of the web application using flask and Sqlite3

## Chapter IV:

Figure	Page	Description
1	27	The software Synoptic of wearable IoT helmet
2	28	Position of different sensors
3	29	Result of DHT22 test using Python3
4	29	Raspberry pi CSI port for the Camera v1.3
5	31	Result of the test of Gyroscope and accelerometer in 6-axis31
6	32	algorithm of the application
7	37	base station interface shows temperature, video in real time and control panel
8	39	Soldier movements on 6-axis using a simple 3D anthropomorphic in real time
9	40	FigIV.9-Standing position of the soldier
10	40	FigIV.10-Left slanted position of soldier

## 2-Abbreviation list:

<b>IoMT</b>	internet of Military Things
<b>IoT</b>	internet of things
<b>OS</b>	operating system
<b>MC</b>	microcontroller
<b>LED</b>	Light emitting diode
<b>SCL</b>	Serial clock Line
<b>SDA</b>	Serial Data Line
<b>GND</b>	Ground
<b>HTML</b>	Hypertext Markup Language
<b>CSS</b>	Cascading Style Sheets
<b>NPM</b>	Node package manager
<b>HTTP</b>	Hypertext Transfer Protocol
<b>IP</b>	internet protocol

## INDEX

<b>Remerciement</b> .....	<b>I</b>
<b>Abstract-résumé-ملخص</b> .....	<b>III</b>
<b>Figures list</b> .....	<b>IV</b>
<b>Abbreviations list</b> .....	<b>V</b>
<b>General introduction</b> .....	<b>1</b>
<b>Chapter I: Generalities</b> .....	<b>3</b>
I-Introduction .....	<b>4</b>
II-Stat of art .....	<b>4</b>
III-IOT projects Comparison table.....	<b>5</b>
III.1-Smart helmet for riders.....	<b>5</b>
III.2-Smart helmet for accidents.....	<b>5</b>
III.3-Wearable sensor and IOT for better medicine FIT-WIT.....	<b>6</b>
III.4-IOT based health monitoring for active and assisted living.....	<b>6</b>
III.5-Foot wear based wearable system smart shoe.....	<b>7</b>
III.6-Wearable glove system.....	<b>7</b>
IV-Conclusion .....	<b>7</b>
<b>Chapter II: Helmet components; hardware description and design</b> .....	<b>8</b>
I- Problematic and solution .....	<b>9</b>
II- Electronic schema.....	<b>9</b>
III-Synoptic.....	<b>10</b>
IV-Hardware description .....	<b>11</b>
IV.1-Raspberry Pi.....	<b>11</b>
IV.1.1-Definition.....	<b>11</b>
IV.1.2- Raspberry Pi choice over Arduino and other Micro controllers...	<b>12</b>
IV.2-CloudIoTCore.....	<b>12</b>
IV.3-actuators.....	<b>12</b>
IV.3.1-Buzzer.....	<b>12</b>

IV.3.2-LED.....	12
IV.4-Sensors.....	13
IV.4.1-Accelerometer and Gyroscope MPU-6050.....	13
IV.4.2-Temperature and Humidity Sensor DHT 22.....	13
IV.4.3-Raspberry Pi Camera V1.3.....	13
V-Project hardware realization.....	14
VI-conclusion.....	14
<b>CHAPTER III: Helmet components; software description and tests .....</b>	<b>15</b>
I-Introduction.....	16
II-Software definition.....	17
II.1-Rasbian: .....	17
II.2-Python.....	17
II.3-Flask.....	17
II.4-SQLite.....	18
II.5- HTML,CSS, JavaScript , Node.js and Npm.....	18
II.5.1-HTML: Hypertext Markup Language .....	18
II.5.2-CSS: Cascading Style Sheets .....	19
II.5.3- Java Script.....	19
II.5.4-Node.js.....	19
II.5.5-Npm.....	19
III-Procedure steps.....	20
III.1-Rasbpian installation.....	20
III.2-Python3 installation.....	21
III.3-Enabling camera and i2c module.....	22
III.4-Install Adafruit Library for Dht22.....	22
III.5-Flask installation.....	22
III.6-SQLite local storage installation and configuration.....	24
III.7-Server, Data base and web page creation.....	24
III.8-Local Network test.....	24



IV-Conclusion.....	25
<b>Chapter IV:Realization of helmet application, controlled by a web server.....</b>	<b>26</b>
I-Introduction: .....	27
II-Sensors tests.....	28
II.1-Temperature and humidity sensor DHT22.....	28
II.2-The Camera Module.....	29
II.3-Accelerometer and Gyroscope i2c MPU-6050 .....	30
III-Organizational Chart.....	32
IV-APPLICATION CODE.....	33
IV.1-First Part: Data Base creation's and rumpling's Script.....	33
IV.1.1-Data base creation.....	33
IV.1.2-Store DHT22 data into our table's database.....	34
IV.1.3-Camera Streaming .....	35
IV.2-Second Part: Lunching web application's Script and camera and Data representing.....	35
IV.2.1-Flask application including camera, DHT22data and led control in case of danger.....	35
IV.2.2-HTML AND CSS CODES.....	36
IV.2.2.1-HTML code for the web server.....	36
IV.2.2.2-CSS code.....	37
IV.3-Third Part: Lunching Nodes JS Application and 3D simulator's Script.....	38
IV.3.1.Lunching the node.js application and show movement of the accelerometer and gyroscope.....	38
IV.3.2-Install Node.js and packages.....	38
IV.3.3 Node.js script.....	39
IV.3.4-Result and discussions.....	40
V-Conclusion.....	40
<b>General Conclusion.....</b>	<b>42</b>
<b>ANNEX.....</b>	<b>43</b>
<b>References.....</b>	<b>50</b>

## General introduction

With the development of the Internet of Things, soldiers have become key nodes of Information collection and resource control on the battlefield. It has become a trend to Develop wearable devices with diverse functions for the military. The interconnection and Collaborative decision-making between combat equipment and battlefield resources is one of the characteristics of the Internet of Military Things (IoMT). The next generation of military Networks will consist of densely deployed battlefield nodes, including human wearable Devices like smart helmet, gilet, shoes and gloves. Command, control, communications, and Intelligence systems will accelerate integration with the IoMT to influence military decision Making in future wars.

Different smart sensors integrated into a soldier's equipment provide a command center with Multidimensional battlefield information Wearable sensors is the basic elements of military Smart devices.

In our study we will be interested in this new trend of interconnected objects used to improve the state of the soldiers in the battlefield. For this, we need to assure security, health and dangers surrounding soldiers which are one of the major concerns of this project. Sensors implemented in wearable smart devices will help to collect information's and send it to the base station , soldiers using smart helmets, Gilets ,Gloves and suits will be connected and controlled to avoid any type of dangers ahead, also it can be controller remotely by military station base.

# General introduction

---

The first chapter presents general introduction about harness the internet of things in military Domain, and state of art; different projects of wearable IoT devices in health care and Military domain and a comparative table contain information's about it.

The second chapter carries about problematic and her proposed solution about the smart wearable devices, in addition to synoptic and different sensors used in this project with details.

The Third chapter consists of different information about software synoptic and the OS used in Raspberry PI, the algorithm used to link different sensors and actuators with the MC and The Python Script of the web page used to collect information From Raspberry Pi.

The Fourth chapter will present tests of algorithm application results given by different hardware parts, data and video shown in monitor, discussion and comments about the different results.

Finally, a general conclusion gives the evaluation performance of our system, ends by possible ameliorations that we can provide to the project to improve the performances and expand it in other application domain.

# **Chapter I: Generalities**

## I-Introduction:

Internet of Things (IoT) technology is the extension of internet network of things or objects. Items then become connected and networked, such as connected watches, gloves and shoes... The Internet of Things is growing up, benefiting from the creation of the Cloud Computing therefore it can be applied in various fields. In this part, we will describe how IoT works, we will mention and describe some IoT projects and compare them.

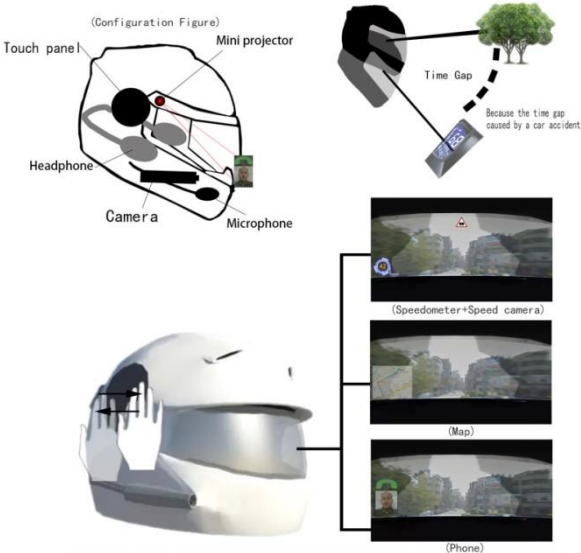
## II-Stat of art:

Project	University/Labo	MC	Sensors	Application	Date
Smart helmet for riders ( Drunken Drive ) [1]	IJAREEIE	Raspberry PI	- Position Sensor - Slot Sensor - Alcohol Sensor - Vibration Sensor	Provide security and safety for riders against accidents	2019
Smart helmet for accidents with Bluetooth and WIFI[2]	IJPAM	Raspberry PI	-Mobile GSM -Speed limit -Temperature sensor -Position sensor ( Bluetooth or WIFI)	-Give notification of the accident area -Encourage driver to ride safe	2018
Wearable sensor and IOT for better medicine FIT-WIT[3]	SPCIJET	Intel Gennino 101 board	-LM35 -pulse rate -Grove-gas sensor -PPG	Health monitoring temperature -heart rate stat -blood pressure and oxygen in blood.	2019
IOT based health monitoring for active and assisted living[4]	Central Michigan University	Raspberry PI	-pulse oximeter -ECG -oral airflow -Temperature Sensor	Wearable IOT system for active and living health care application	2017
Foot wear based wearable system smart shoe[5]	MDPI	Raspberry PI	-Accelerometer -Pressure sensor -GPS -Vibration -Tri-axial force	- activity sports -Bodyweight -Navigation	2016
Wearable glove system[6]	University of Nottingham Ningbo China	ATMega32U4	-PPG -IMU Orientation -Accelerometer -Magnetometer -Gyroscope	Driving safety	2016

Fig I.1-different IoT systems

**III-IOT projects Comparison table:**

**III.1-Smart helmet for riders:** The conventional helmet is used for the safety of driver’s head. It does not serve any other purposes in case of any untimely accidents. The major cause for the loss of lives in accidents is due to the delay to reach the hospital. A smart helmet is a special idea which makes motorcycle driving safer than before.



**Fig I.2-smart helmet of riders with camera**

**III.2-Smart helmet for accidents:** This helmet makes rider to feel comfortable as well as with high protection and security. This smart helmet works on raspberry pi 3controller which is WIFI based, acts as a station for the networking system. Bluetooth and raspberry pi 3 was interfaced with cloud based services.

**III.3-Wearable sensor and IOT for better medicine FIT-WIT:**

E-health provides information on disease prevention, detecting early symptoms, and monitoring the patient's condition based on medical parameters from a far distance. Internet of things became the main concept in this system, which combines wearable sensors, communication systems, and mobile user interfaces. Reliable and valid system, easily carried, help the doctor to monitor patients from far distance expectantly to overcome the problems.

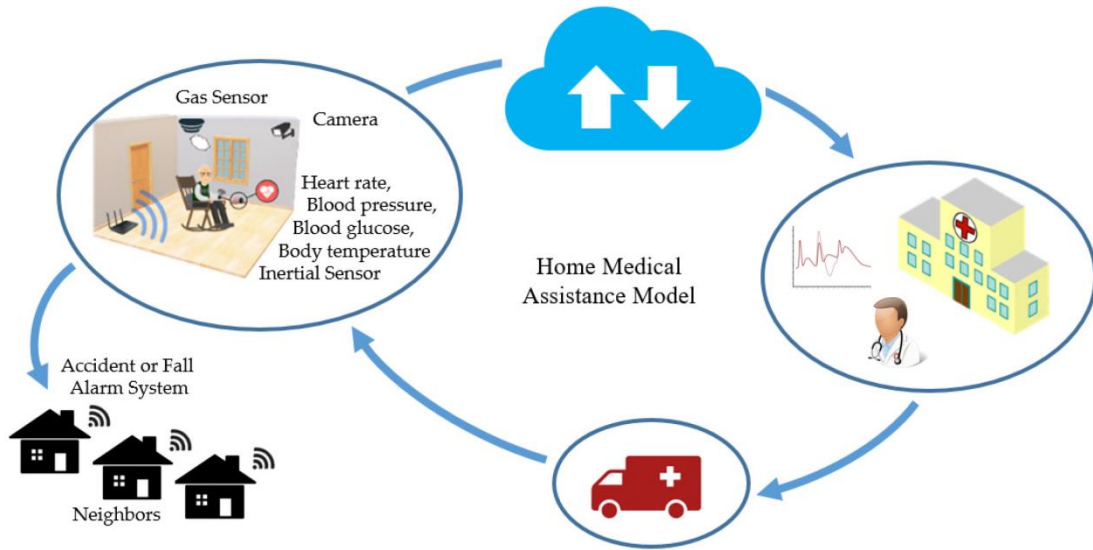


Fig I.3-smart IoT system for old people

**III.4-IOT based health monitoring for active and assisted living:**

Health monitoring for active and assisted living is one of the paradigms that can use the IoT advantages to improve the elderly lifestyle.

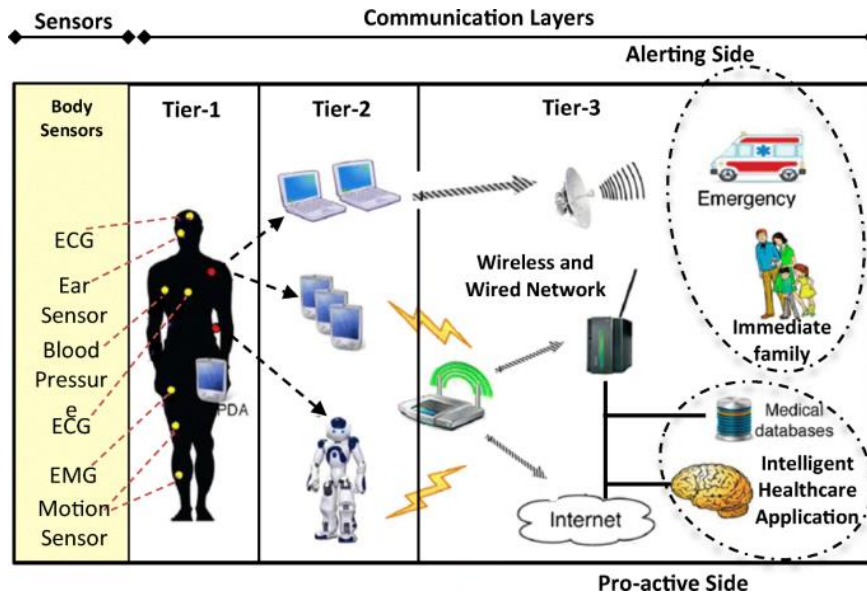


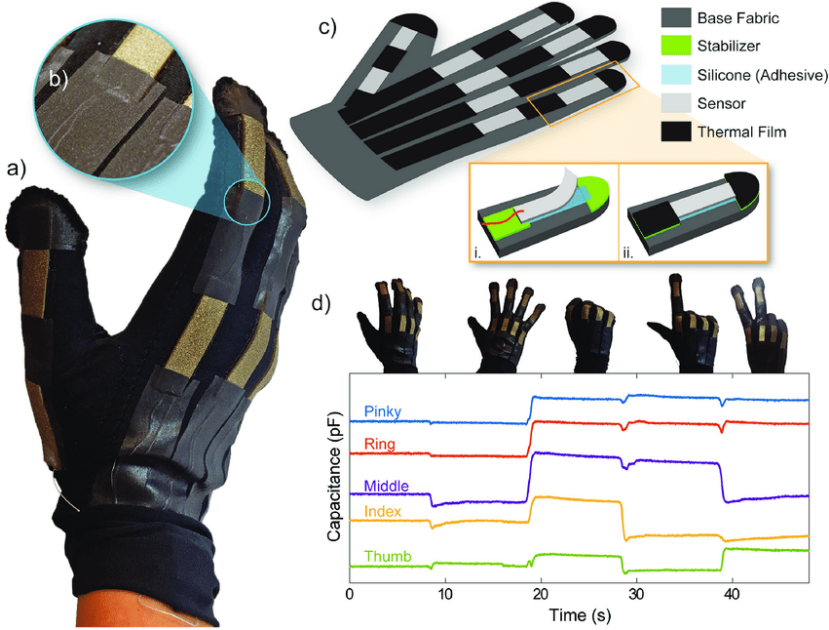
Fig I.4-IoT system of health monitoring and assisted living

**III.5-Foot wear based wearable system smart shoe:**

Footwear is an integral part of daily life. Embedding sensors and electronics in footwear for various different applications, their systems range from simple step counting solutions to more advanced systems intended for use in rehabilitation programs for disabled subjects.

**III.6-Wearable glove system:**

Develop a system to detect driver stress events in real time. The driver’s stress is estimated by the use of physiological signals and steering wheel motion analysis. The steering wheel motion is analyzed by driver’s hand moving characteristic.



**Fig I.5 -wearable glove smart system**

**IV-Conclusion:**

In this chapter we have seen some different IoT project and we have compared them especially in terms of application and the sensors it contains, we can notice clearly that the majority of projects focus on medicine and sports domain but we rarely find it in military one, so we decided to work with our project in this field, and that what we going to do in next chapters.



# **Chapter II: Helmet components; hardware description and design**

## I- Problematic and solution:

In any country, soldiers are always the frontline defense. Nevertheless, they always give their best by defending the nation by risking their lives. It is always the case that a soldier is injured on a battlefield barbarously. Threats surrounding soldiers and stuffs used like the Gilet or Helmet don't assure full protection, because of this reason they are supposed to compromise about the security of their lives

The objective of this project is to provide an intelligent security system to help soldiers dominate the battlefield with connected wearable objects of defense (helmet, Gilet, and suit), providing security by monitoring the soldier health parameters and threats detection system designed in it by using IOT

The goal here is to show the facilities offered by such an intelligent system.

## II- Electronic schema:

We made this first initial schema for the first time but we couldn't find all sensors so we replaced some of them. We didn't use a pulse rate sensor because of non-availability of components, and because there is no digital input in the raspberry pi we needed an ADC chip so we replaced LM35 by a DHT22 Humidity and temperature sensor.

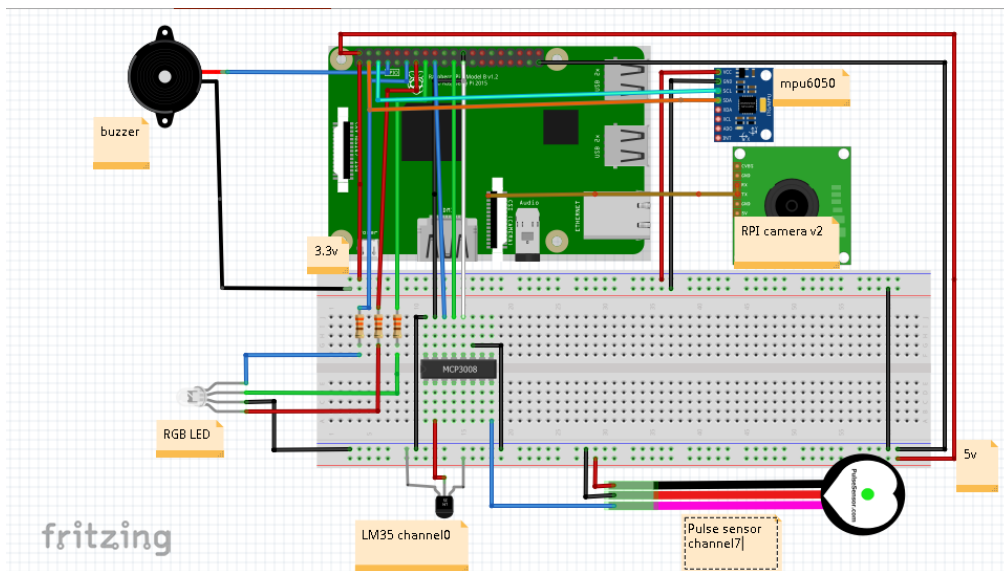
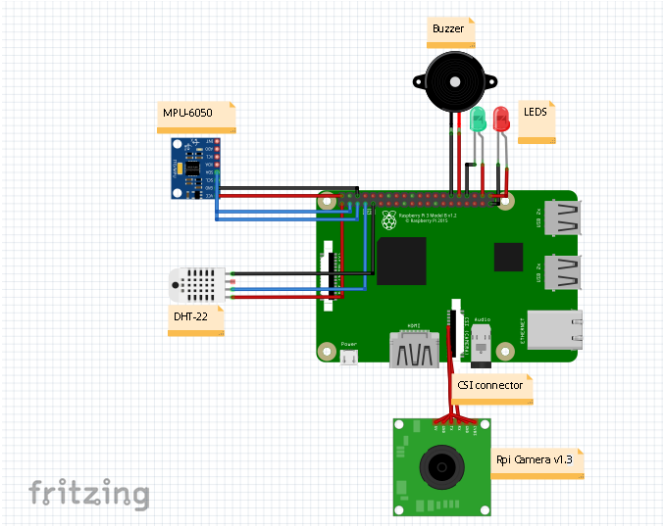
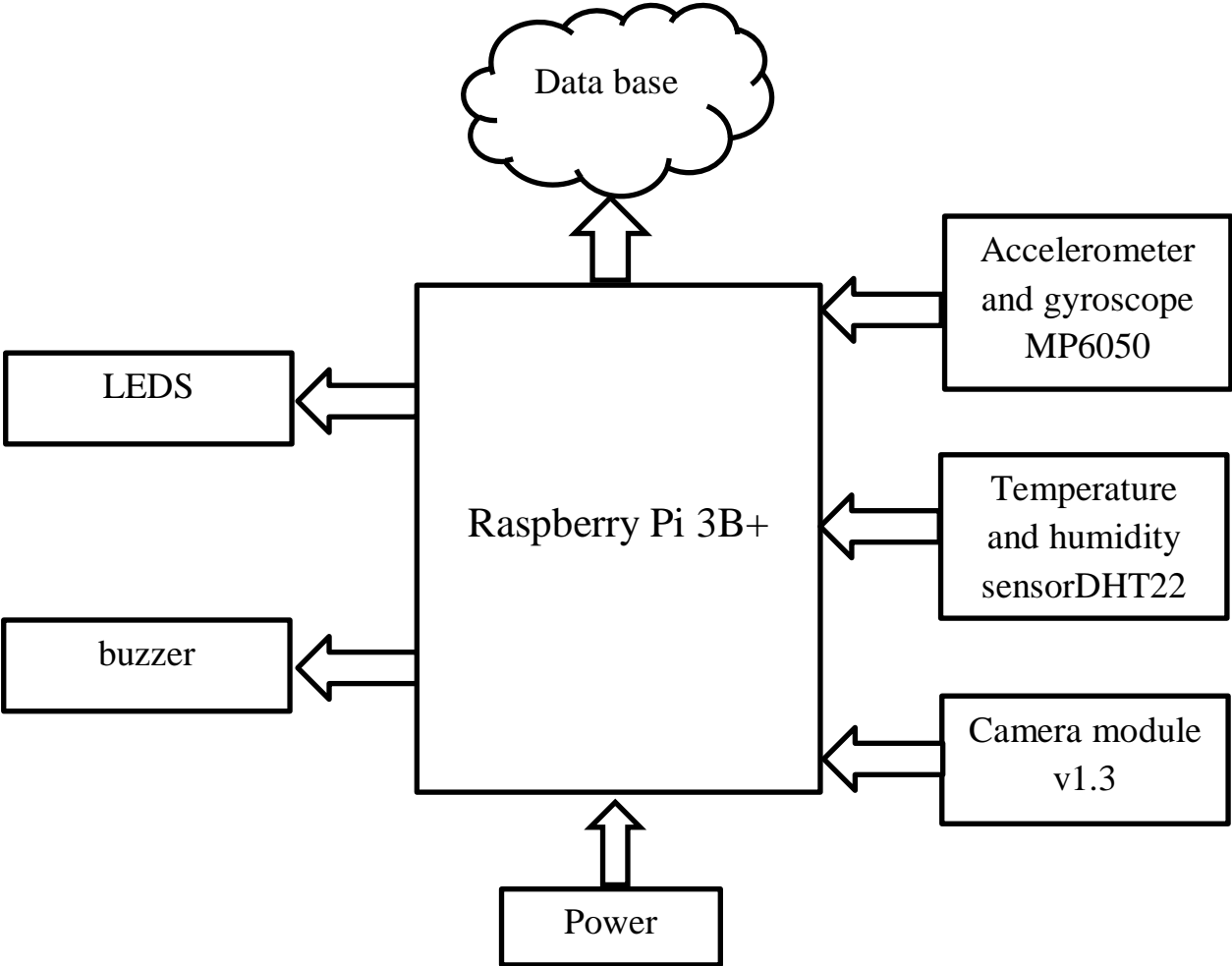


Fig II.1- The first electronic schema of different hardware components



FigII.2-The final electronic schema of different hardware components

III-Synoptic:

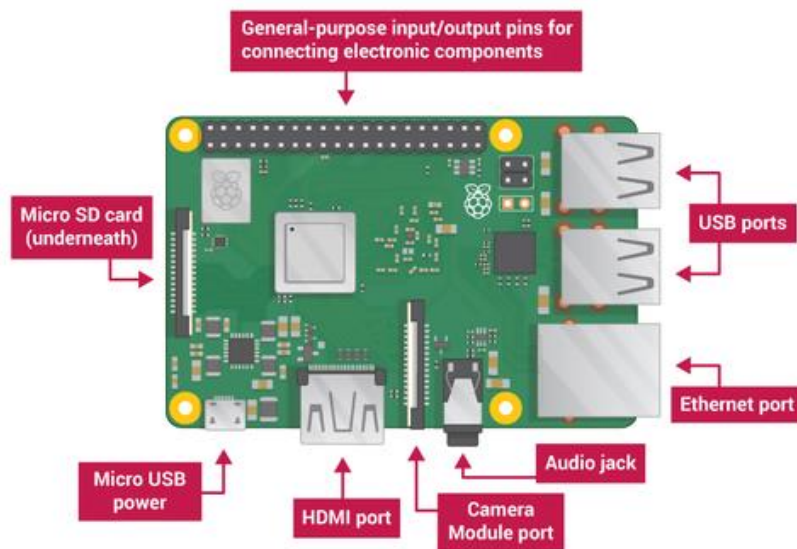


FigII.3 - hardware synoptic of soldier's Wearable device

### IV-Hardware description:

#### IV.1-Raspberry Pi:

**IV.1.1-Definition:** is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote teaching of basic computer science in schools and in developing countries. The original model became far more popular than anticipated, selling outside its target market for uses such as robotics. It does not include peripherals or cases. However, some accessories have been included in several official and unofficial bundles. The organization behind the Raspberry Pi consists of two arms. The first two models were developed by the Raspberry Pi Foundation. After the Pi Model B was released, the Foundation set up Raspberry Pi Trading, with Eben Upton as CEO, to develop the third model, the B+.<sup>[7]</sup>



**Fig II.4 -different ports of the Raspberry pi**

#### IV.1.2-Raspberry Pi choice over Arduino and other Micro controllers:

Raspberry Pi and Arduino are quite different boards. Each board has its own advantages and disadvantages, one of the main advantages that made us to choose the first one ( RPI ) is powerfulness, in which he capable of doing multiple tasks at a time like a computer and this is the important thing that we need in this project where sensors need to be controlled from a web page over internet, also Pi can be converted into a webserver and data base, beside that Raspberry pi is 40 times faster than Arduino also as we have stated earlier that it has memory, processor, USB ports, Ethernet port etc...

In term of networking Raspberry Pi has the built in Ethernet port, through which you can directly connect to the networks. Even Internet can easily be run on Pi using some USB Wi-Fi dongles. While in Arduino, it's very difficult to connect to network. External hardware's need to be connected and properly addressed using code, to run network using Arduino. External

Boards called “**Shields**” needs to be plugged in, to make Arduino, as functional as Pi, with a proper coding to handle them.

In addition to all this, the table below will show the others advantages of Raspberry Pi over Arduino:

<b>Raspberry Pi 3</b>	<b>Arduino</b>
It is a mini computer with Raspbian OS. It can run multiple programs at a time.	Arduino is a microcontroller, which is a part of the computer. It runs only one program again and again.
It requires complex tasks like installing libraries and software for interfacing sensors and other components	It is very simple to interface sensors and other electronic components to Arduino.
Raspberry Pi can be easily connected to the internet using Ethernet port and USB Wi-Fi dongles.	Arduino requires external hardware to connect to the internet and this hardware is addressed properly using code.
Raspberry Pi has 4 USB ports to connect different devices.	Arduino has only one USB port to connect to the computer.
The Recommended programming language is python but C, C++, Python, ruby are pre-installed.	Arduino uses C/C++.
512 MB memory	0.002 MB Memory
700 MHz Clock Speed	16 MHz clock speed

**FigII.5-comparison between raspberry pi and Arduino**

**IV.2-CloudIoTCore:** our IoT devices, Helmet and Gilet need storage to save information and make data processing, which needs to be stored in data base better than device itself. In our case we will use a local one.

### **IV.3-actuators:**

**IV.3.1-Buzzer:** the buzzer has necessary role in the soldier helmet it will make a warning alarm if he is not in a secure position in the battlefield those tones of the buzzer depend on the stat of the temperature sensor DHT22. Also the buzzer is mainly used to emit a sound when temperature is more than 37.5.

**IV.3.2-LED:** two different LED colors are used in the smart helmet to notify the soldier automatically in danger's case, in which the lights switching between white and red in specific temperature conditions that will be clarified later, also it used by controllers in base station to order the soldier in battlefield remotely via IOT local network considering and notify him from a possible danger considering that they have the possibility to see the back view of soldier via helmet camera.

### IV.4-Sensors:

**IV.4.1-Accelerometer and Gyroscope MPU-6050:** is a device that measure acceleration, which is the rate of change of the velocity of an object. They measure in meters per second squared ( $m/s^2$ ) or in G-forces (g). Accelerometers can measure acceleration on 3-axis. The MPU6050 consist of 3-axis Gyroscope/Accelerometer with Micro Electro Mechanical System (MEMS) technology.

MPU6050 sensor module is complete 6-axis Motion Tracking Device. It combines 3-axis Gyroscope, 3-axis Accelerometer and Digital Motion Processor all in small package. It has I2C bus interface to communicate with the microcontrollers.

The accelerometer sensor helps to calculate the acceleration, the position of the soldier and his state when he is walking, crouching or laying down when variation of the acceleration in the 3 axes is big that will detects the true state of the soldier in battle field. The data will be sent to the BS so they can treat precaution.<sup>[8]</sup>

The accelerometer has 8 pins but in this case we will use only 4:

**SCL:** Serial Clock pin. Connect this pin to RPI SCL pin.

**SDA:** Serial Data pin. Connect this pin to RPI SDA pin.

**GND:** Ground pin. Connect this pin to ground connection.

**VCC:** Power supply pin. Connect this pin to +5V DC supply.

**IV.4.2-Temperature and Humidity Sensor DHT 22:** there are too many sensors that we can use with the raspberry pi to measure the temperature of the soldier in the battle field.

The DHT22 is a versatile and low-cost Temperature sensor that can also calculate the temperature of an area.

This sensor has a relatively long transmission distance, allowing the sensor to transmit data through wires up to 20m away from the Raspberry Pi.

As a bonus, the DHT22 is a digital sensor with an inbuilt analog to digital converter. The converter makes it a lot easier to connect the sensor to the Raspberry Pi as you do not need to deal with any additional chips.<sup>[9]</sup>

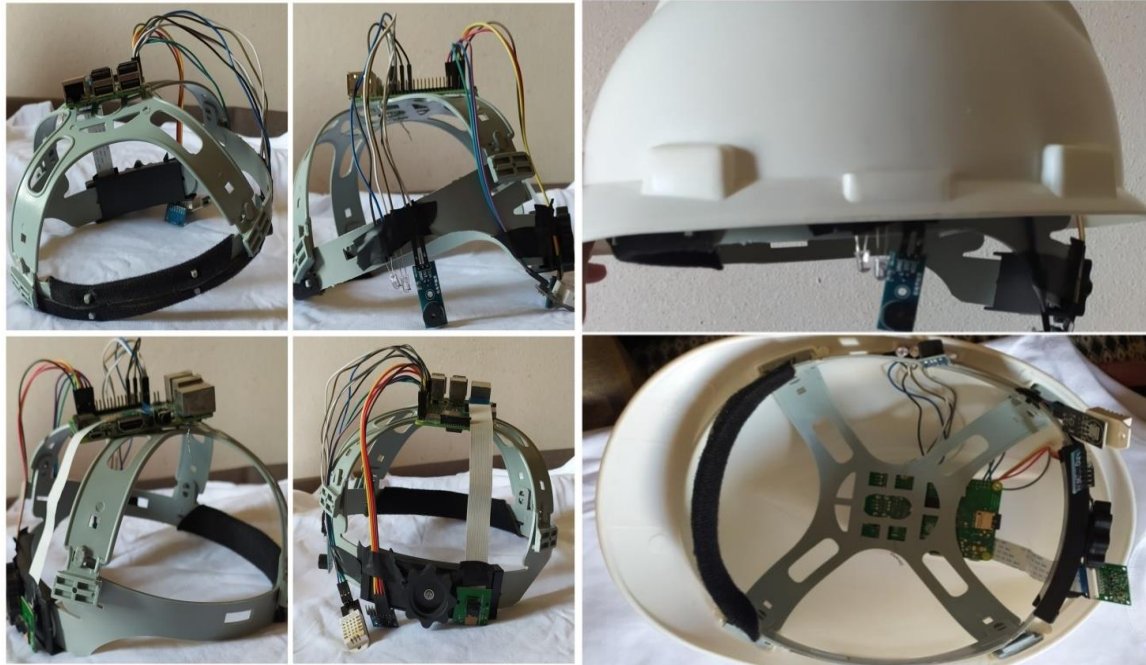
**IV.4.3-Raspberry Pi Camera V1.3:** is a High Definition camera module compatible with all Raspberry Pi models. The camera module connects to the Raspberry Pi board via the CSI connector.

The camera V1.3it's able to deliver a crystal clear 5MP resolution image or 1080p HD video recording at 30fps.

The Camera used in the project by placing it behind the helmet to provide vision to base station a back view of the soldier on which will be directed and alerted to possible danger.

### V-Project hardware realization:

After putting all components together we made the helmet of the soldier



**FigII.6-construction of the smart helmet**

### VI-conclusion:

In this chapter we have focused on hardware side in which we have defined the available sensors which we can use them with specific software and algorithms to solve the problematic presented above.

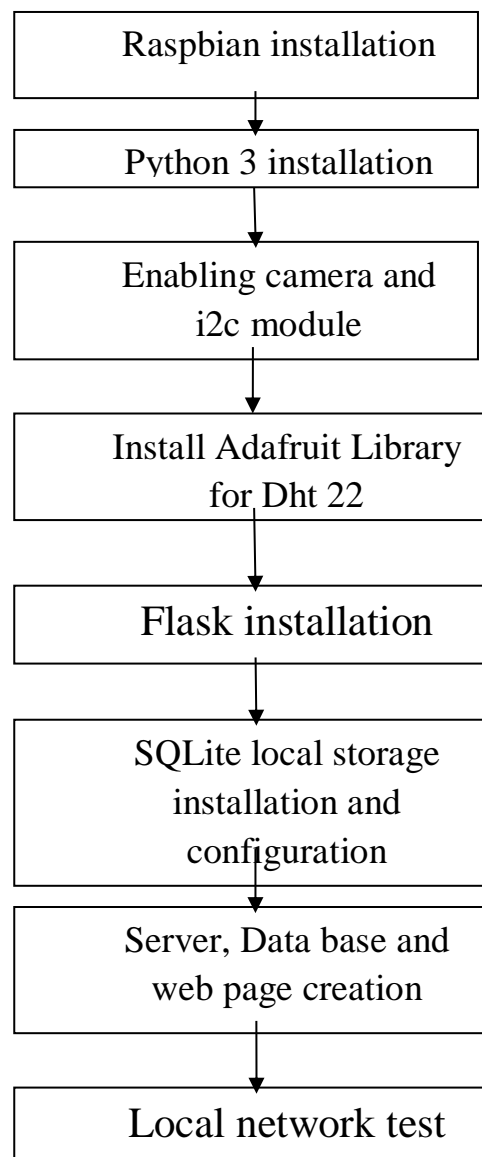
**CHAPTER III:  
Helmet components;  
software description  
and tests**



### **I-introduction:**

this chapter focus on application , we will show in detailed way the planning and methods that we used to build an local IoT network consisting of server, data base and web page using raspberry pi 3 B+ , python , Flask, SQLite , Html , CSS ,JavaScript, Node.js and npm following procedure bellow :

### **Procedure organizational chart:**



### **II-Software definition:**

#### **II.1-Rasbian:**

is a free operating system based on Debian optimized for the Raspberry Pi hardware. An operating system is the set of basic programs and utilities that make your Raspberry Pi run. However, Raspbian provides more than a pure OS: it comes with over 35,000 packages; pre-compiled software bundled in a nice format for easy installation on your Raspberry Pi.

The initial build of over 35,000 Raspbian packages, optimized for best performance on the Raspberry Pi, was completed in June of 2012. However, Raspbian is still under active development with an emphasis on improving the stability and performance of as many Debian packages as possible. <sup>[10]</sup>

#### **II.2-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.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective. <sup>[11]</sup>

**II.3-Flask:** This HTTP server has been developed in Python3, using a popular framework called Flask; it is classified as a micro framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, and upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more frequently than the core Flask program. <sup>[12]</sup>

### **Flask choice:**

There are two well-known web frameworks developed for Python: aiohttp and Flask. We chose the latter because it included more interesting login support, but the project could have been developed with either of those. Actually, Flask is a micro framework, but it supports extensions for extra functionalities. It is also interesting for this project the fact that it uses html templates, which will be explained later.

**II.4-SQLite:** is a relational database management system (RDBMS) contained in a Library. In contrast to many other database management systems, SQLite is not a client–server database engine. Rather, it is embedded into the end program.

SQLite is ACID-compliant and implements most of the SQL standard, generally following PostgreSQL syntax. However, SQLite uses a dynamically- and weakly-typed SQL syntax that does not guarantee the domain integrity. This means that one can, for example, insert a string into a column defined as an integer. SQLite will attempt to convert data between formats where appropriate, the string "123" into an integer in this case, but does not guarantee such conversions, and will store the data as-is if such a conversion is not possible.

There are many options in the market and probably the 2 most used with Raspberry Pi and sensors are MySQL and SQLite. Here, SQLite is probably the most suitable choice, because it is server less, lightweight, open source and supports most SQL code.

### **II.5- HTML, CSS, JavaScript, Node.js and Npm:**

We have to give instructions to the computer. It's not enough just type the text that should appear on the screen (as we would in a treatment text); we must also indicate where to place this text, insert images, make links between pages, etc.

**II.5.1-HTML: Hypertext Markup Language** is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by *tags*, written using angle brackets. Tags such as `<img/>` and `<input/>` directly introduce content into the page. Other tags such as `<p>` surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), former maintainer of the HTML and

current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.[13]

**II.5.2-CSS: Cascading Style Sheets** is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separated file, and reduce complexity and repetition in the structural content.

Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device.

The name *cascading* comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable. <sup>[14]</sup>

**II.5.3- Java Script:** JavaScript is a text-based programming language used both on the client-side and server-side that allows you to make web pages interactive. Where HTML and CSS are languages that give structure and style to web pages, JavaScript gives web pages interactive elements that engage a user.

Incorporating JavaScript improves the user experience of the web page by converting it from a static page into an interactive one. To recap, JavaScript adds **behavior** to web pages.

JavaScript is mainly used for web-based applications and web browsers. But JavaScript is also used beyond the Web in software, servers and embedded hardware controls. <sup>[15]</sup>

**II.5.4-Node.js:** is an open-source, cross-platform, JavaScript runtime environment (Framework) that executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web-application development around a single programming language, rather than different languages for server- and client-side scripts. <sup>[16]</sup>

**II.5.5-Npm:** is a package manager for the JavaScript programming language. It is the default package manager for the JavaScript runtime environment Node.js. It consists of a command line client, also called npm, and an online database of public and paid-for private packages, called the npm registry. The registry is accessed via the client, and the available packages can be browsed and searched via the npm website. The package manager and the registry are managed by npm, Inc.

## III-Procedure steps:

### III.1-Raspbian installation:

#### Step 1: Formatting the microSD card:

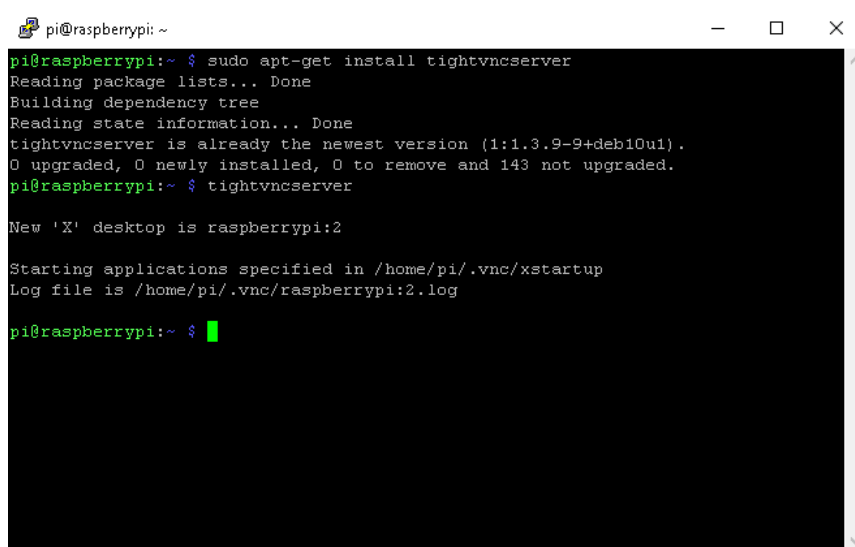
The initial step was to reformat the microSD, because mostly new microSD cards come with some irrelevant files on them. Therefore reformatting it completely deletes and clears all files on the microSD.

#### Step 2: Download and installing Raspbian

Raspbian was then downloaded and installed onto the microSD using a disk imager. Once it this was done, the microSD card was plugged it into the Raspberry Pi. Once the installation was completed the Raspbian automatically began to boot. The operating system was then configured.

#### Step 3: Initializing Raspberry Pi

The microSD card was inserted into the card socket of the raspberry Pi 3b+. An RJ45 Ethernet cable was used to connect the Raspberry Pi to the laptop. Opening the SSH port with putty and install the VNC Server on it then we used a viewer to show Raspberry pi desktop using the local IP address of raspberry pi. Alternatively, Wi-Fi adapter of the Raspberry Pi could have been used to connect the system to the network The 5.1 volts 2.5 amps micro USB power supply was used power up the Raspberry Pi to boot it.



```
pi@raspberrypi: ~  
pi@raspberrypi:~$ sudo apt-get install tightvncserver  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
tightvncserver is already the newest version (1:1.3.9-9+deb10u1).  
0 upgraded, 0 newly installed, 0 to remove and 143 not upgraded.  
pi@raspberrypi:~$ tightvncserver  
New 'X' desktop is raspberrypi:2  
Starting applications specified in /home/pi/.vnc/xstartup  
Log file is /home/pi/.vnc/raspberrypi:2.log  
pi@raspberrypi:~$ █
```

Fig-III.1- installing VNC server on raspberry pi

#### Step 5: Configuration of Raspberry Pi 3

Upon completion of the boot process, the location, date, and time is done were configured to suit the local setting using the command:

```
pi@raspberrypi:~$ sudo raspi-config
```

-After reconfiguring the Raspberry Pi, the system was rebooted. Once the Raspberry Pi was restarted, it was ready to be used.

```
Raspberry Pi 3 Model B Plus Rev 1.3
Raspberry Pi Software Configuration Tool (raspi-config)
1 Change User Password Change password for the 'pi' user
2 Network Options       Configure network settings
3 Boot Options          Configure options for start-up
4 Localisation Options Set up language and regional settings to match your
5 Interfacing Options  Configure connections to peripherals
6 Overclock             Configure overclocking for your Pi
7 Advanced Options     Configure advanced settings
8 Update               Update this tool to the latest version
9 About raspi-config   Information about this configuration tool

<Select>                                <Finish>
```

**Fig-III.2 configuration interface of the raspberry pi**

**III.2-Python3 installation:** It might not be necessary to update the Raspbian, but we do it to avoid problems we use the following code to update your Raspbian:

```
pi@raspberrypi:~ $ sudo apt-get update
```

Once the update process is done, and Raspbian is already up to date, we can run the following code to start the installation process of Python 3:

```
pi@raspberrypi:~ $ sudo apt-get install python3-pip
```

```
pi@raspberrypi:~ $ sudo apt-get update
Get:1 http://raspbian.raspberrypi.org/raspbian buster InRelease [15.0 kB]
Get:2 http://archive.raspberrypi.org/debian buster InRelease [32.6 kB]
Get:3 http://raspbian.raspberrypi.org/raspbian buster/main armhf Packages [13.0
MB]
Get:4 http://archive.raspberrypi.org/debian buster/main armhf Packages [331 kB]
Fetched 13.4 MB in 33s (408 kB/s)
Reading package lists... Done
pi@raspberrypi:~ $ sudo apt-get install python3-pip
Reading package lists... Done
Building dependency tree
Reading state information... Done
python3-pip is already the newest version (18.1-5+rpt1).
0 upgraded, 0 newly installed, 0 to remove and 182 not upgraded.
```

**Fig III.3 updating Raspbian and installing Python3**

## III.3-Enabling camera and i2c module:

I2C is a very commonly used standard designed to allow one chip to talk to another. So, since the Raspberry Pi can talk I2C we can connect it to a variety of I2C capable chips and modules. The I2C bus allows multiple devices to be connected to your Raspberry Pi, each with a unique address that can often be set by changing jumper settings on the module. It is very useful to be able to see which devices are connected to your Pi as a way of making sure everything is working.

to install i2c support and camera module trough raspberry pi configuration interface we use the following command :

```
pi@raspberrypi:~ $ sudo raspi-config
```

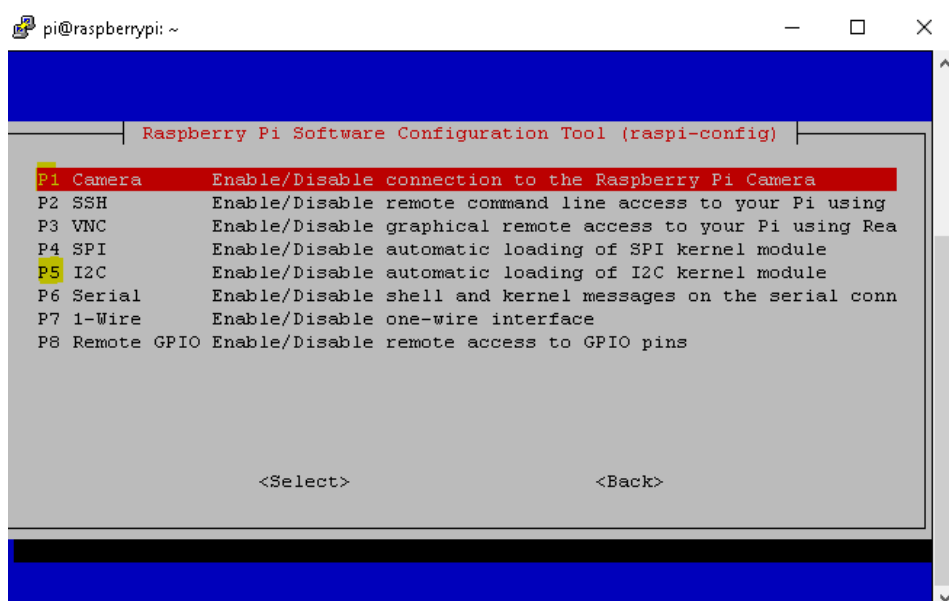


Fig III.4 -enabling i2c protocol and Camera module

**III.4-Install Adafruit Library for Dht22:** Using pip, we will install Adafruit's DHT library to the Raspberry Pi We will be using this Python library to interact with our DHT22 Humidity/Temperature sensor. We used following command to install the DHT library to Raspberry Pi.

```
pi@raspberrypi:~ $ sudopip3 install Adafruit_DHT
```

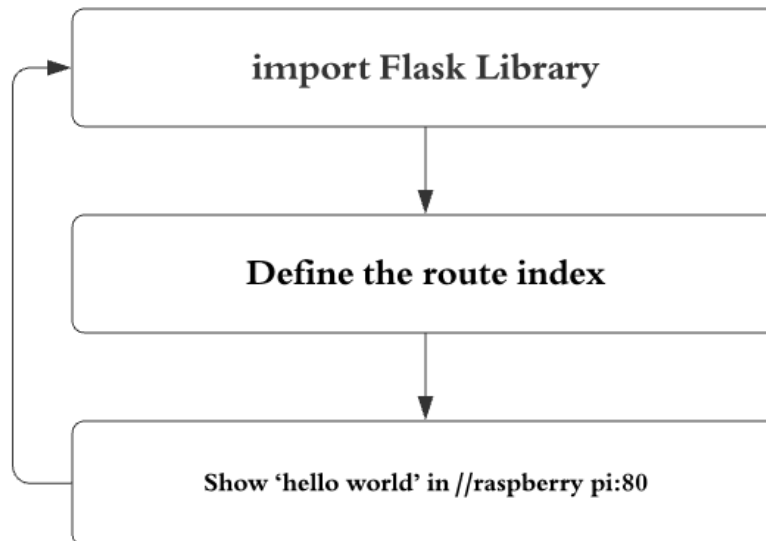
**III.5-Flask installation:** Modules can be downloaded as packages from the Python Package Index-(Pypi) and installed on the raspberry pi automatically. To install the module we use the command:

```
pi@raspberrypi:~ $ pip3 install flask
```

-After installing flask we can create the local web page and display it on the browser using the IP address of the raspberry pi with the port 80:

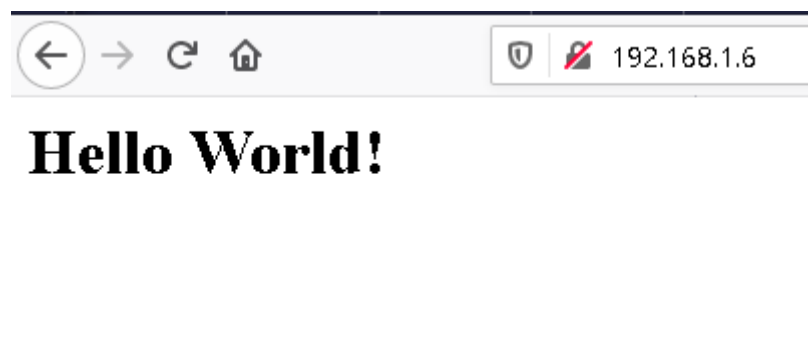
**Example for a local web page using flask that shows “Hello World”:**

**First Script: Algorithm for flask application test**



After creating the python3 code using flask we can run it in console using the following command and see the results in web browser:

```
pi@raspberrypi:~ $ sudo pyhton3 testflask.py
```



**FigIII.5-result of flask application**



**III.6-SQLite local storage installation and configuration:** the general idea will be collect data from the sensors and store them in a database that we create after installing SQLite3 using the command:

```
pi@raspberrypi:~ $ sudo apt-get install sqlite3
```

### III.7-Server, Data base and web page creation

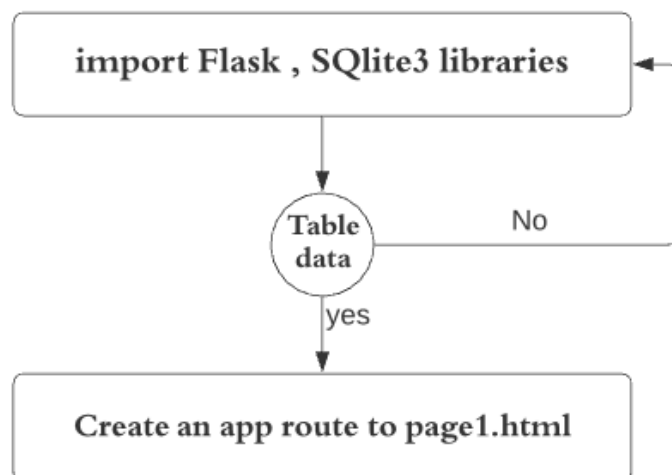
For that we create the data base using SQLite3 in the console using the command bellow after that we will create a table store data in it.

The previous table is created using the SQLite3 shell:

```
pi@raspberrypi:~ $ sqlite3database.db
sqlite>CREATE TABLE data1 (data1,data2);
sqlite>INSERT INTO data VALUES (1,10);
```

### III.8-Local Network test:

#### Second Script: Local network test using Flask and sqlite3



Now our table is created we can use the database with flask to show values in the web page python3 using flask and sqlite3 libraries:

-the data received will be showed in the web browser page after lunched the flask server:



**Fig III.6-** result of the web application using flask and Sqlite3

**The index.html files used in this web page:**

```
<head>
<title>example for sqlite and flask</title>
</head>
<body>
  <h1>data stored in data.db</h1>
  <h3> data1 ==> {{ data1 }} </h3>
  <h3> data2 ==> {{ data2 }} </h3>
</body>
</html>
```

## **IV-conclusion:**

The design of a project can be done in several ways to meet the same specifications. Choosing the right solution is essential and depends on several criteria.

Choosing the correct methods was depending of available sensors, information and acquired knowledge in term of software and programming.

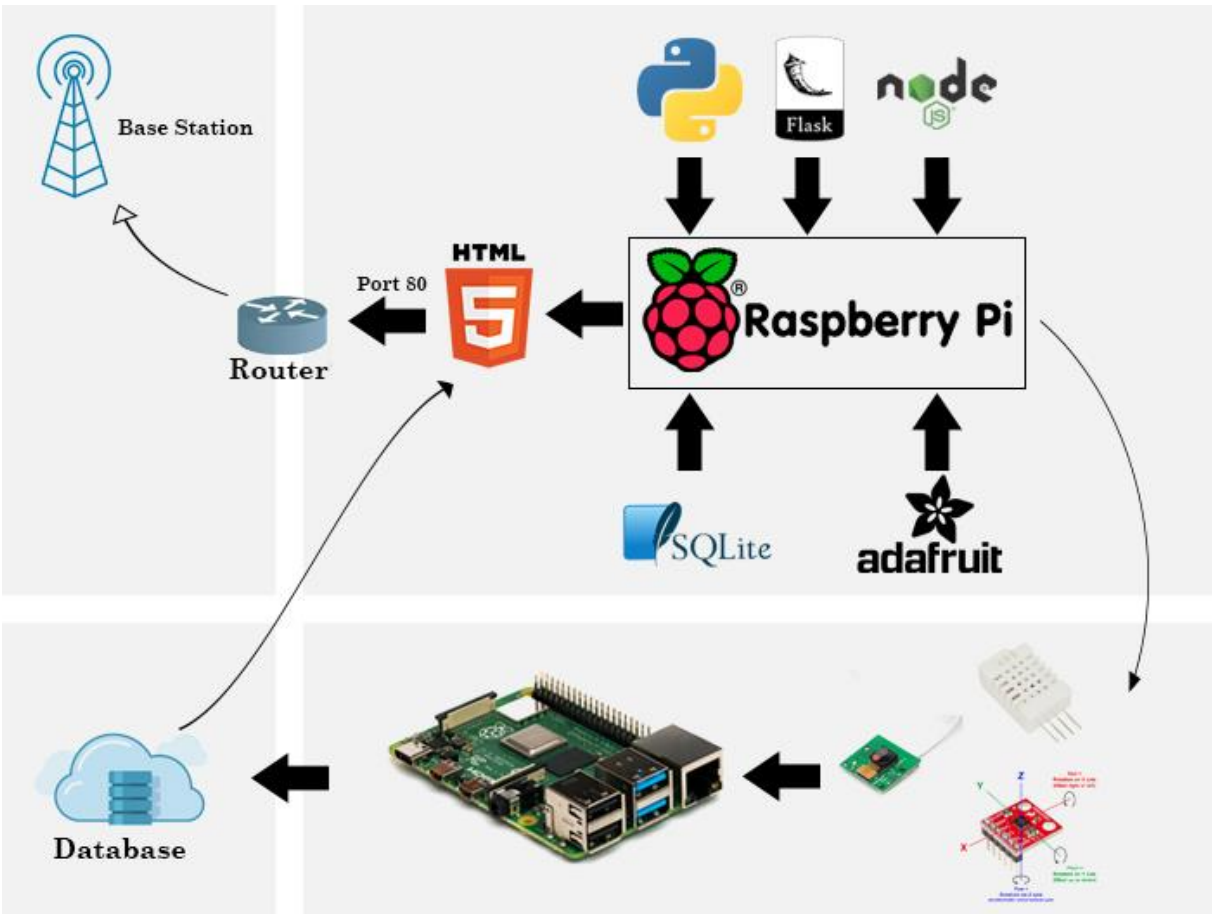
We explained in this chapter the method that we will follow to solve the problematic and we tested it with some basic application, so we can use them to concept our project.

**Chapter IV:**  
**Realization of helmet**  
**application,**  
**controlled by a web**  
**server**

**I-Introduction:**

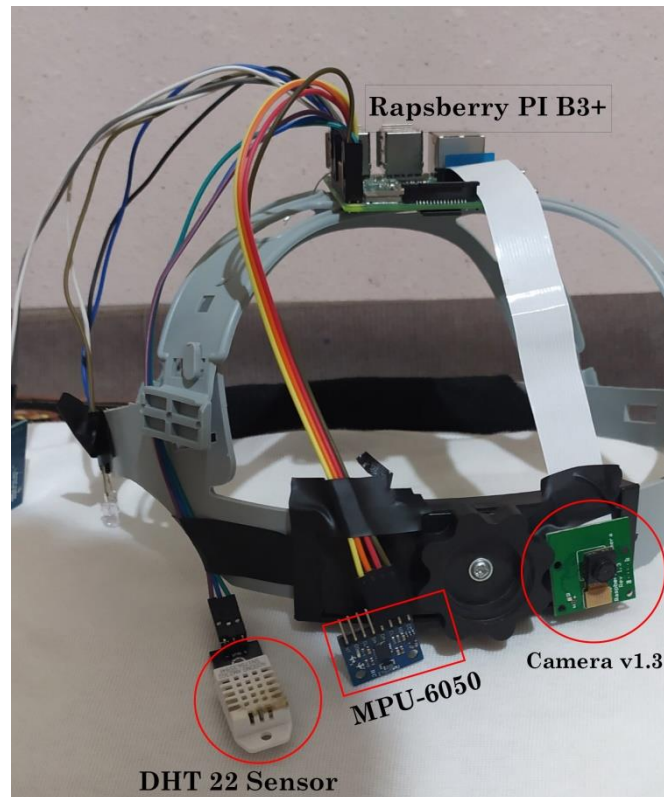
The final chapter in our thesis will contain the most important part of project in which we will test our equipment's one by one and collect them in final program that will be displayed in detail, also all other programs that we used in this project.

In the other hand we will discuss the results, and based on it we'll talk about how far we can go if we keep progressing on this helmet in future, finally we'll complete this chapter with final conclusion.



**FigIV.1 -The software Synoptic of wearable IoT helmet**

**II-Sensors tests:** the following figure shows the real position of sensors in the helmet

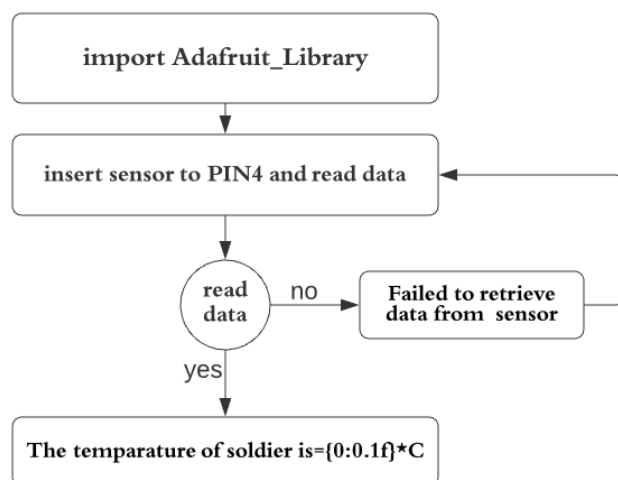


**FigIV.2-Position of different sensors**

### II.1-temperature and humidity sensor DHT22:

After installing Adafruit library in the previous chapter, now we can use the sensor and measure the temperature and humidity:

#### Third Script: DHT22 Test application



## Chapter IV: Realization of helmet application, controlled by a web server

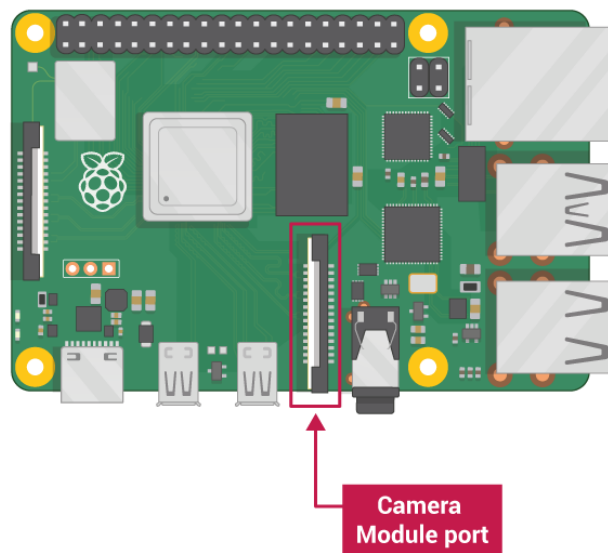
After running the code using python3 we can see that our sensor starts to measure temperature of the soldier in Celsius. The result will be shown and displayed in the terminal print screen, the results above show the Instantaneous temperature and renewed each second.

```
pi@raspberrypi: ~  
pi@raspberrypi:~$ sudo python3 dht1.py  
The temperature of soldier is=37.1*C  
The temperature of soldier is=36.9*C  
The temperature of soldier is=36.9*C  
The temperature of soldier is=36.9*C  
The temperature of soldier is=36.8*C  
The temperature of soldier is=36.8*C  
The temperature of soldier is=36.8*C  
The temperature of soldier is=36.8*C  
The temperature of soldier is=36.8*C  
The temperature of soldier is=36.8*C  
The temperature of soldier is=36.8*C  
The temperature of soldier is=36.7*C  
The temperature of soldier is=36.7*C
```

**FigIV.3-Result of DHT22 test using Python3**

### II.2-the Camera Module:

After enabling the camera module in configuration interface now we can plug it in the CSI connector and use it:



**Fig IV.4 Raspberry pi CSI port for the Camera v1.3**

## Chapter IV: Realization of helmet application, controlled by a web server

In Raspbian there are built-in functions for the camera module .With those commands it is possible to take pictures and record videos. One of these built-in functions or commands is called “**raspistill**” for taking pictures and “**raspivid**” for videos.

```
raspistill -o battlefield.jpg
```

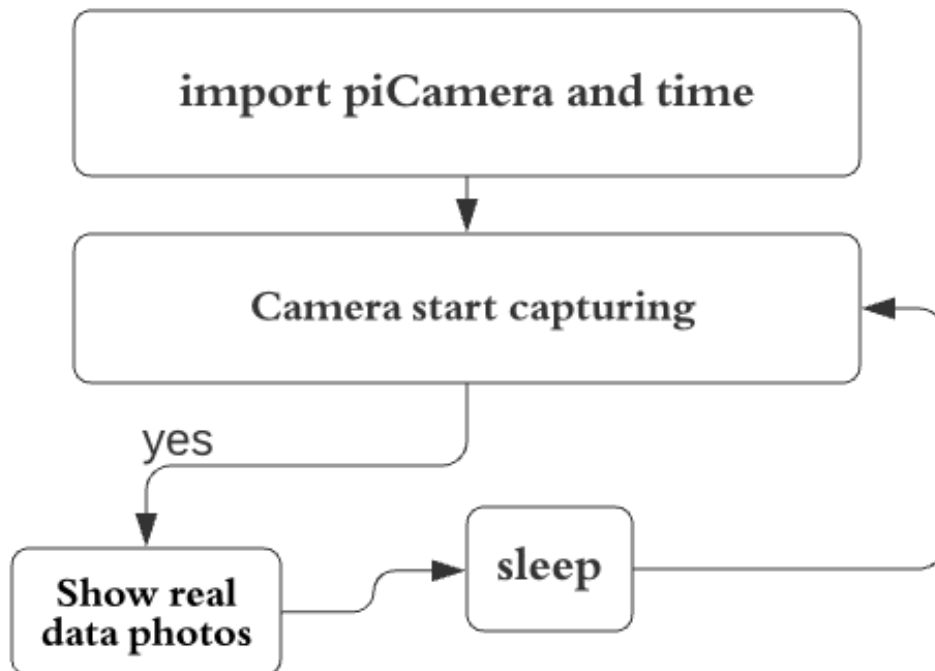
-We can record videos using the following command:

```
raspivid -o battlefield.h264
```

After typing the command above into terminal a preview window is started up. The preview window is running for 5 seconds, and then Raspberry pi takes the picture or the video and saves it. We used `-o` in the command for specifying the output filename

There are many possible options we can use such as `-w` for width, `-h` for height and `-q` for quality.

### Fourth Script: camera module test



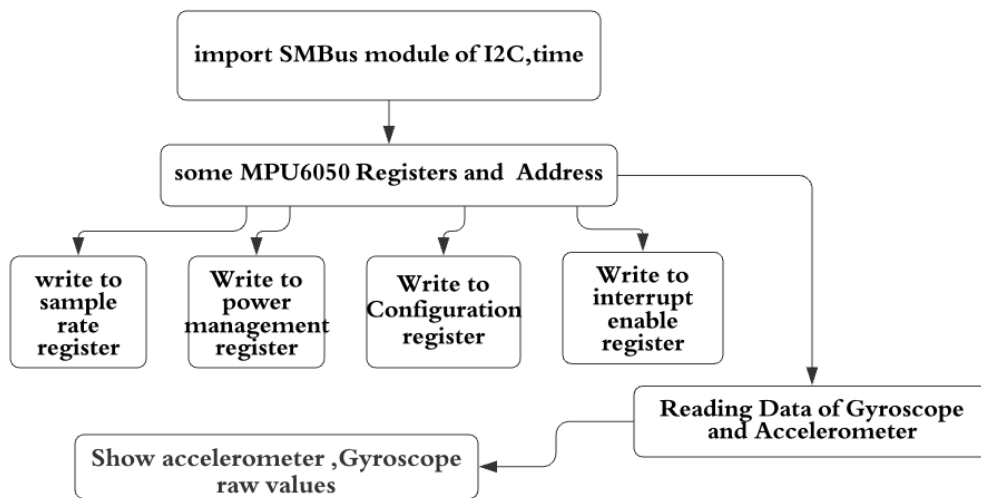
We can control the camera via the command line or via a script in python

## II.3-Accelerometer and Gyroscope i2c MPU-6050:

After installing the **SMBUS** library that provides the Circuit Python support and after enabling I2C on platform and verifying that Python 3 is running.

We can use the following code to import the necessary modules and initialize the I2C connection with the sensor:

### Fifth Script: MPU-6050 accelerometer gyroscope test



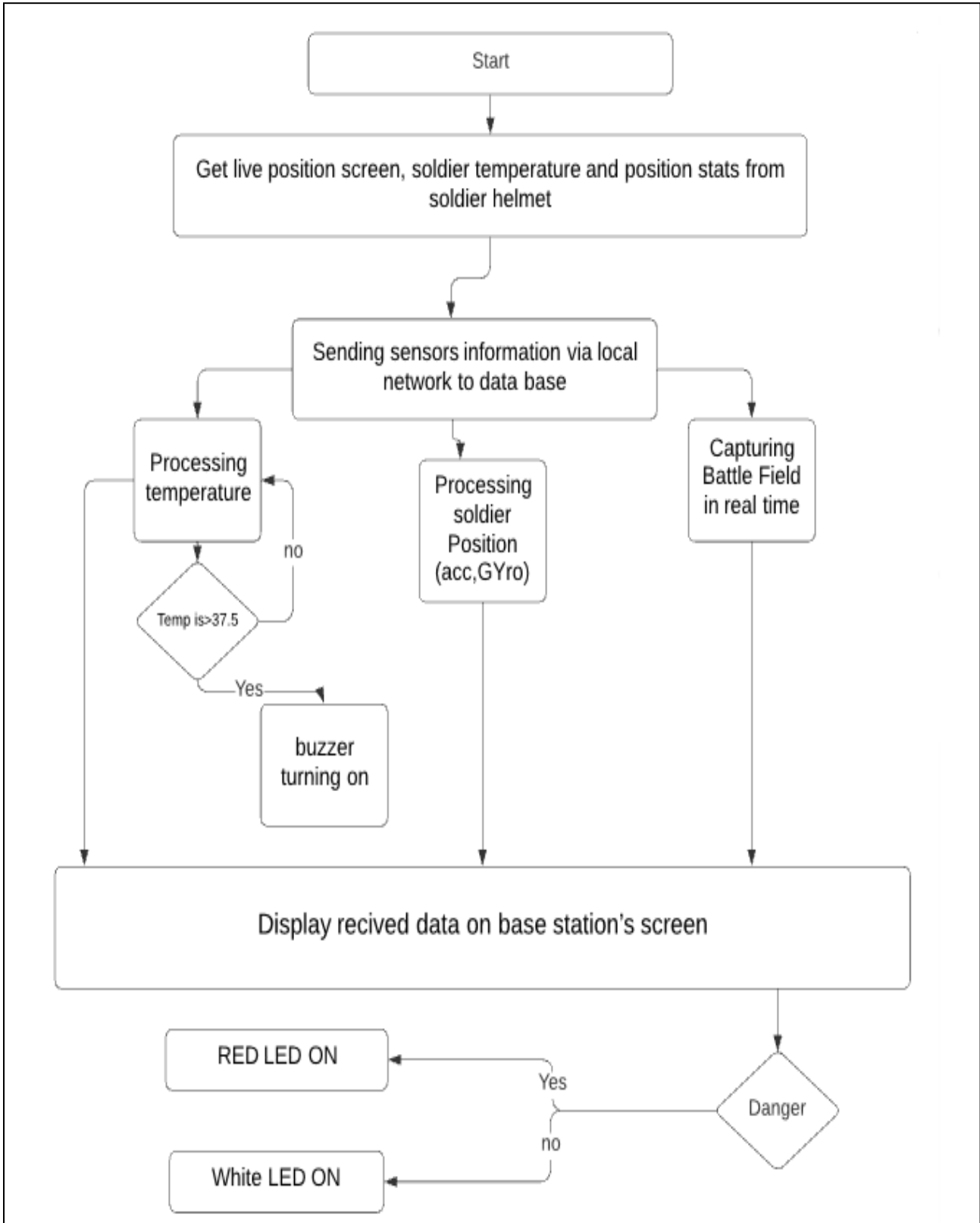
-The result after running the code using python:

```
pi@raspberrypi:~$ sudo python3 testgyro.py
Reading Data of Gyroscope and Accelerometer
Gx=-0.02 °/s Gy=0.13 °/s Gz=0.02 °/s Ax=-0.46 g Ay=-0.85 g Az=0.08 g
Gx=1.93 °/s Gy=0.18 °/s Gz=0.03 °/s Ax=-0.47 g Ay=-0.84 g Az=0.07 g
Gx=1.95 °/s Gy=0.15 °/s Gz=0.03 °/s Ax=-0.46 g Ay=-0.84 g Az=0.07 g
Gx=-0.01 °/s Gy=0.17 °/s Gz=0.04 °/s Ax=-0.46 g Ay=-0.83 g Az=0.08 g
Gx=-0.01 °/s Gy=0.22 °/s Gz=0.02 °/s Ax=-0.47 g Ay=-0.83 g Az=0.08 g
Gx=-0.02 °/s Gy=0.16 °/s Gz=0.02 °/s Ax=-0.46 g Ay=-0.84 g Az=0.07 g
Gx=-1.95 °/s Gy=0.18 °/s Gz=0.03 °/s Ax=-0.46 g Ay=-0.83 g Az=0.07 g
Gx=0.02 °/s Gy=0.18 °/s Gz=0.05 °/s Ax=-0.46 g Ay=-0.85 g Az=0.07 g
Gx=-1.95 °/s Gy=0.16 °/s Gz=0.02 °/s Ax=-0.46 g Ay=-0.84 g Az=0.07 g
Gx=0.02 °/s Gy=0.18 °/s Gz=0.03 °/s Ax=-0.46 g Ay=-0.84 g Az=0.08 g
```

FigIV.5 -Result of the test of Gyroscope and accelerometer in 6-axis



III-Organizational Chart :

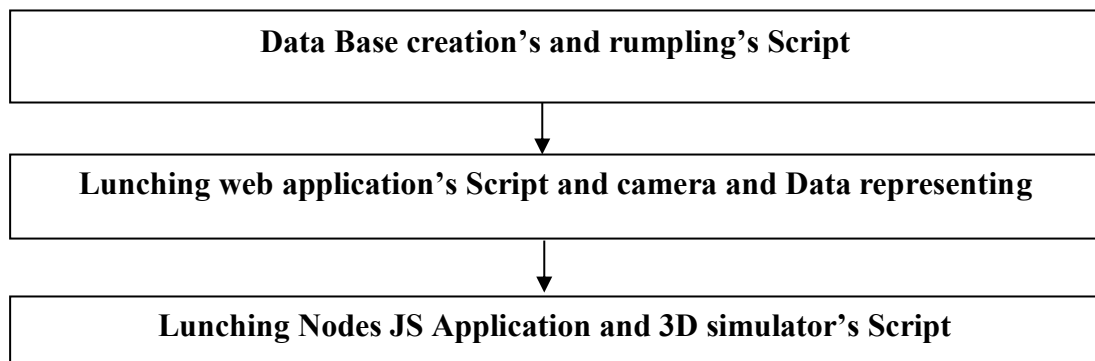


FigIV.6-algorithm of the application

## IV-APPLICATION CODE:

- By using the methods that we mentioned in previous chapter , now our sensor results and camera Instantaneous pictures can be uploaded via local network to the station control base to be available to viewed and simple to analyzed , below we will explain how we applied Chapter 3 methods with our sensors captures to obtain the final results .Capture real data (air temperature and relative humidity) using a DHT22 sensor; Load those data on a local database, built with SQLite3

To facilitate the work, we will representing our final code by divide it into 3 parts , each one has its own code attached , by following the procedure bellow :



### IV.1-First Part: Data Base creation's and rumpling's Script

#### IV.1.1-creating data base:

First we need to create our data base using sqlite3 for saving the dht22received dataat the current time:

```
import sqlite3 as lite
import sys
con = lite. Connect('sensorsData.db')
with con:
    cur = con.cursor()
    cur.execute("DROP TABLE IF EXISTS DHT_data")
    cur.execute("CREATE TABLE DHT_data(timestamp DATETIME, temp NUMERIC, hum
    NUMERIC)")
```

### IV.1.2-store DHT22 data into our table's database:

After creating our database we call a function that will return the captured data by the DHT22 sensor. Take those data (temperature and humidity) and passing them to another function that insert them, together with actual date and time, to our table. The program will continuously capture data, feeding them in our database. We included a buzzer that will warn the soldier if the temp exceed 37.5 (body's normal temperature)

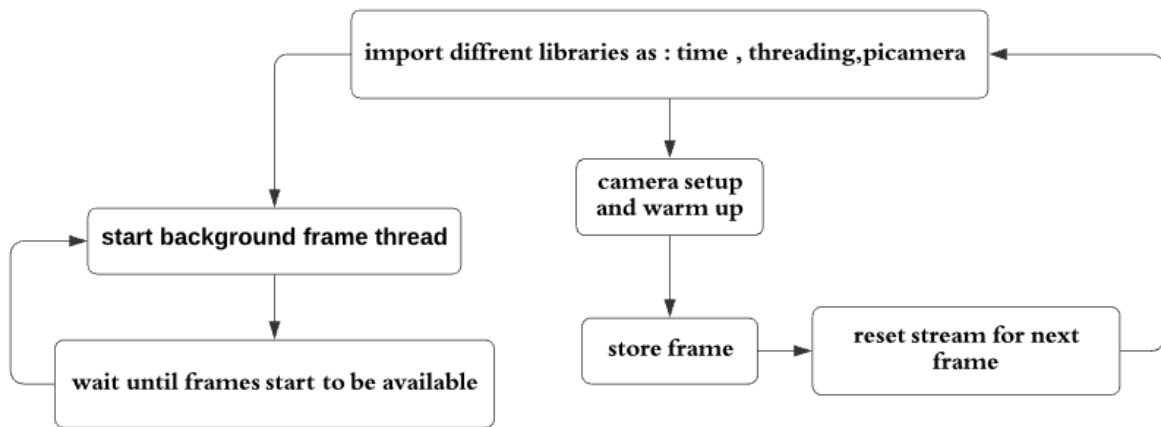
```
import time
import sqlite3
import Adafruit_DHT
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(26,GPIO.OUT)
BUZZER=26
dbname='sensorsData.db'
sampleFreq = 5 # time in seconds ==> Sample each 5sec
# get data from DHT sensor
def getDHTdata():
    DHT22Sensor = Adafruit_DHT.DHT22
    DHTpin = 4
    hum, temp = Adafruit_DHT.read_retry(DHT22Sensor, DHTpin)
    if hum is not None and temp is not None:
        hum = round(hum)
        if temp > 37.5:
            GPIO.output(BUZZER,1)
        else:
            GPIO.output(BUZZER,0)
        temp = round(temp, 1)
        return temp, hum
# log sensor data on database
def logData (temp, hum):
    conn=sqlite3.connect(dbname)
    curs=conn.cursor()
    curs.execute("INSERT INTO DHT_data values(datetime('now'), (?), (?))", (temp, hum))
    conn.commit()
    conn.close()
# main function
def main():
    while True:
        temp, hum = getDHTdata()
        logData (temp, hum)
        time.sleep(sampleFreq)
# ----- Execute program
main()
```

# Chapter IV: Realization of helmet application, controlled by a web server

## IV.1.3-Camera Streaming:

For this code we will initialize it for our flask application this code will work as a library for our camera's functions such as frames rotation and camera resolution.

### Seventh Script: The Camera library algorithm

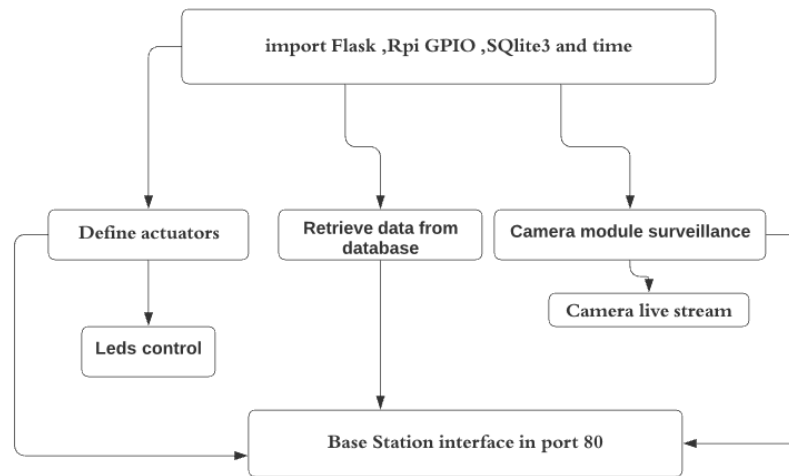


## IV.2-Second Part: Lunching web application's Script and camera and Data representing

The intelligent helmet that we made as we mentioned in previous chapters contain camera module which capture sequential photos in the second , using the script bellow we can send those frames to the control base station so we can follow the state of the soldier in real time and protect him against attacks behind him. Also we programmed an control panel used by the base station for backing up soldier, this control panel uses 2 lighting LEDs make them blink On/Off in case of detecting possible danger Those two property are presented in one script attached bellow

### IV.2.1-Flask application including camera, DHT22data and led control in case of danger:

## Sixth Script: Flask web application



### IV.2.2-HTML AND CSS CODES:

Sensor's revived data can be dense and disorganized , that can be very difficult to be read by controllers in station base and that results disturbance and slow in making the decisions and the orders, for that reason we designed an simple interface using HTML and CSS Code bellow, that provide in single page as simple way:

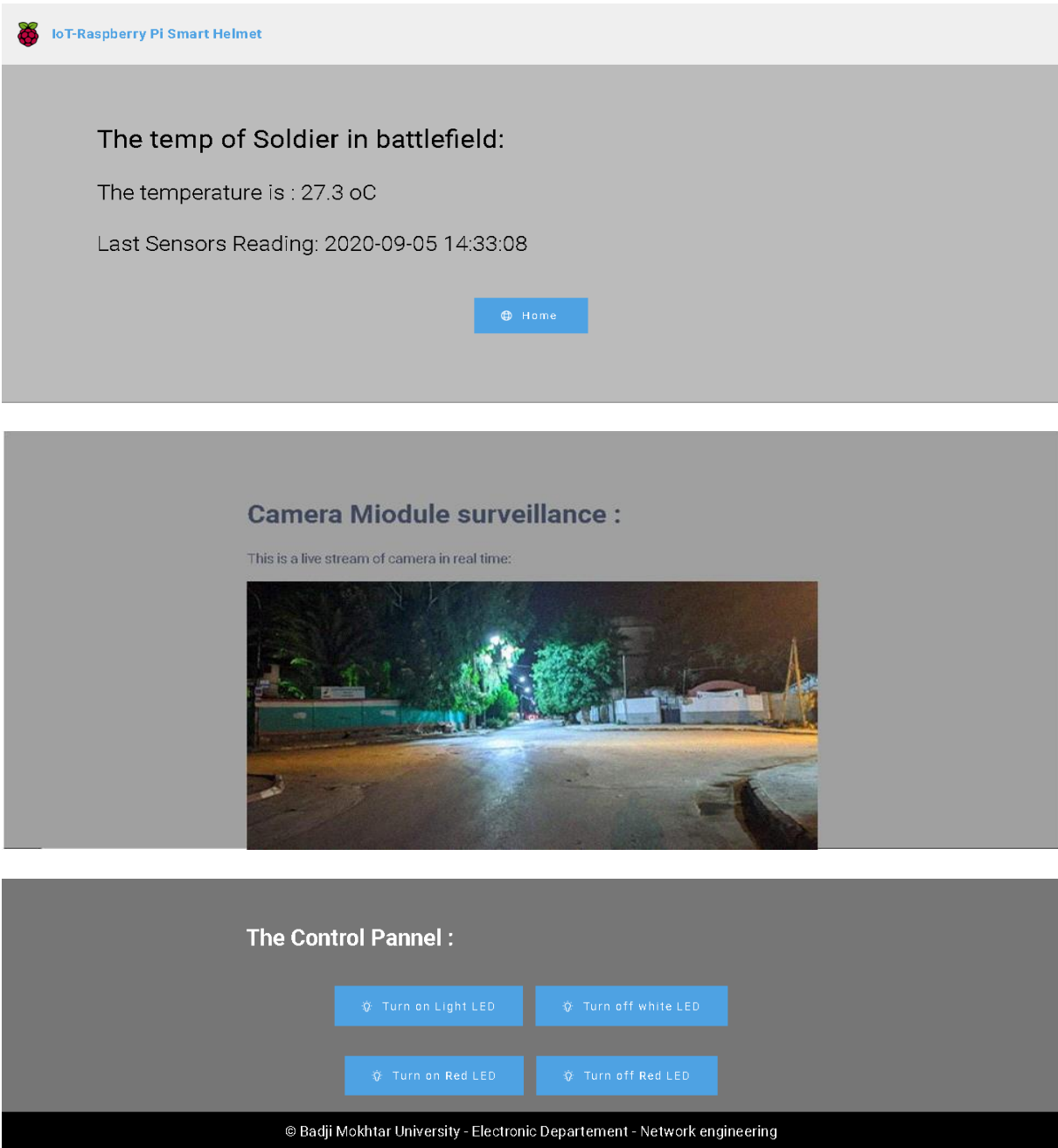
- The Battlefield arena's live Streaming
- Soldier instantaneous temperature
- LED and Buzzer control panel

#### IV.2.2.1-HTML code for the web server:

```
<head>
<link rel="stylesheet" href='../static/style.css'/>
</head>
<body>
<h3>DHT22 TEMPERATURE ==> {{ temp }} oC</h3>
<h3> Last Sensors Reading: {{ time }} ==><a href="/" class="button">REFRESH</a>
<h3><a href="{{ url_for('led') }}">leds</a></li></h3>
<h3><imgsrc="{{ url_for('video_feed') }}" width="300" height="300"></h3>
<h1>Actuators</h1> <br>
<h3>
Turn on red LED ==>
<a href="/ledYlw/on" class="button">TURN ON</a>
<a href="/ledYlw/off" class="button">TURN OFF</a>
</h3>
</body>
</html>
```

# Chapter IV: Realization of helmet application, controlled by a web server

**IV.2.2.2-Web server interface:** using html code of our web application showed in page 36 we get the following result:



**FigIV.7-base station interface shows temperature, video in real time and control panel**

-Our web application interface will now contain the data base information, It shows the soldier temperature in real time and the last time which the sensors captures Also we can see live stream of battlefield on the same page , and we can control the LEDs to inform the soldier in dangers cases

**IV.3-Third Part: Lurching Nodes JS Application and 3D simulator’s Script**

- To track soldier’s body, if he’s up,down, crouching ..., which inform us that the soldier still awake and alive, we used a node.js with specific code bellow to design an simple 3D anthropomorphic, simulator to soldier body, related to Gyroscope moves which is also related to soldier’s moves. This simulation will be added to the principal control interface of base station which we made before using HTML and Css.

**IV.3.1lurching the node.js application and show movement of the accelerometer and gyroscope:**

After detecting the accelerometer address: At address 0x68 (hexadecimal) there is an I2C device - in our case it is the MPU 6050 gyroscope.

```
pi@raspberrypi:~ $ sudo i2cdetect -y 1
 0 1 2 3 4 5 6 7 8 9 a b c d e f
00:  -----
10:  -----
20:  -----
30:  -----
40:  -----
50:  -----
60:  ----- 68 -----
70:  -----
```

**IV.3.2-Install Node.js and packages:**

Before they can start the server, we first have to install Node.JS and the package manager npm :

```
pi@raspberrypi:~ $ sudo apt-get install gitnodejsnpm --yes
```

After that we can clone the GitHubprogram and go to the directory and download the required libraries for our accelerometer using npm node.js and after that we install the nmp to our raspberry pi

```
pi@raspberrypi:~ $ git clone https://github.com/tutRPi/Raspberry-Pi-WebGL-Gyroscope
pi@raspberrypi:~ $ install npm
```

## Chapter IV: Realization of helmet application, controlled by a web server

### IV.3.3 node.js script:

```
var express = require('express');
var gyroscopeController = require('./controllers/gyroscopeController');

require('dotenv').config()

var app = express();
app.use(express.static('public'));

app.get('/', function (req, res) {
  res.sendFile('index.html');
});

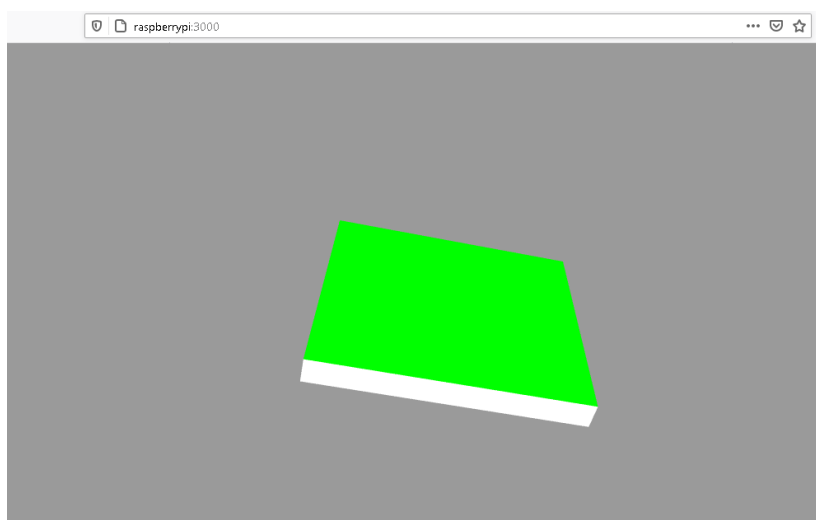
app.get('/api/data', gyroscopeController.data);

app.listen(process.env.PORT || 3000 , function(){
  console.log("Listening on port " + (process.env.PORT || 3000))
})
```

-Our application will be on port 3000 on browser we use the command:

```
pi@raspberrypi:~/mputest $ node index.js
```

-the result of our 3D anthropomorphic system is shown in the next figure:



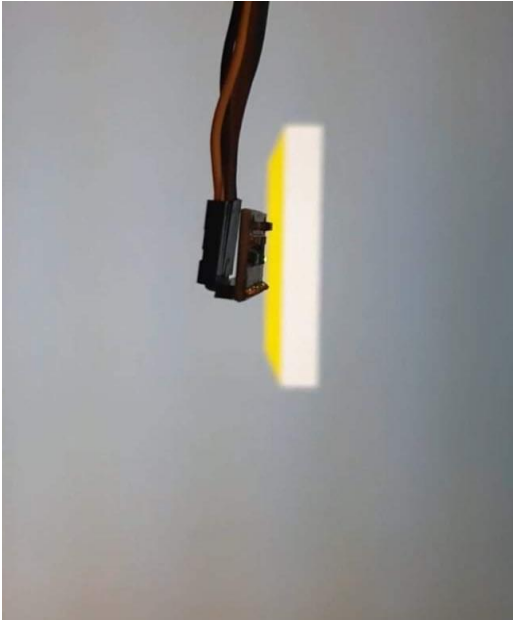
**FigIV.8-Soldier movements on 6-axis using a simple 3D anthropomorphic in real time**



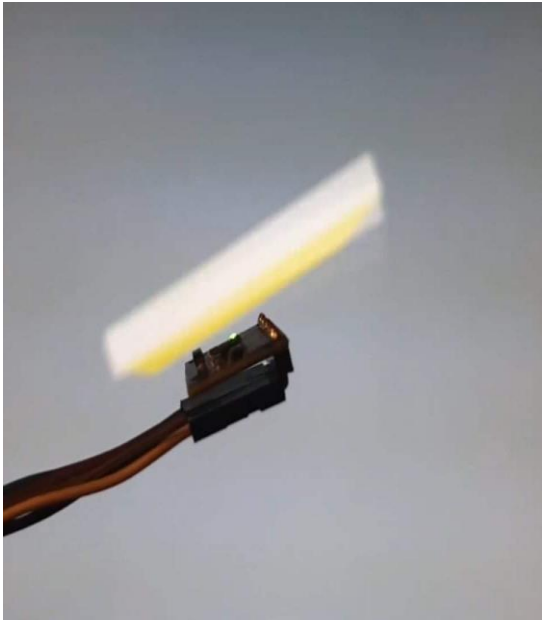
# Chapter IV: Realization of helmet application, controlled by a web server

### IV.3.4-Result and discussions:

The gyroscope and accelerometer proved to be a highly effective motion sensor for physical activity Assessment, the anthropomorphic simulates the same moves of soldier (gyroscope sensor on helmet) behavior if he is standing or prone based on X, Y and Z axis in data processing as shown in figures bellow, The mount of acceleration in Z axe shows if the soldier is falling too so the base station can send an alert via the control panel. Meanwhile we are unable to differentiate both jogging and walking pattern due to the ability of the sensor for directing direction and since both jogging and walking activity are conducted in the same direction. The mount of acceleration in Z axe shows if the soldier is falling too so the base station can send an alert via the control panel.



FigIV.9-Standing position of the soldier



FigIV.10-Left slanted position of soldier

### V-Conclusion:

By using some electronic materials and tools ,and after tests we have fabricated an IOT object, a smart helmet that help the soldier by scanning and covering the battlefield area and we have succeeded to connect it remotely via local network to military base station that can make decisions and order the soldier via a server web .

The biggest problem we faced in this project is the acquisition of electronic material because of the exceptional circumstances of Corona pandemic, So that was supposed to add a pulse

## Chapter IV: Realization of helmet application, controlled by a web server

---

rate sensor to the helmet for more soldier's health tracking also an reliable temperature sensor with big resistance, so we were forced to satisfied with the available sensors and materials to success this experience hoping to having the chance to develop it in the future in other best conditions.

In conclusion we can say that the IOT objects and the smart helmet especially are revolutionaries systems destined to evolve even further in the future.

## General Conclusion:

In this context and as part of our master's formation in telecommunications at Badji Mokhtar University, each student must participate in the PFE.

We chose between several themes and then formed groups for each theme, and to put

By applying our acquired knowledge and improving our skills we undertook to work on IOT objects which addresses the design and integration of a smart soldier helmet

Our study allowed us to better understand the components and technologies affected by the field of the Internet of Things.

In our project we have implemented a system Embodied in a smart helmet connected via a local network remotely to a base control station, represented in interface web, it can receive the health status of the soldier who wearing that helmet and the information of battle field which covered by its camera, also the station can send orders and inform the soldier in dangers cases, so it works in bidirectional way and remotely, and this is the main importance of internet of things.

We were not content with just the theoretical side, but we moved to the practical one also, in which we designed a real helmet that works with high efficiency.

During this work we have employed several modules that we have studied in previous years such as: IP routing, broadband networks, Linux, web technologies, basic electronics, Documentary research and thesis design.

In terms of skills acquired, this project allowed us to:

- ✓ Know how to program a microcontroller ( Raspberry Pi )
- ✓ Know the PYTHON software and be able to make the connection between the Raspberry card and this software to acquire data and also to control in return.
- ✓ Create an HTML page and server using python and flask , which are very helpful in learning data, image processing ...
- ✓ To face practical problems, real problems not only in theory,
- ✓ Know how to manage them and most importantly find reliable solutions to meet these problematic.

This did not prevent having encountered many constraints and obstacles in particular the lack of electronic components, which was limited because of Covid-19 pandemic.

We also encountered programming and configuration problems of the various modules electronic because it was our first experience in this field.

Finally, we hope through our work to provide practical validation of these Techniques and donate a good cause to better explore this area of the Internet of Things.

**ANNEX:****The First Script:**

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def index():
    return '<h1>Hello World!</h1>'

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=80)
```

**The Second Script:**

```
from flask import Flask, render_template, request
app = Flask(__name__)
import sqlite3
# Retrieve data from database
def getData():
    conn=sqlite3.connect('../data.db')
    curs=conn.cursor()
    for row in curs.execute("SELECT * FROM data1"):
        data1 = row[0]
        data2 = row[1]
    conn.close()
    return data1, data2
# main route
@app.route("/")
def index():
    data1,data2 = getData()
    templateData = {
        'data1': data1,
        'data2': data2,
    }
    return render_template('index.html', **templateData)
@app.route("/page1")
def page1():
    return render_template("page1.html")

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80, debug=False)
```

**The Third Script:**

```
import Adafruit_DHT

DHT_SENSOR = Adafruit_DHT.DHT22
DHT_PIN = 4

while True:
    humidity, temperature = Adafruit_DHT.read_retry(DHT_SENSOR, DHT_PIN)
    if humidity is not None and temperature is not None:
        print("The temperature of soldier is={0:0.1f}*C".format(temperature, humidity))
    else:
        print("Failed to retrieve data from humidity sensor")
```

## The Fourth Script:

```
from picamera import PiCamera
from time import sleep
camera = PiCamera()

camera.start_preview()
sleep(5)
camera.stop_preview()
```

## The Fifth Script:

```
import smbus          #import SMBus module of I2C
from time import sleep #import

#some MPU6050 Registers and their Address
PWR_MGMT_1 = 0x6B
SMPLRT_DIV = 0x19
CONFIG     = 0x1A
GYRO_CONFIG = 0x1B
INT_ENABLE = 0x38
ACCEL_XOUT_H = 0x3B
ACCEL_YOUT_H = 0x3D
ACCEL_ZOUT_H = 0x3F
GYRO_XOUT_H = 0x43
GYRO_YOUT_H = 0x45
GYRO_ZOUT_H = 0x47
def MPU_Init():
    #write to sample rate register
    bus.write_byte_data(Device_Address, SMPLRT_DIV, 7)

    #Write to power management register
```

```
bus.write_byte_data(Device_Address, PWR_MGMT_1, 1)

#Write to Configuration register
bus.write_byte_data(Device_Address, CONFIG, 0)

#Write to Gyro configuration register
bus.write_byte_data(Device_Address, GYRO_CONFIG, 24)

#Write to interrupt enable register
bus.write_byte_data(Device_Address, INT_ENABLE, 1)

def read_raw_data(addr):
    #Accelerometer and Gyro value are 16-bit
    high = bus.read_byte_data(Device_Address, addr)
    low = bus.read_byte_data(Device_Address, addr+1)

    #concatenate higher and lower value
    value = ((high << 8) | low)

    #to get signed value from mpu6050
    if(value > 32768):
        value = value - 65536
    return value

bus = smbus.SMBus(1)      # or bus = smbus.SMBus(0) for older version boards
Device_Address = 0x68 # MPU6050 device address

MPU_Init()

print (" Reading Data of Gyroscope and Accelerometer")

while True:

    #Read Accelerometer raw value
    acc_x = read_raw_data(ACCEL_XOUT_H)
    acc_y = read_raw_data(ACCEL_YOUT_H)
    acc_z = read_raw_data(ACCEL_ZOUT_H)

    #Read Gyroscope raw value
    gyro_x = read_raw_data(GYRO_XOUT_H)
    gyro_y = read_raw_data(GYRO_YOUT_H)
    gyro_z = read_raw_data(GYRO_ZOUT_H)

    #Full scale range +/- 250 degree/C as per sensitivity scale factor
    Ax = acc_x/16384.0
    Ay = acc_y/16384.0
```

```
Az = acc_z/16384.0

Gx = gyro_x/131.0
Gy = gyro_y/131.0
Gz = gyro_z/131.0

print ("Gx=%.2f" %Gx, u'\u00b0'+ "/s", "\tGy=%.2f" %Gy, u'\u00b0'+ "/s", "\tGz=%.2f"
%Gz, u'\u00b0'+ "/s", "\tAx=%.2f g" %Ax, "\tAy=%.2f g" %Ay, "\tAz=%.2f g" %Az)
sleep(1)
```

### The sixth script:

```
# Calling libraries and initializing part
from flask import Flask, render_template, request, Response
app = Flask(__name__)
from camera_pi import Camera
import sqlite3
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

#define actuators GPIOs
ledRed = 18
ledYlw = 20
#initialize GPIO status variables
ledRedSts = 0
ledYlwSts = 0
# Define led pins as output
GPIO.setup(ledRed, GPIO.OUT)
GPIO.setup(ledYlw, GPIO.OUT)
# turn leds OFF
GPIO.output(ledRed, GPIO.LOW)
GPIO.output(ledYlw, GPIO.LOW)
```

**# Retrieve data from database**

```
defgetData():
    conn=sqlite3.connect('../sensorsData.db')
    curs=conn.cursor()
    for row in curs.execute("SELECT * FROM DHT_data ORDER BY
timestamp DESC LIMIT 1"):
        time = str(row[0])
        temp = row[1]
        hum = row[2]
    conn.close()
    return time, temp, hum
```

**# DHT22 data**

```
@app.route("/")
def index():
    time, temp, hum = getData()
    templateData = {
        'time': time,
        'temp': temp,
        'hum': hum
    }
    returnrender_template('index.html', **templateData)
```

**# Camera module surveillance**

```
@app.route('/camera')
def cam():
    """Video streaming home page."""
    timeNow = time.asctime( time.localtime(time.time()) )
    templateData = {
        'time': timeNow
    }
    returnrender_template('camera.html', **templateData)
```

```
def gen(camera):
    """Video streaming generator function."""
    while True:
        frame = camera.get_frame()
        yield (b'--frame\r\n'
b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
    @app.route('/video_feed')
    defvideo_feed():
        """Video streaming route. Put this in the src attribute of an img tag."""
        return Response(gen(Camera()),
```



```
mimetype='multipart/x-mixed-replace; boundary=frame')

# Leds control
@app.route("/leds")
def led():
    # Read Sensors Status
    ledRedSts = GPIO.input(ledRed)
    ledYlwSts = GPIO.input(ledYlw)

    templateData = {
        'title' : 'GPIO output Status!',
        'ledRed' : ledRedSts,
        'ledYlw' : ledYlwSts,
    }

    returnrender_template('index.html', **templateData)

@app.route("/<deviceName>/<action>")
defaction(deviceName, action):
    ifdeviceName == 'ledRed':
        actuator = ledRed
    ifdeviceName == 'ledYlw':
        actuator = ledYlw

    if action == "on":
        GPIO.output(actuator, GPIO.HIGH)
    if action == "off":
        GPIO.output(actuator, GPIO.LOW)

    ledRedSts = GPIO.input(ledRed)
    ledYlwSts = GPIO.input(ledYlw)

    templateData = {
        'ledRed' : ledRedSts,
        'ledYlw' : ledYlwSts,
    }
    returnrender_template('index.html', **templateData)

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80, debug=False)
```

### The seventh Script:

```
import time
importio
```

```
import threading
importpicamera

class Camera(object):
    thread = None # background thread that reads frames from camera
    frame = None # current frame is stored here by background thread
    last_access = 0 # time of last client access to the camera

    def initialize(self):
        ifCamera.thread is None:
            Camera.thread = threading.Thread(target=self._thread)
            Camera.thread.start()
        whileself.frame is None:
            time.sleep(0)
        defget_frame(self):
            Camera.last_access = time.time()
            self.initialize()
            returnself.frame
        @classmethod
        def _thread(cls):
            withpicamera.PiCamera() as camera:
                camera.resolution = (320, 240)
                camera.hflip = True
                camera.vflip = True
                camera.start_preview()
                time.sleep(2)
                stream = io.BytesIO()
                for foo in camera.capture_continuous(stream, 'jpeg',
                    use_video_port=True):
                    stream.seek(0)
                    cls.frame = stream.read()
                    stream.seek(0)
                    stream.truncate()
                iftime.time() - cls.last_access > 10:
                    break
```

## References :

- [1] Bessy Benny, Gissmon Babu , Lijamol Mathew, Minna Eldho, Jinto Mathew, Robin George, "Smart Helmet, an intelligent safety for riders using Raspberry Pi and sensors for Drunken drive with GPS tracking" , IJAREEIE , Vol. 8, Issue 6, June 2019.
- [2] Mr.Sethuram rao, Vishnupriya.S.M, Mirnalini.Y , Padmapriya.R.S , " THE HIGH SECURITY SMART HELMET USING INTERNET OF THINGS", IJPAM, Volume 119. 14439-14450, No 12 2018
- [3] Musab Ali, Arisa Olivia, Mouhammad Saad, Sidiq Hidayat, "Wearable sensor and internet of things technology for better medical science", IJET, January 2019
- [4] Ahmed Abdelgawad, Ahmed Khattab, Kumar Yelamarthi, " IOT based health monitoring for active and assisted living" , Lecture Notes of the Institute for Computer Sciences · July 2017
- [5] Nagaraj Hegde, Edward Sazonov, " Foot wear based wearable system smart shoe", Department of Electrical and Computer Engineering, The University of Alabama; Tuscaloosa, AL 35487, USA; 10 August 2016
- [6] Boon Giin Lee , "Stress Events Detection of Driver by wearable Glove system" IEEE Sensors Journal · November 2016
- [7] Geunsik Lim, 30 Aug 2019, Raspberry PI , <<https://github.com/bjpublic/raspberrypi/wiki>>
- [8] Lokeshc, 2018, ElectronicSensors, <<https://www.electronicwings.com/sensors-modules/mpu6050-gyroscope-accelerometer-temperature-sensor-module>>
- [9] Emmet, 13 mai ,2019, Raspberry Pi Humidity Sensor using the DHT22 < <https://pimylifeup.com/raspberry-pi-humidity-sensor-dht22/>>
- [10] Welcome to Rasbian, n.d, BytemarkHosting <<https://www.raspbian.org>>
- [11] WhatIsPython, n.d, <<https://www.python.org/doc/essays/blurb>>
- [12] Flask Foreword". Archived from the original on 2017-11-17 <[https://en.wikipedia.org/wiki/Flask\\_\(web\\_framework\)](https://en.wikipedia.org/wiki/Flask_(web_framework))>
- [13] World Wide Web Consortium. December 18, 1997. Archived from the original on July 5, 2015. Retrieved July 6, 2015
- [14] "What is CSS?". World Wide Web Consortium. Archived from the original on 2010-11-29. Retrieved 2010-12-01.
- [15] Hack Reactor, October 18, 2018, WhatIsJava, <<https://www.hackreactor.com/blog/what-is-javascript-used-for>>
- [16] gcuomo (24 October 2013). "JavaScript Everywhere and the Three Amigos (Into the wild BLUE yonder!)". www.ibm.com.