Misr Higher Institute
for Engineering & Technology

Department of Communications
and Computer Engineering Department

Using EEG Headset based on AI and IoT techniques to help the handicapped

A project submitted in partial fulfillment for the

requirements of the degree of B. Sc. in

Computer Engineering

By

Ahmed Mohammed Abo Saif
Ahmed Hossam Mahmoud
Aia Magdi Saadawi
Alaa Mohammed Abdelfattah
Dina Mohammed Ismail
Hossam Mohammed Ashour
Kerolos Sadek Gerges
Nada Abdo Alnajdi
Nourhan Ashraf Khafagy
Rajaa Abdelfattah Alzahaby
Shimaa Mahmoud Zakaria

Supervisor

Assist. Prof. Dr./ Yasser Alawady

Eng./ Omar Atef Sesa

Communications and Computer Engineering Department

Misr Higher Institute for Engineering & Technology

2023

Project Group Members

- Ahmed Abo Saif

- Aia Saadawi

- Dina Ismail

- Kerolos Sadek

- Nourhan Ashraf

- Shimaa Mahmoud

- Ahmed Hossam

- Alaa Mohammed

- Hossam Ashour

- Nada Abdo

- Rajaa Alzahaby

Dedicated to our colleague

Ahmed Magdy

May allah receive him in his mercy

Acknowledgement

First of all, all thanks to Allah for giving us the strength and patience to go through this project.

We introduce Great thanks to our supervisor

Dr. Yasser Al-Awady

For his great help, valuable information, precious guidance

We are so thankful to

Eng. Omar Atef Sesa

For his tremendous helpful discussions, guidance, support, and hearty encouragement

Finally, an honorable mention goes to the entire team for their outstanding effort and to our families for supporting us in completing this project.

Abstract

This project makes intention to provide a comfortable environment for disabled people and make their lives more independent and natural as possible. It consists of three systems: self-driving car, smart home, and game. The idea is to get signals from the human mind through the headset, and then send it to the microcontroller to read and translate the event of the different applications.

Systems utilized in the project will be listed on the website as options for the user to choose from when he first visits, after which the user will activate the system. Additionally, information like temperature and heartbeat vital signs are some instances of data that will be presented on the website when the user takes any action. An alert will be sent out via the website if any sensor issues arise.

Table of Contents

List of contents

List of Contents

Acknowledgement ..... **i**

Abstract ..... **ii**

List of Figures ..... **ix**

List of Abbreviations ..... **xii**

Chapter 1: Introduction

1.1 Overview 2

1.2 Reason for the idea 3

1.3 Idea 3

1.4 Project Infograph 4

1.5 Documentation layout 5

Chapter 2: Hardware Components

2.1 Overview 7

2.2 Components 7

2.2.1 Controllers 8

2.2.1.1 Overview 8

2.2.1.2 Components 8

2.2.1.2.1 Atmega32-8bit AVR 8

2.2.1.2.2 PIC18f452 9

2.2.1.2.3 ESP8285 10

2.2.2 Sensors and Inputs 10

2.2.2.1 Overview 10

2.2.2.2 Components 11

2.2.2.2.1 HC-05 Bluetooth module 11

2.2.2.2.2 MQ-4 Gas sensor module 11

2.2.3 Actuators 11

2.2.3.1 Overview 11

2.2.3.2 Components 12

2.2.3.2.1 Relay 12

2.2.3.2.2 LED 12

2.2.3.2.3 LDR module 13

2.2.3.2.4 LCD Graphical screen 13

2.2.3.2.5 Servo motor 14

2.2.3.2.6 Motor driver 14

List of contents

2.2.3.2.7 BCI Headset 15
2.2.4 Power 16
2.2.4.1 Overview 16
2.2.4.2 Components 16
2.2.4.2.1 Battery 16
Chapter 3: Embedded Systems
3.1 Brief of embedded system 18
3.2 Characteristics of embedded system 18
3.3 Types of embedded system 19
3.3.1 Based on the performance and functional requirement 19
3.3.1.1 Real-Time embedded system 19
3.3.1.2 Stand-Alone embedded system 19
3.3.1.3 Networked embedded system 20
3.3.1.1 Mobile embedded system 20
3.3.2 Microcontroller performance-based embedded system 20
3.3.2.1 Small-Scale embedded system 20
3.3.2.2 Medium-Scale embedded system 20
3.3.2.3 Sophisticated embedded system 20
3.4 Embedded system architectures 21
3.4.1 Hardware architectures 21
3.4.1.1 Von-Neumann architecture 21
3.4.1.2 Harvard architecture 22
3.4.2 Software architectures 23
3.5 How do embedded system work 23
3.6 Applications of embedded system 24
3.7 Advantages and disadvantages of embedded system 25
3.8 Our driver codes 26
3.8.1 Flowchart describes car driver code 26
3.8.2 Flowchart describes smart home driver code 27
3.8.3 Flowchart describes sensors driver code 28
3.9 Visual diagrams 29
3.9.1 Smart home visual diagram 29
3.9.2 Smart car visual diagram 29
Chapter 4: Artificial Intelligence

List of contents

4.1 What is AI 31

4.2 Types of AI 31

4.2.1 Narrow AI 31

4.2.2 General AI 31

4.2.3 Super AI 32

4.3 Machine learning 32

4.2.1 Supervised learning 33

4.2.2 Unsupervised learning 33

4.2.3 Reinforcement learning 33

4.4 Deep learning 34

4.5 libraries used 34

4.5.1 Seaborn 35

4.5.2 NumPy 35

4.5.3 Cortex 35

4.5.4 Pandas 35

4.5.5 Tensorflow 35

4.5.6 Scikit learn 35

4.5.7 Matplotlib 36

4.6 Algorithms 36

4.6.1 Decision Tree Classifier 37

4.6.1.1 Why use Decision Trees? 38

4.6.2 Support Vector Classification (SVC) 38

4.6.2.1 Types of SVM 38

4.6.3 SGD Classifier 39

4.6.4 Random Forest Classifier 39

4.6.5 Naive Bayes Classifier 39

4.6.6 Linear Discriminant Analysis (LDA) 40

4.6.7 Ada Boost 40

4.6.8 Deep Neural Network (DNN) 40

4.6.9 Perception 40

4.6.10 K-Nearest Neighbors (KNN) 41

4.6.10.1 Advantages of KNN Algorithm 41

4.6.10.2 Disadvantages of KNN Algorithm 41

4.7 Project AI 42

List of contents

4.8 Model check 44

4.8.1 DNN 44

4.8.2 SVC 45

4.8.3 Model check 45

4.8.3.1 SGD classifier 46

4.8.3.2 Decision Tree classifier 46

4.8.3.3 Random Forest classifier 47

4.8.3.4 LDA classifier 48

4.8.3.5 Ada Boost 48

4.8.3.6 Perception 49

4.8.3.7 Native Bayes 49

4.8.3.8 KNN 50

4.9 Trials 51

4.9.1 Polynomial features 51

4.10 Algorithm with highest accuracy 53

Chapter 5: Internet of Things (IOT)

5.1 What is the internet of things 55

5.2 What is the history of the internet of things 55

5.3 Characteristics of IOT 56

5.4 IOT infogragh 59

5.5 Firebase overview 59

Chapter 6: Website & Mobile Applications

6.1 Front-End development 61

6.2 Programming languages for front-end development 61

6.2.1 HTML 61

6.2.2 CSS 61

6.2.3 TypeScript 61

6.3 Front-End development framework 61

6.3.1 Angular 61

6.4 Common Front-End development tasks 61

6.5 Back-End development 62

6.6 Back-End development languages and framework 62

6.6.1 Firebase 62

6.6.2 Node.js 62

List of contents

6.7 Common back-End development tasks 63

6.8 Website interface 64

6.8.1 Home page 64

6.8.2 Utilized technologies 65

6.8.2.1 Artificial Intelligence 65

6.8.2.2 Embedded systems 66

6.8.2.3 Web & mobile application 66

6.8.2.4 Hardware 67

6.8.3 Applications 67

6.8.3.1 Jump Games 67

6.8.3.2 The robot car 68

6.8.3.3 Smart home 68

6.8.4 Admin dashboard 70

6.9 Mobile application 71

6.9.1 Flutter 71

6.9.2 Flutter front-end 71

6.9.3 Flutter concepts 71

Chapter 7: Applications

7.1 Overview 74

7.2 Applications 74

7.2.1 Smart home 74

7.2.1.1 Internal System 74

7.2.1.2 External System 74

7.2.2 Smart car 75

7.2.3 Jump game 77

Chapter 8: Conclusion & Future Work

8.1 Conclusion 79

8.2 Future Work 79

References

List of figures

List of Figures

Chapter 1: Introduction

Fig 1.1 Project Idea 3

Fig 1.2 Project infograph 4

Chapter 2: Hardware Components

Fig 2.1 Components Categories..... 7

Fig 2.2 Atmega32-8bit AVR..... 8

Fig 2.3 ATmega32-8bit Microcontroller Pin Datagram 9

Fig 2.4 PIC18f452 9

Fig 2.5 ESP8285..... 10

Fig 2.6 HC-05 Bluetooth Module 11

Fig 2.7 MQ-4 Gas Sensor Module 11

Fig 2.8 Relay 12

Fig 2.9 LED 12

Fig 2.10 Lifetime Hours 13

Fig 2.11 LDR Module 13

Fig 2.12 LCD Graphical Screen 13

Fig 2.13 Servo Motor 14

Fig 2.14 L289N Motor Driver 14

Fig 2.15 Block Diagram of Motor Driver 15

Fig 2.16 Emotive Insight 5-channel EEG Headset 15

Chapter 3: Embedded Systems

Fig 3.1 Types of Embedded System..... 19

Fig 3.2 The Basic Architecture of Embedded System..... 21

Fig 3.3 The Von Neumann Architecture of Embedded System..... 22

Fig 3.4 Harvard Architecture of Embedded System..... 23

Fig 3.5 Applications of Embedded System 24

Fig 3.6 Flowchart describes car driver code..... 26

Fig 3.7 Flowchart describes smart home driver code 27

Fig 3.8 Flowchart describes sensors driver code 28

Fig 3.9 Smart home visual diagram..... 29

Fig 3.10 Smart car visual diagram 29

List of figures

Chapter 4: Artificial Intelligence

Fig 4.1 libraries 34

Fig 4.2 Algorithms 36

Fig 4.3 General structure of a decision tree 37

Fig 4.4 Data acquisition flowchart 42

Fig 4.5 Data before preparation 43

Fig 4.6 Data after preparation 43

Fig 4.7 AI flowchart 44

Fig 4.8 Trying DNN 44

Fig 4.9 OneVsOne classifier 45

Fig 4.10 Model check 45

Fig 4.11 Rest of model check code 46

Fig 4.12 SGD classifier train result 46

Fig 4.13 SGD classifier test result 46

Fig 4.14 Decision tree classifier train result 47

Fig 4.15 Decision tree classifier test result 47

Fig 4.16 Random forest classifier train result 47

Fig 4.17 Random forest classifier test result 48

Fig 4.18 LDA classifier train result 48

Fig 4.19 LDA classifier test result 48

Fig 4.20 Ada Boost classifier train result 48

Fig 4.21 Ada Boost classifier test result 49

Fig 4.22 Perception classifier train result 49

Fig 4.23 Perception classifier test result 49

Fig 4.24 Native Bayes classifier train result 49

Fig 4.25 Native Bayes classifier test result 50

Fig 4.26 KNN code 50

Fig 4.27 KNN code 50

Fig 4.28 Polynomial features ROC carve 51

Fig 4.29 Polynomial features confusion matrix 51

Fig 4.30 Features selection ROC carve 51

Fig 4.31Features selection confusion matrix 51

Fig 4.32 KNN result 52

Fig 4.33 KNN ROC carve 52

List of figures

Fig 4.34 KNN confusion matrix 52

Chapter 5: Internet of Things (IOT)

Fig 5.1 IOT infograph 58

Fig 5.2 Firebase overview 58

Chapter 6: Website & Mobile Applications

Fig 6.1 Back-End 61

Fig 6.2 Web application 62

Fig 6.3 Home page section 1 63

Fig 6.4 Home page section 2 63

Fig 6.5 Home page section 3 64

Fig 6.6 Artificial Intelligence technology 64

Fig 6.7 Embedded system technology 65

Fig 6.8 Web & mobile application technology 65

Fig 6.9 Hardware technology 66

Fig 6.10 Game 66

Fig 6.11 Robotic car 67

Fig 6.12 Smart home 67

Fig 6.13 Door state 68

Fig 6.14 Light state 68

Fig 6.15 Admin dashboard 69

Fig 6.16 Mobile App interface 1 71

Fig 6.17 Mobile App interface 2 71

Fig 6.18 Mobile App interface 3 71

Fig 6.19 Mobile App interface 4 71

Chapter 7: Applications

Fig 7.1 Smart home 74

Fig 7.2 Smart car photo 1 74

Fig 7.3 Smart car photo 2 75

Fig 7.4 Game application 75

List of Abbreviation

Abbreviation	Name
BCI	Brain Computer Interface
IoT	Internet of Things
AI	Artificial Intelligence
EEG	Electroencephalography
UART	universal asynchronous receiver-transmitter
CSV	Comma Separated Values
JSM	Junos Subscriber Management
MQTT	Message Queuing Telemetry Transport
R/W	Read or Write
CMOS	Complementary Metal Oxide Semiconductor
RISC	Reduced Instruction Set Computer
MIPS	Million instructions per second
MHz	Mega Hertz
RAM	Random-Access Memory
EEPROM	Electrically Erasable Programmable Read-Only Memory
USART	Universal Synchronous Asynchronous Receiver Transmitter
ADC or A/D	Analogue to Digital Converter
DIP	Dual In-Line Packet
QFN	quad flat no-lead package
MLF	multilateral (nuclear) force.
QTFP	Quicken Tax Freedom Project
RC	Radio Control
CPU	Central Processing Unit
PLVD	Programmable low voltage detection
MSSP	Master Synchronous Serial Port
WDT	Watchdog timer
ICSP™	In-Circuit Serial Programming™
ICD	In-Circuit Debug
TCP	Transmission control protocol
IP	Internet protocol
MiB	Mebibyte
IEEE	Institute of Electrical and Electronics Engineers
TR	Turkey
LNA	Licensed Nursing Assistant
GPIO	General Purpose Input/Output
WEP	Wired Equivalent Privacy

List of abbreviations

WPA	Wi-Fi Protected Access
SPI	Statistical Performance Indicators
I^2C	Inter-Integrated Circuit
I^2S	Integrated Inter-IC Sound Bus
DMA	Direct Memory Access
EDR	Endpoint Detection and Response
ISM	International Safety Management
Mbps	Megabits per Second
MOS	Metal Oxide Semiconductor
CNG	Compressed Natural Gas
LED	light-emitting diode
LCD	liquid-crystal display
CMS	Content Management System
DRL	Daytime running lamp
EEG	Electroencephalogram
IMU	Inertial Measurement Unit
API	Application Programming Interface
RTOS	real-time operating system
DSP	digital signal processors
FPGA	field-programmable gate arrays
ASIC	application-specific integrated circuits
GPU	Graphics processing unit
LCS	League Championship Series
PS4	Playstation 4
MRI	Magnetic resonance imaging
CT	Computed tomography
ECG	electrocardiogram
HTML	Hypertext Markup Language
CSS	Cascading style sheets
MVC	model-view-controller
OS	Operating Systems
iOS	iPhone operating system
UI	User Interface
LDR	Light Dependent Resistor
RFID	Radio Frequency Identification
M2M	Machine-to-Machine
ANI	Artificial narrow intelligence
AGI	Artificial general intelligence
ASI	Artificial super intelligence
NLP	Natural Language Processing
PDE	Partial Differential Equation

List of abbreviations

Sklearn	Scikit-learn
SVM	Support Vector Machine
SVC	Support Vector Classification
LDA	Linear Discriminant Analysis
DNN	Deep Neural Network
KNN	K-Nearest Neighbors
IMEI	International Mobile Equipment Identity
SMEs	Small and medium-sized enterprises
ML	Machine Learning

Chapter1

Introduction

Chapter 1

Introduction

1.1 Overview

Loss of motor function in one or more muscles is referred to as paralysis. If there is sensory injury, paralysis may also be accompanied by sensation loss in the affected area.

More people than previously believed are affected by paralysis, which has an impact on communication—the most crucial component of daily living that meets a person's requirements.

Disabled people may fall victim to mental and behavioral pitfalls such as:

- Believing they have no control over their lives.
- Rely on others more than necessary.
- Consider the worst-case scenario.
- Aim to highlight their limitations.

So, in addition to our goal of giving the disabled a comfortable atmosphere and attempting to make their life as autonomous and natural as possible, we consider the issue and work to find a solution.

And, with the growth of technology and artificial intelligence, we applied the concept of BCI and combined IOT and AI approaches to create a project that receives signals from the human mind through a headset, converting it to numerical data by cortex, and then sending it to the microcontroller to read and translate the event of each application:

- By receiving signals and employing automation, you can operate appliances at your home by simply thinking about them.
- Also, you can drive your car the way you think.
- We don't forget how essential fun is in our lives, so we worked on creating a game that can be played by your EEG signals.

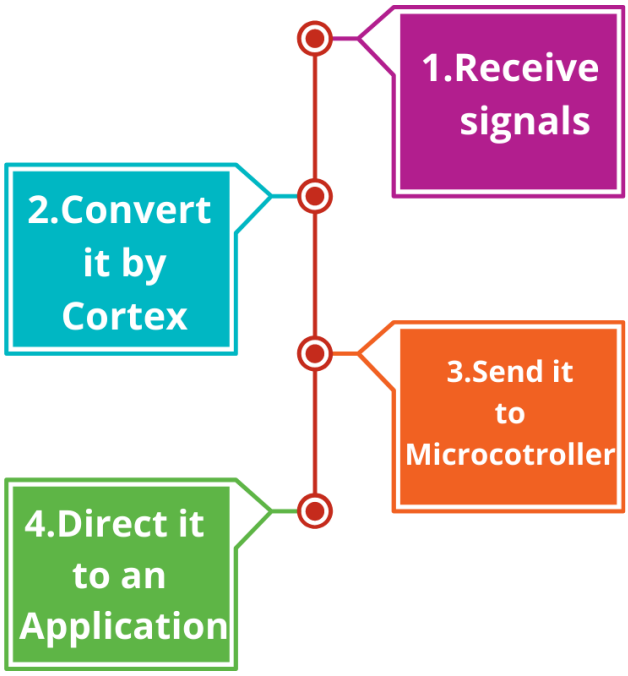


Figure 1.1 project idea

1.2Reason for the idea:

We know that disabled are facing challenges:

- There were difficulties to manage their home even simple things as turn on lights or open doors …etc.
- They couldn't enjoy their life enough.
- Also, they have more driving risks than others.

1.3Idea:

- Suitability for multiple paralysis types.
- Make disables life as autonomous and natural as possible.
- Using brain signals to control your home without making any effort.
- Controlling your car, the way you think.
- Have fun while using your mind's signals to play our designed game.
- Using many types of sensors to implement security in our home model.
- Availability of web application and mobile application to provide a simultaneous simulation of what you think.
- Gathering many technologies in one place.

1.4 Project infograph :

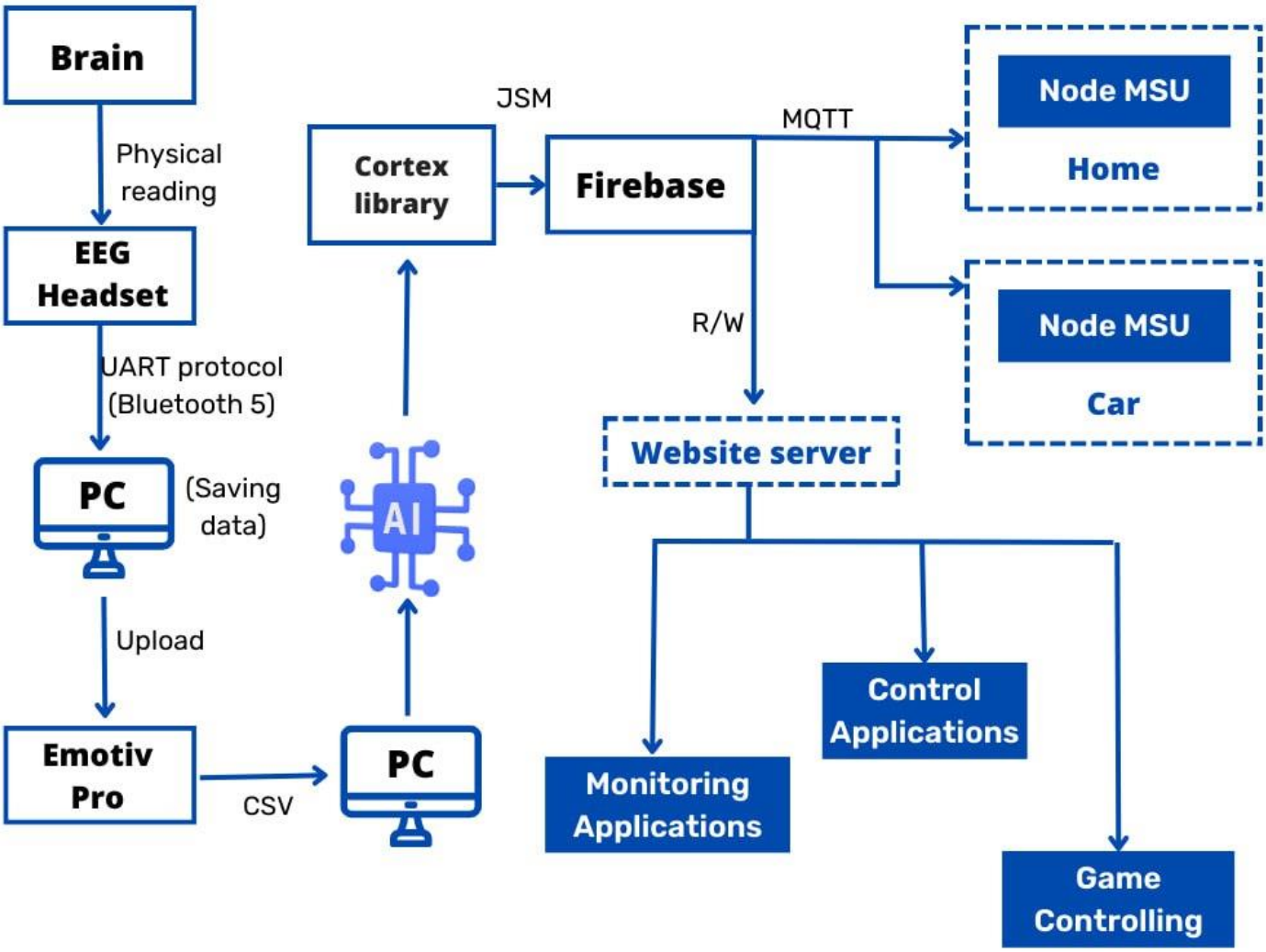


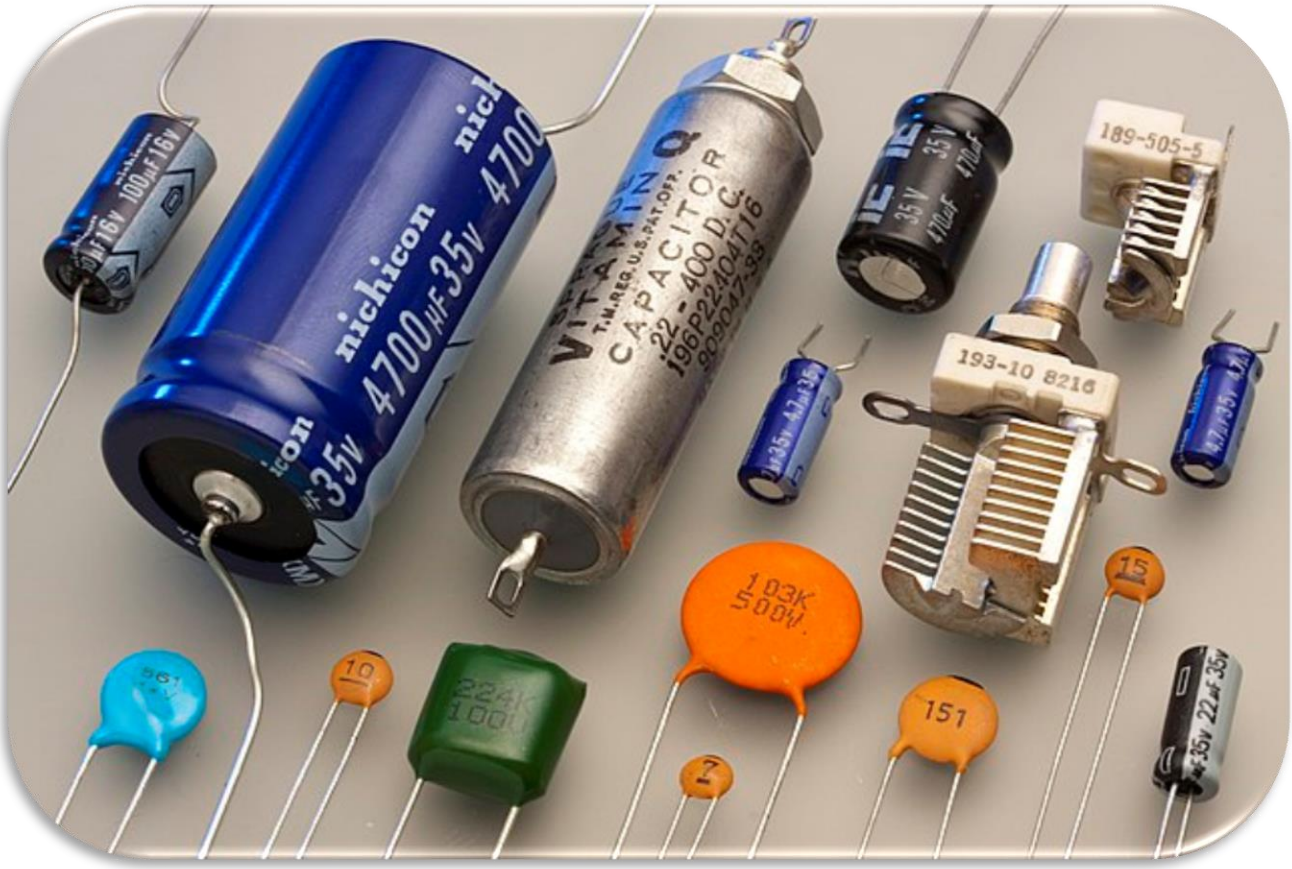
Figure 1.2 project infograph

1.5 Documentation layout:

This project is organized as follows; chapter 2 will discuss components used in details. Chapter 3 will elaborate on the software design steps, a detailed description of how the user deals with the hardware, and some charts describing our drivers' programming. Chapter 4 will show how to use deep learning to detect real-time facial emotions. Chapter 5 will show some techniques of IOT. Chapter 6 will emphasize the web application design and its dashboard connected to the database. Chapter 7 will show the mechanical design, our schematic of circuits, and the final design of our robot. Finally, chapter 8 will give a conclusion of the work and the references used.

Chapter 2

Hardware Components



Chapter 2

Hardware Components

2.1 Overview

Hardware components are the building blocks of any project.

So, in this chapter we will introduce our components and display their functions and features.

2.2 Components

Our components are divided into 4 categories.

- Controllers
- Sensors
- Actuators
- Power

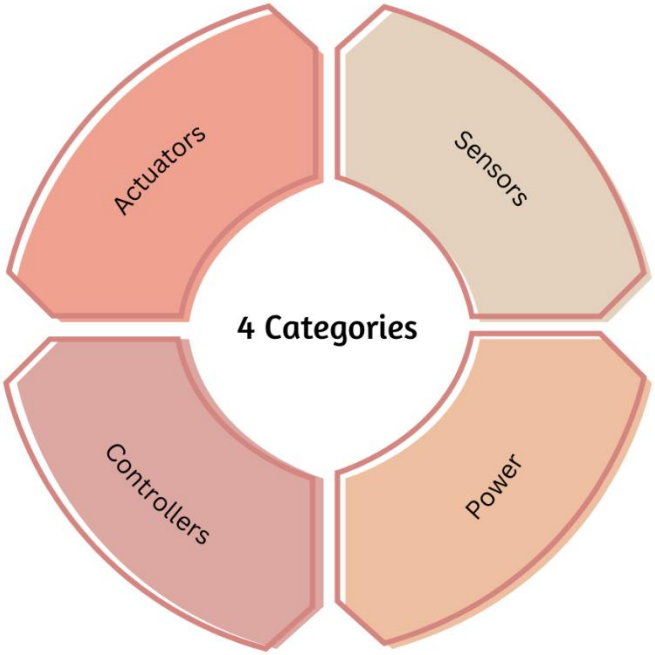


Figure 2.1 Components Categories

2.2.1 Controllers

2.2.1.1 Overview

An Electronic Controller uses electrical signals and digital algorithms to perform its receptive, comparative and corrective functions.

2.2.1.2 Components

2.2.1.2.1 ATmega32-8bit AVR

ATmega32 microcontroller is a type of low-power 8-bit CMOS microcontroller. The architecture of this microcontroller is based on enhanced RISC architecture. ATmega achieves throughputs of 1 MIPS/MHz by executing powerful instruction, which allows the designer to optimize power consumption and speed.

Features

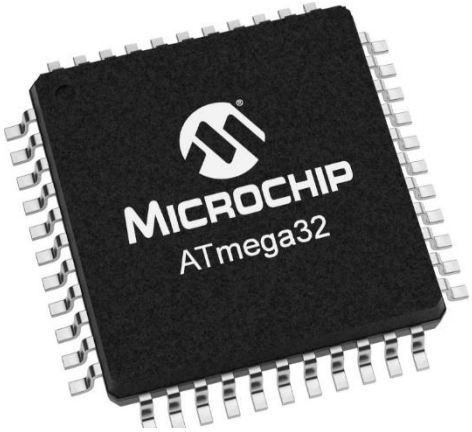


Figure 2.2 ATmega32-8bit

- Kilo bytes of internal Static RAM.
- 32 X 8 general working purpose registers.
- 32 Kilo bytes of in system self-programmable flash program memory.
- 1024 bytes EEPROM.
- Programmable serial USART.
- 8 Channel, 10-bit ADC.
- One 16-bit timer/counter with separate prescaler, compare mode and capture mode.
- Available in 40 pin DIP, 44-pad QFN/MLF and 44-lead QTFP.
- Two 8-bit timers/counters with separate prescalers and compare modes.
- 32 programmable I/O lines.
- In system programming by on-chip boot program.
- Master/slave SPI serial interface.
- PWM channels.
- Programmable watch dog timer with separate on-chip oscillator.
- External and internal interrupt sources.
- Six sleep modes: Idle, ADC noise reduction, power-save, power-down, standby and extended standby.
- Power on reset and programmable brown-out detection.

- Internal calibrated RC oscillator

ATmega32 Microcontroller Pin diagram

It consists of 40-pin Dual Inline Package (DIP) of microcontroller integrated circuit.

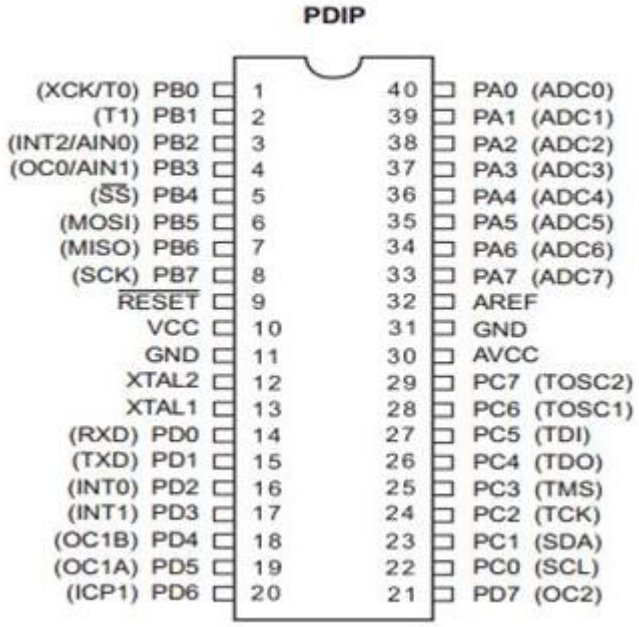


Figure 2.3 ATmega32-8bit Microcontroller pin datagram

2.2.1.2.2 PIC18f452

PIC18f452 a high-performance Enhanced Flash Microcontroller with 8 channels of 10-bit Analogue-to-digital (A/D) converter.

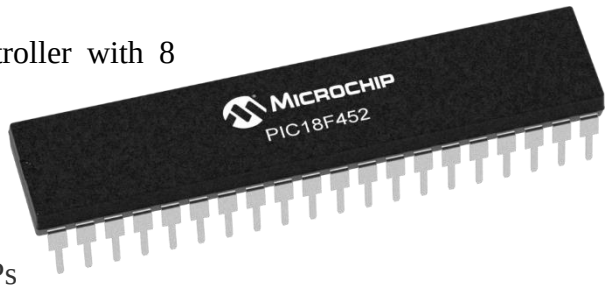


Figure 2.4 PIC18f452

Features:

- High performance RISC CPU - Up to 10MIPs operation & 16-bit wide instructions, 8-bit wide data path.
- Peripheral - 3 external interrupt pins, master synchronous serial port (MSSP) module.
- Analogue - Programmable low voltage detection (PLVD) and programmable brown-out reset (BOR).
- Flash/data EEPROM retention of >40 years.
- Self-reprogrammable under software control.
- Watchdog timer (WDT) with its own on-chip RC oscillator for reliable operation.
- Programmable code protection.

- Power saving sleep mode.
- Single supply 5V In-Circuit Serial Programming™ (ICSP™) via two pins.
- In-Circuit Debug (ICD) via two pins.
- CMOS technology - Low power, high speed flash/EEPROM technology.

2.2.1.2.3 ESP8266

The ESP8266 is a low-cost Wi-Fi microchip, with built-in TCP/IP networking software, and microcontroller capability.

Features:

- Processor: L106 32-bit RISC microprocessor core based on the Tensilica Diamond Standard 106Micro running at 80 or 160 MHz.
- Memory:
  - 32 KiB instruction RAM
  - 32 KiB instruction cache RAM
  - 80 KiB user-data RAM
  - 16 KiB ETS system-data RAM
- External QSPI flash: up to 16 MiB is supported (512 KiB to 4 MiB typically included).

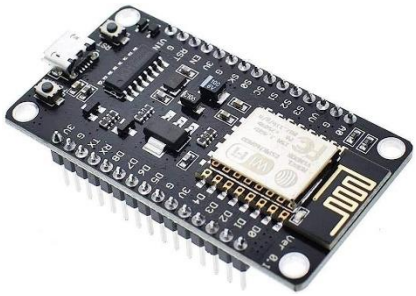


Figure 2.5 ESP8266

- IEEE 802.11 b/g/n Wi-Fi
  - Integrated TR switch, balun, LNA, power amplifier and matching network
  - WEP or WPA/WPA2 authentication, or open networks
- 17 GPIO pins
- Serial Peripheral Interface Bus (SPI)
- I²C (software implementation)
- I²S interfaces with DMA (sharing pins with GPIO)
- UART on dedicated pins, plus a transmit-only UART can be enabled on GPIO2
- 10-bit ADC (successive approximation ADC)

2.2.2 Sensors and Inputs

2.2.2.1 Overview

A sensor is a device that detects and responds to some type of input from the physical environment.

2.2.2.2 Components

2.2.2.2.1 HC-05 Bluetooth module

The HC-05 is a class 2 Bluetooth module designed for transparent wireless serial communication.

Figure2.6 HC-05 Bluetooth module

Features:

- Bluetooth v2.0+EDR
- 2.4GHz ISM band frequency
- Supported baud rate: 9600 (default), 19200,38400, 57600, 115200, 230400, 460800.
- Speed: Asynchronous: 2.1Mbps (Max) / 160 kbps, Synchronous: 1Mbps/1Mbps
- Power supply: 3.6V to 6V DC

2.2.2.2.2 MQ-4 Gas sensor module

It is a Metal Oxide Semiconductor (MOS) type Gas Sensor mainly used to detect the Methane (CNG) gas concentration in the air at home.

Features:

- Good sensitivity to Combustible gas in wide range.
- High sensitivity to CH_4, Natural gas.
- Small sensitivity to alcohol, smoke.
- Fast response Stable and long life.
- Simple drive circuit.

Figure 2.7 MQ-4 Gas sensor module

2.2.3 Actuators

2.2.3.1 Overview

An actuator is a part of a device or machine that helps it to achieve physical movements by converting energy, often electrical, air, or hydraulic, into mechanical force.

2.2.3.2 Components

2.2.3.2.1 Relay

A Relay is a simple electromechanical switch. While we use normal switches to close or open a circuit manually, a Relay is also a switch that connects or disconnects two circuits. But instead of a manual operation, a relay uses an electrical signal to control an electromagnet, which in turn connects or disconnects another circuit.

Features:

- Lighted Indicator
- Mechanical Indicator
- Test Button
- Resistor
- Debounce Delay
- Magnetic Blowout
- Reverse Coil Polarity

Figure 2.8 Relay

2.2.3.2.2 LED

A light-emitting diode (LED) is a semiconductor device that emits light when an electric current flows through it. When current passes through an LED, the electrons recombine with holes emitting light in the process. LEDs allow the current to flow in the forward direction and blocks the current in the reverse direction.

Features:

- A solid-state light technology made with Light Emitting Diodes.
- More energy efficient that incandescent, fluorescent, or other lights.
- Operates at a much lower temperature (cool to the touch).
- Environmentally Friendly.
- Durable and capable of long hours of use (see chart).

Figure 2.9 LED

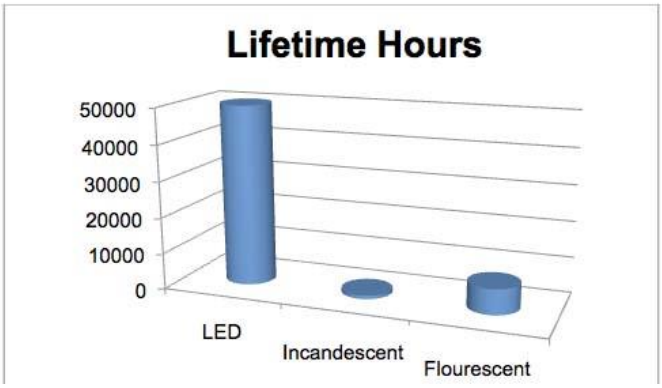


Figure 2.10 Lifetime hours

2.2.3.2.3 LDR module

LDR sensor module is a low-cost digital sensor as well as analog sensor module, which is capable to measure and detect light intensity.

Figure 2.11 LDR module

Features:

- Can detect ambient brightness and light intensity.
- Adjustable sensitivity (via blue digital potentiometer adjustment).
- Operating voltage 3.3V-5V.
- Output Type
  - Analogue voltage output - A0.
  - Digital switching outputs (0 and 1) -D0.
- With fixed bolt hole for easy installation.
- Small board PCB size: 3cm * 1.6cm.
- Power indicator (red) and the digital switch output indicator (green).
- Using LM393 comparator chip, stable.

2.2.3.2.4 LCD Graphical screen

A liquid-crystal display (LCD) is a flat-panel display or another electronically modulated optical device that uses the light-modulating properties of liquid crystals. Liquid crystals do not emit light directly, instead using a backlight or reflector to produce images in color or monochrome.

Figure 2.12 LCD Graphical screen

Features:

- Excellent Resolution.
- Brightness.
- Contrast.
- Operating Voltage: 5V.
- Low power consumption.

2.2.3.2.5 Servo Motor

A servo motor is a self-contained electrical device, that rotate parts of a machine with high efficiency and with great precision.

Features:

- It is able to operate at a wide range of speeds—both high and low—without overheating.
- Maintain sufficient torque at zero speed to hold a load in place.
- Maintain a constant velocity, despite changes in the amount of torque acting on the system.
- A rotation detector (encoder) is mounted on the motor and feeds the rotation position/speed of the motor shaft back to the driver.
- It is capable of accurate rotation angle and speed control.
- It allows for precise control of angular or linear position, velocity, and acceleration.

Figure 2.13 Servo Motor

2.2.3.2.6 Motor Driver

Motor driver is used to control motion of a motor and its direction by feeding current accordingly. Output of a motor driver is in digital form so it uses PWM (Pulse Width Modulation) to control speed of a motor. Motor Driver are basically current amplifiers followed by input signals. It can also drive inductive loads such as relays, solenoids, transformer etc. We used L289N Motor Driver because of the following features.

Features:

- It is a high voltage (up to 46V), high current (up to 4A) dual full-bridge driver designed to accept standardTTL logic levels.

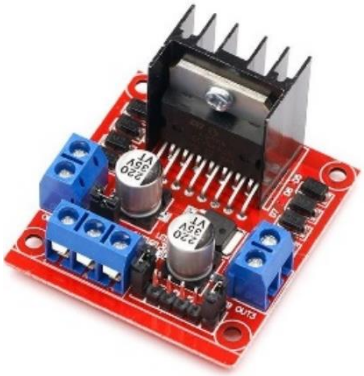


Figure 2.14 L289N Motor Driver

- It is good for battery powered smart cars, toy cars, robots. This is small size, Low weight, Low-cost driver module.
- It works at 2V to 10V supply and maximum output current is 1.5A

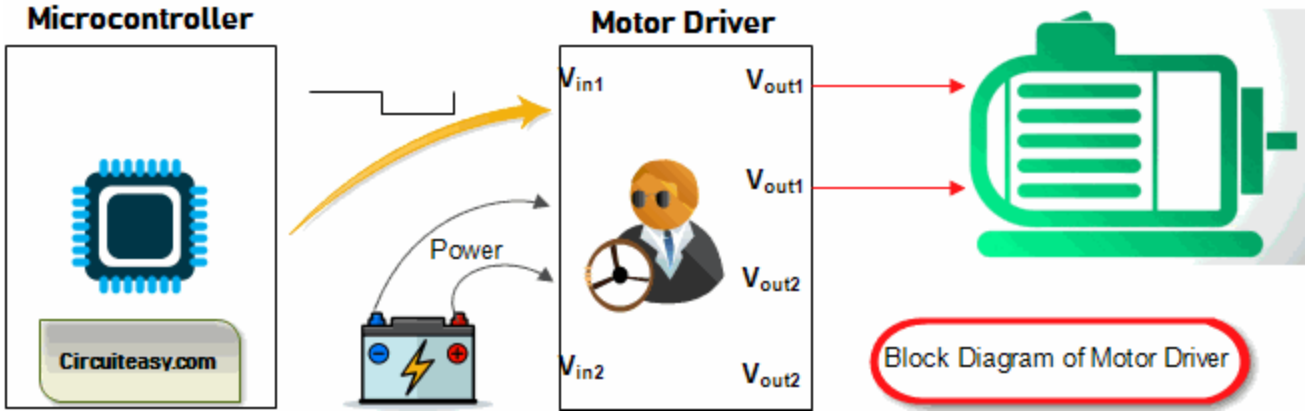


Figure 2.15 Block Diagram of Motor Driver

2.2.3.2.7 BCI Headset

Figure 2.16 Emotive Insight 5-channel EEG Headset

EMOTIV INSIGHT 5-channel mobile EEG boasts advanced electronics that are fully optimized to produce clean, robust signals anytime, anywhere.

Features:

- Improved radio connectivity, updated antenna with Bluetooth 5 support.
- Improved battery life for up to 20hrs.
- Includes Protective Travel Case.
- EEG sensors
  - **5 channels:** AF3, AF4, T7, T8, Pz
  - **2 references:** CMS/DRL references on left mastoid process.
  - **New three prong gummy sensor:** for better hair penetration on Pz.
  - **Sensor material:** Hydrophilic semi-dry polymer.
- Connectivity
  - **Wireless:** Bluetooth Low Energy – Improved radio connectivity, updated antenna with Bluetooth 5 support.
- EEG signals
  - **Sampling rate:** 128 samples per second per channel.

- **Resolution:** 16 bits with 1 LSB = 0.1275µV.
- **Frequency response:** 0.5-43Hz, digital notch filters at 50Hz and 60Hz.
- **Filtering:** Built in digital 5th order Sinc filter.
- **Dynamic range (input referred):** 8400 µV(pp).
- **Coupling mode:** AC coupled.

- Motion sensors
  - **IMU part:** ICM-20948.
  - **Accelerometer:** 3-axis +/-8g.
  - **Gyroscope:** 3-axis +/-2000 dps (converted to 4 normalised quaternions).
  - **Magnetometer:** 3-axis +/- 12 gauss.
  - **Sampling rate:** 32 Hz.
  - **Resolution:** 16 bits.
- Power
  - **Battery:** Internal Lithium Polymer battery 480mAh.
  - **Battery life:** Improve battery life up to 20hrs on a single charge.
- Detections
  - **Mental commands:** neutral + up to 4 pretrained items per training profile.
  - **Performance metrics:** Excitement, Engagement, Relaxation, Interest, Stress, Focus.
  - **Facial Expressions:** Blink, Wink L/R, Surprise, Frown, Smile, Clench.

2.2.4 Power

2.2.4.1 Overview

It is the rate at which electrical energy is transferred by an electric circuit.

2.2.4.2 Components

2.2.4.2.1 Battery

We used 12v battery and 5v battery both at the smart home, 5V for ESP8285 and 12V for motors.

Chapter 3

Embedded systems

Chapter 3

Embedded systems

3.1 Brief of embedded system

An embedded system is a combination of computer hardware and software designed for a specific function. Embedded systems may also function within a larger system. The systems can be programmable or have a fixed functionality. It is embedded as part of a complete device often including electrical or electronic hardware and mechanical parts. Because an embedded system typically controls the physical operations of the machine that it is embedded within, it often has real-time computing constraints. Complexities range from a single microcontroller to a suite of processors with connected peripherals and networks; from no user interface to complex graphical user interfaces. The complexity of an embedded system varies significantly depending on the task for which it is designed. Embedded systems range in size from portable personal devices such as digital watches and MP3 players to bigger machines like home appliances, industrial assembly lines, robots, transport vehicles, traffic light controllers, and medical imaging systems.

3.2 Characteristics of embedded system

- Task Specific
- Strict Design Parameters
- Efficiency
- Microprocessor Or Microcontroller Based
- Limited Memory
- Reliability
- Real Time
- Compact Design
- Minimal Power Dissipation
- Sophisticated Functionality
- Minimal User Interface
- Safety Factor
- Cost-Effective

3.3 Types of Embedded Systems

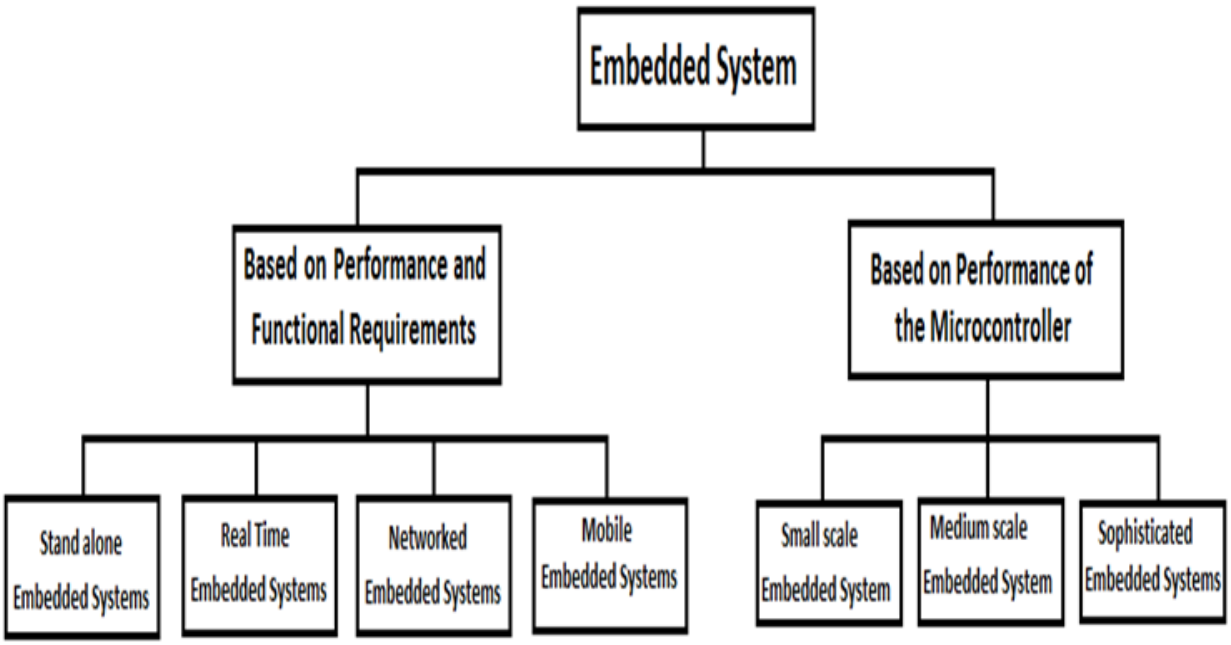


Figure 3.1 Types of the embedded systems

3.3.1 based on the performance and functional requirement

3.3.1.1 Real-Time Embedded Systems:

A Real-Time Embedded System provides output within a defined specific time. That is, real-time embedded systems are designed and created to perform some specific work in a pre-specified time. There are two types of Real-Time Embedded Systems. They are:

a. Soft Real-Time Embedded Systems
b. Hard Real-Time Embedded Systems

3.3.1.2 Stand-Alone Embedded Systems:

Stand-Alone Embedded Systems are those that can work by themselves i.e. they are self-sufficient and do not depend on a host system. Stand-alone embedded systems are made in a way such that input is received, processed, and thereafter the desired output is produced.

3.3.1.3 Networked Embedded Systems:

Networked Embedded Systems depend on a connected network to perform their assigned tasks.

3.3.1.4 Mobile Embedded Systems:

Mobile Embedded Systems are those that are small-sized and can be used in smaller devices. They are used in mobile phones and digital cameras because of their small size. They often have memory constraints and lack a good user interface.

3.3.2 Microcontroller Performance-Based Embedded System

3.3.2.1 Small-Scale Embedded System:

Small Scale Embedded System is normally designed and created using an 8-bit microcontroller. This microcontroller can be battery activated.

3.3.2.2 Medium Scale Embedded System:

Medium Scale Embedded System uses a single 16-bit or 32-bit microcontroller or multiple microcontrollers linked together. These systems have a lot of hardware as well as software complexities, hence are not preferred by many.

3.3.2.3 Sophisticated Embedded System:

Sophisticated Embedded System often functions on multiple algorithms that result in complexities in both hardware and software. They often need a processor that is the configurable and logic array that can be programmed.

3.4 Embedded System Architectures

An embedded system consists of two main parts: embedded hardware and embedded software. The embedded hardware primarily includes the processor, memory, bus, peripheral devices, I/O ports, and various controllers. The embedded software usually contains the embedded operating system and various applications.

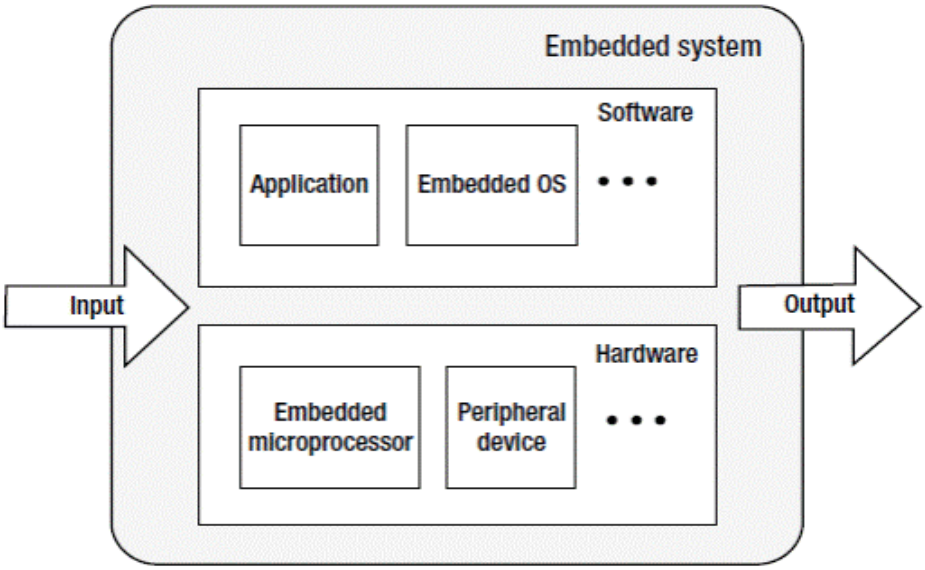


Figure 3.2 the basic architecture of an embedded system

In the embedded system, the hardware and software often collaborate to deal with various input signals from the outside and output the processing results in some form. The input signal may be an ergonomic device (such as a keyboard, mouse, or touch screen) or the output of a sensor circuit in another embedded system. The output may be in the form of sound, light, electricity, or another analog signal, or a record or file for a database.

3.4.1 Hardware Architectures

There are basically two types of architecture that apply to embedded systems: Von Neumann architecture and Harvard architecture.

3.4.1.1 Von Neumann Architecture

The Von Neumann architecture was first proposed by computer scientist John von Neumann. In this architecture, one data path or bus exists for both instruction and data. As a result, the CPU does one

operation at a time. It either fetches an instruction from memory or performs a read/write operation on data. So an instruction fetch and a data operation cannot occur simultaneously, sharing a common bus. Von-Neumann architecture supports simple hardware. It allows the use of a single, sequential memory. Today's processing speeds vastly outpace memory access times, and we employ a very fast but small amount of memory (cache) local to the processor.

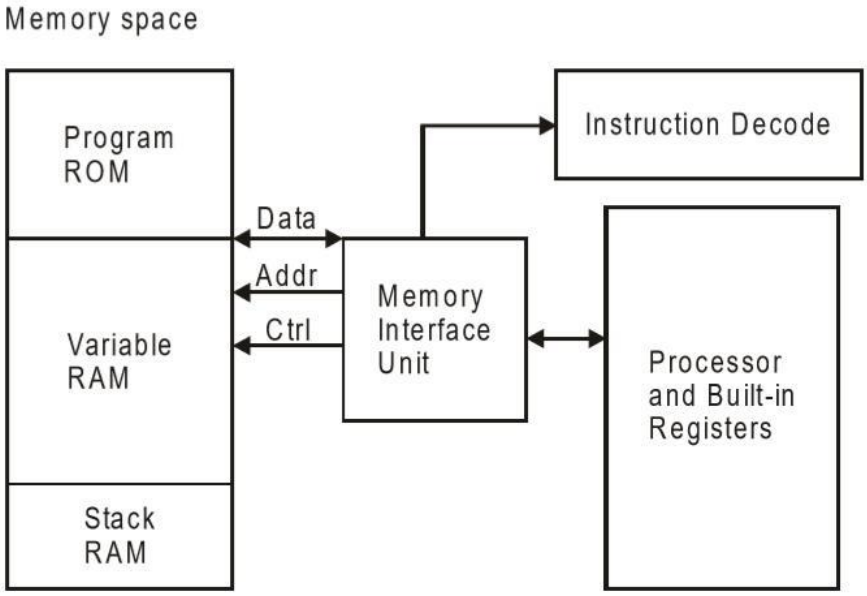


Figure 3.3 the von-Neumann architecture of an embedded system

3.4.1.2 Harvard Architecture

The Harvard architecture offers separate storage and signal buses for instructions and data. This architecture has data storage entirely contained within the CPU, and there is no access to the instruction storage as data. Computers have separate memory areas for program instructions and data using internal data buses, allowing simultaneous access to both instructions and data. Programs needed to be loaded by an operator; the processor could not boot itself. In Harvard architecture, there is no need to make the two memories share properties.

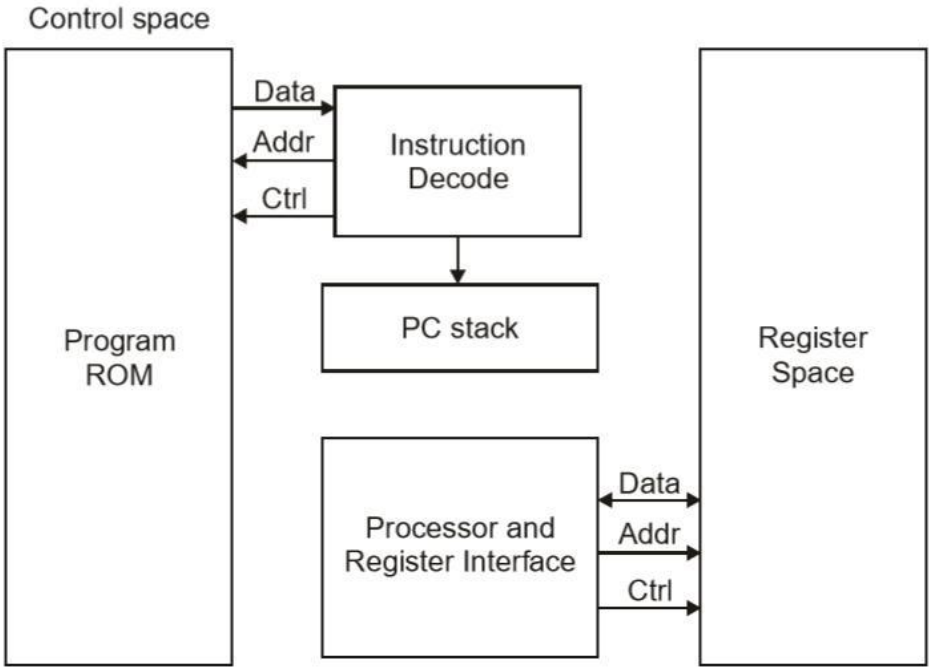


Figure 3.4 Harvard architecture of an embedded system

3.4.2 Software Architectures

There are several common embedded system software architectures, which become necessary as embedded systems grow and become more complex in scale. These include:

- Simple control loops called subroutines, which manage a specific part of the hardware or embedded programming.
- Interrupt controlled systems have two loops: a main one and a secondary one. Interruptions in the loops trigger tasks.
- Cooperative multitasking is essentially a simple control loop located in an application programming interface (API).
- Preemptive multitasking or multithreading is often used with an RTOS and features synchronization and task-switching strategies.

3.5 How do Embedded Systems Work

Microcontrollers play a significant role in the working of embedded systems. Apart from that, digital signal processors (DSP), field-programmable gate arrays (FPGA), GPU technology, application-specific integrated circuits (ASIC), and gate arrays also enable the embedded systems to work efficiently. The firmware is stored in a ROM or memory chip that runs with limited computer hardware resources.

If we go with the steps on how embedded systems work, then:

- A user first gives the analog input through sensors, keypads, touch buttons, etc.
- Following that, the system gives out the output by processing the information and converting it into a digital one using a monitor, LCS, etc.

3.6 Applications of Embedded Systems

Figure 3.5 Applications of embedded system

- Consumer electronics: Televisions and digital cameras, computer printers, video game consoles, and home entertainment systems like PS4.
- Household appliances: Refrigerators, washing machines, microwave ovens, air conditioners.
- Medical equipment: Scanners like those for MRI, CT, ECG machines, and devices to monitor blood pressure and heartbeat.
- Automobiles: Fuel injection systems, anti-lock braking systems, music and entertainment systems.
- Industrial applications: Assembly lines, systems for feedback, systems for data collection.
- Aerospace: Systems for navigation and guidance, GPS.
- Communications: Routers, satellite phones.

3.7 Advantages and Disadvantages of embedded system

There are some important advantages of embedded systems are given below:

- The embedded system is easy for mass production.
- The embedded system is highly reliable.
- It has very few interconnections.
- The embedded system is small in size.
- The embedded system has less expensive.
- It has a fast operation.
- It has improved product quality.
- It optimizes the use of system resources.
- It has low power operation.

There are some important disadvantages of embedded systems are given below:

- The embedded systems are hard for maintaining as it is used and throw device.
- It has no technological improvement.
- Less power supply durability if it is battery operated.
- It has hard to take backup of embedded files.

3.8 Our driver codes

3.8.1 Flowchart describes car driver code

Car driver code refers to the software code that controls the behavior of a car's electronic systems and allows it to be driven. This code is often written in a low-level programming language, and is responsible for managing the car's engine, transmission, brakes, steering, and other systems.

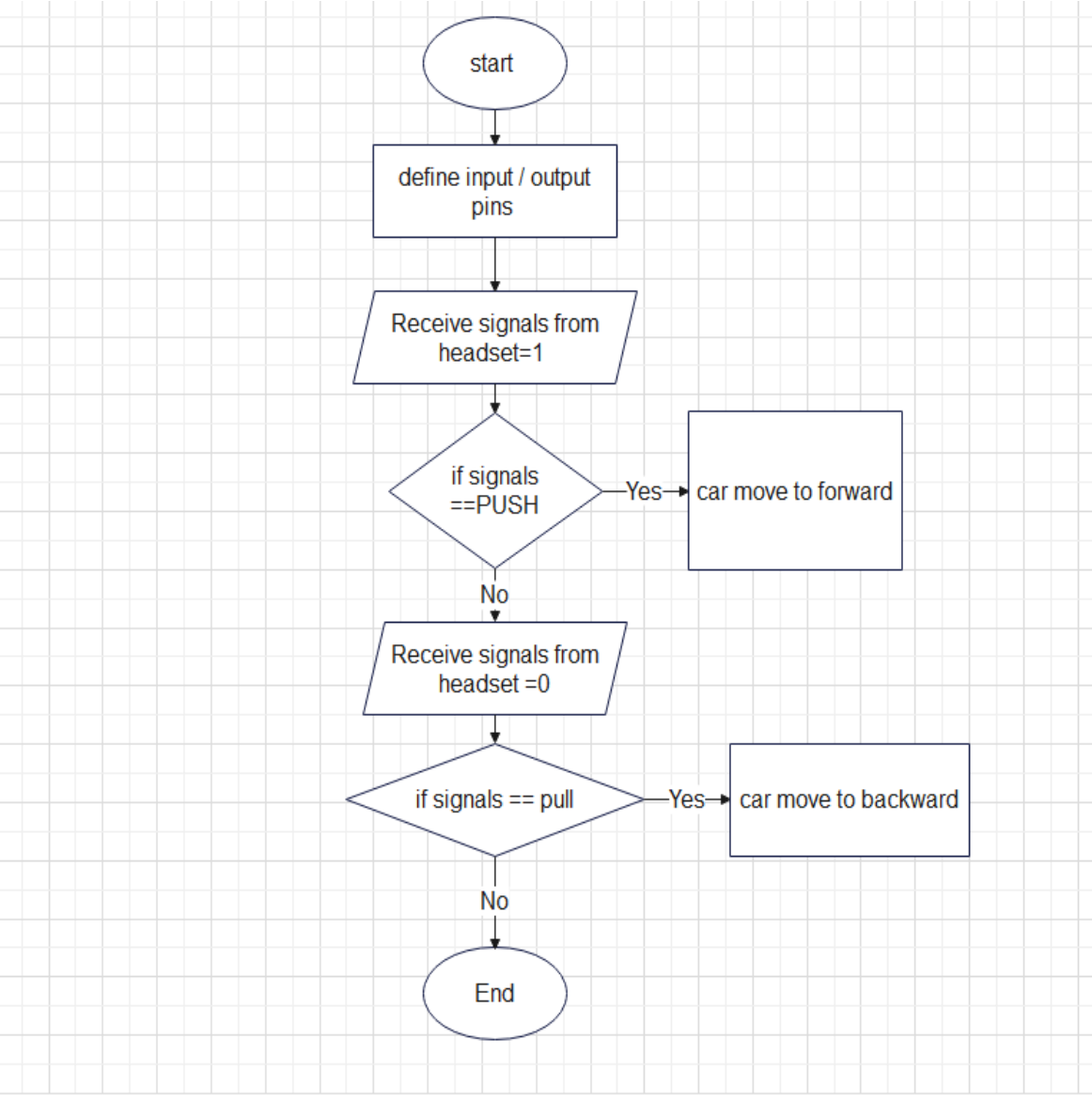


Figure 3.6 Car driver code flowchart

3.8.2 Flowchart describes the smart home driver code

Software code known as "smart home driver code" regulates how a smart home's electronic systems behave and enables users to interface with and manage their home automation equipment.

Typically, a controller is used by the driver code to interface with smart home appliances. It might also communicate with the sensor.

Features like speech recognition, scheduling, and automation may be included in the code.

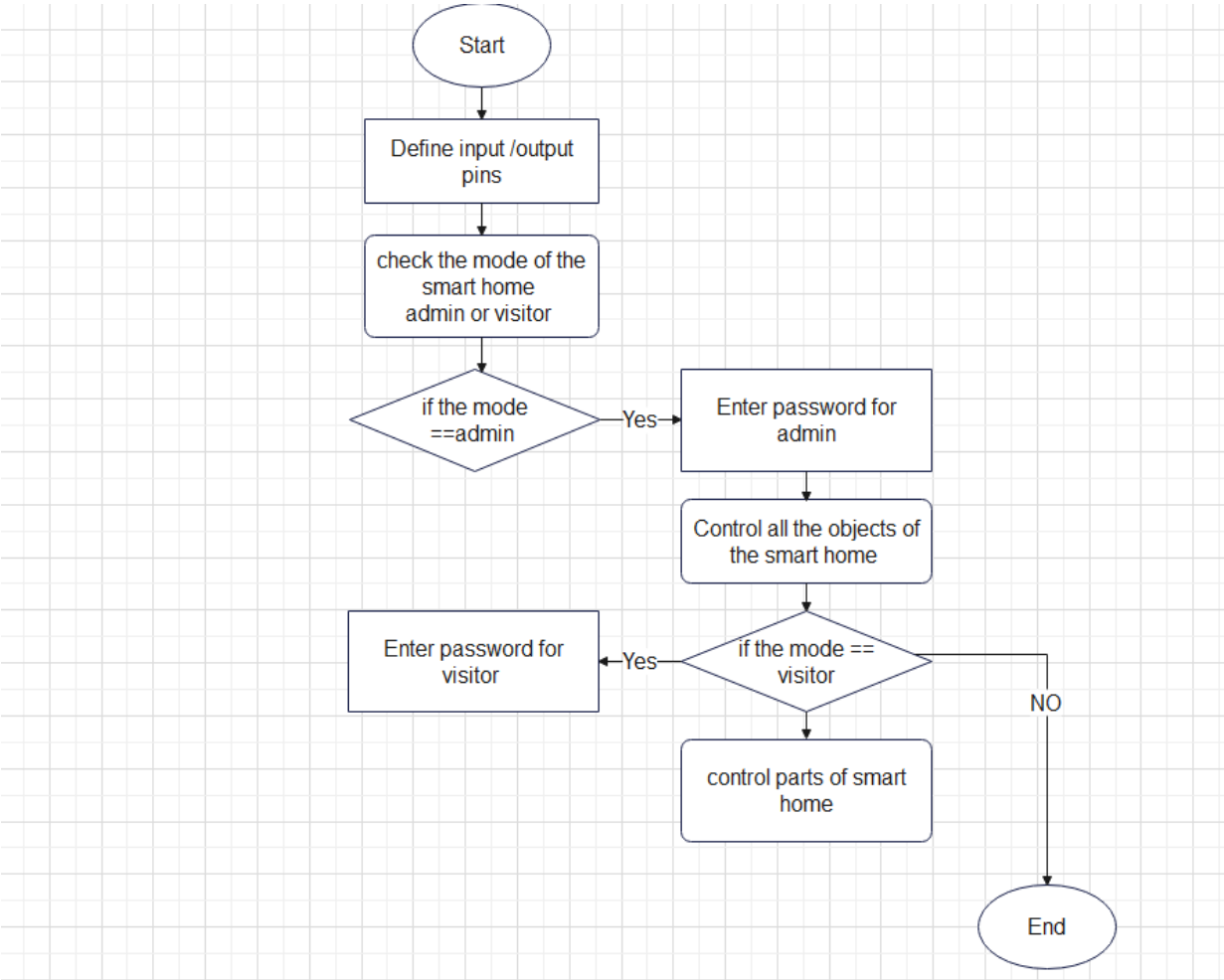


Figure 3.7 Smart home driver code flowchart

3.8.3 Flowchart describes sensors driver code

We use pic18f452 which supports 3 sensors. The temperature sensor has a certain temperature limit. If the temperature equals to 37, the fan will be turned on, and if it is less than 37, the fan will be turned off. Also, in the ultrasonic sensor, the distance must be checked, if the distance equals 3cm the car way is changed, and if it is less, the car will stop. In the gas sensor, if there is a gas leak, alarm will be triggered.

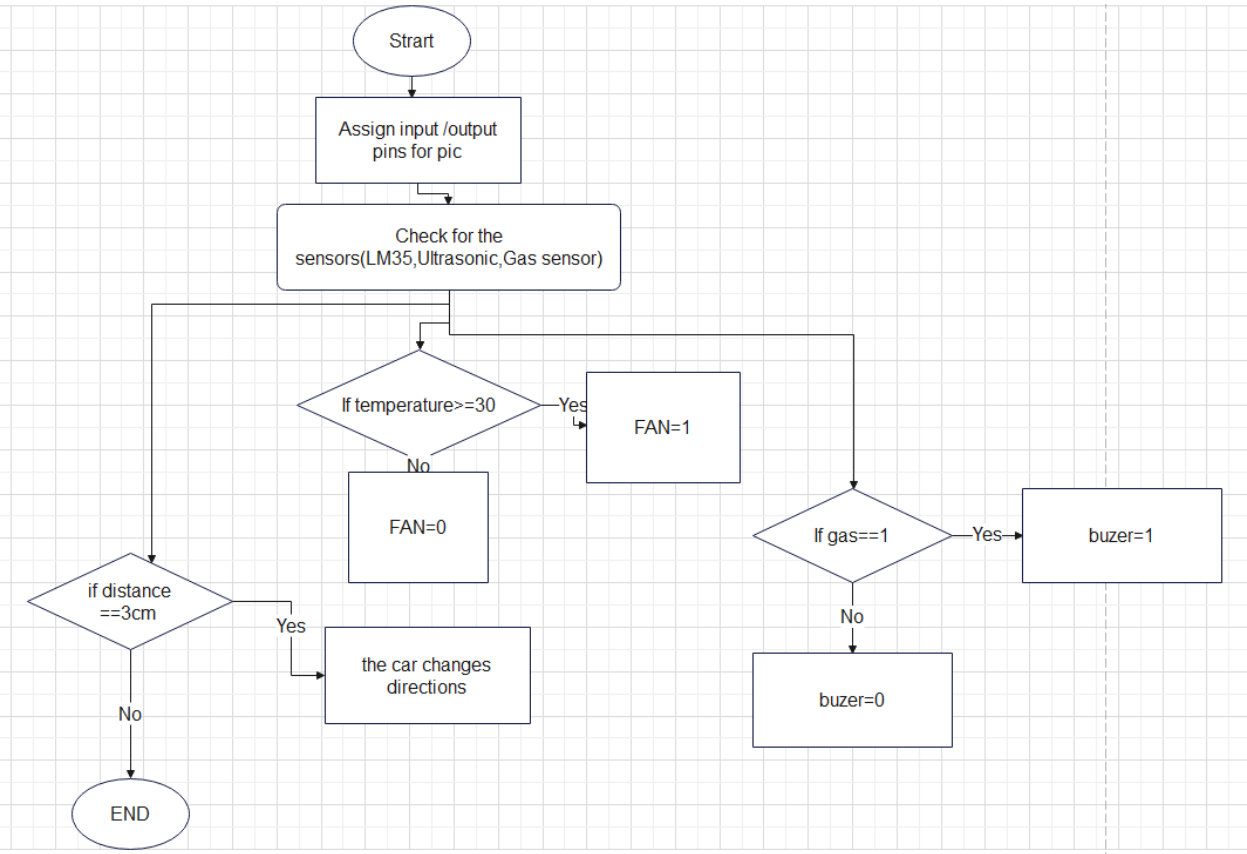


Figure 3.8 sensors driver code flowchart

3.9 Visual Diagrams

3.9.1 Smart home visual diagram

Here, we visually declare the internal and external systems that control the entire house.

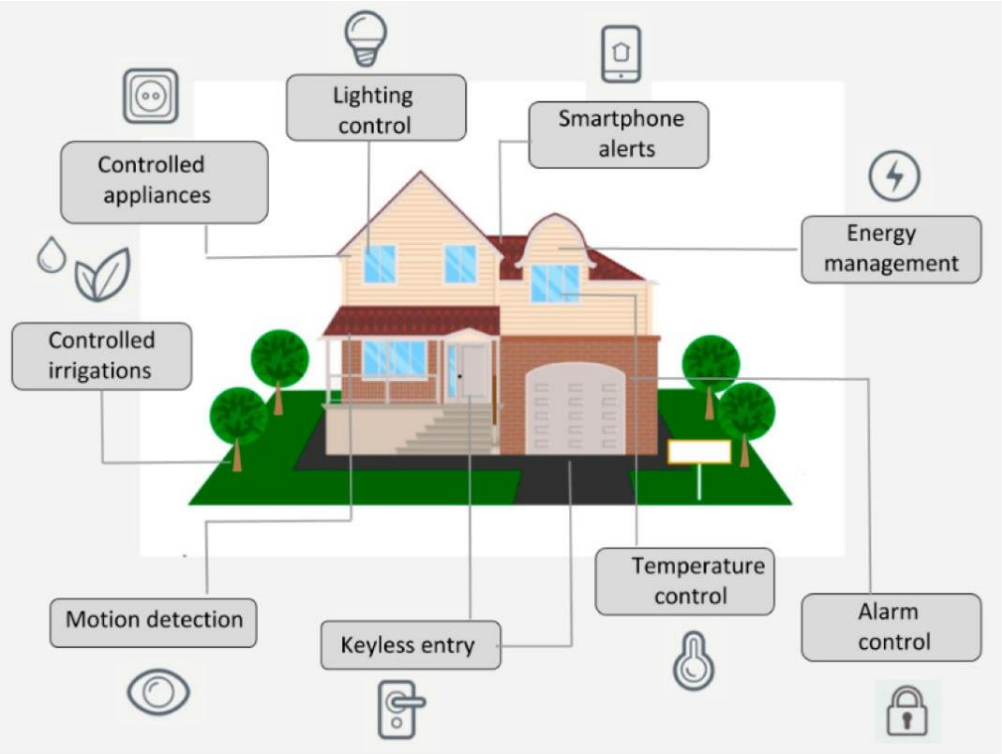


Figure 3.9 Smart home visual diagram

3.9.2 Smart car visual diagram

We used a node MCU in our smart automobile to get data and perform actions. Here is a visual declaration to the other elements as well.

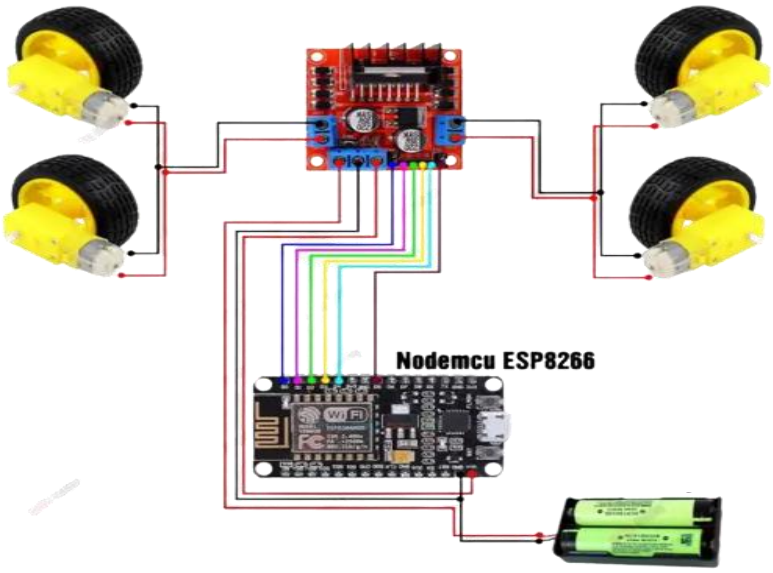
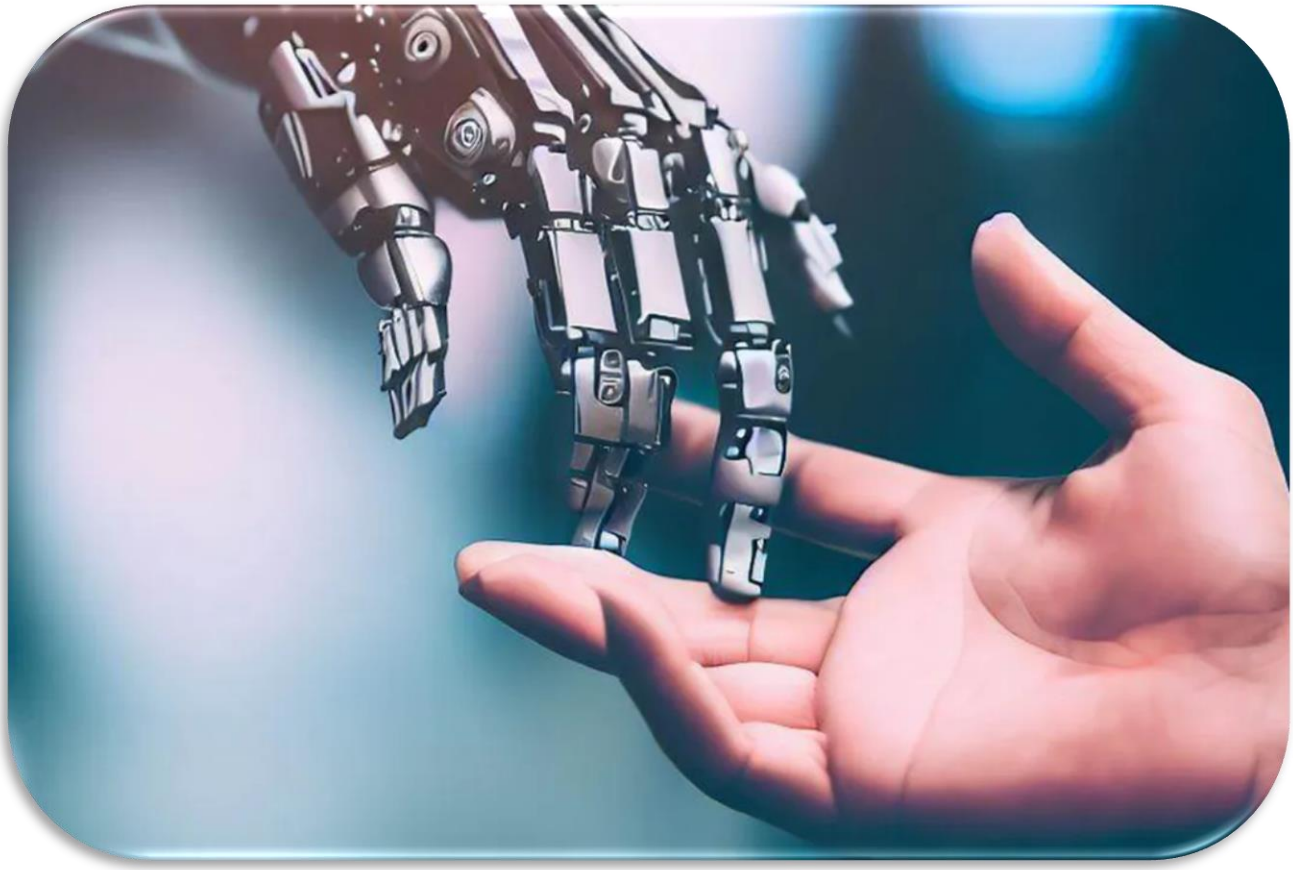


Figure 3.9 Smart car visual diagram

Chapter 4

Artificial Intelligence



Chapter 4
Artificial Intelligence

4.1 What is AI?

AI is a concept that has been around, formally, since the 1950s, when it was defined as a machine's ability to perform a task that would've previously required human intelligence. This is quite a broad definition and one that has been modified over decades of research and technological advancements.

When you consider assigning intelligence to a machine, such as a computer, it makes sense to start by defining the term 'intelligence' -- especially when you want to determine if an artificial system is truly deserving of it.

4.2 Types of AI

Artificial intelligence can be divided into three widely accepted subcategories: narrow AI, general AI, and super AI.

4.2.1 Narrow AI

Artificial narrow intelligence (ANI) is crucial to voice assistants, such as Siri, Alexa, and Google Assistant. This category includes intelligent systems that have been designed or trained to carry out specific tasks or solve particular problems, without being explicitly designed to do so.

ANI might often be referred to as weak AI, as it doesn't possess general intelligence, but some examples of the power of narrow AI include the above voice assistants, and also image-recognition systems, technologies that respond to simple customer service requests, and tools that flag inappropriate content online.

ChatGPT is an example of ANI, as it is programmed to perform a specific task, which is to generate text responses to the prompts it is given.

4.2.2 General AI

Artificial general intelligence (AGI), also known as strong AI, is still a hypothetical concept as it involves a machine understanding and performing vastly different tasks based on its accumulated experience. This type of

intelligence is more on the level of human intellect, as AGI systems would be able to reason and think like a human.

Like a human, AGI would potentially be able to understand any intellectual task, think abstractly, learn from its experiences, and use that knowledge to solve new problems. Essentially, we're talking about a system or machine capable of common sense, which is currently not achievable with any form of available AI.

Developing a system with its own consciousness is still, presumably, a fair way in the distance, but it is the ultimate goal in AI research.

4.2.3 Super AI

Artificial super intelligence (ASI) is a system that wouldn't only rock humankind to its core, but could also destroy it. If that sounds straight out of a science fiction novel, it's because it kind of is: ASI is a system where the intelligence of a machine surpasses all forms of human intelligence, in all aspects, and outperforms humans in every function.

An intelligent system that can learn and continuously improve itself is still a hypothetical concept. However, it's a system that, if applied effectively and ethically, could lead to extraordinary progress and achievements in medicine, technology, and more.

4.3 Machine Learning

The biggest quality that sets AI aside from other computer science topics is the ability to easily automate tasks by employing machine learning, which lets computers learn from different experiences rather than being explicitly programmed to perform each task. This capability is what many refer to as AI, but machine learning is actually a subset of artificial intelligence.

Machine learning involves a system being trained on large amounts of data, so it can learn from mistakes, and recognize patterns in order to accurately make predictions and decisions, whether they've been exposed to the specific data or not.

Examples of machine learning include image and speech recognition, fraud protection, and more. One specific example is the image recognition system when users upload a photo to Facebook. The social media network can analyze the image and recognize faces, which leads to recommendations to tag different friends. With time and practice, the system hones this skill and learns to make more accurate recommendations.

It is generally split into two main categories: supervised, and unsupervised learning.

4.3.1 Supervised learning

This is a common technique for teaching AI systems by using many labelled examples that have been categorized by people. These machine-learning systems are fed huge amounts of data, which has been annotated to highlight the features of interest -- you're essentially teaching by example.

If you wanted to train a machine-learning model to recognize and differentiate images of circles and squares, you'd get started by gathering a large dataset of images of circles and squares in different contexts, such as a drawing of a planet for a circle, or a table for a square, for example, complete with labels for what each shape is.

The algorithm would then learn this labeled collection of images to distinguish the shapes and its characteristics, such as circles having no corners and squares having four equal sides. After it's trained on the dataset of images, the system will be able to see a new image and determine what shape it finds.

4.3.2 Unsupervised learning

In contrast, unsupervised learning uses a different approach, where algorithms try to identify patterns in data, looking for similarities that can be used to categorize that data.

An example might be clustering together fruits that weigh a similar amount or cars with a similar engine size. The algorithm isn't set up in advance to pick out specific types of data; it simply looks for data with similarities that it can group, for example, grouping customers together based on shopping behavior to target them with personalized marketing campaigns.

4.3.3 Reinforcement learning

In reinforcement learning, the system attempts to maximize a reward based on its input data, basically going through a process of trial and error until it arrives at the best possible outcome.

Consider training a system to play a video game, where it can receive a positive reward if it gets a higher score and a negative reward for a low score. The system learns to analyze the game and make moves, and then learns solely from the rewards it receives, reaching the point of being able to play on its own and earn a high score without human intervention.

Reinforcement learning is also used in research, where it can help teach autonomous robots about the optimal way to behave in real-world environments.

4.4 Deep learning

Part of the machine-learning family, deep learning involves training artificial neural networks with three or more layers to perform different tasks. These neural networks are expanded into sprawling networks with a large number of deep layers that are trained using massive amounts of data.

Deep-learning models tend to have more than three layers, and can have hundreds of layers. It can use supervised or unsupervised learning or a combination of both in the training process.

Because deep-learning technology can learn to recognize complex patterns in data using AI, it is often used in natural language processing (NLP), speech recognition, and image recognition.

4.5 Libraries used

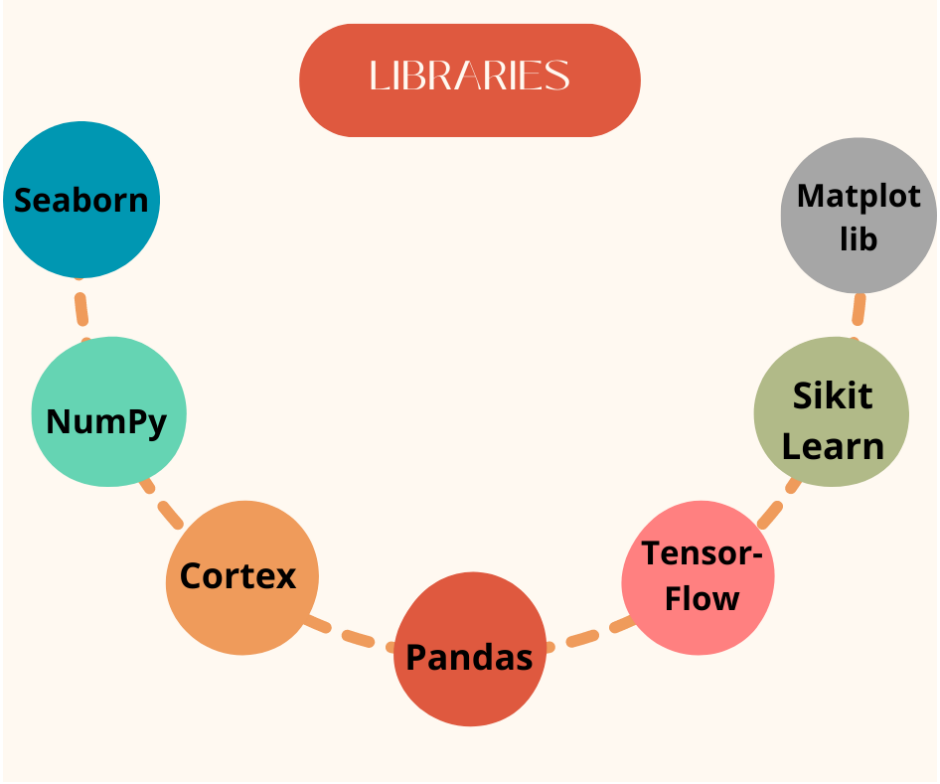


Figure 4.1 Libraries

4.5.1 Seaborn

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures.

Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them.

4.5.2 NumPy

NumPy is its de-facto standard Python programming language library that supports large, multi-dimensional arrays and matrices, and comes with a vast collection of high-level mathematical functions to operate on these arrays.

4.5.3 Cortex

It is a deep learning neural network that creates robust 3D spatial data from a wide variety of capture devices including Lidar cameras, the Matterport Pro2, 360 cameras, and even smartphones. Cortex makes our all-in-one 3D data platform the most powerful on the market.

4.5.4 Pandas

Python Pandas is an open-source toolkit which provides data scientists and analysts with data manipulation and analysis capabilities using the Python programming language. The Pandas library is very popular in the preprocessing phase of machine learning and deep learning. But now you can do more with it.

4.5.5 Tensorflow

It can train and run deep neural networks for handwritten digit classification, image recognition, word embeddings, recurrent neural networks, sequence-to-sequence models for machine translation, natural language processing, and PDE (partial differential equation)-based simulations. Best of all, TensorFlow supports production prediction at scale, with the same models used for training.

4.5.6 Scikit Learn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression,

clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

4.5.7 Matplotlib

Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. Matplotlib is written in Python and makes use of NumPy, the numerical mathematics extension of Python. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter. It can be used in Python and IPython shells, Jupyter notebook and web application servers also.

4.6 Algorithms

While a general algorithm can be simple, AI algorithms are by nature more complex. AI algorithms work by taking in training data that helps the algorithm to learn. How that data is acquired and is labeled marks the key difference between different types of AI algorithms.

At the core level, an AI algorithm takes in training data (labeled or unlabeled, supplied by developers, or acquired by the program itself) and uses that information to learn and grow. Then it completes its tasks, using the training data as a basis. Some types of AI algorithms can be taught to learn on their own and take in new data to change and refine their process. Others will need the intervention of a programmer in order to streamline.

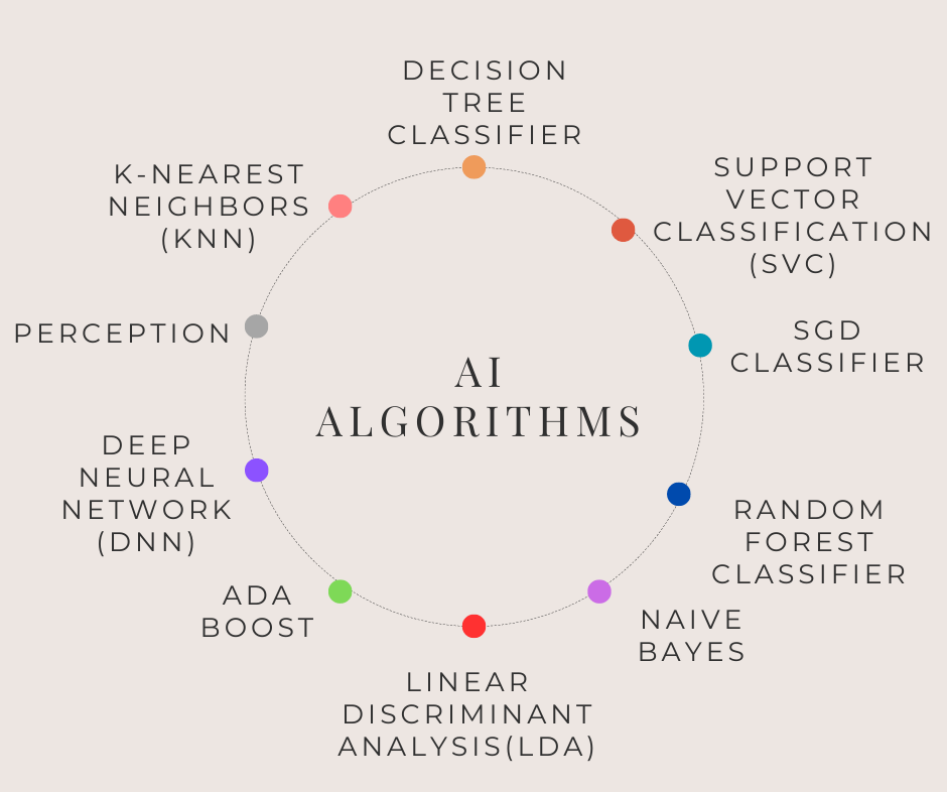


Figure 4.2 Algorithms

4.6.1 Decision Tree Classifier

- Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.
- In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.

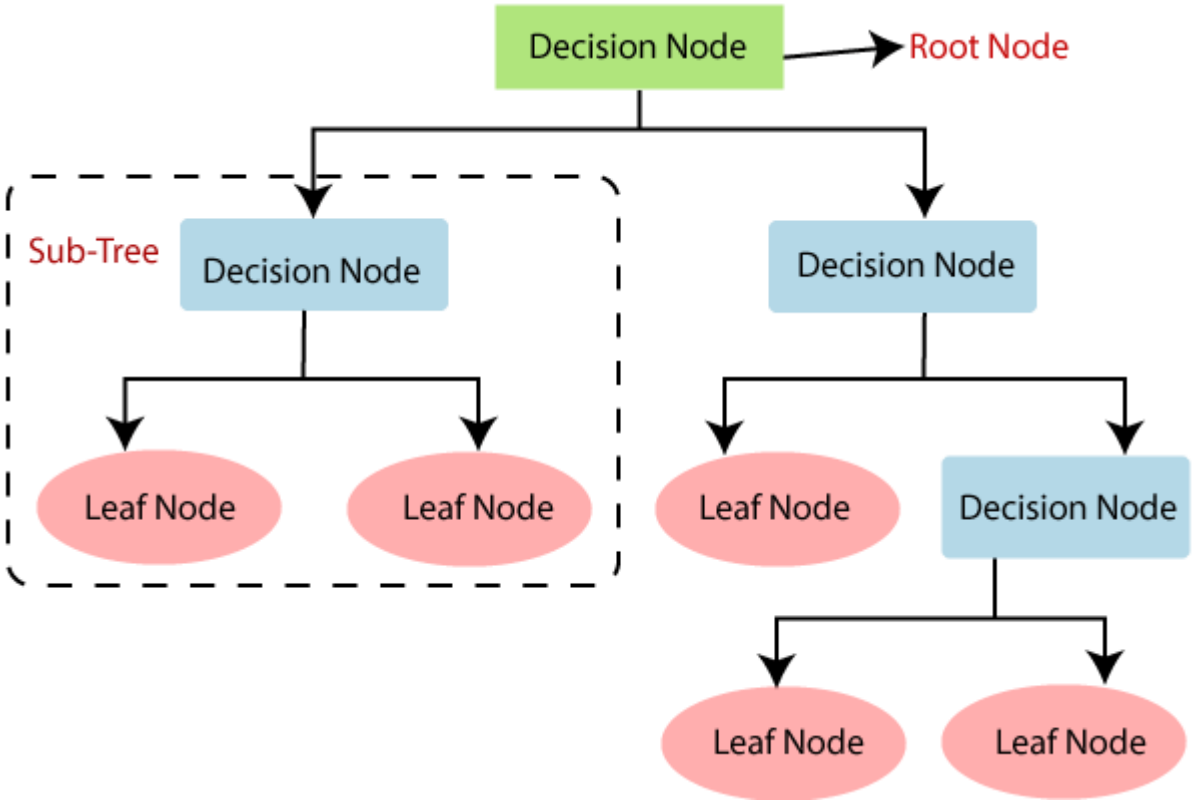


Figure 4.3 General Structure of a decision tree

4.6.1.1 Why use Decision Trees?

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

- Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- The logic behind the decision tree can be easily understood because it shows a tree-like structure.

4.6.2 Support Vector Classification (SVC)

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

4.6.2.1 Types of SVM

SVM can be of two types:

- Linear SVM: Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- Non-linear SVM: Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

4.6.3 SGD Classifier

The SGD Classifier is a type of linear classifier that uses stochastic gradient descent (SGD) optimization to minimize the loss function and learn the optimal weights for the model. It is a simple and efficient algorithm that can be used for both binary and multiclass classification problems.

The SGD Classifier works by iteratively updating the weights of the model based on the gradient of the loss function with respect to the weights. The loss function used in the SGD Classifier is typically the hinge loss for linear SVM classification or the log loss for logistic regression classification.

During each iteration, the SGD Classifier randomly selects a subset of the training data (a mini-batch) to compute the gradient of the loss function. This random selection of the data is what makes the algorithm stochastic.

The SGD Classifier is a popular choice for large-scale machine learning problems because it can handle large datasets efficiently and can be parallelized easily. It is also well-suited for online learning, where the model can be updated continuously as new data becomes available.

However, the SGD Classifier can be sensitive to the learning rate and other hyperparameters, so careful tuning is necessary to obtain good results. Additionally, because it relies on a stochastic approximation of the gradient, it can be less stable than other optimization methods like batch gradient descent.

4.6.4 Random Forest Classifier

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the max_samples parameter if bootstrap=True (default), otherwise the whole dataset is used to build each tree.

4.6.5 Naive Bayes Classifier

The Naïve Bayes classifier is a supervised machine learning algorithm, which is used for classification tasks, like text classification. It is also part of a family of generative learning algorithms, meaning that it seeks to model the distribution of inputs of a given class or category. Unlike discriminative classifiers, like logistic regression, it does not learn which features are most important to differentiate between classes.

4.6.6 Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis, or LDA, is a machine learning algorithm that is used to find the Linear Discriminant function that best classifies or discriminates or separates two classes of data points. LDA is a supervised learning algorithm, which means that it requires a labelled training set of data points in order to learn the Linear Discriminant function.

The following are some of the benefits of using LDA:

- LDA is used for classification problems.
- LDA is a powerful tool for dimensionality reduction.
- LDA is not susceptible to the "curse of dimensionality" like many other machine learning algorithms.

4.6.7 Ada Boost

AdaBoost, also called Adaptive Boosting, is a technique in Machine Learning used as an Ensemble Method. The most common estimator used with AdaBoost is decision trees with one level which means Decision trees with only 1 split. These trees are also called **Decision Stumps.**

4.6.8 Deep Neural Network (DNN)

Deep Neural Networks (DNNs) have become a promising solution to inject AI in our daily lives from self-driving cars, smartphones, games, drones, etc. In most cases, DNNs were accelerated by server equipped with numerous computing engines, e.g., GPU, but recent technology advance requires energy-efficient acceleration of DNNs as the modern applications moved down to mobile computing nodes. Therefore, Neural Processing Unit (NPU) architectures dedicated to energy-efficient DNN acceleration became essential. Despite the fact that training phase of DNN requires precise number representations, many researchers proved that utilizing smaller bit-precision is enough for inference with low-power consumption.

4.6.9 Perception

Machine perception is the capability of a computer to take in and process sensory information in a way that's similar to how humans perceive the world. It may rely on sensors that mimic common human senses — sight, sound, touch, taste — as well as taking in information in ways that humans cannot.

Sensing and processing information by a machine generally requires specialized hardware and software. It's a multistep process to take in and then convert or translate raw data into the overall scan and detailed selection of focus by which humans (and animals) perceive their world.

4.6.10 K-Nearest Neighbors (KNN)

- K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.
- It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data

4.6.10.1 Advantages of KNN Algorithm

- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

4.6.10.2 Disadvantages of KNN Algorithm

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

4.7 Project AI

Firstly, data acquisition begins by collecting data after the user puts on the headset and launches the emotiv pro programme. In order to be dealt with, the data is stored as physical signals. This information is now available for use after being uploaded to the company's cloud.in csv format,

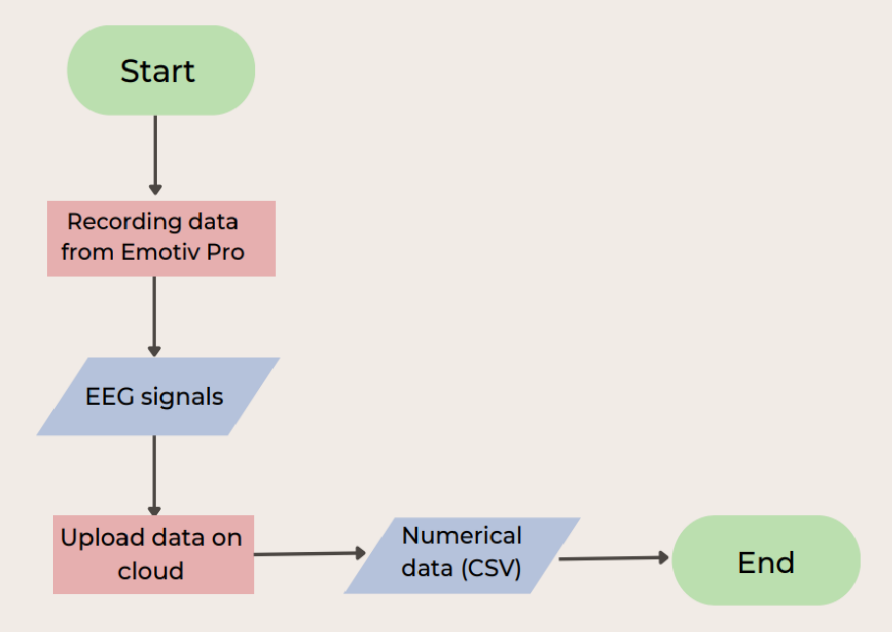


Figure 4.4 Data Acquisition Flowchart

And now we are able to work with the data in .csv format. Then, only the data (which are the 5 columns) that will affect the categorization alone are used to prepare the data.

Data before preparation:

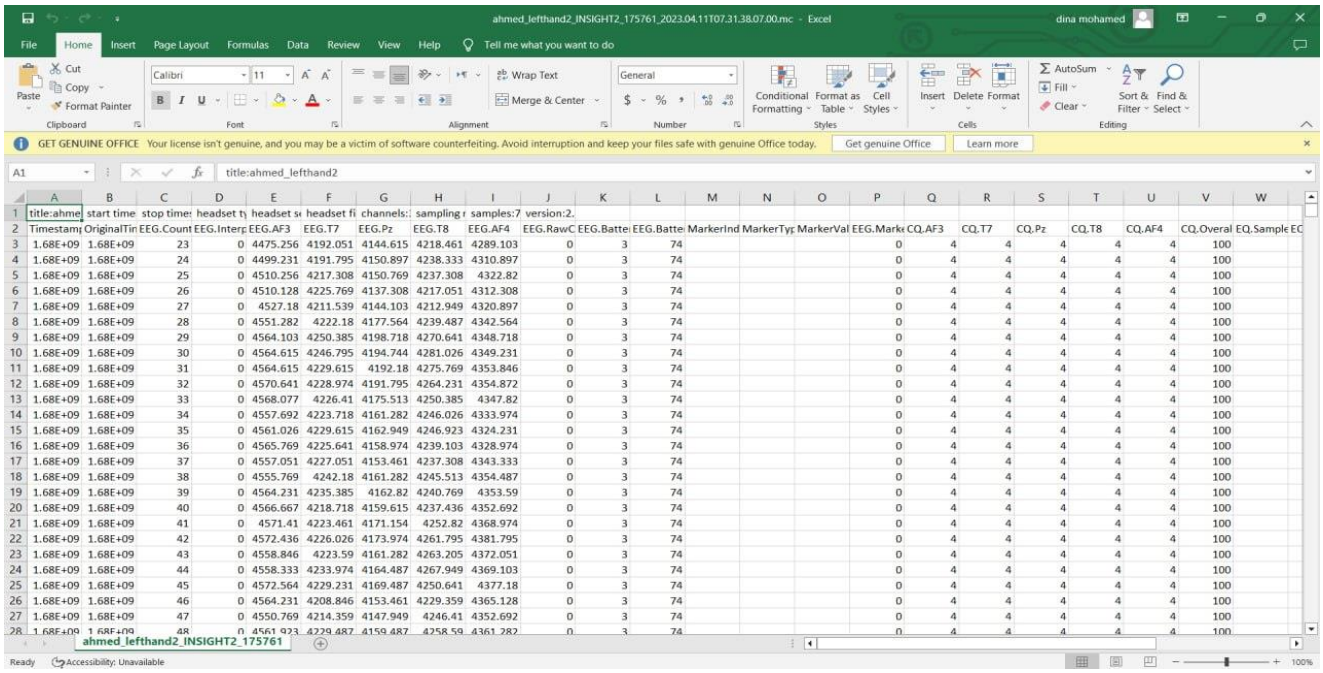


Figure 4.5 Data before preparation

Data after preparation:

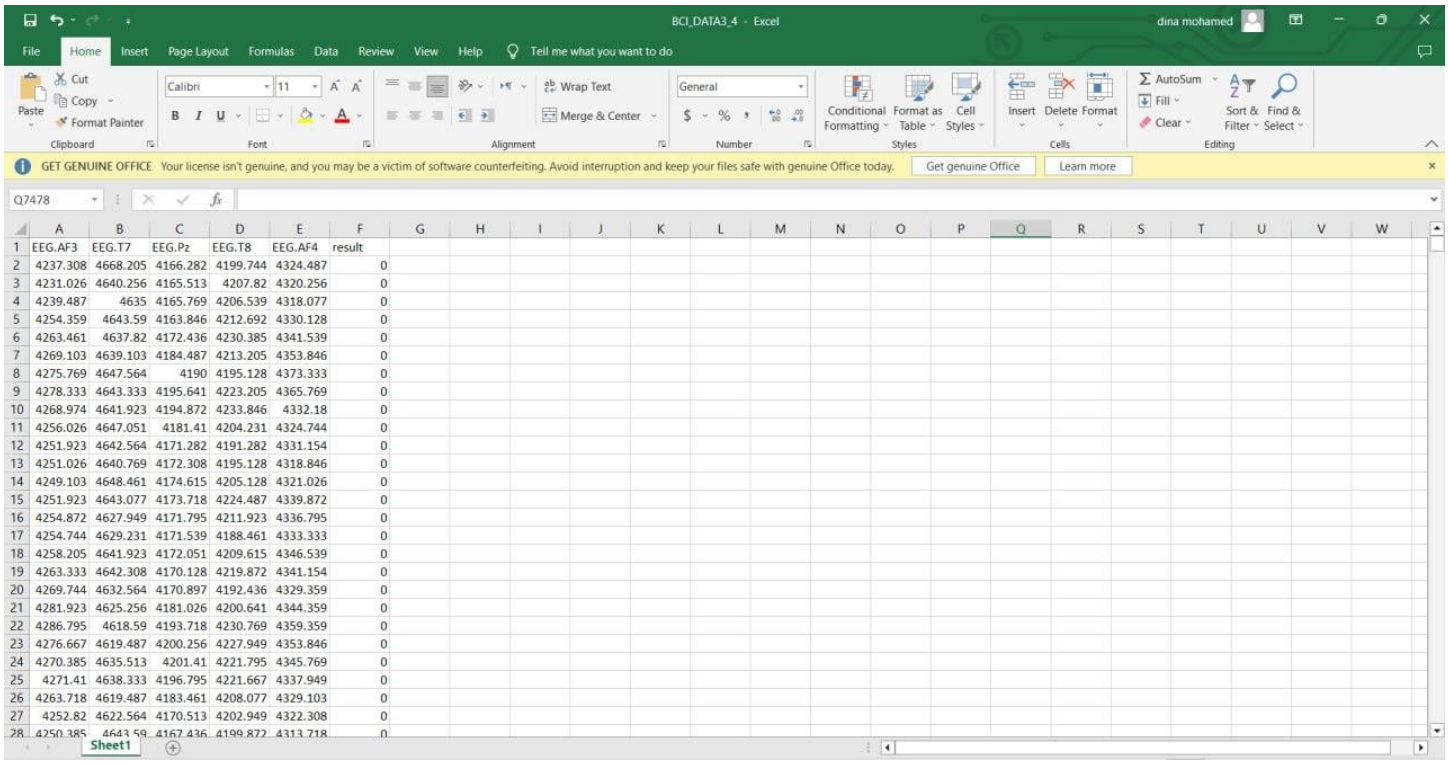


Figure 4.6 Data after preparation

The classification phase is next carried out, and finally the model check step. In order to find the algorithm that provides the highest accuracy of the data used, the algorithms are compared (the KNN algorithm provides the highest accuracy), and the algorithm with the highest accuracy is then used in the metrics module process to produce the confusion matrix and Roc curve.

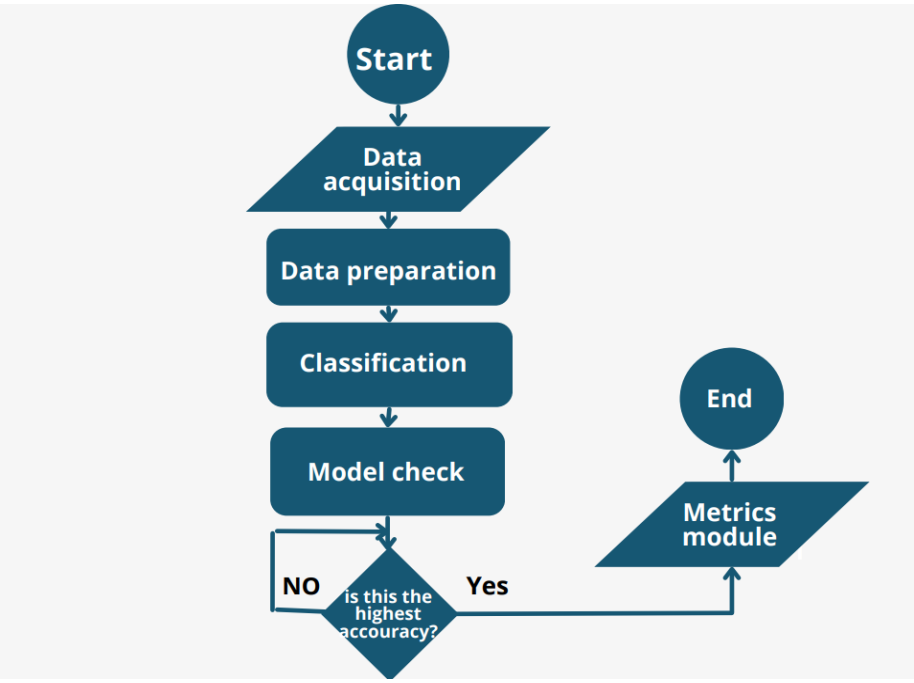


Figure 4.7 AI Flowchart

4.8 Model check

DNN and SVC are removed from the model check and executed separately because they take a long time to run.

4.8.1 DNN

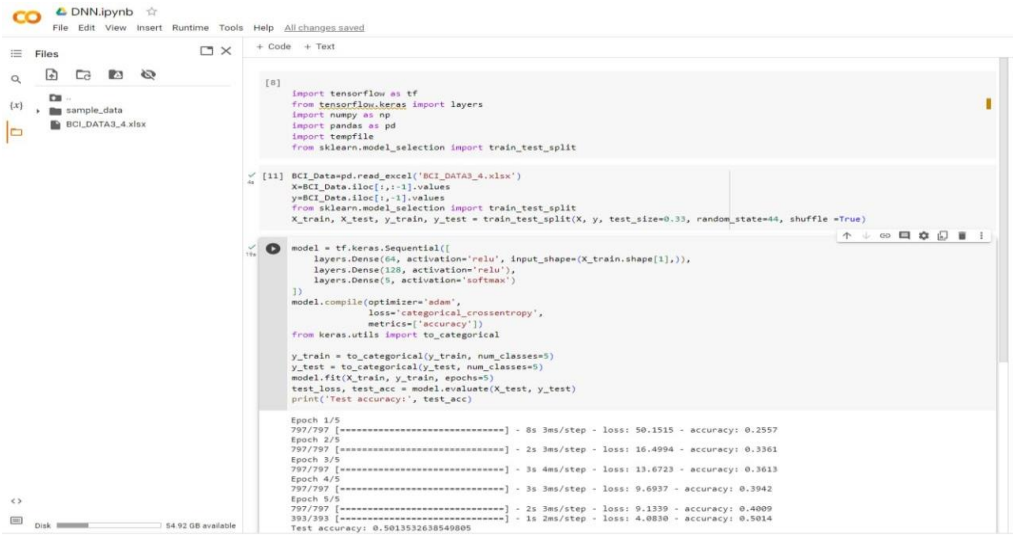


Figure 4.8 Trying DNN

4.8.2 SVC

```
import pandas as pd
from sklearn.multiclass import OneVsOneClassifier
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
convert_Data=pd.read_excel('C:/Users/pc/.spyder-py3/BCI Project/BCI_DATA3_4.xlsx')
X=convert_Data.iloc[:,:-1].values
y=convert_Data.iloc[:,-1].values
#-----------------------------------------------------
#Splitting data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=44, shuffle =True)
#-----------------------------------------------------
clf = OneVsOneClassifier(SVC(kernel='linear', C=1))
# train the classifier on training data
clf.fit(X_train, y_train)
# test the classifier on testing data
y_pred = clf.predict(X_test)
# evaluate the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Figure 4.9 OneVsOne Classifier

By trying OneVsOne Classifier which is one type of SVC, we reach accuracy = 0.6486228307594332

4.8.3 Model Check

```
import pandas as pd
from sklearn.linear_model import Perceptron
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score
from sklearn.naive_bayes import GaussianNB
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
BCI_Data=pd.read_excel('C:/Users/pc/.spyder-py3/BCI Project/BCI_DATA3_4.xlsx')
#-----------------------------------------------------
X=BCI_Data.iloc[:,:-1].values
y=BCI_Data.iloc[:,-1].values
scaler = PolynomialFeatures(degree=3, include_bias=True, interaction_only=False)
X = scaler.fit_transform(X)
#-----------------------------------------------------
#Splitting data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=44, shuffle =True)
#-----------------------------------------------------
```

Figure 4.10 Model check

```
#model check
model1 =  SGDClassifier(penalty='l2',loss='squared_loss',learning_rate='optimal',random_state=33)
model2 = DecisionTreeClassifier(criterion='gini',max_depth=3,random_state=33)
model3=RandomForestClassifier(criterion='gini',n_estimators=10,max_depth=3,random_state=33)
model4=GaussianNB()
model5 = KNeighborsClassifier(n_neighbors= 5,weights ='uniform', algorithm='auto')
model5.fit(X_train, y_train)
model6 = LinearDiscriminantAnalysis(solver='svd',tol=0.0001)
model6.fit(X_train, y_train)
model7=AdaBoostClassifier(n_estimators=10,learning_rate=1)
model7.fit(X_train,y_train)
model8 = Perceptron()
model8.fit(X_train,y_train)
models = [model1 , model2 , model3, model4,model5,model6,model7,model8]
x=0
for m in models:
    x+=1

    for n in range(3,6):
        print('the train result of model : ' , m ,' for cv value ',n,' is ' , cross_val_score(m, X_train, y_train, cv=n))
    print('====================================')

x=0
for m in models:
     x+=1

     for n in range(3,6):
         print('the test result of model  : ' , m ,' for cv value ',n,' is ' , cross_val_score(m, X_test, y_test, cv=n))
     print('**********************************')
```

Figure 4.11 Rest of Model check code

4.8.3.1 SGD Classifier

Train result:

```
the train result of model :
SGDClassifier(loss='squared_loss', random_state=33)  for cv
value  3  is  [0.18209622 0.18221386 0.19644748]
the train result of model :
SGDClassifier(loss='squared_loss', random_state=33)  for cv
value  4  is  [0.20812422 0.19165621 0.20937892 0.19858824]
the train result of model :
SGDClassifier(loss='squared_loss', random_state=33)  for cv
value  5  is  [0.20799843 0.20466575 0.19858851 0.27882353
0.24431373]
------------------------------------
```

Figure 4.12 SGD Classifier train result

Test result:

```
the test result of model  :
SGDClassifier(loss='squared_loss', random_state=33)  for cv
value  3  is  [0.10410697 0.20515882 0.20300931]
the test result of model  :
SGDClassifier(loss='squared_loss', random_state=33)  for cv
value  4  is  [0.18306272 0.20503025 0.2022293  0.18312102]
the test result of model  :
SGDClassifier(loss='squared_loss', random_state=33)  for cv
value  5  is  [0.18304815 0.20931158 0.06528662 0.23447452
0.18312102]
------------------------------------
```

Figure 4.13 SGD Classifier test result

4.8.3.2 Decision Tree Classifier

Train result:

```
the train result of model :
DecisionTreeClassifier(max_depth=3, random_state=33)  for cv
value  3  is  [0.6845077  0.68427244 0.69062463]
the train result of model :
DecisionTreeClassifier(max_depth=3, random_state=33)  for cv
value  4  is  [0.68773526 0.67612923 0.68883312 0.69458824]
the train result of model :
DecisionTreeClassifier(max_depth=3, random_state=33)  for cv
value  5  is  [0.68986473 0.67653401 0.68535581 0.68509804
0.69980392]
```

Figure 4.14 Decision Tree Classifier train result

Test result:

```
the test result of model  :
DecisionTreeClassifier(max_depth=3, random_state=33)  for cv
value  3  is  [0.67239733 0.67470743 0.67231908]
the test result of model  :
DecisionTreeClassifier(max_depth=3, random_state=33)  for cv
value  4  is  [0.6758994  0.67335244 0.67579618 0.67834395]
the test result of model  :
DecisionTreeClassifier(max_depth=3, random_state=33)  for cv
value  5  is  [0.67608436 0.67727815 0.66679936 0.68351911
0.6727707 ]
**********
```

Figure 4.15 Decision Tree Classifier test result

4.8.3.3 Random Forest Classifier

Train result:

```
the train result of model :
RandomForestClassifier(max_depth=3, n_estimators=10,
random_state=33)  for cv value  3  is  [0.6554523  0.6547465
0.66192213]
the train result of model :
RandomForestClassifier(max_depth=3, n_estimators=10,
random_state=33)  for cv value  4  is  [0.65166248 0.64632999
0.66311167 0.66823529]
the train result of model :
RandomForestClassifier(max_depth=3, n_estimators=10,
random_state=33)  for cv value  5  is  [0.66065477 0.65163693
0.66261517 0.66490196 0.66823529]
==========
```

Figure 4.16 Random Forest Classifier train result

Test result:

```
the test result of model  :
RandomForestClassifier(max_depth=3, n_estimators=10,
random_state=33)  for cv value  3  is  [0.65711557 0.65392883
0.66252687]
the test result of model  :
RandomForestClassifier(max_depth=3, n_estimators=10,
random_state=33)  for cv value  4  is  [0.6663483  0.65902579
0.65414013 0.66050955]
the test result of model  :
RandomForestClassifier(max_depth=3, n_estimators=10,
random_state=33)  for cv value  5  is  [0.66931954 0.67051333
0.64769108 0.67078025 0.65764331]
************************************
```

Figure 4.17 Random Forest Classifier test result

4.8.3.4 LDA Classifier

Train result:

```
the train result of model :  LinearDiscriminantAnalysis()
for cv value  3  is  [0.69427126 0.69580049 0.69544759]
the train result of model :  LinearDiscriminantAnalysis()
for cv value  4  is  [0.69557716 0.68961731 0.69918444
0.69709804]
the train result of model :  LinearDiscriminantAnalysis()
for cv value  5  is  [0.69731425 0.68672809 0.69535385
0.69862745 0.69705882]
____________________________________
```

Figure 4.18 LDA Classifier train result

Test result:

```
the test result of model  :  LinearDiscriminantAnalysis()
for cv value  3  is  [0.69269341 0.70097922 0.69835204]
the test result of model  :  LinearDiscriminantAnalysis()
for cv value  4  is  [0.70105062 0.6864056  0.7111465
0.6888535 ]
the test result of model  :  LinearDiscriminantAnalysis()
for cv value  5  is  [0.69558297 0.70274572 0.69585987
0.70621019 0.69187898]
************************************
```

Figure 4.19 LDA Classifier test result

4.8.3.5 Ada Boost

Train result:

```
the train result of model :
AdaBoostClassifier(learning_rate=1, n_estimators=10)  for cv
value  3  is  [0.36125162 0.39265969 0.26855664]
the train result of model :
AdaBoostClassifier(learning_rate=1, n_estimators=10)  for cv
value  4  is  [0.39570263 0.35759097 0.39491844 0.40078431]
the train result of model :
AdaBoostClassifier(learning_rate=1, n_estimators=10)  for cv
value  5  is  [0.35600863 0.32366203 0.37972946 0.37039216
0.39705882]
```

Figure 4.20 Ada Boost Classifier train result

Test result:

```
the test result of model  :
AdaBoostClassifier(learning_rate=1, n_estimators=10)  for cv
value  3  is  [0.48352436 0.40219728 0.55123   ]
the test result of model  :
AdaBoostClassifier(learning_rate=1, n_estimators=10)  for cv
value  4  is  [0.39891754 0.3272843  0.44235669 0.49267516]
the test result of model  :
AdaBoostClassifier(learning_rate=1, n_estimators=10)  for cv
value  5  is  [0.45324314 0.58058098 0.51671975 0.47730892
0.32285032]
```

Figure 4.21 Ada Boost Classifier test result

4.8.3.6 Perception

Train result:

```
the train result of model :  Perceptron()  for cv value  3
is  [0.38983649 0.3767792  0.3803082 ]
the train result of model :  Perceptron()  for cv value  4
is  [0.40244668 0.30928482 0.42769762 0.40894118]
the train result of model :  Perceptron()  for cv value  5
is  [0.31523231 0.22172123 0.3799255  0.40588235 0.24490196]
```

Figure 4.22 perception Classifier train result

Test result:

```
the test result of model  :  Perceptron()  for cv value  3
is  [0.36700096 0.34272749 0.3926439 ]
the test result of model  :  Perceptron()  for cv value  4
is  [0.39318688 0.37917861 0.38025478 0.21751592]
the test result of model  :  Perceptron()  for cv value  5
is  [0.37445285 0.40588938 0.2977707  0.30175159 0.41640127]
```

Figure 4.23 perception Classifier test result

4.8.3.7 Naive Bayes

Train result:

```
the train result of model :  GaussianNB()  for cv value  3
is  [0.52958475 0.53252559 0.54040701]
the train result of model :  GaussianNB()  for cv value  4
is  [0.53121079 0.52525094 0.54156211 0.54164706]
the train result of model :  GaussianNB()  for cv value  5
is  [0.53303274 0.5249951  0.5318565  0.5427451  0.54      ]
```

Figure 4.24 naive bayes Classifier train result

Test result:

```
**************************************
the test result of model  :  GaussianNB()  for cv value  3
is  [0.52913085 0.53427275 0.52973489]
the test result of model  :  GaussianNB()  for cv value  4
is  [0.53517988 0.52371856 0.54458599 0.5156051 ]
the test result of model  :  GaussianNB()  for cv value  5
is  [0.53601273 0.53203343 0.52507962 0.53821656 0.51671975]
```

Figure 4.25 naïve bayes Classifier test result

4.8.3.8 KNN

KNN Code:

```python
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import label_binarize
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import zero_one_loss
convert_Data=pd.read_excel('C:/Users/pc/.spyder-py3/BCI Project/BCI_DATA3_4.xlsx')
X=convert_Data.iloc[:,:-1].values
y=convert_Data.iloc[:,-1].values
#-----------------------------------------------------
#Splitting data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=44, shuffle =True)
#-----------------------------------------------------
KNNClassifierModel = KNeighborsClassifier(n_neighbors= 5,weights ='uniform', algorithm='auto')
KNNClassifierModel.fit(X_train, y_train)
#Calculating Score
print('KNNClassifierModel Train Score is : ' , KNNClassifierModel.score(X_train, y_train))
print('KNNClassifierModel Test Score is : ' , KNNClassifierModel.score(X_test, y_test))
#Calculating Prediction
y_pred = KNNClassifierModel.predict(X_test)
y_pred_prob = KNNClassifierModel.predict_proba(X_test)
# Compute predicted labels for test data
y_pred = KNNClassifierModel.predict(X_test)
# Compute confusion matrix
cm = confusion_matrix(y_test, y_pred)
print('Confusion Matrix is : \n', cm)
# Plot confusion matrix
sns.heatmap(cm)
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.title('Confusion matrix')
plt.show()
# Convert labels to one-hot encoded vectors
y_test_bin = label_binarize(y_test, classes=[0, 1, 2, 3, 4])
# Compute probabilities for each class
probas = KNNClassifierModel.predict_proba(X_test)
```

Figure 4.26 KNN Code

```python
# Compute probabilities for each class
probas = KNNClassifierModel.predict_proba(X_test)
# Compute ROC curve and ROC area for each class
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(5):
    fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], probas[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])
# Compute micro-average ROC curve and ROC area
fpr["micro"], tpr["micro"], _ = roc_curve(y_test_bin.ravel(), probas.ravel())
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
# Plot ROC curves
plt.figure()

lw = 2
plt.plot(fpr["micro"], tpr["micro"], color='darkorange',
         lw=lw, label='micro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["micro"]))
for i in range(5):
    plt.plot(fpr[i], tpr[i], lw=lw,
             label='ROC curve of class {0} (area = {1:0.2f})'
             ''.format(i, roc_auc[i]))

plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic (ROC) curve')
plt.legend(loc="lower right")
plt.show()
AccScore=accuracy_score(y_test,y_pred)
print('Accuracy Score is:',AccScore)
ZeroOneLossValue=zero_one_loss(y_test, y_pred)
print('Misclassification Rate:',ZeroOneLossValue)
```

Figure 4.27 KNN Code

4.9 Trails

4.9.1 Polynomial features

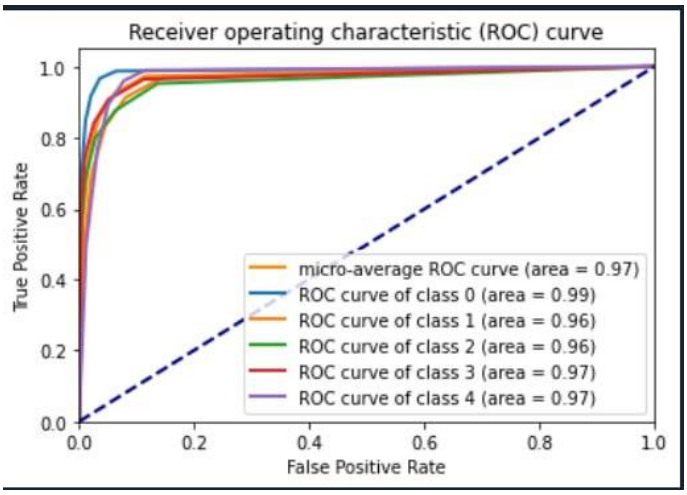


Figure 4.28 Polynomial features ROC curve

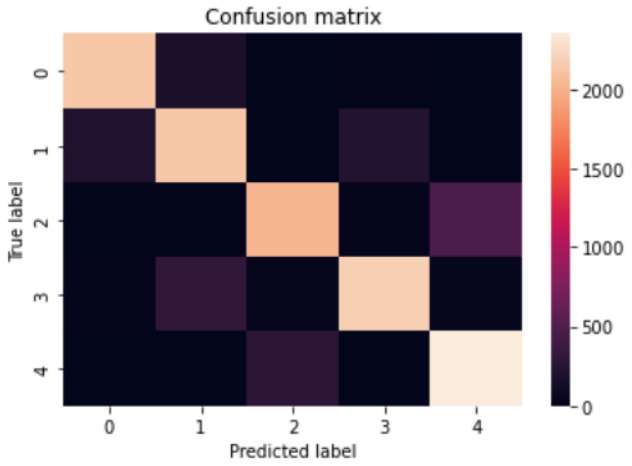


Figure 4.29 Polynomial features Confusion Matrix

Accuracy score=(TP+TN)/total=85.94%

Misclassification rate=(FP+FN)/total=14.05

4.9.2 Feature Selection by using Random Forest

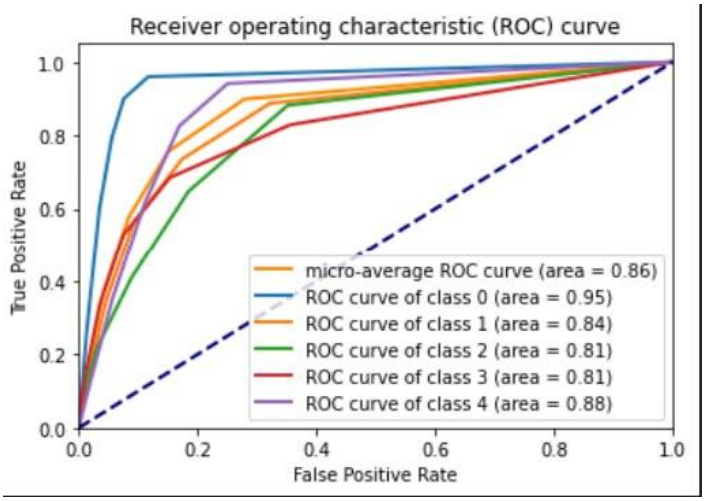


Figure 4.30 Feature Selection ROC curve

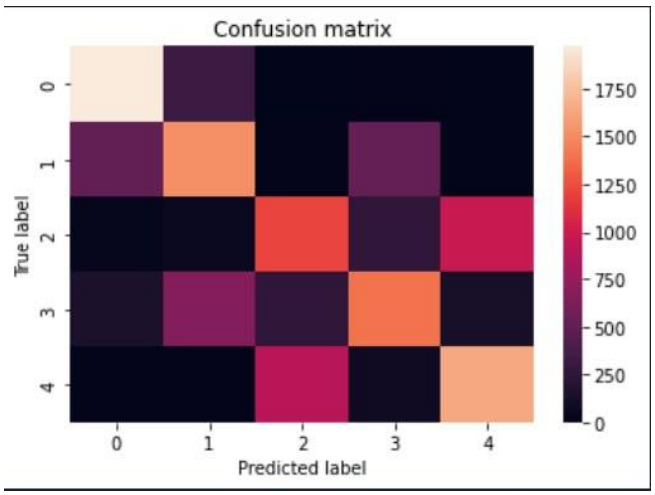


Figure 4.31 Feature Selection Confusion Matrix

Accuracy score=(TP+TN)/total=61.34%

Misclassification rate=(FP+FN)/total=38.65%

4.10 Algorithm with highest accuracy

KNN Algorithm

Result:

```
In [5]: runfile('C:/Users/pc/.spyder-py3/BCI Project/KNN.py',
wdir='C:/Users/pc/.spyder-py3/BCI Project')
KNNClassifierModel Train Score is :  0.9197741442183273
KNNClassifierModel Test Score is :  0.872472536220347
Confusion Matrix is :
 [[2145  144    0   11    0]
 [ 201 2145    2  190    3]
 [   0    0 2061   21  431]
 [  22  262   23 2232   38]
 [   0    0  248    6 2377]]
Accuracy Score is: 0.872472536220347
Misclassification Rate: 0.12752746377965296
```

Figure 4.32 KNN result

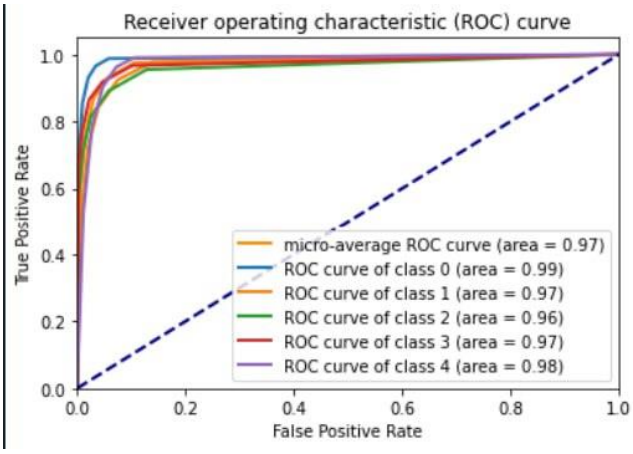


Figure 4.33 KNN ROC curve

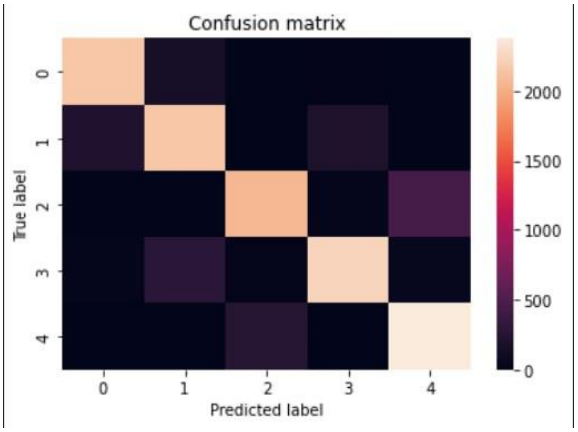


Figure 4.34 KNN Confusion Matrix

Chapter 5

Internet of Things

Chapter 5
Internet of Tings

5.1 What is the Internet of Things?

The Internet of Things, or IoT, refers to the billions of physical devices around the world that are now connected to the internet, all collecting and sharing data. Thanks to the arrival of super-cheap computer chips and the ubiquity of wireless networks, it's possible to turn anything, from something as small as a pill to something as big as an aeroplane, into a part of the IoT. Connecting up all these different objects and adding sensors to them adds a level of digital intelligence to devices that would be otherwise dumb, enabling them to communicate real-time data without involving a human being. The Internet of Things is making the fabric of the world around us more smarter and more responsive, merging the digital and physical universes.

5.2 What is the history of the Internet of Things?

The idea of adding sensors and intelligence to basic objects was discussed throughout the 1980s and 1990s (and there are arguably some much earlier ancestors), but apart from some early projects -- including an internet-connected vending machine -- progress was slow simply because the technology wasn't ready. Chips were too big and bulky and there was no way for objects to communicate effectively.

Processors that were cheap and power-frugal enough to be all but disposable were needed before it finally became cost-effective to connect up billions of devices. The adoption of RFID tags -- low-power chips that can communicate wirelessly -- solved some of this issue, along with the increasing availability of broadband internet and cellular and wireless networking. The adoption of IPv6 -- which, among other things, should provide enough IP addresses for every device the world (or indeed this galaxy) is ever likely to need -- was also a necessary step for the IoT to scale.

Kevin Ashton coined the phrase 'Internet of Things' in 1999, although it took at least another decade for the technology to catch up with the vision.

"The IoT integrates the interconnectedness of human culture -- our 'things' -- with the interconnectedness of our digital information system -- 'the internet.' That's the IoT," Ashton told ZDNet.

Adding RFID tags to expensive pieces of equipment to help track their location was one of the first IoT applications. But since then, the cost of adding sensors and an internet connection to objects has continued to fall,

and experts predict that this basic functionality could one day cost as little as 10 cents, making it possible to connect nearly everything to the internet.

The IoT was initially most interesting to business and manufacturing, where its application is sometimes known as machine-to-machine (M2M), but the emphasis is now on filling our homes and offices with smart devices, transforming it into something that's relevant to almost everyone. Early suggestions for internet-connected devices included 'blogjects' (objects that blog and record data about themselves to the internet), ubiquitous computing (or 'ubicomp'), invisible computing, and pervasive computing. However, it was Internet of Things and IoT that stuck.

5.3 Characteristics of IoT (Internet of Things)

The way we define and demonstrate IoT is often explained by the characteristics of the Internet of Things (IoT). Here is a comprehensive list of all the significant characteristics of IoT:

- Connectivity

  One of the most crucial characteristics of IoT infrastructure is connectivity. IoT things must stay connected to the infrastructure of IoT. There should be an all-time assurance that anyone can connect anytime, anywhere they want.

  For instance, the connection between internet gadgets like sensors and routers and the connection between one person and another via internet devices like smartphones.

- Interoperable Communication Protocols

  Seamless, interoperable communication is one of the key features of the Internet of Things. With the rising popularity of IoT, these IoT devices are now more interconnected than before. This is one of the most important characteristics of IoT

  This characteristics of IoT ensures that’s the communication with one another work more seamlessly and efficiently. These devices/gadgets are connected to the world of the internet by leveraging cloud service. This gives room for seamless communication between devices.

- Identity of Things

  It’s a well-known fact that anyone’s unique characteristic is their identity. For instance, all smartphones come with a unique IMEI number. This is true for all other smart devices on the planet.

Identifying the things on the internet and people becomes easier with this unique number. Thus, it's safe to say the Identity of Things is among the most crucial characteristics of IoT.

- Scalability

Considering scalability in the system design has become more crucial than ever due to the surging ubiquity of the IoT. In the technical world, scalability refers to a system's capability to grow without adversely impacting its performance. Today, scalability is among the most popular characteristics of IoT.

With the help of this characteristics of IoT, businesses can leverage the power of IoT by scaling it to fit their business. For instance, IoT can help hotel businesses track the rooms and amenities used by different guests. It also assists SMEs by sending targeted messages and offers on their behalf to customers.

- Dynamic or Self-Adapting

The dynamic nature is another key characteristic of IoT as it requires one to become self-adaptive to comprehend the modifications and upgrades happening around it and respond accordingly. For instance, the camera device was originally introduced to the world to take pictures.

Today, the camera comes with many features that let you adjust pictures based on the light. Thus, being dynamic and self-adaptive is among the crucial characteristics of IoT that one cannot avoid.

- Architecture

There's no doubt in saying that architecture is one of the key characteristics of IoT. There are plenty of products and manufacturers in the IoT that leverages the architecture to support their gadgets or devices. The significance of architecture varies due to the surge in device numbers. Its capacity to accommodate a wide range of devices, protocols, and technologies is the reason behind this. The architecture is primarily in charge of ensuring that the devices cooperate and communicate with one another. This IOT characteristics plays a crucial role in preventing cross-interference between the devices.

- Safety

The importance of the Internet of Things can be fully comprehended with its safety. When every user's device is connected to the internet, there will be a high risk of the users' confidential personal information getting compromised. This will result in data loss for the user.

Thus, one of the major challenges that remain is data security. Apart from that, the size of the equipment involved is enormous. Even IoT networks can get exposed to such a risk. Thus, equipment safety is one of the crucial characteristics of IoT.

- Intelligence

IoT devices' intelligence is one of the many characteristics of IoT. Basic object-level interactions augment the IoT networks' collective intelligence. It is necessary to transform the acquired data into information that the devices can use. These devices must continue to gather information to sustain and improve this intelligence.

If you seek interest in adding or enhancing intelligence to IoT devices and make a career out of it, you must learn program languages like ML. Thus, an AI & ML Course will be the right path to begin this journey.

- Embedded Sensors and Actuators

This is one of the unique or different characteristics of IoT. The IoT relies on sensors to identify and quantify environmental changes to produce data that may be used to report on the environment's condition or even to interact with it.

With sensing technologies, it is possible to create capabilities that accurately represent knowledge of the physical world and its inhabitants. Although sensory information is an analog input from the real world, it can offer a profound understanding of our complex environment.

- Data

This feature of IOT helps in gathering of the data for future prediction. For example, "How much calorie does our body consume/day. This feature helps to regulate the number of calories each day. Therefore, we can use the data collected through these devices in multiple ways and make our lives a lot easier.

5.4 IoT infograph

Signals are converted to numerical data by the cortex, which is then posted to Firebase using the JSM protocol. From there, data can be sent to smart home and car via node MCU using the MQTT protocol, and a web server can also get the data to control all applications, monitor them, and control our game.

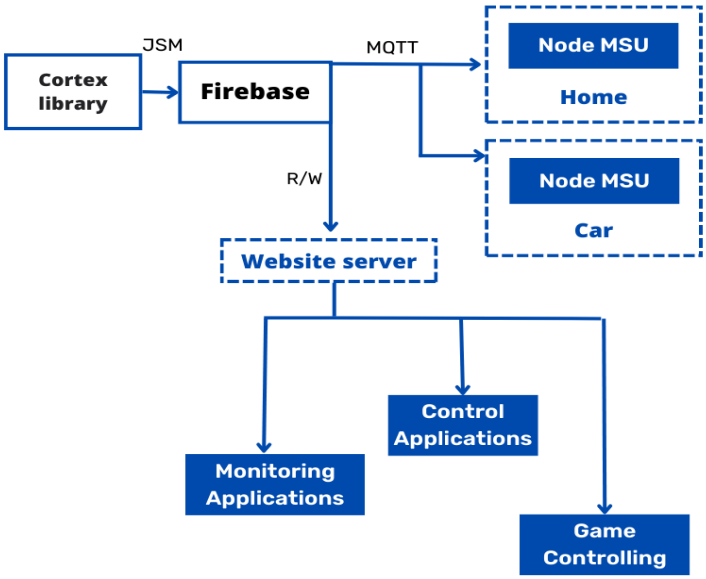


Figure 5.1 IoT infograph

5.5 Firebase overview

From the headset, data is transferred to the Firebase by using the Firebase configuration, and then all our applications (smart home, smart car, game) can receive data and perform the required action, and also our web application can get and upload data from/ to the firebase.

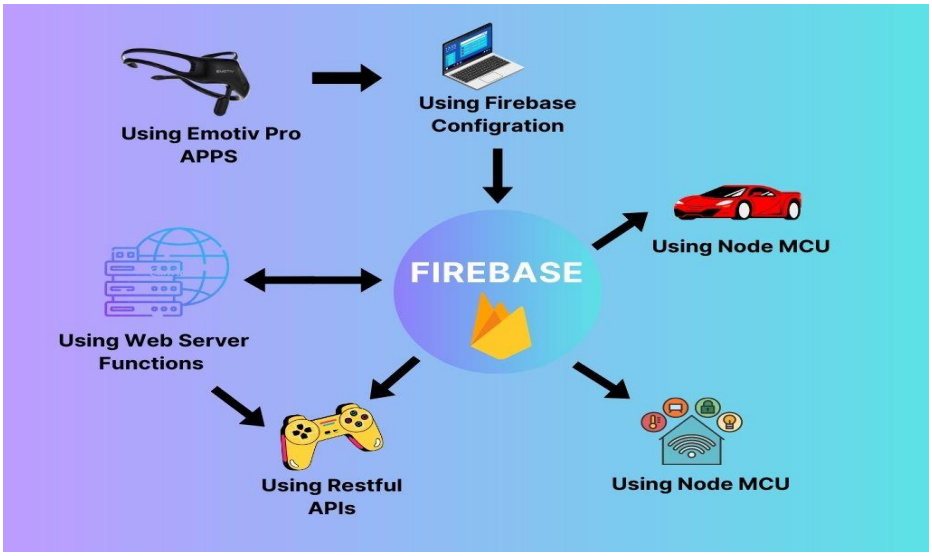


Figure 5.2 Firebase overview

Chapter 6

Website & Mobile Applications

Chapter 6

Website & Mobile Applications

6.1 Front-End development

It is the process of building components that interact with users. Examples are the user interface, buttons, user-entered data, websites, and user experience (UX) features. The front end aims at meeting user requirements and delivering a positive user experience.

6.2 Programming Languages for Front-End Development

Below are the most common programming languages for developing the front end.

6.2.1 HTML

The Hypertext Markup Language (HTML) programming language, which defines the structure and meaning of web content, is a building block for front-end development. Through HTML, browsers display text or load elements, rendering webpages, which contain hyperlinks and links to other webpages, for users.

6.2.2 CSS

Cascading style sheets (CSS) is the standard language that specifies how to display HTML content: fonts, foreground, and background colors, etc. With CSS, you can control the design layout and its components for various devices like desktops, tablets, and smartphones. Examples of the components are the header, body, footer, content, asides, and sections.

6.2.3 TypeScript

TypeScript is a solid programming language that builds on JavaScript. TypeScript allows specifying the types of data being passed around within the code, and has the ability to report errors when the types don't match.

TypeScript is a strongly typed, object-oriented, compiled programming language that builds on JavaScript. It is a superset of the JavaScript language, designed to give you better tooling at any scale.

The lead architect behind TypeScript is Anders Hejlsberg, designer of C# at Microsoft. TypeScript is open source, backed by Microsoft, and considered both a language and a set of tools.

6.3 Front-End development framework

6.3.1 Angular

Angular is an open-source, JavaScript framework written in TypeScript. Google maintains it, and its primary purpose is to develop single-page applications. You can deliver highly dynamic results through the HTML syntax. Angular adopts an efficient modular approach and follows the MVC architecture, which divides the website structure into three parts: model, view, and controller (MVC).

6.4 Common front-end developer tasks

A frontend developer would be expected to take on any of the following:

- Designing and overseeing the user experience of an app or website.
- Using programming languages such as HTML, CSS, and Typescript to create the front-end.
- Maintaining and updating the user interface.
- Developing new tools to improve interaction with the website or app.
- Carrying out testing and debugging.

6.5 Back-end development

It is the process of building the components for running the application behind the scenes. Examples are components for data storage, infrastructure, integration with external systems, and code written in one or more programming languages. Users cannot access the back end.

6.6 Back-End development languages and framework

6.6.1 Firebase

Firebase is a backend-as-a-service platform from Google for mobile and web app development. The platform offers a range of tools and services for building, managing and scaling apps, including a real-time database, user authentication, hosting, cloud storage and more.

In addition, Firebase offers a range of analytics and reporting tools that can help developers understand how their users interact with their app and make data-driven decisions.

6.6.2 Node.js

Node.js is a server-side platform based on the JavaScript Engine in Google Chrome. This is a cross-platform runtime environment for developing server-side and networking applications that are open source. Node.js programs are written in JavaScript and run on the Node.js runtime on OS X, Microsoft Windows, and Linux. Node.js also comes with a big library of JavaScript modules, which makes developing Node.js web applications much easier.

Figure 6.1 Back-End

6.7 Common back-end developer tasks

- To design effective and efficient solutions, they must first gain a thorough grasp of the website's performance demands and goals.
- Application Programming Interfaces (APIs) development and administration.
- Develop data acceptance and storage solutions for websites, particularly for those involved in payment processing.
- Writing, testing, and maintaining development solutions for code-related problems are all part of the job.
- To identify new features, and communicate effectively with developers, designers, and system administrators.
- Create a website architecture by utilizing correct product lifecycles approaches, such as Agile Scrum and frameworks.
- Organize the system logic.
- Provide remedies to difficulties with the system.
- Debug and troubleshoot apps.

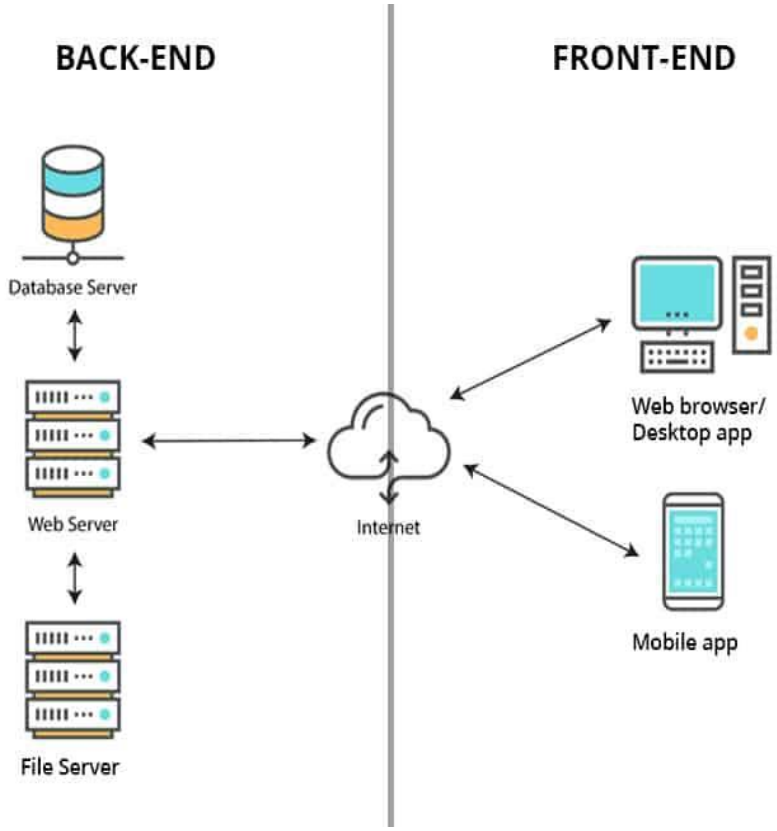


Figure 6.2 web Application

6.8 Website Interface

6.8.1 Home page

Both the activities that have been carried out and solutions to the issues raised in the project are presented. The page is divided into three sections:

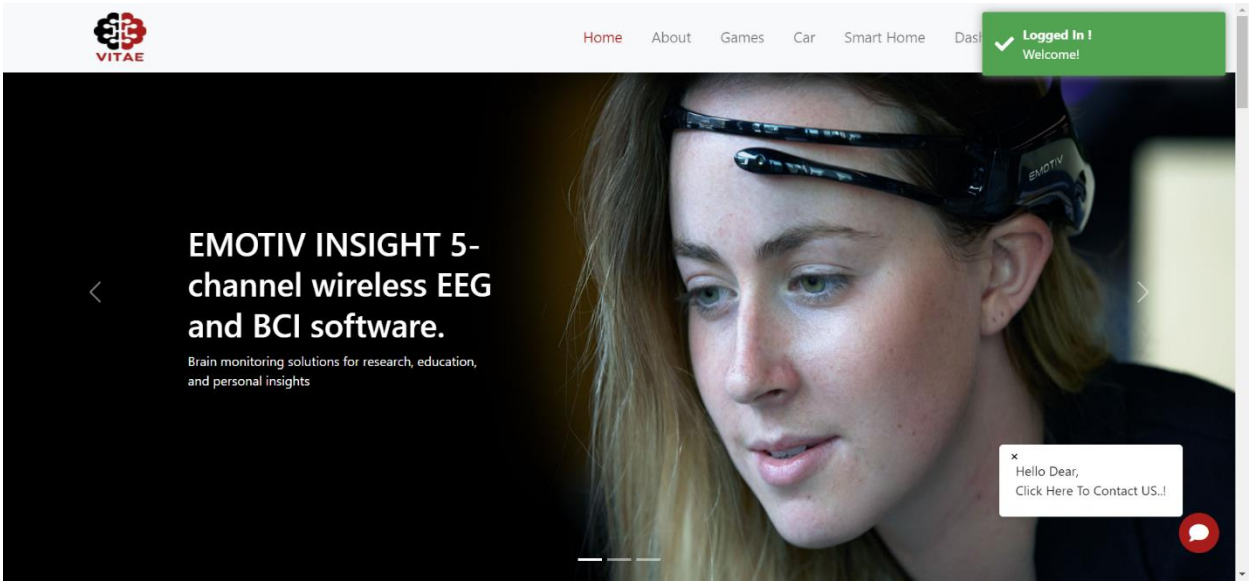


Figure 6.3 Home page section 1

Figure 6.4 Home page section 2

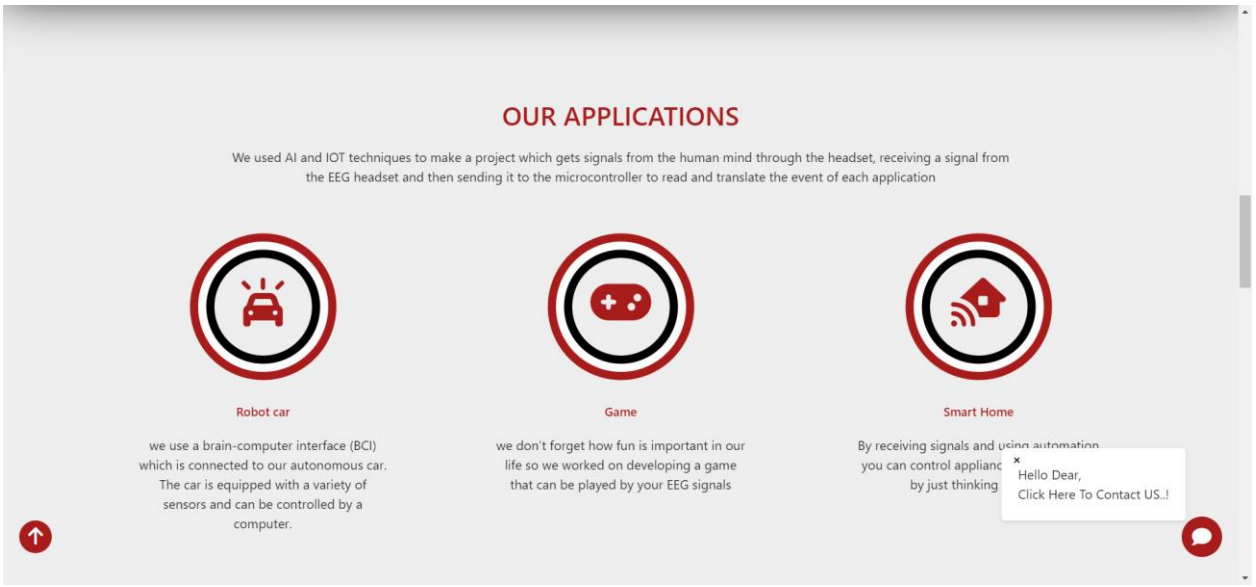


Figure 6.5 Home page section 3

6.8.2 Utilized technologies

The technologies we employed for the project and their respective functions are appeared in the second page.

6.8.2.1 Artificial Intelligence

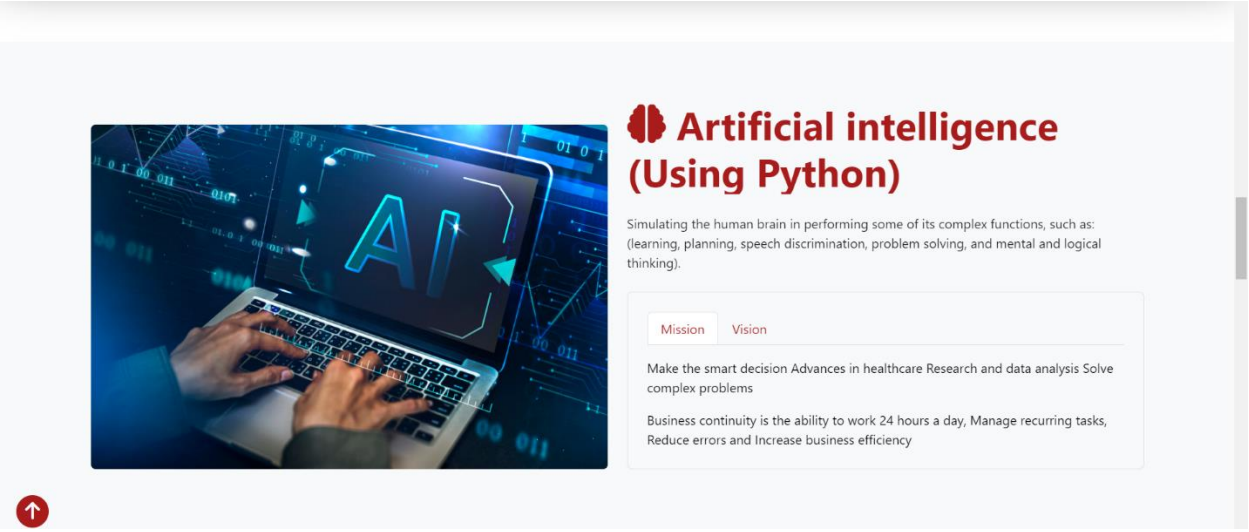


Figure 6.6 Artificial Intelligence Technology

6.8.2.2 Embedded System

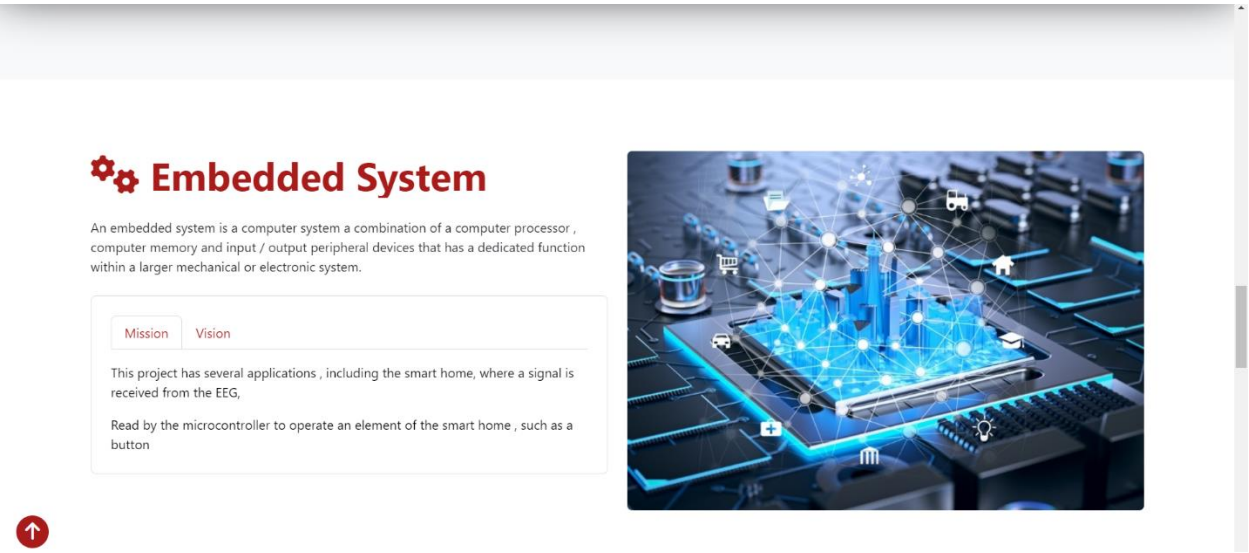


Figure 6.7 Embedded System Technology

6.8.2.3 Web & Mobile Application

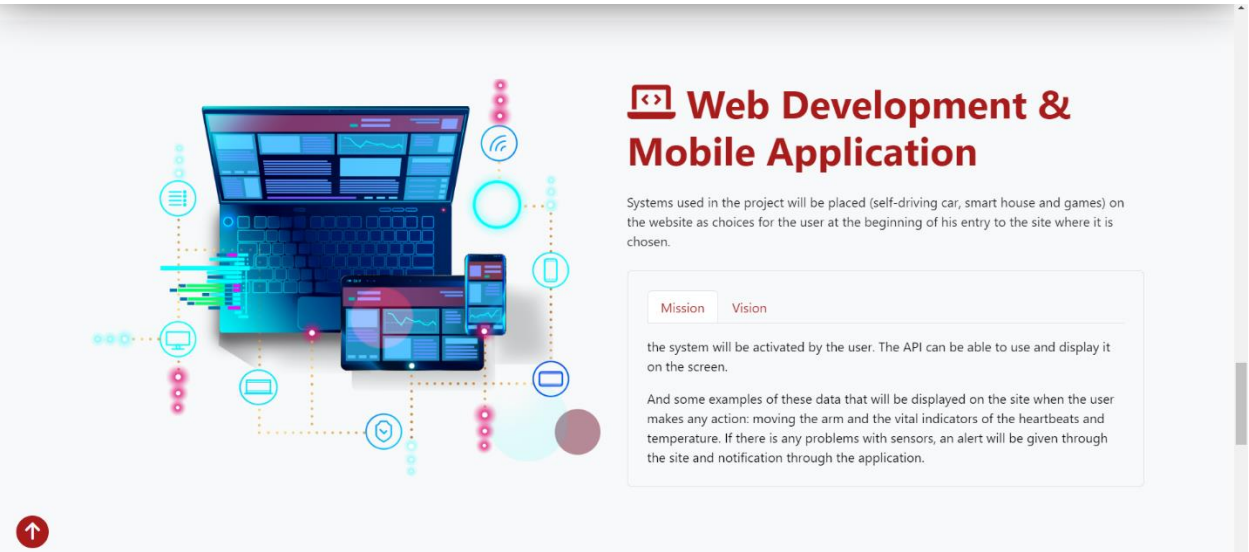


Figure 6.8 Web & Mobile Application Technology

6.8.2.4 Hardware

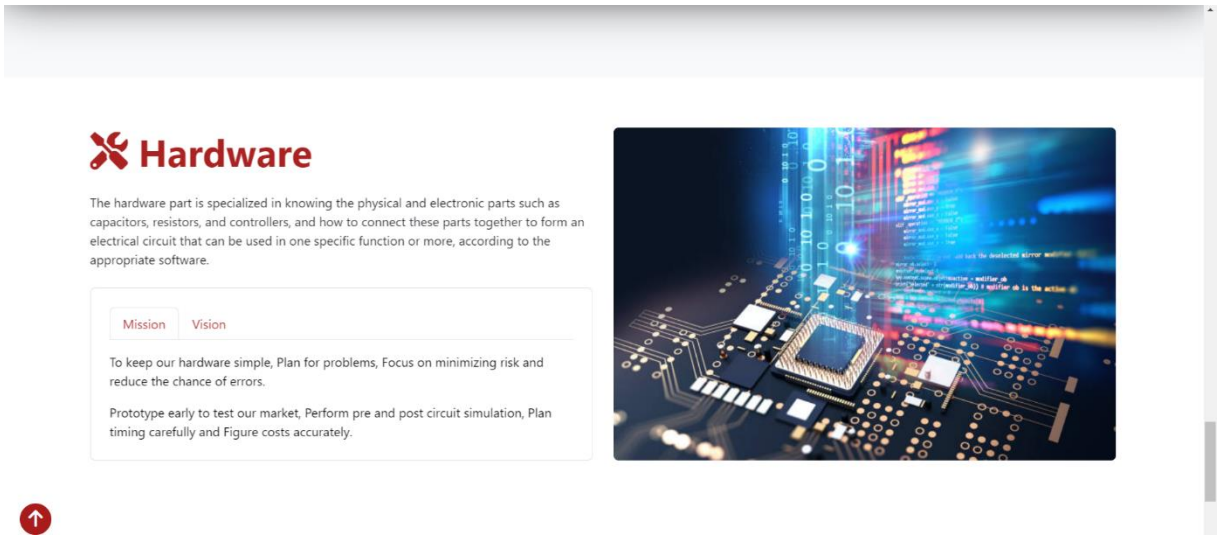


Figure 6.9 Hardware Technology

6.8.3 Applications

Applications that implemented in the project.

6.8.3.1 Jump games

There is a display and supervision of the game control.

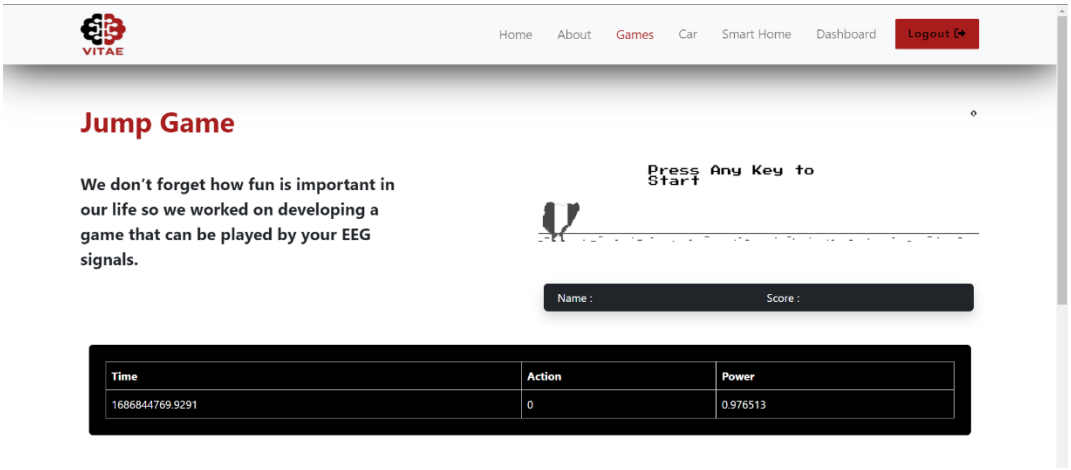


Figure 6.10 Game

6.8.3.2 The Robot Car

Here is a simulation for the actual movement of the car.

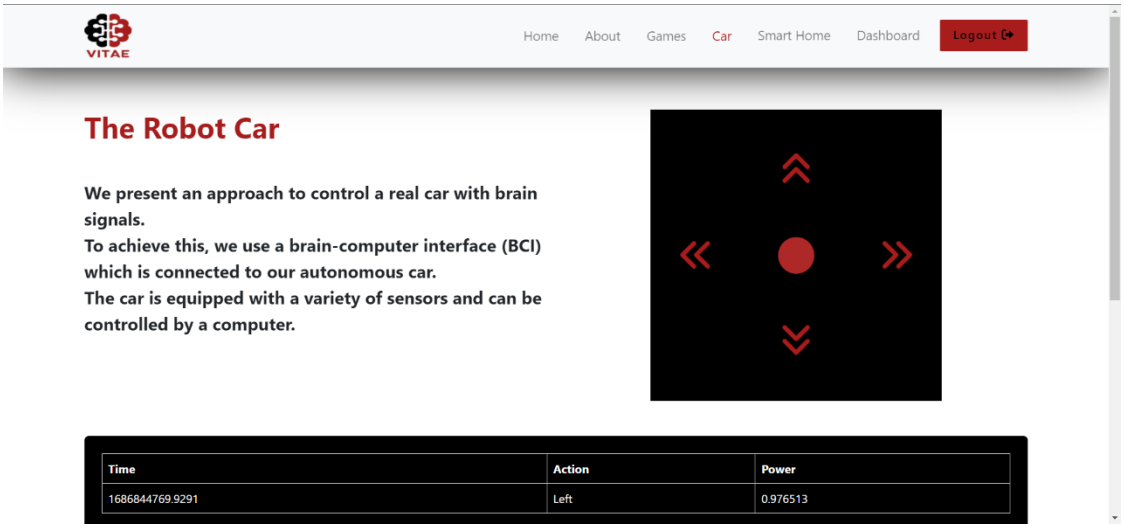


Figure 6.11 Robotic car

6.8.3.3 Smart home

Home appliances can be displayed here with its instant state.

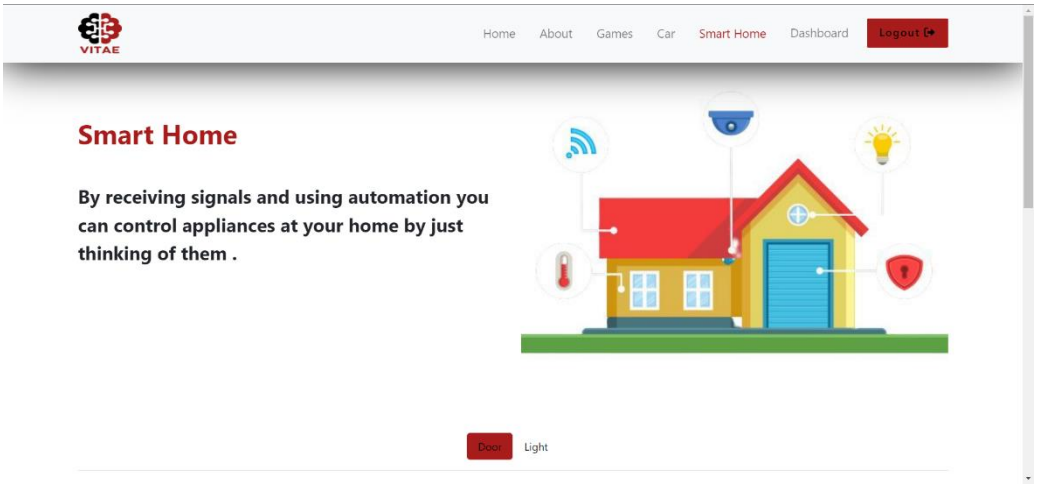


Figure 6.12 Smart home

This section display the state of door whether it open or close.

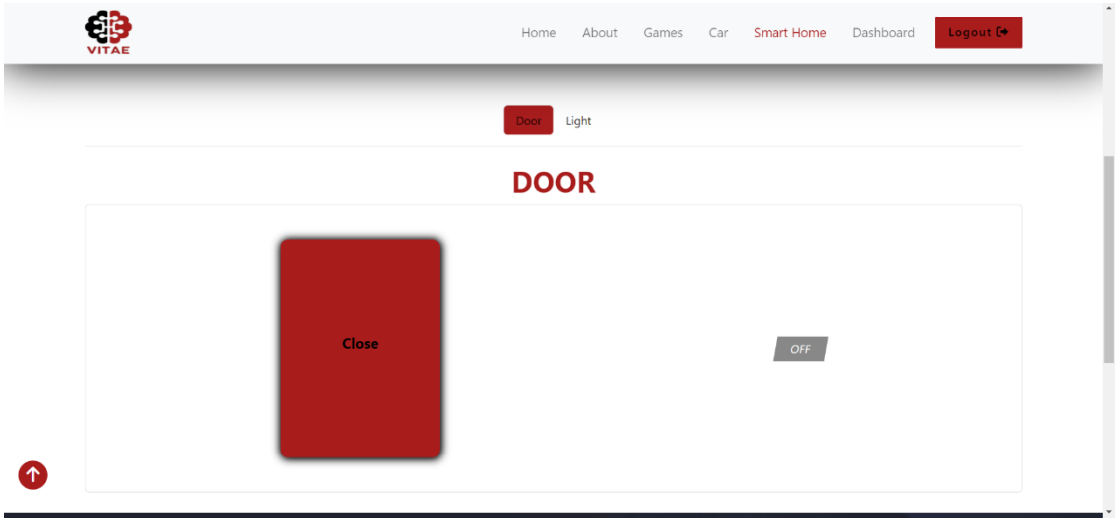


Figure 6.13 door state

This section display the state of light whether it on or off.

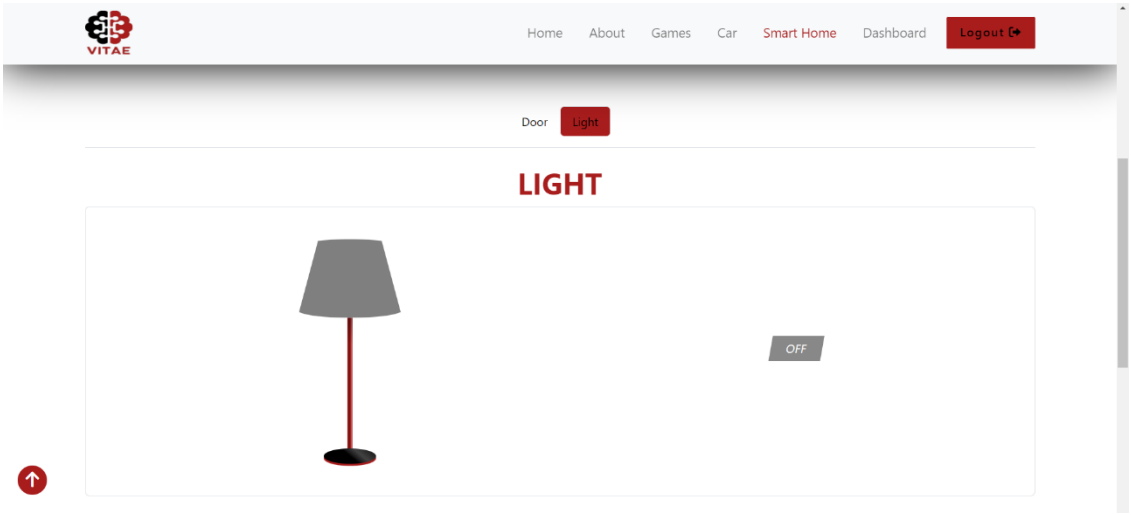


Figure 6.14 light state

6.8.4 Admin dashboard

Appears only for admin and includes all users' account that can be controlled by the admin.

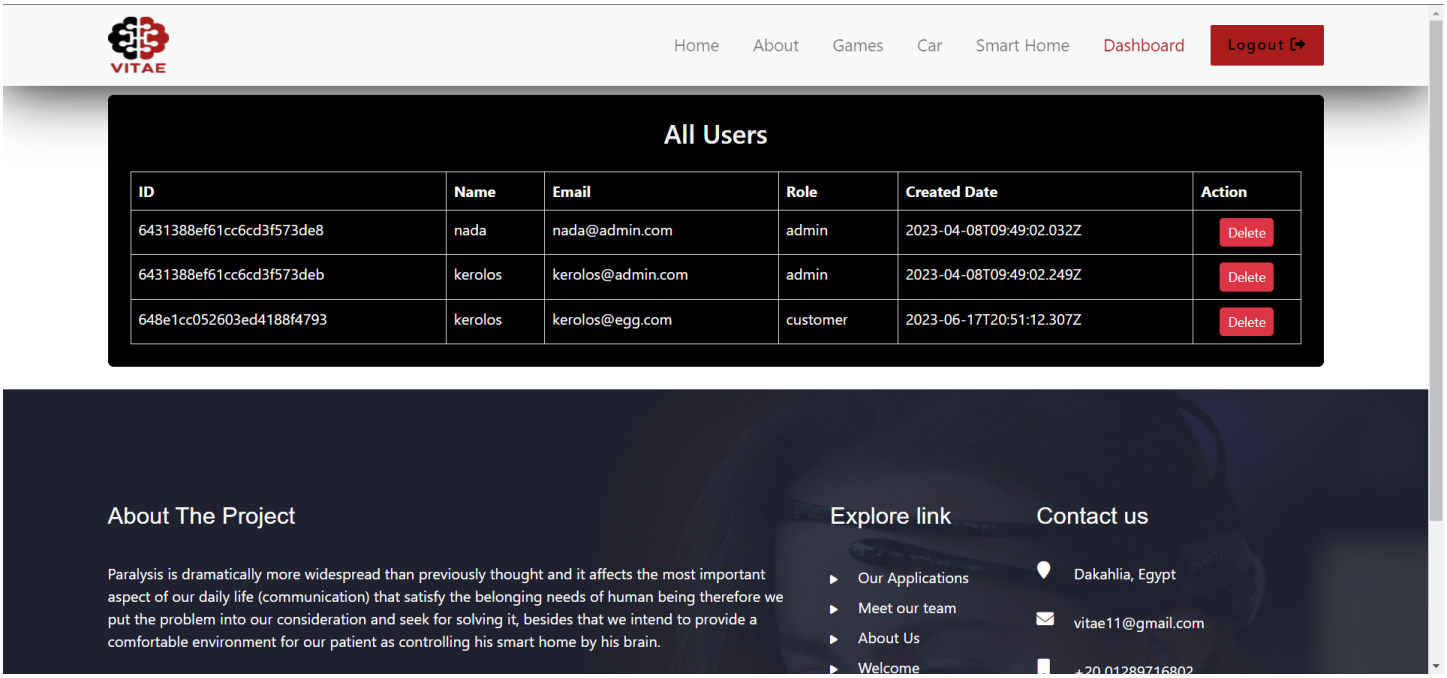


Figure 6.15 Admin dashboard

6.9 Mobile Application

6.9.1 Flutter

Flutter is an open-source mobile application development framework created by Google. It allows developers to build high-performance, visually attractive mobile applications for both Android and iOS platforms from a single codebase. Flutter uses the Dart programming language, which was also created by Google, and provides a rich set of pre-built widgets and tools to help developers create beautiful and responsive user interfaces.

One of the key features of Flutter is its "hot reload" capability, which allows developers to see changes to the code reflected in the app almost instantly, without having to rebuild the entire app. This makes the development process faster and more efficient.

Flutter also offers a wide range of plugins and packages that developers can use to add additional functionality to their apps, such as geolocation, camera integration, and more. With Flutter, developers can create apps that run on both Android and iOS devices, as well as web and desktop platforms, making it a versatile and powerful tool for mobile app development

6.9.2 Flutter Frontend

In Flutter, the user interface (UI) of the app is built using widgets, which are the building blocks of the UI. Flutter provides a wide range of pre-built widgets that developers can use to create beautiful and responsive user interfaces.

6.9.3 Flutter concepts

As a frontend developer using Flutter, there are a few essential concepts and skills that you should familiarize yourself with:

- Dart

The very first thing you should know before learning Flutter is Dart. Dart is a programming language developed by Google that is used to build applications for the web, mobile, desktop, and backend. Flutter, Google's mobile app development framework, is built on top of Dart, so if you want to develop Flutter apps, you will need to use Dart.

- Widgets

In Flutter, everything is a widget. Image, icon, text, and even row, column, and padding are all considered widgets in Flutter. Understanding how to use and compose different types of widgets is essential for building Flutter apps.

- Layout

Flutter provides several layout widgets, such as Container, Row, and Column, that you can use to arrange and size your app's user interface elements. Here are examples of container, row, and column Flutter widget codes.

- Styling

Flutter provides a number of ways to style your app's user interface, including using theme data, inline styles, and custom widgets.

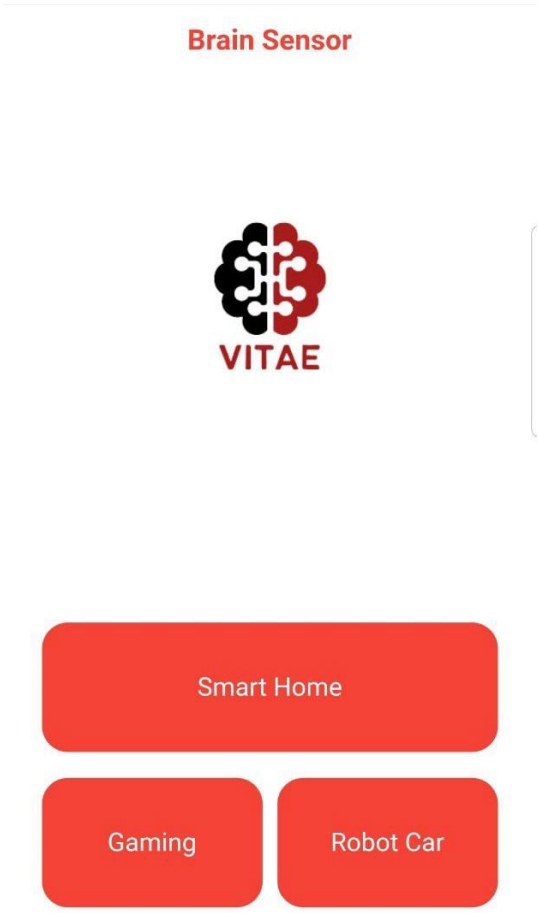


Figure 6.16 Mobile App. Interface1

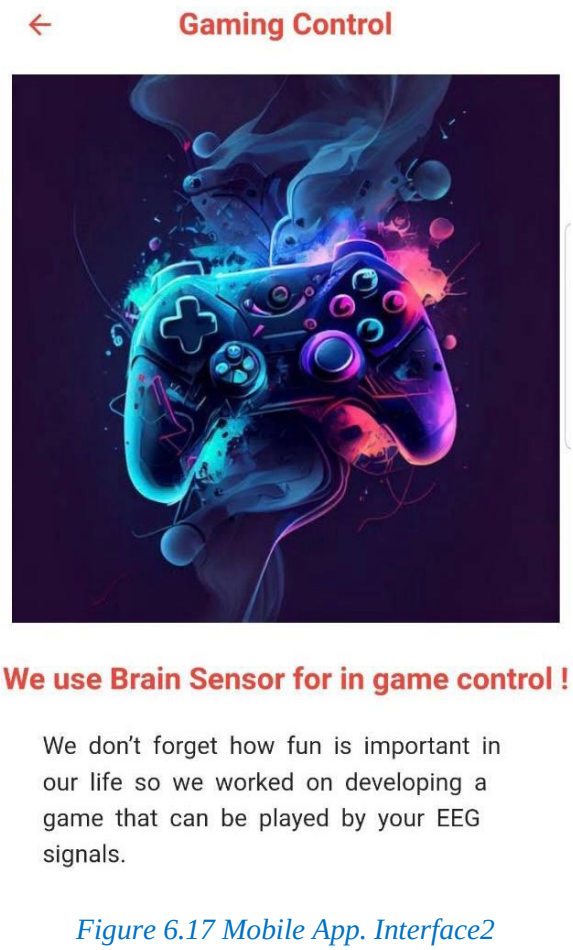


Figure 6.17 Mobile App. Interface2

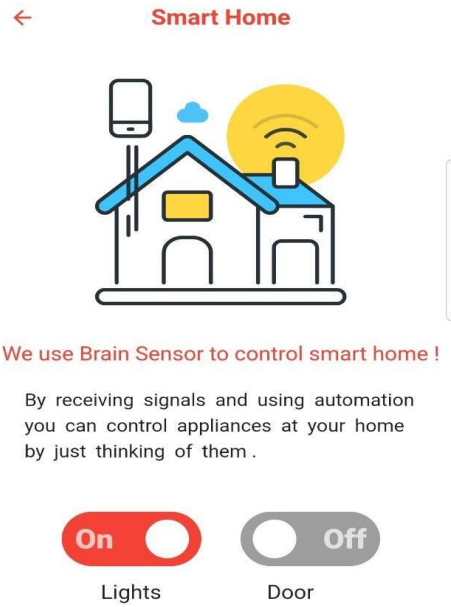


Figure 6.18 Mobile App. Interface3

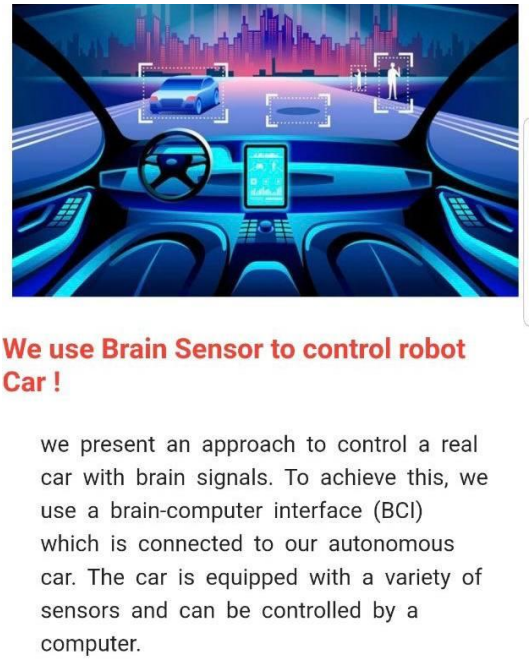


Figure 6.19 Mobile App. Interface4

Chapter 7

Applications

Chapter 7

Applications

7.1 Overview

To be more involved in people's lives and provide more assistance, we have implemented three applications which are smart home, smart car, game.

7.2 Applications

7.2.1 Smart home

Since our smart home has two systems—an internal system and an exterior system—we aim to assist humans in their homes by regulating household appliances.

7.2.1.1 Internal system

We have a fully automated internal system:

- With the use of LDR, we can set the light to turn on automatically at night and off automatically throughout the day.
- To detect a gas leak and sound an alert, MQ-5
- LM-35 to detect the temperature, and the fan will turn on or off based on that measurement
- An ultrasonic sensor to make garage auto entry easier
- The system password input and the sensor status are displayed on an LCD panel.
- Enter the system password using the keypad.

7.2.1.2 External system

Our external system that depends on signals coming out from the headset:

We receive human brain impulses from the headset and transform them into numerical data using the cortex. We then upload this data to the firebase, which then sends it to the Node MCU to control turning on and off of lights and opening and closing of doors.

Figure 7.1 Smart home

7.2.2 Smart car

The Node MCU in our smart car implements the desired action by receiving signals from the firebase. Our smart automobile has a forward motion and a rotation.

Figure 7.2 Smart car photo 1

Figure 7.3 Smart car photo 2

7.2.3 Jump Game

The game application receive signals from the firebase and control the game directly.

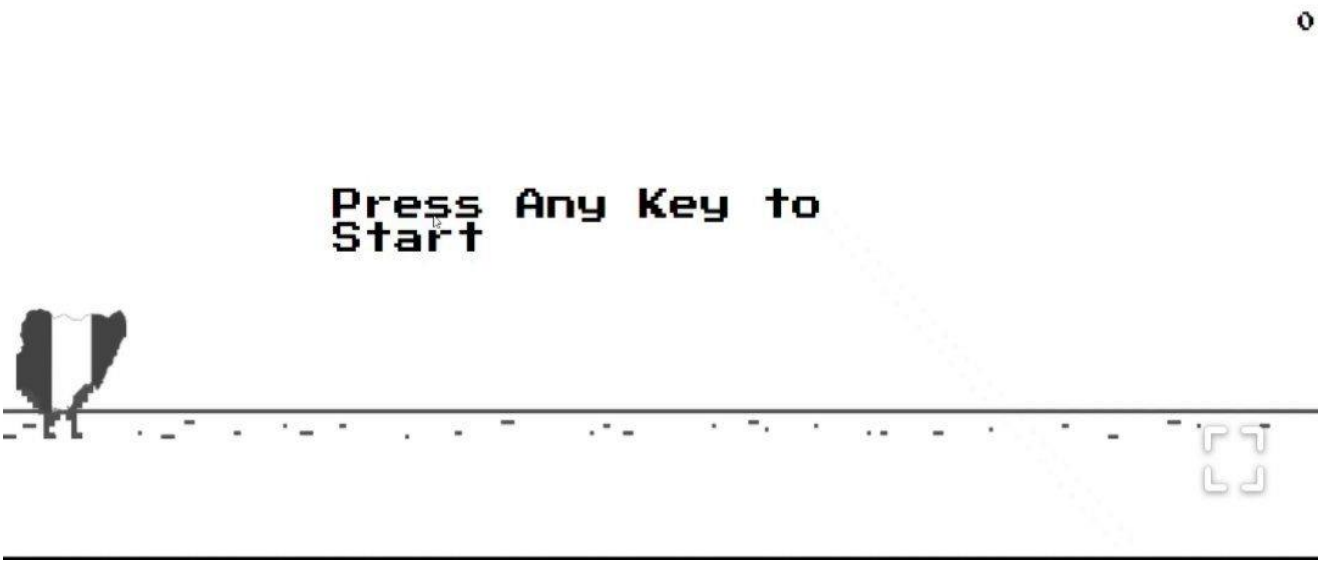


Figure 7.4 Game Application

Chapter 8

Conclusion and Future work

Chapter 8

Conclusion & Future Work

8.1 Conclusion

In conclusion we discovered that paralysis is more common than previously believed and impacts communication, the most crucial component of daily life that meets a person's requirements. In order to give the disabled a comfortable atmosphere and to make their life as independent and natural as possible, we consider the issue and work to find a solution.

We discovered that in order to control anything in your life, you must either create it yourself, if it is smart you can use your mobile phone our just say what you want and it will be executed, and it will be done. As a result, we sought to make it easier for you to get what you want done.

So, In order to create a project that receives signals from the human mind via the headset and sends them to the microcontroller so it can read and translate the events of each application, we employed AI and IOT approaches.

By utilizing automation and receiving signals, you may operate your home's appliances simply by thinking about them, driving the way you believe you should, and we labored to create a game that can be played using EEG signals.

8.2 Future Work

Future work will focus on

- Expanding the applications.
- Increasing the precision of signals obtained from the brain.

References

References

[1] Schaumont, P. R. (2010). A Practical Introduction to Hardware/Software Codesign, Springer.

a. Excellent book with disciplined design of hardware and software for embedded applications.

[2] E. A. Lee and P. Varaiya, *Structure and Interpretation of Signals and Systems*, Addison-Wesley, 2003.

a. Background material on signals and systems. Review of EECS 20. Available in PDF in pre-publication version.

[3] M. Barr, Anthony Massa, *Programming Embedded Systems*, second edition, O'Reilly, 2006. Available on line at http://www.oreilly.com/catalog/embsys2/.

a. A practical, how-to book.

[4] G. C. Buttazzo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*, Second Edition, Springer, 2005.

a. The book to read about RTOS scheduling (EDF and rate monotonic scheduling, for example). Buttazzo is the top expert in the field and excellent writer. On reserve in the Engineering Library.

[5] Schaumont, P. R. (2010). A Practical Introduction to Hardware/Software Codesign, Springer.

a. Excellent book with disciplined design of hardware and software for embedded applications.

[6] E. A. Lee and P. Varaiya, *Structure and Interpretation of Signals and Systems*, Addison-Wesley, 2003.

a. Background material on signals and systems. Review of EECS 20. Available in PDF in pre-publication version.

[7] M. Barr, Anthony Massa, *Programming Embedded Systems*, second edition, O'Reilly, 2006. Available on line at http://www.oreilly.com/catalog/embsys2/.

a. A practical, how-to book.

[8] G. C. Buttazzo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*, Second Edition, Springer, 2005.

a. The book to read about RTOS scheduling (EDF and rate monotonic scheduling, for example). Buttazzo is the top expert in the field and excellent writer. On reserve in the Engineering Library.

[9] Approaches based on artificial intelligence and the internet of intelligent things to prevent the spread of COVID-19: scoping review,Aya Sedky Adly, Afnan Sedky Adly, Mahmoud Sedky Adly,Journal of medical Internet research 22 (8), e19104, 2020. [8] Development and Design Principles of the Optimal Computer-and Roboticaided Healthcare Systems for the Diagnosis, Treatment, and Rehabilitation, Vitali Guitberg, Gennady Kouzaev,TechRxiv, 2020

[10] American Association for Artificial Intelligence (AAAI), Welcome to AI Topics, 2003, http://www.aaai.org/AITopics/ -- a Web-based library of introductory information about various areas of artificial intelligence; altogether, a resource with links to hundreds (thousands?) of sites, organized by an easy-to-use, interactive index.

References

[11] George Luger, Artificial Intelligence: Structures and Strategies for Complex Problem Solving, Fourth Edition Addison-Wesley, 2002 -- a well-respected introduction to artificial intelligence, as witnessed by its being in its fourth edition.

[12] Peter Norvig, AI on the Web, http://aima.cs.berkeley.edu/ai.html -- a list of over 800 links on various aspects of artificial intelligence.

[13] Nils J. Nilsson, Artificial Intelligence: A New Synthesis, Morgan Kaufmann Publishers, 1998 -- another fine introductory textbook on artificial intelligence.

[14] Stuart Russell and Peter Norvig, Artificial Intelligence: A Modern Approach, Second Edition, Prentice-Hall, 2003 -- the leading introductory textbook in the field.

[15] American Association for Artificial Intelligence, Expert Systems, http://www.aaai.org/AITopics/html/expert.html -- an on-line index of materials, including tutorials on the subject. Highly recommended as a starting point for readings on the subject.

[16] Virginia Barker and Dennis O'Connor "Expert Systems for Configuration at Digital: XCON and Beyond", *Communications of the ACM*, Volume 32, Number 3, March 1989, pp. 298-317.

[17] Daniel Bobrow et al, "Expert Systems: Perils and Promise", Volume 29, Number 9, September 1986, pp. 880-894.

[18] Joseph Giarratano and Gary Riley, Expert Systems: Principles and Programming, Third Edition Brooks/Cole Publishers, 1998.

[19] Frederick Hayes-Roth, "The Knowledge-Based Expert System: A Tutorial", *IEEE Computer*, Volume 18, Number 9, September 1984, pp. 11-28.

[20] Frederick Hayes-Roth, "Rule-Based Systems", *Communications of the ACM*, Volume 28, Number 9, September 1985, pp. 921-932.

[21] Peter Jackson, Introduction to Expert Systems, Third Edition, Addison-Wesley, 1998.

[22] Gary Riley, CLIPS: A Tool for Building Expert Systems, 2002. (a Web site that provides software and support for building expert systems; the software is based in standard C for portability)

[23] Scandia National Laboratories, Jess: the Rule Engine for the Java Platform, 2003. (a Java-based expert system and environment, originally based on CLIPS)

[24] Henry Walker, Vikram Subramaniam, and Ivan Sykes, "An Expert System to Place Incoming Students in Math and CS Classes", *Journal of Computer Science Education*, Volume 3, Number 3, 1992, pp. 223-232.

[25] [37] Fowler M (2003) UML distilled: a brief guide to the standard object modeling language, 3rd edn. Addison-Wesley Professional, Boston Google Scholar [38] Haller S (2010) The things in the internet of things. Tokyo: s.n Google Scholar

References

[26] Acharya, S, Fifer, MS, Benz, HL, Crone, NE, Thakor, NV. Electrocorticographic amplitude predicts finger positions during slow grasping motions of the hand. J Neural Eng. 2010 Aug;7(4):046002.Google Scholar

[27] Andersen, RA, Hwang, EJ, Mulliken, GH. Cognitive neural prosthetics. Annu Rev Psychol. 2010;61:169–90, C1–3.CrossRef Google Scholar PubMed

[28] Anderson, C, Sijercic, Z. Classification of EEG signals from four subjects during five mental tasks. In Solving Engineering Problems with Neural Networks: Proceedings of the Conference on Engineering Applications in Neural Networks (EANN'96), 1996, Bulsari, AB, Kallio, S, and Tsaptsinos, D (eds.), pp. 407–14.

[29] Ayaz, H, Shewokis, PA, Bunce, S, Schultheis, M, Onaral, B. Assessment of cognitive neural correlates for a functional near infrared-based brain computer interface system. Augmented Cognition, HCII, 2009;LNAI 5638, pp. 699–708.Google Scholar

[30] Babiloni, C, Carducci, F, Cincotti, F, Rossini, PM, Neuper, C, Pfurtscheller, G, Babiloni, F. Hum an movement-related potentials vs desynchronization of EEG alpha rhythm: a high-resolution EEG study. Neuroimage. 1999 Dec;10(6):658–65.CrossRef Google Scholar

[31] Barber, D. Bayesian Reasoning and Machine Learning. Cambridge University Press, 2012.Google Scholar

[32] Bayliss, JD. Use of the evoked potential P3 component for control in a virtual apartment. IEEE Trans Neural Syst Rehabil Eng. 2003;11(2):113–16.Google Scholar

[33] Bear, MF, Connors, BW, Paradiso, MA. Neuroscience: Exploring the Brain., 3rd ed., Lippincott Williams & Wilkins, Baltimore, MD, 2007.Google Scholar

[34] Bell, AJ, Sejnowski, TJ. An information-maximization approach to blind separation and blind deconvolution. Neural Computation. 1995;7:1129–59.CrossRefGoogle Scholar

[35] Bell, CJ, Shenoy, P, Chalodhorn, R, Rao, RPN. Control of a humanoid robot by a noninvasive brain-computer interface in humans. J Neural Eng. 2008 Jun;5(2):214–20.CrossRef Google Scholar

[36] Bellavista, P, Corradi, A, Giannelli, C. Evaluating filtering strategies for decentralized handover prediction in the wireless internet. Proc. 11th IEEE Symposium Computers Commun., 2006.Google Scholar

[37] P. I. Grammatikis, P. G. Sarigiannidis, I. D. Moscholios, "Securing the Internet of Things: Challenges, threats, and solutions", Internet of Things, Vol. 5, 2019, pp. 41–70

[38] K. Ashton, et al., "That internet of things thing", Radio Frequency Identification (RFID) Journal Vol. 22, Issue 7, 2009, pp. 97–114

[39] P.P. Ray, "A survey on Internet of Things architectures", Journal of King Saud University – Computer and Information Sciences, Vol. 30, 2018, pp. 291–319

References

[40] M. M. Noor, W. H. Hassan, "Current research on Internet of Things (IoT) security: A survey", Computer Networks, Vo. 148, 2019, pp. 283–294

[41] Y. Ismail, "IoT and Smart Home Automation", book, IntechOpen Publisher, 2019

[42] https://www.intechopen.com/chapter/pdf-download/65738

[43] Z. Shouran, A. Ashari, T. Priyambodo, "Internet of Things (IoT) of Smart Home: Privacy and Security", International Journal of Computer Applications, Vol. 182, No. 39, February 2019, pp. 0975 – 8887

[44] D. Mocrii, Y. Chen, P. Musilek, "IoT-based smart homes: A review of system architecture, software, communications, privacy and security", Internet of Things Vol.1, Issue 2, 2018, pp. 81–98

[45] *"ESP8266 Overview". Espressif Systems.* Retrieved 2017-10-02.

[46] https://www.javatpoint.com/atmega32-avr-microcontroller

[47] https://www.techtarget.com/whatis/definition/sensor

[48] https://components101.com/sensors/mq-4-methane-gas-sensor-pinout-datasheet

[49] https://www.progressiveautomations.com/pages/actuators

[50] https://www.electronicshub.org/what-is-relay-and-how-it-works/

[51] https://forum.digikey.com/t/relay-features/9453

[52] https://www.electroduino.com/ldr-sensor-module-how-ldr-sensor-works/

[53] https://realpars.com/servo-motor/

[54] https://www.linearmotiontips.com/servo-motor-basic/

[55] https://www.circuiteasy.com/what-is-a-motor-driver

[56] https://www.investopedia.com/terms/s/smart-home.asp

[57] https://www.circuiteasy.com/what-is-a-motor-driver

[58] https://www.heavy.ai/technical-glossary/embedded-systems

[59] https://en.wikipedia.org/wiki/Embedded_system

[60] https://www.techtarget.com/iotagenda/definition/embedded-system

[61] https://www.theengineeringprojects.com/2021/06/characteristics-of-embedded-systems.html

[62] https://electricalfundablog.com/embedded-system-characteristics-types-advantages-disadvantages/

[63] https://radixweb.com/blog/embedded-systems-concept-worth-understanding-implementing

[64] https://ebrary.net/22041/computer_science/typical_architecture_embedded_system

[65] https://www.tutorialspoint.com/embedded_systems/es_architectures.htm

[66] https://www.polytechnichub.com/advantages-disadvantages-embedded-systems/

[67] https://www.techtarget.com/iotagenda/definition/embedded-system

[68] https://matterport.com/cortex-ai

References

[69] https://seaborn.pydata.org/tutorial/introduction

[70] https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

[71] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[72] https://www.ibm.com/topics/naive-bayes

[73] https://vitalflux.com/linear-discriminant-analysis-lda-concepts-examples/

[74] https://www.sciencedirect.com/topics/engineering/deep-neural-network

[75] What is machine perception? How artificial intelligence (AI) perceives the world | VentureBeat

[76] https://herovired.com/learning-hub/blogs/characteristics-of-iot/

ملخص عربي

يتأثر عدد أكبر مما كان يُعتقد سابقًا بالاعاقة الجسدية، والذي له تأثير على كل جوانب حياته ويتسبب له في عدم القدرة على توفير احتياجاته اليومية مما يؤثر سلبا عليه نفسيا وجسديا ويؤثر على قدرته على التواصل - وهو العنصر الأكثر أهمية في الحياة اليومية.

لذلك بالإضافة إلى هدفنا المتمثل في منح ذوي الاحتياجات جوًا مريحًا ومحاولة جعل حياتهم مستقلة وطبيعية قدر الإمكان، فإننا ننظر في المشكلة ونعمل على إيجاد حل.

فلذلك مع نمو التكنولوجيا والذكاء الاصطناعي، قمنا بتطبيق مفهوم واجهة الدماغ والحاسوب (BCI)ودمجنا أساليب انترنت الاشياء IOTوالذكاء الاصطناعي AI لإنشاء مشروع يتلقى إشارات من العقل البشري من خلال سماعة رأس، وإرسال إشارة EEGإلى متحكم دقيق، والذي يقوم بعد ذلك بقراءة وترجمة حدث كل تطبيق:

- من خلال تلقي الإشارات واستخدام الأتمتة ، يمكنك تشغيل الأجهزة في منزلك بمجرد التفكير فيها.
- يمكنك أيضًا قيادة سيارتك بالطريقة التي تفكر فيها.
- لا ننسى مدى أهمية المتعة في حياتنا ، لذلك عملنا على إنشاء لعبة يمكن أن تلعبها إشارات EEG الخاصة بك.