Mobile Health Revolution:
The MobiCare Approach to Patient Support

Supervisor:

Dr. Ayman Soliman

كلية الهندسة ببنها

FACULTY OF ENGINEERING- BENHA

Presentation Content

Introduction

- Introduction Our Team
- introduction to the project
- Why this idea was chosen (the problem it solves)

Problem Statement

- What challenges do people with disabilities face with traditional wheelchairs?
- The importance of a smart, remotely controlled and health-monitored wheelchair

1 2 3 4

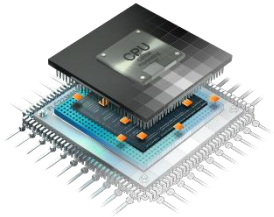
Objectives

- Facilitate mobility and control
- Monitor health conditions

System Overview

- How the System Works
- Devices used
- Block Diagram

Presentation Content



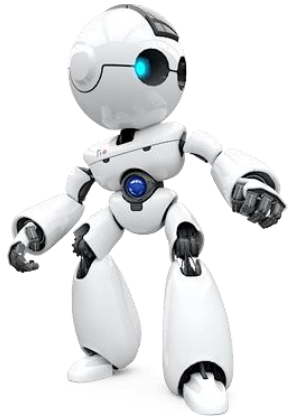
Embedded Section

- Embedded Components
- Problems and Solutions

Embedded & AI Integration

- Integration between Embedded Systems and Artificial Intelligence

5 6 7 8



AI Section

- Voice (How to build the model)
- Testing for voice model
- Face direction Tracking
- EYE direction Tracking

Mobile App Section

- UI
- Technology (Flutter)
- Database (Firebase)

Presentation Content

Mobile & Embedded Link

- Integration between Mobile Applications and Embedded Systems

Conclusion

- Summary of the idea
- Key outcomes and achievements
- Is the project scalable or ready for development?

9 10 11

Testing & Results

- Photos and videos of the testing phase
- Data samples and system performance evaluation

Introduction to the Project

In our project, we sought to solve a problem that affects many people: how to help patients and people with special needs move easily without relying on anyone else. We designed and implemented a smart wheelchair, called MobiCare, that uses modern technologies such as artificial intelligence and mobile applications to control the wheelchair's movement in various ways, such as voice, eye tracking, or even a mobile app. The project's primary goal is to improve patients' quality of life and facilitate their movement in a safe and simple manner.

Why This Idea Was Chosen (Problem It Solves)

Many individuals with physical disabilities face daily challenges in using traditional wheelchairs, especially those who cannot operate them manually. Most available solutions are either too expensive, lack intelligent control, or do not support health monitoring features. This project aims to address these issues by creating a cost-effective, intelligent, and accessible system that improves mobility, enables remote control, and allows caregivers or doctors to monitor the user's health in real-time.

What Challenges Do People with Disabilities Face with Traditional Wheelchairs?

Lack of physical ability to control the wheelchair manually

No remote-control options

No health monitoring features

There is no way to track the user’s vital signs, making it difficult to detect health issues early.

Lack of safety and independence

People using traditional wheelchairs often feel dependent onothers to help them move, which takes away their sense ofindependence and privacy in daily life.

High cost of existing smart solutions

Advanced smart wheelchairs on the market are often expensive and not accessible for everyone.

The Importance of a Smart Wheelchair

Enables remote control through mobile application

Users or caregivers can move the wheelchair without physical effort, enhancing convenience and safety.

Integrates health monitoring sensors

Vital signs like heart rate or temperature can be tracked in real-time, allowing quick response from caregivers or doctors.

Improves user independence

Smart features reduce reliance on others, helping users regain a sense of control over their mobility.

Supports real-time communication with medical staff

Data can be sent instantly to doctors or family members for better decision-making and follow-up.

Integrates AI and embedded systems to enable smart, accessible, and health-aware wheelchair control.

Objectives Of The Project

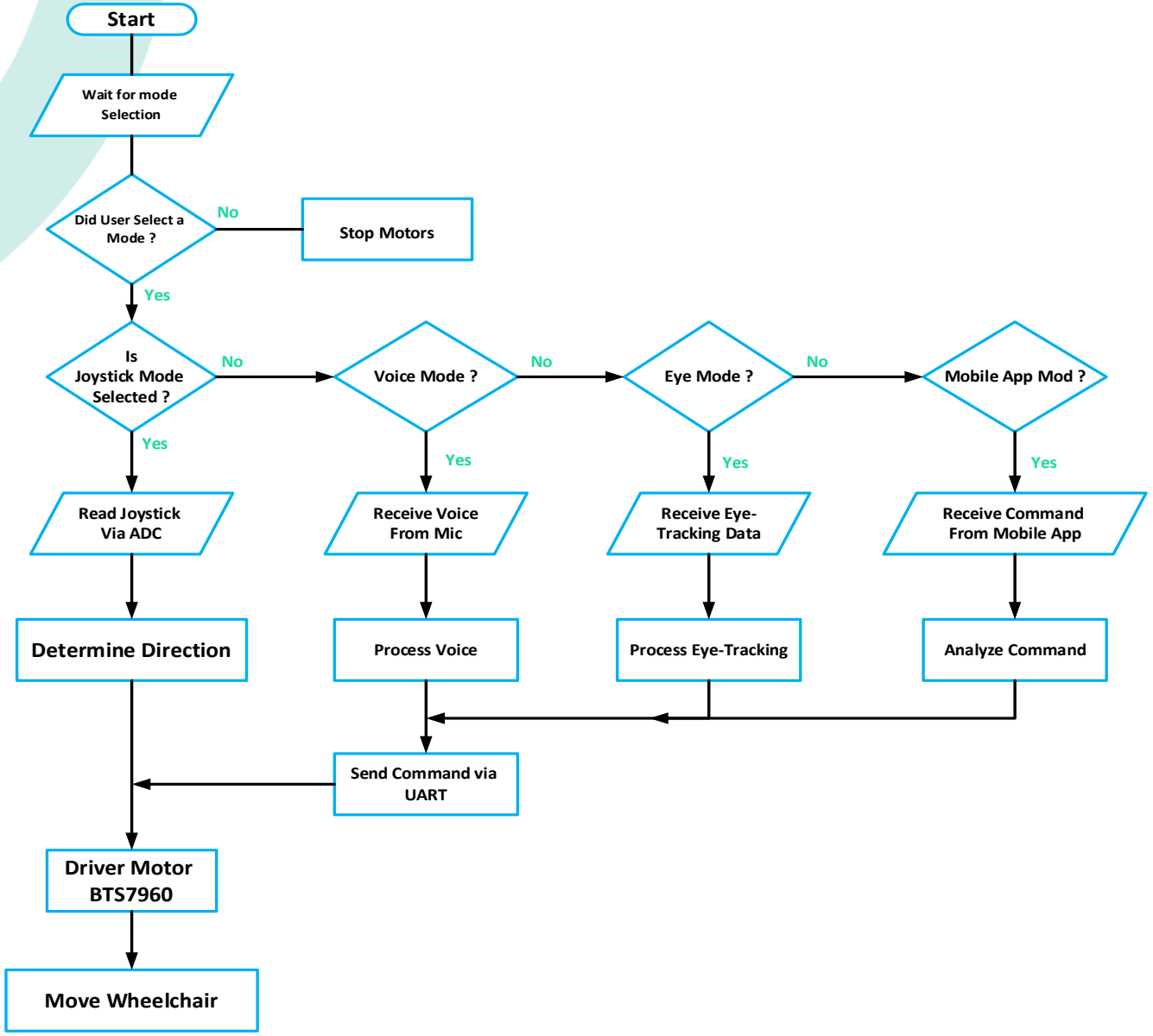
Facilitate Mobility and Control

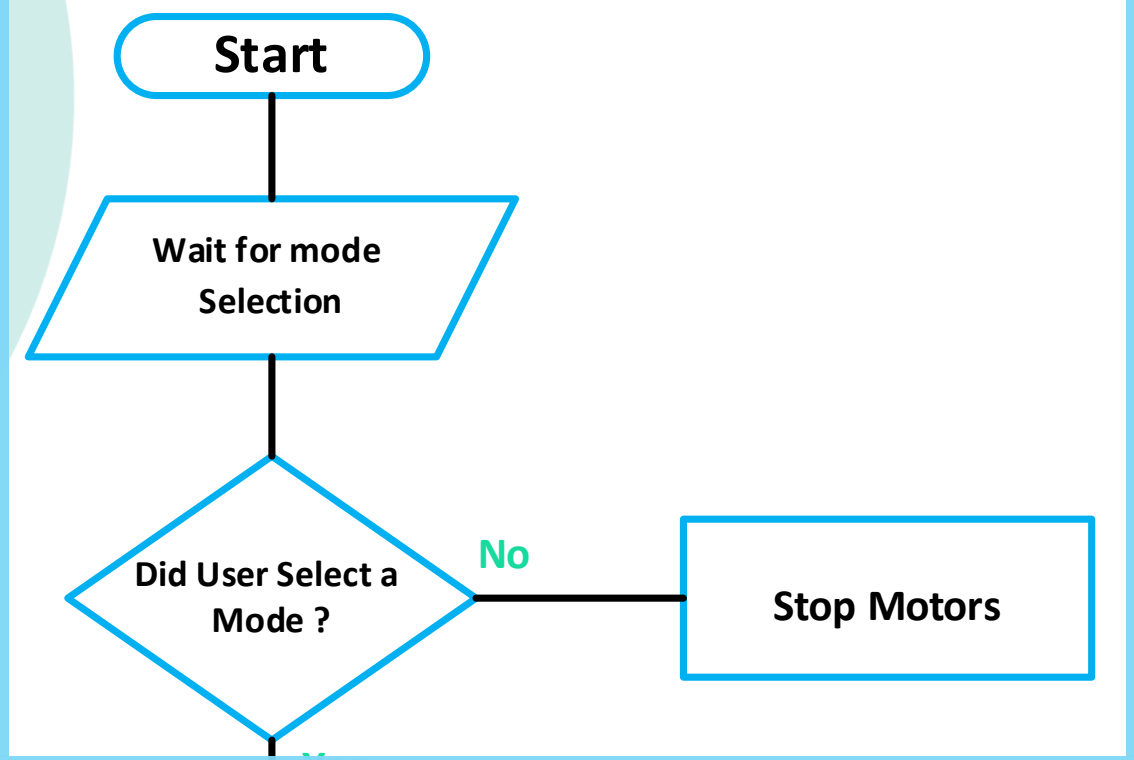
- Remote control via mobile application.
- Support for users with limited motor abilities.
- More flexibility and freedom.

Monitor Health Conditions

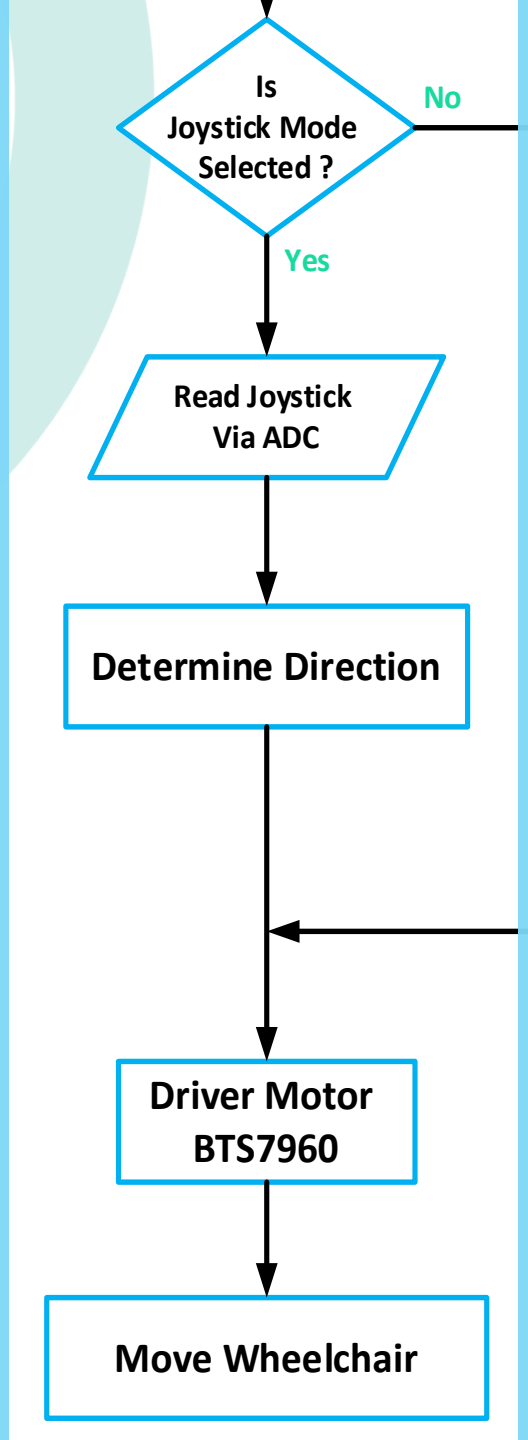
- Real-time health monitoring using sensors.
- Automatic data collection and storage.
- Improved safety and medical response.

How the System Works ?





1) **Mode Selection:**
The user selects the operating mode using push buttons connected to the STM32 microcontroller.

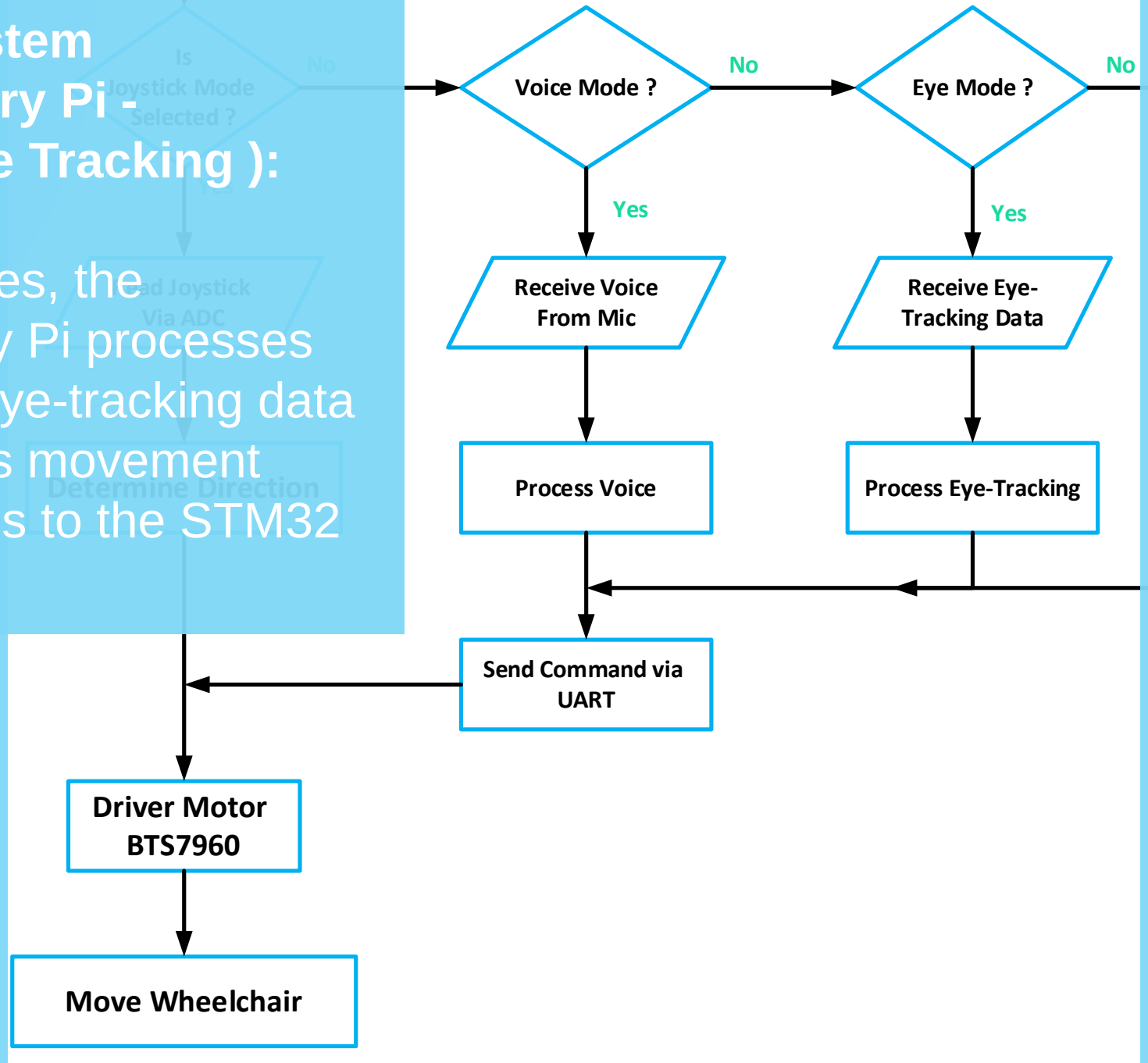


2) Manual Mode (Joystick):

- When the joystick mode is selected, the STM32 reads the joystick analog values (X, Y) through the ADC.
- Then, it determines the direction based on the joystick position
- After that, The STM32 receives the movement commands and controls the DC motors through the BTS7960 motor driver to move the wheelchair accordingly.

3) AI System (Raspberry Pi - Voice/Eye Tracking):

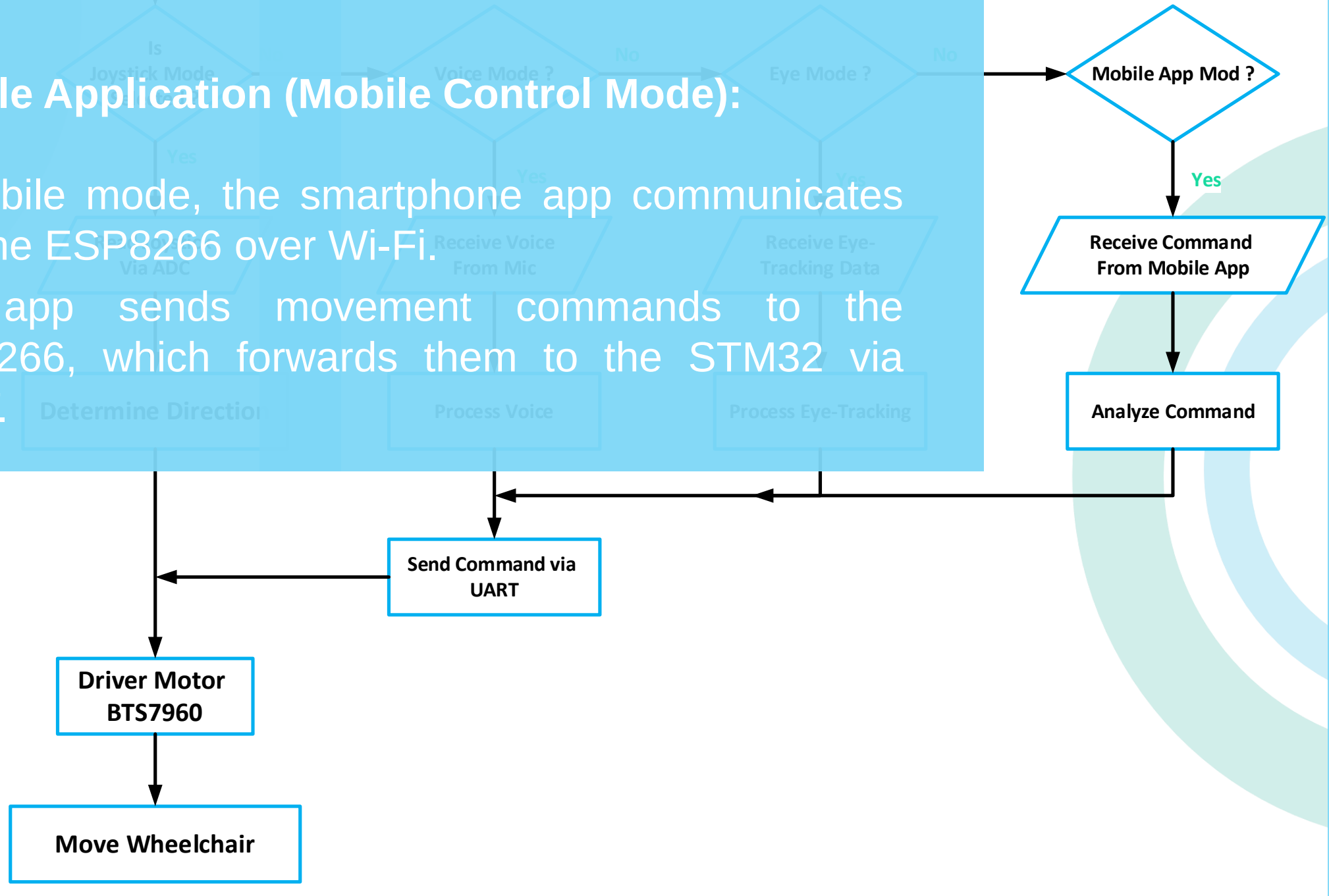
In AI modes, the Raspberry Pi processes voice or eye-tracking data and sends movement commands to the STM32 via UART.



4) Mobile Application (Mobile Control Mode):

In mobile mode, the smartphone app communicates with the ESP8266 over Wi-Fi.

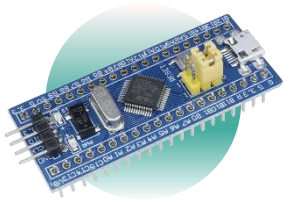
The app sends movement commands to the ESP8266, which forwards them to the STM32 via UART.



5) **Health Monitoring (ESP8266 + Firebase):**
   Health sensors (such as heartbeat and temperature) are connected to the ESP8266.

Devices used

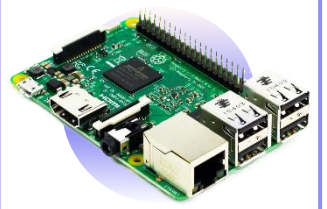
STM32 MCU



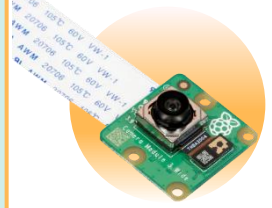
DC MOTOR

Joystick Module

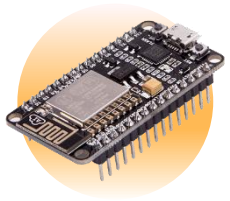
Raspberry Pi 4



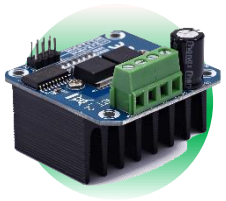
Raspberry CAM



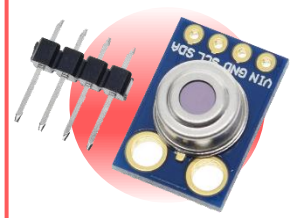
ESP WIFI Module



BTS7960 MOTOR Driver



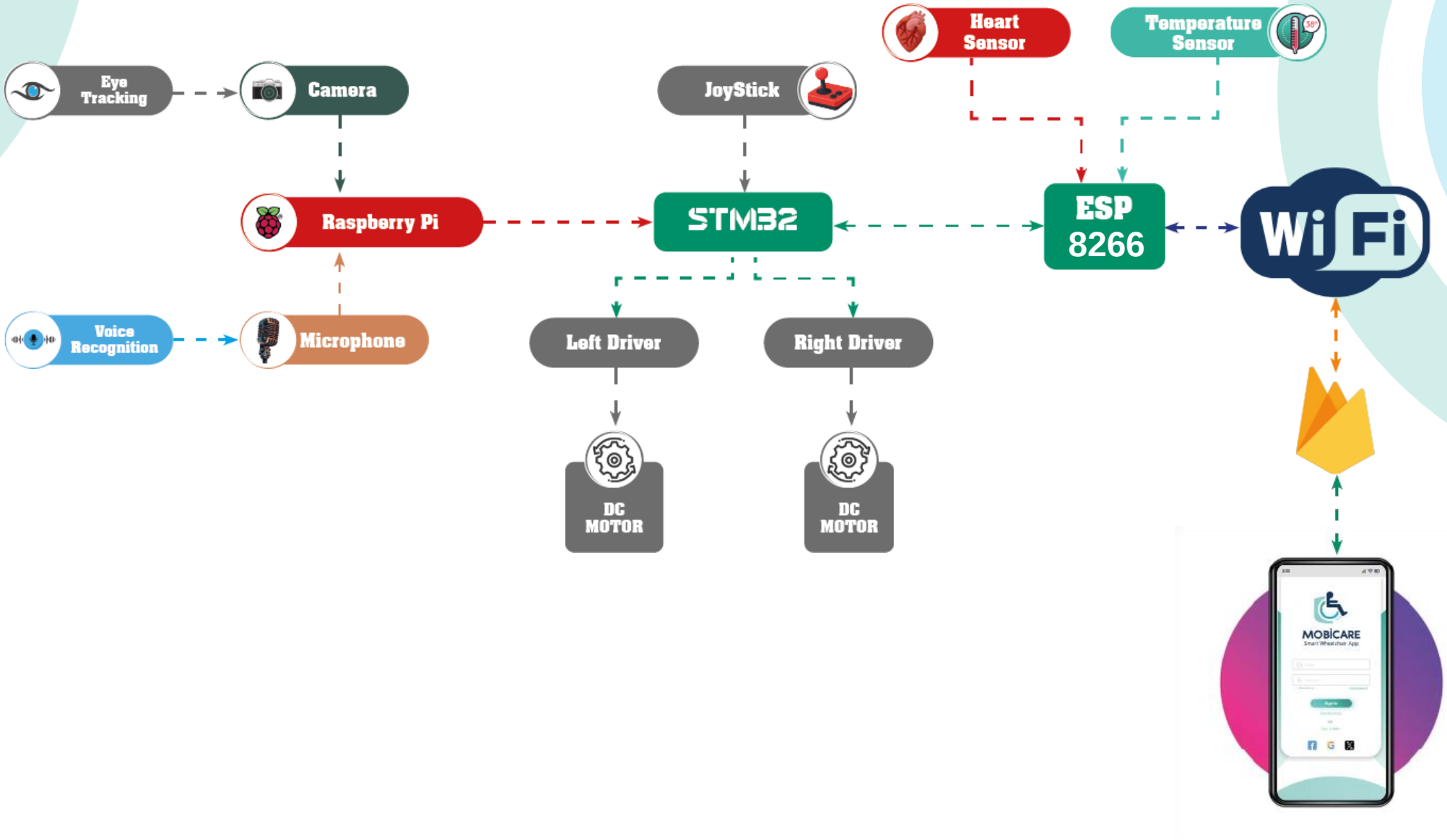
Temp MIx90614



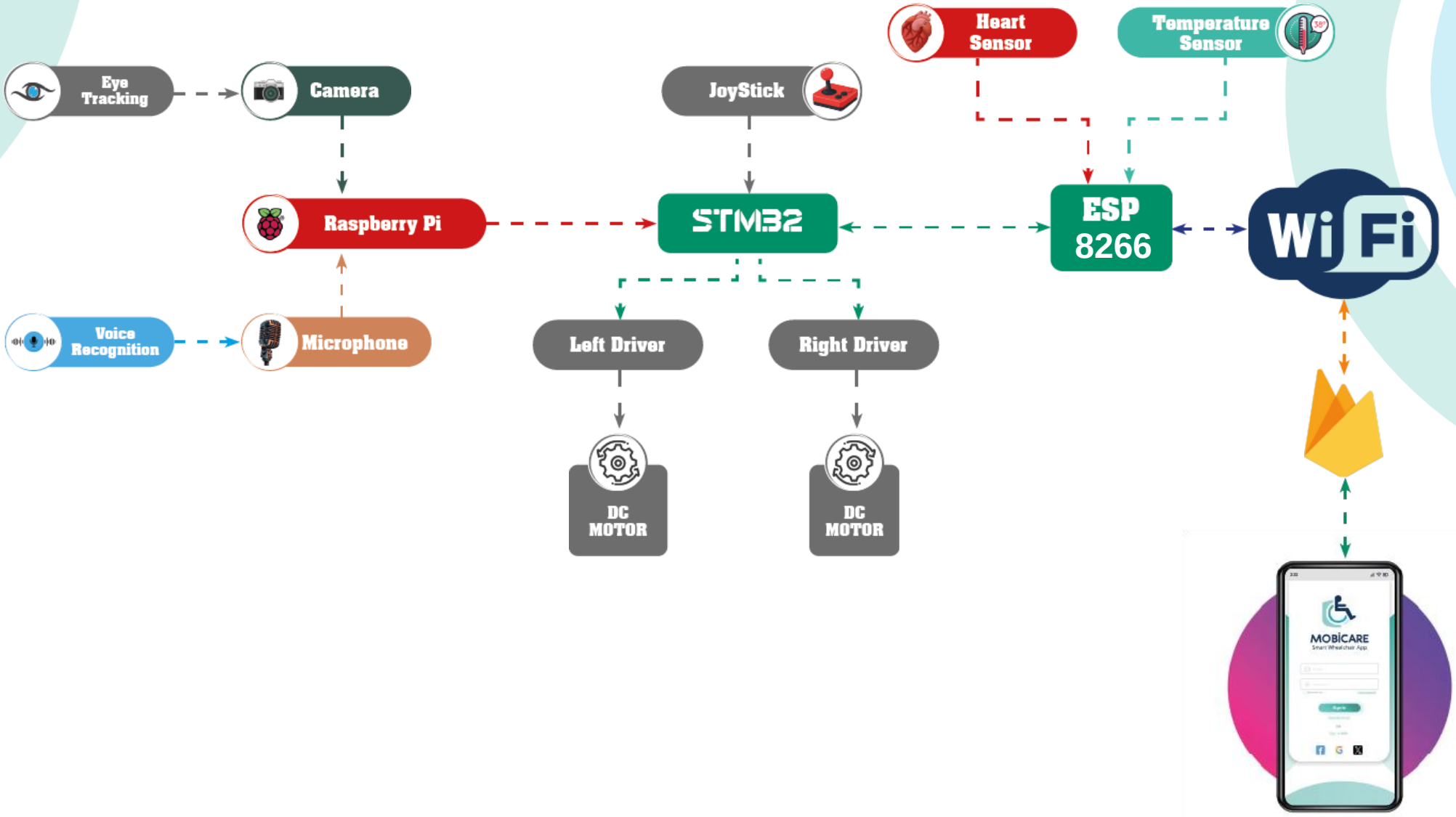
Heartbeat Max30102

Microphone

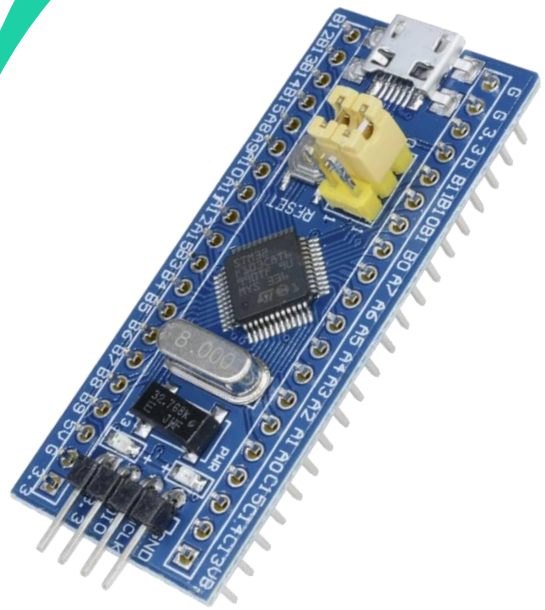
Block Diagram



Block Diagram



STM32 Microcontroller



Overview:

STM32 is a fast and efficient microcontroller, widely used in embedded systems.

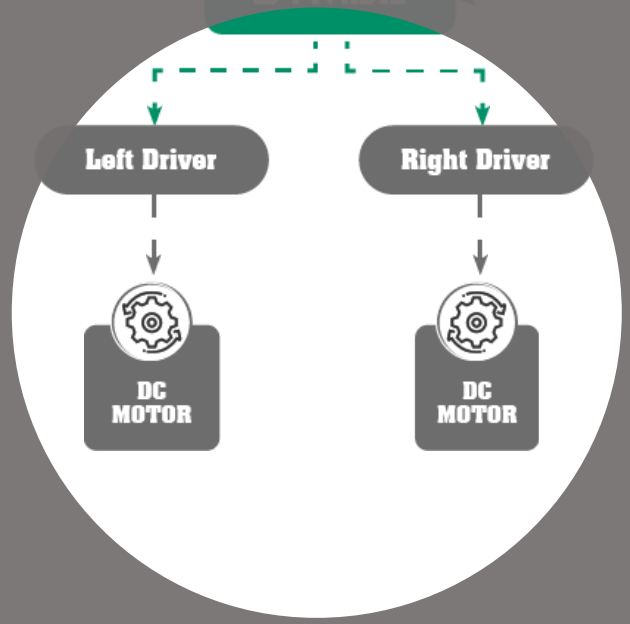
In Our Project:

- It works as the main brain of the system.
- Reads input signals from the push buttonsto detect the user's chosen mode and movement direction.
- Sends control signals to the BTS7960 motor driver, managing the wheelchair’s movement in real-time.

BTS7960 Motor Driver

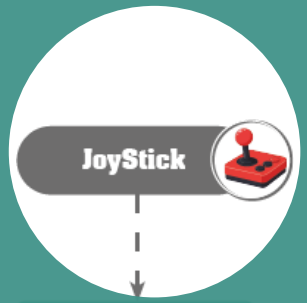
Why We Used It?

- Used to control the direction of the DC motors in the wheelchair.
- Contains an H-Bridge circuit that works like a set of electronic switches (transistors) to reverse the motor's direction.
- Can handle high current, making it suitable for powerful wheelchair motors 43A.



JOYSTICK (Manual Mode)

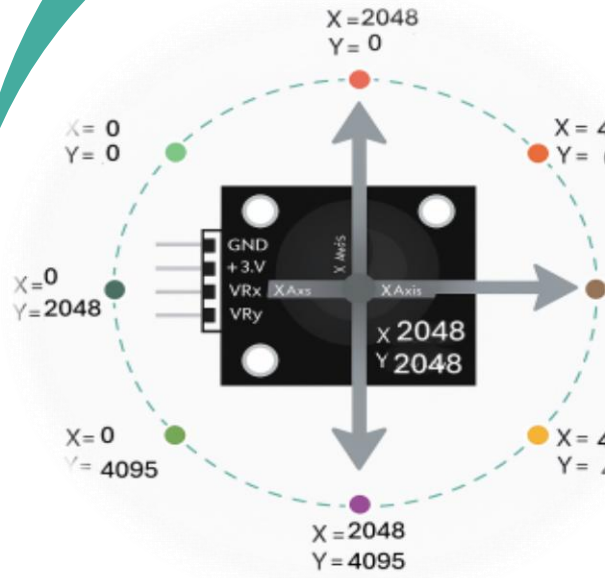
How We Read Direction ?



- The joystick outputs voltage values depending on its movement:
- X and Y values range from ~0 to 4095 (if STM32 ADC is 12-bit).

- THRESHOLD_LOW = 1000
- THRESHOLD_HIGH = 3000

Direction	ADC Condition
LEFT	X < THRESHOLD_LOW
RIGHT	X > THRESHOLD_HIGH
UP	Y < THRESHOLD_LOW
DOWN	Y > THRESHOLD_HIGH
CENTER	Values in between (no movement)



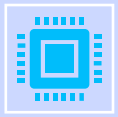
Problems and Solutions

Problems and Solutions

ATmega32A had unstable ADC & limited UART.

STM32F103 has 3 UARTs & better ADC handling (12 bit vs 10 bit).

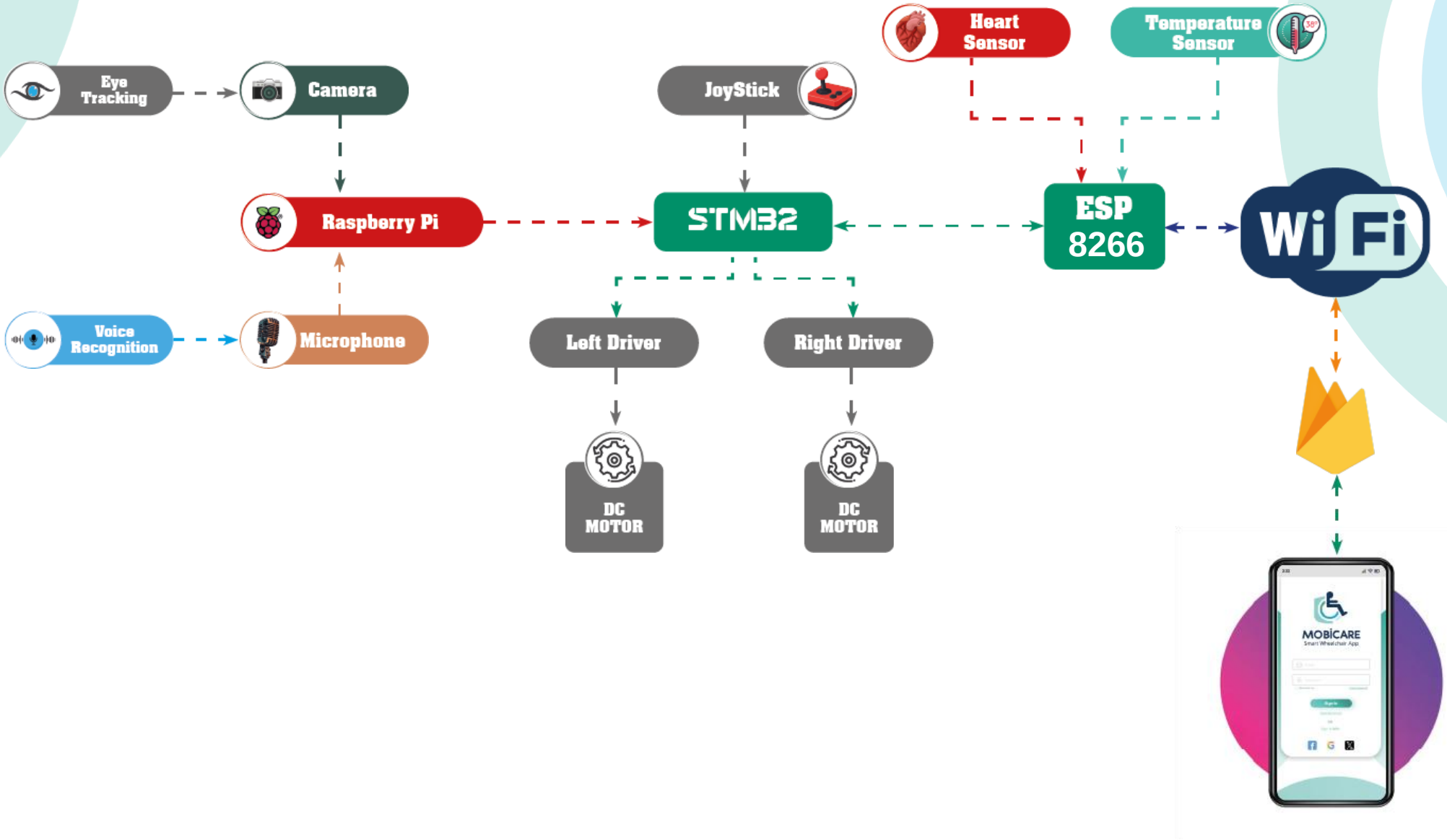
STM32 runs faster (72MHz vs 16MHz).



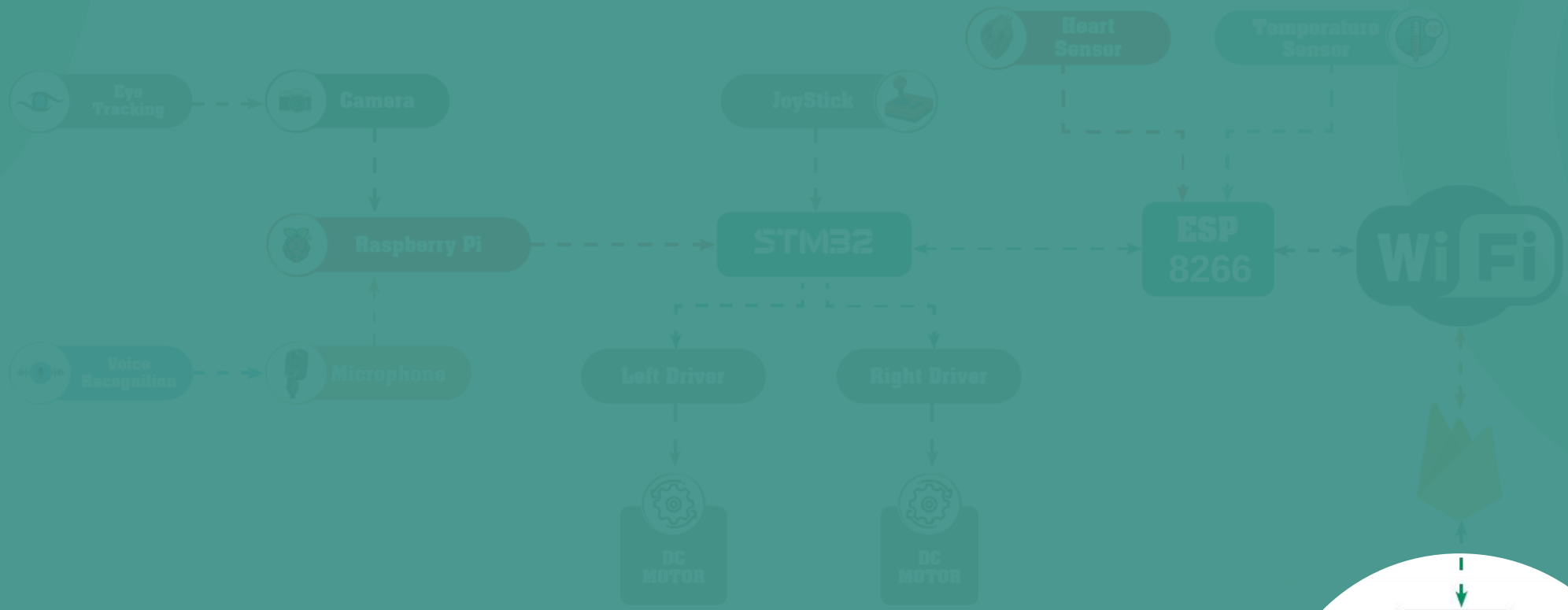
Easier to add new features (e.g., sensors, communication) cause of its has more I/O pins.

Cheaper & more available in the market.

Block Diagram



Block Diagram



UI

UI

Code

UI

Code

Database

UI

UI

Why Is UI Important?

Planning Ahead

Clear Design Vision

UI

Why Is UI Important?

Faster Build Time

Simplified Development

UI

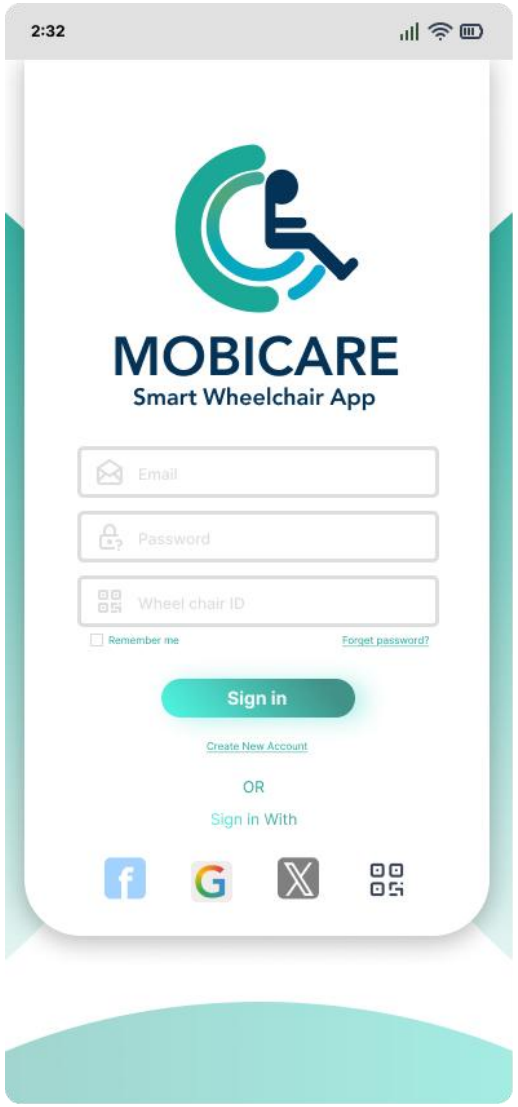
Why Is UI Important?

Understanding User Needs

Overview

Light Mode Design Screens

Light Mode Design Screens



Sign in Screen

Modern & Accessible Design:

Clean, user-friendly layout with soft colors (white, teal, blue) reflecting trust and innovation.

Structured Layout:

Central login card with rounded corners and soft shadow. Background features dynamic oval shapes for visual balance.

Clarity & Simplicity:

- Logo and app name at top for branding.
- Input fields with icons and placeholders.
- Highlighted “Sign in” button in gradient green.
- Subtle links: “Forgot password?”, “Remember me”.

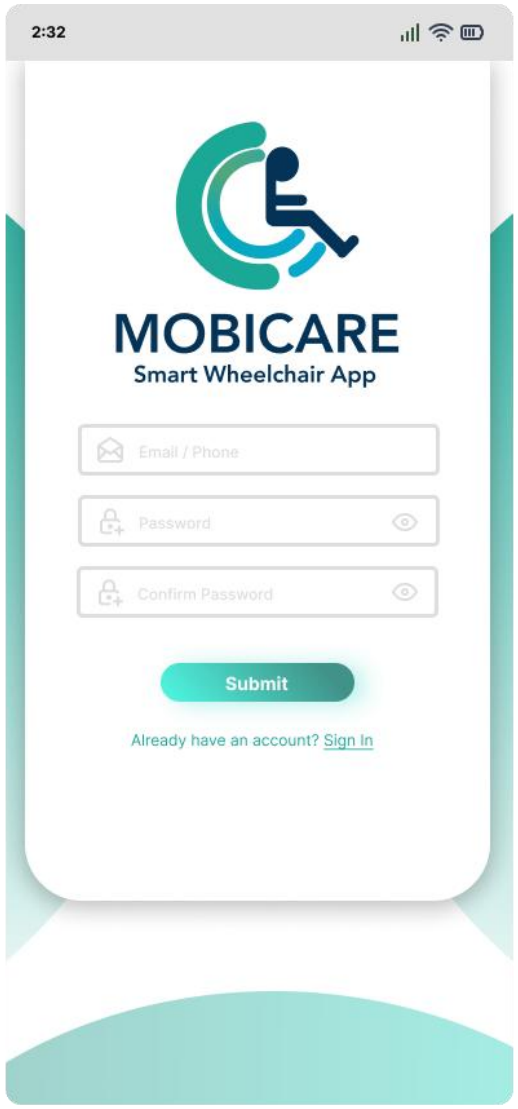
Visual Hierarchy:

- Bold typography for headings; light font for guidance. Clear button contrast and logical arrangement.

Accessibility Focus:

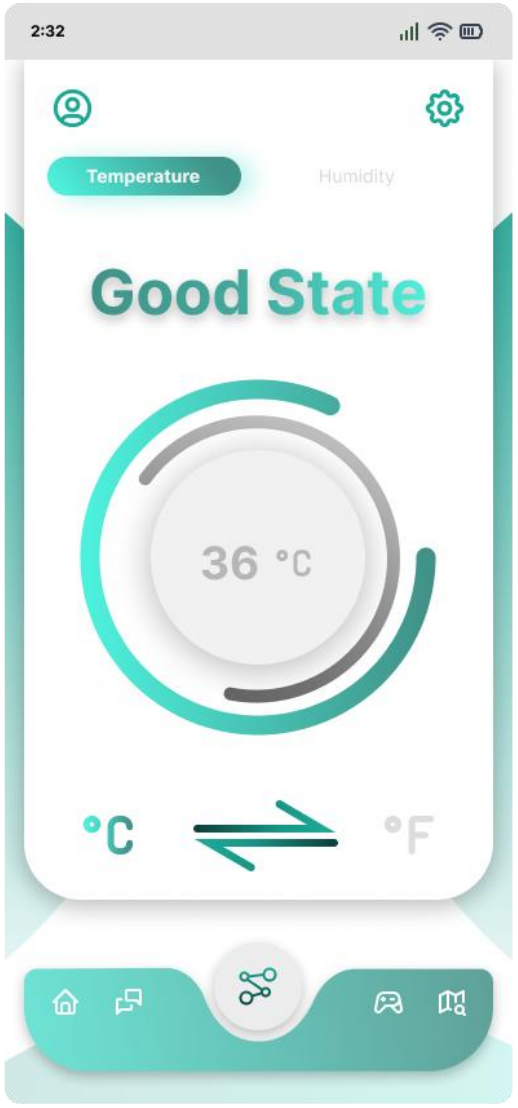
- Large, easy-to-tap fields. Icons aid recognition. Strong text/background contrast for readability.

Light Mode Design Screens



This screen shares many of the same design principles as the Sign-In Screen, such as clarity, consistency, and accessibility. However, it is uniquely tailored to support new user registration, offering additional input fields and features specific to account creation.

Light Mode Design Screens



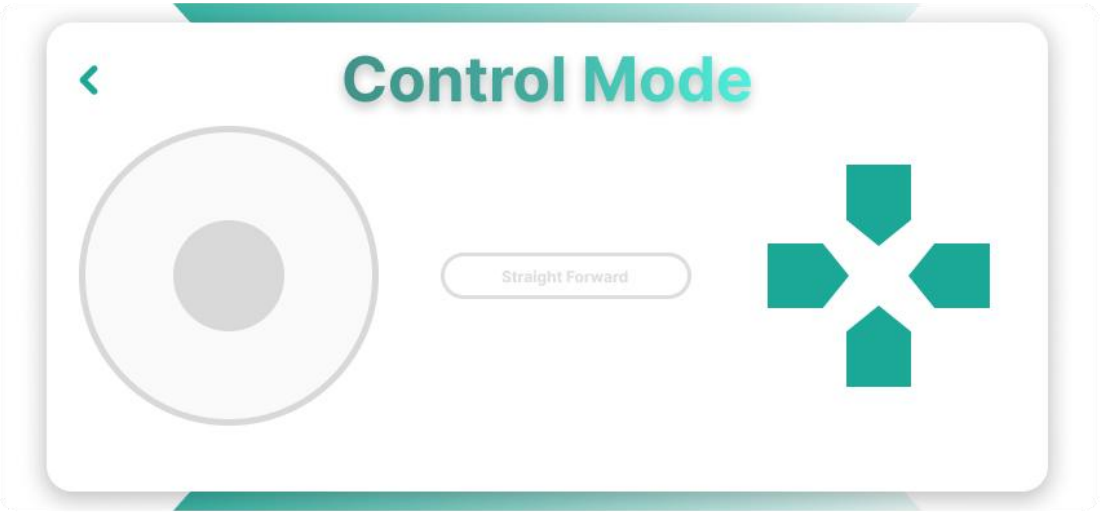
Visual Overview:

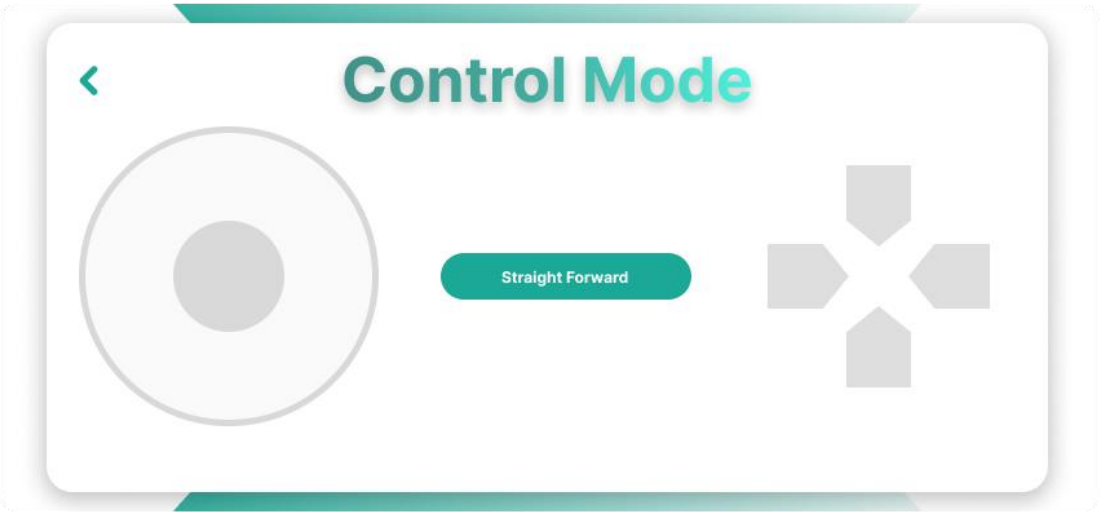
Clean, modern, and minimalistic design, Soft gradient background (teal & white), Focused on clarity, comfort, and professionalism.

Key Screen Components:

- **Top Bar:** Profile Icon (top-left), Settings Icon (top-right).
- **Temperature / Humidity Toggle:** Highlighted tab shows current selection
- **Health Status Indicator:** Displays "Good State" in bold font with gradient effect
- **Temperature Gauge:** Circular design showing 36°C with dynamic ring
- **Unit Switch:** Double-arrow icon toggles °C / °F for flexibility
- **Bottom Navigation Bar:** Icons: Home (active), Connectivity, Main Control, Control Mode, Maps.

Light Mode Design Screens



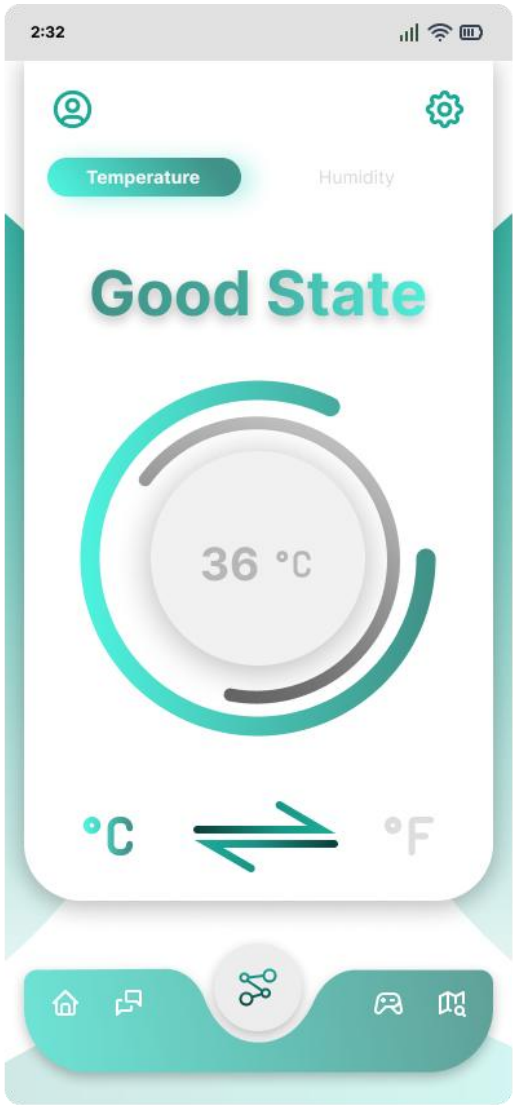


A back button (arrow icon) is positioned on the top left, allowing users to return to the previous screen.

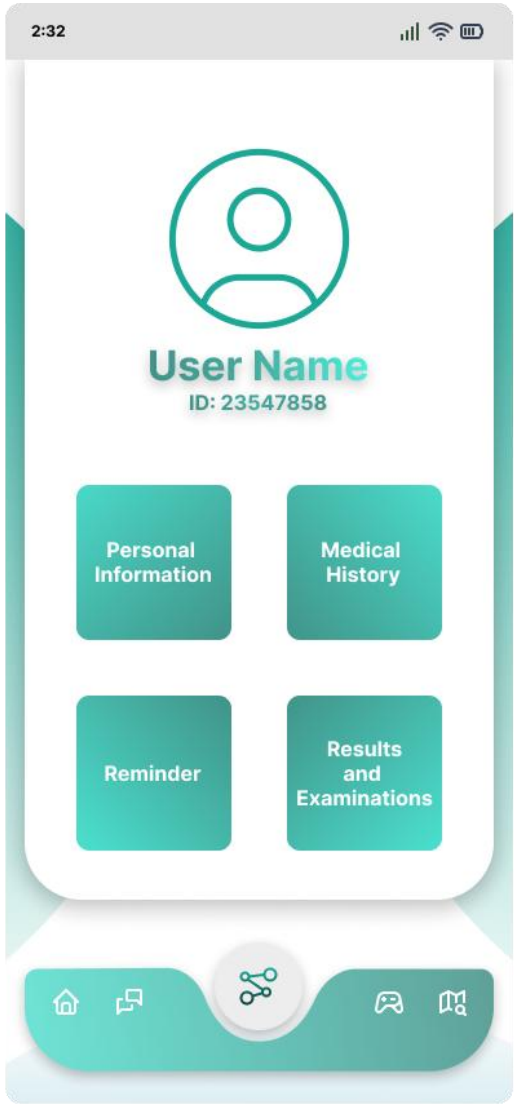
The left section contains a joystick-like circular control, likely used for analog movement control.

The right section features a directional button layout, which includes six directional arrows, indicating precise movement controls for the wheelchair.

Light Mode Design Screens



Light Mode Design Screens



Person Screen

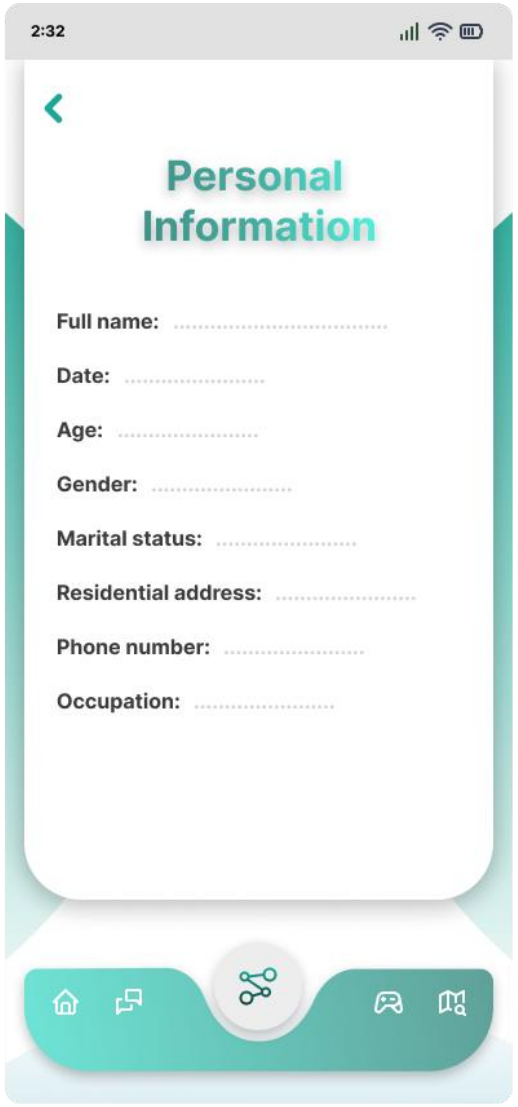
Screen Component:

- Patient Avatar & Name.
- Patient ID.

Feature Buttons:

- Patient Information.
- Medical History.
- Reminder.
- Results and Examinations.

Light Mode Design Screens

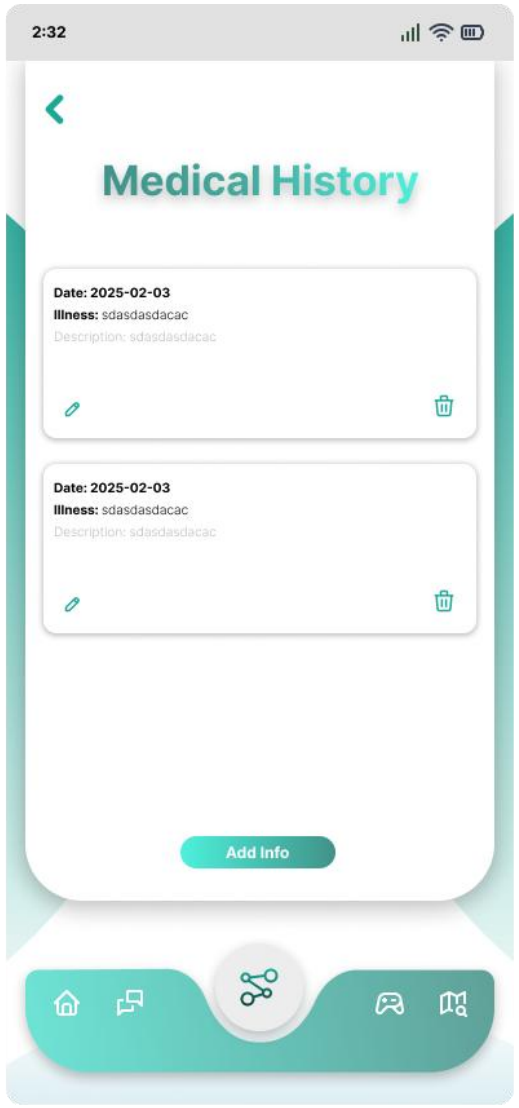


Personal Information Screen

Screen Component:

- Full name.
- Date.
- Age.
- Gender.
- Marital status.
- Residential address.
- Phone number.
- Occupation.

Light Mode Design Screens



Medical History Screen

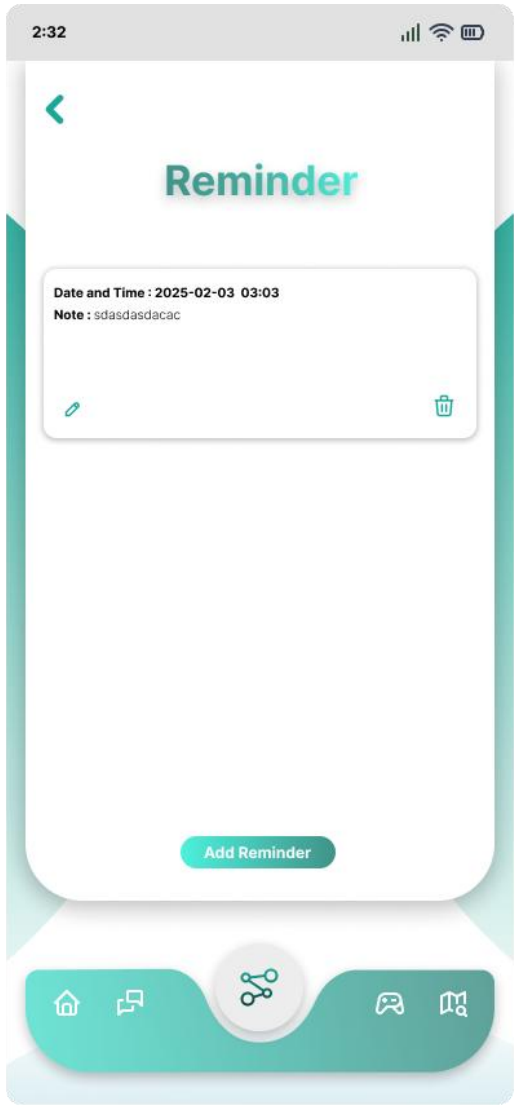
Each medical record is inside a rounded card, maintaining a consistent visual style. The details include:

- Date of record.
- Illness name (bolded for emphasis).
- Description (in a lighter font).

Each card has two action icons:

- Edit (pencil icon) – Allows modifying the record.
- Delete (trash icon) – Enables removal of the record.

Light Mode Design Screens



Reminder Screen

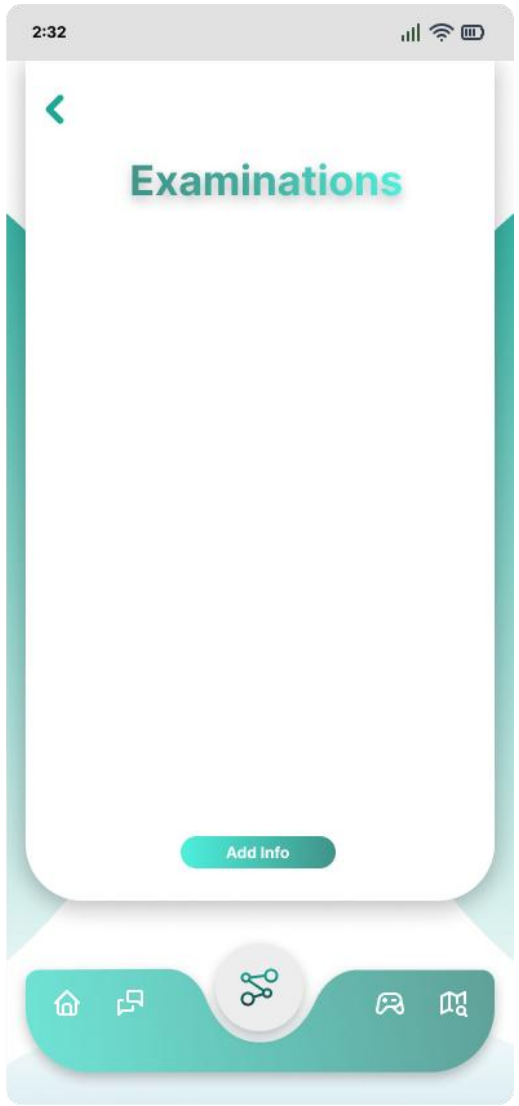
Each Reminder record is inside a rounded card, maintaining a consistent visual style. The details include:

- Date and Time: Displays when the reminder is set.
- Note: Provides a brief description of the reminder.

Each card has two action icons:

- Edit (pencil icon) – Allows modifying the record.
- Delete (trash icon) – Enables removal of the record.

Light Mode Design Screens



Examinations Screen

Each Examinations record is inside a rounded card, maintaining a consistent visual style. The details include:

- Date Field: Used to enter the date of the medical examination.
- Illness Field: Specifies the illness being examined.
- Description Field: Allows users to add additional notes about the examination.
- Upload Files to Google Drive Button: Enables users to upload medical reports or test results directly to Google Drive, enhancing cloud storage and accessibility of medical records.

Each card has two action icons:

- Edit (pencil icon) – Allows modifying the record.
- Delete (trash icon) – Enables removal of the record.

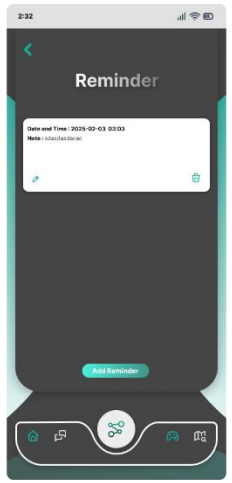
Light Mode Design Screens

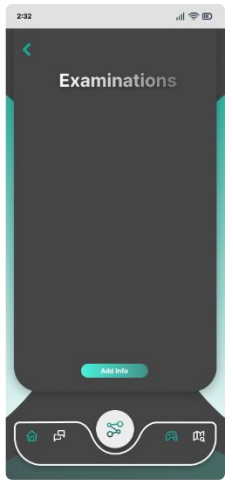
Setting Page:

Page Components:

Dark Mode Option.	Back Button (Top Left Corner).
Logout Option.	Team Member Page.

Dark Mode Design Screens



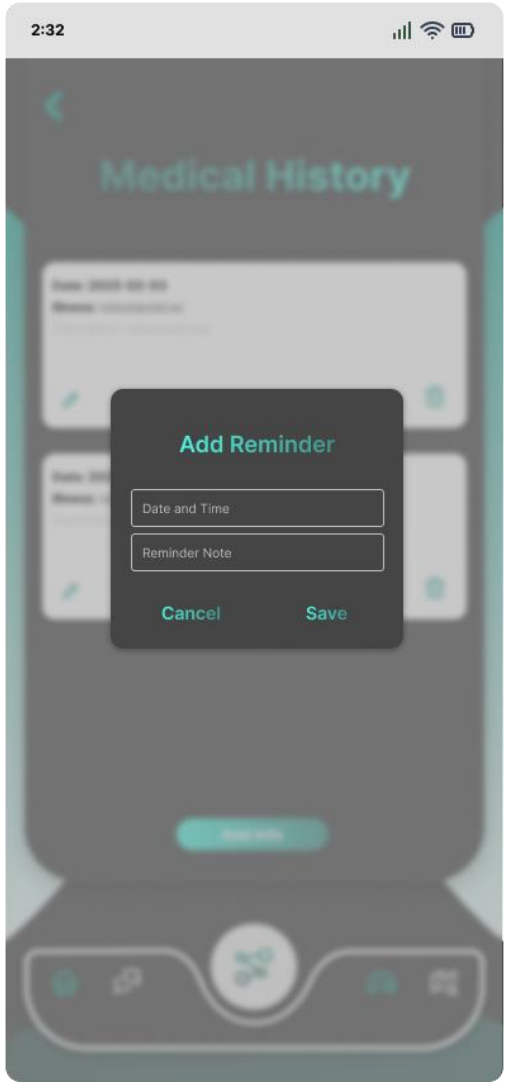


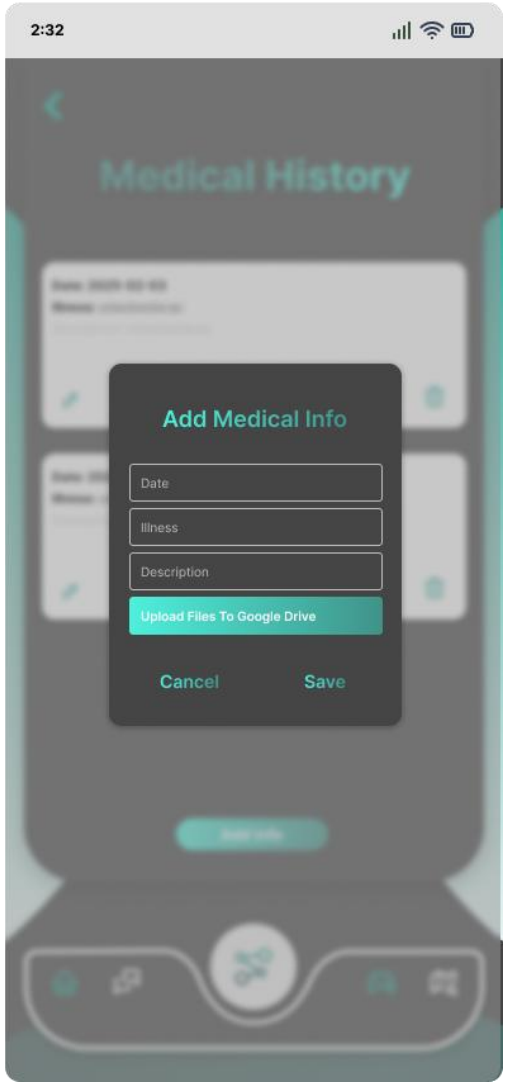
Dark Mode Design Screens

Dark Mode Design Screens

Dark Mode Design Screens

Dark Mode Design Screens





UI

UI

Code

Database

Code

Code

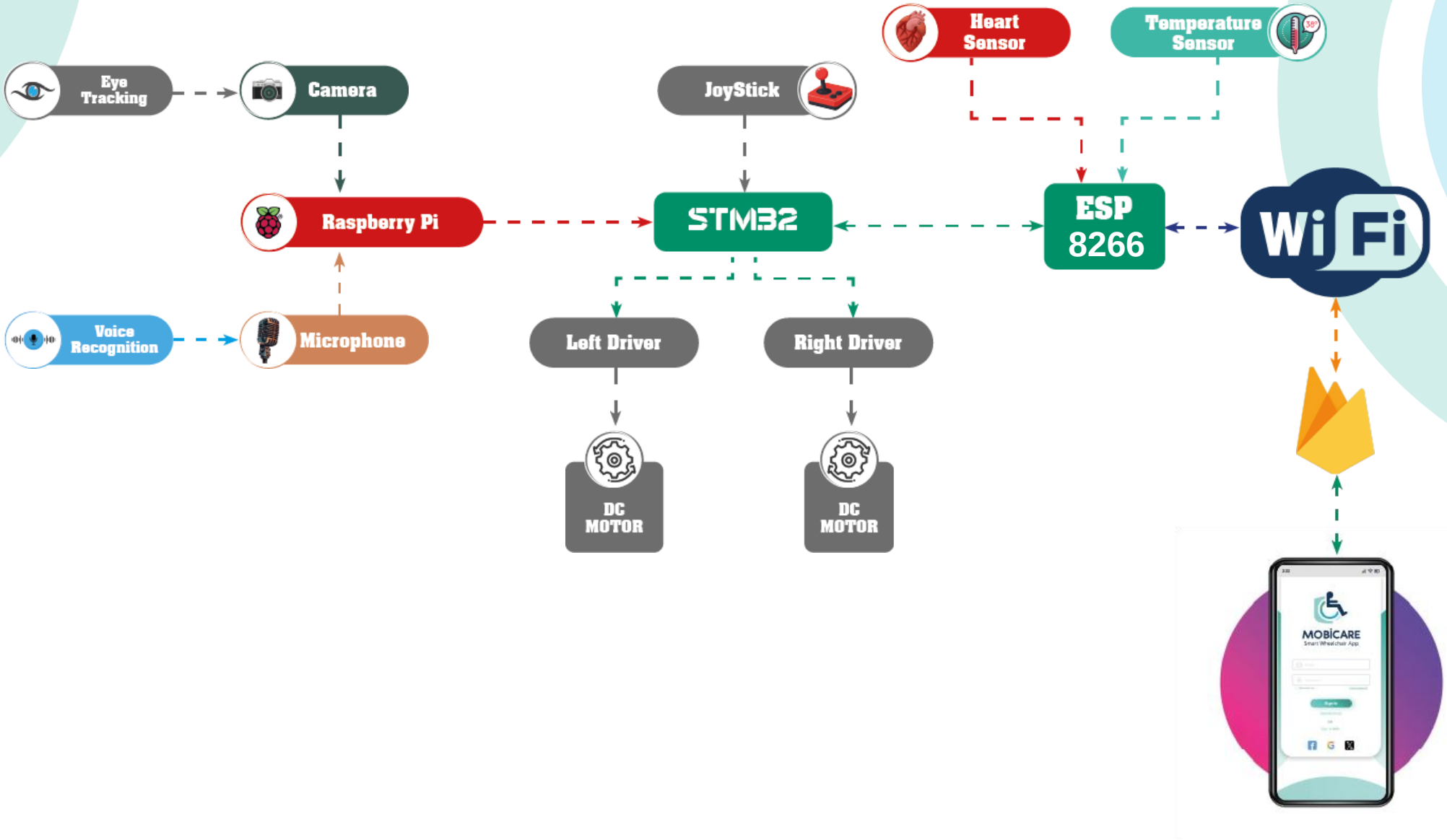
Flutter

Flutter

Why Flutter ?

Flutter

Block Diagram



Block Diagram

UI

UI

Code

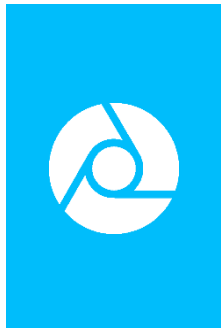
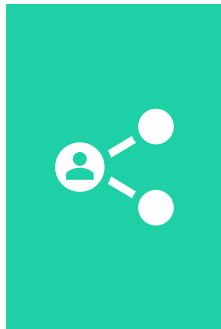
Database

Database

Database

Firebase



Plugins Used

SVG

FIREBASE AUTH

Handles user authentication with email/password, Google, and other sign-in providers.

URL LAUNCHER

Enables the app to open external URLs (like websites, email, or phone dialers) from within the app.

FIREBASE CORE

The core library that connects your Flutter app to Firebase services. Required for any Firebase functionality.

SVG

FIREBASE AUTH

Handles user authentication with email/password, Google, and other sign-in providers.

CLOUD FIRESTORE

Connects to Firebase Cloud Firestore, a real-time cloud-based NoSQL database.

FIREBASE CORE

The core library that connects your Flutter app to Firebase services. Required for any Firebase functionality.

SVG

FIREBASE AUTH

Handles user authentication with email/password, Google, and other sign-in providers.

CLOUD FIRESTORE

Connects to Firebase Cloud Firestore, a real-time cloud-based NoSQL database.

GOOGLE SIGN IN

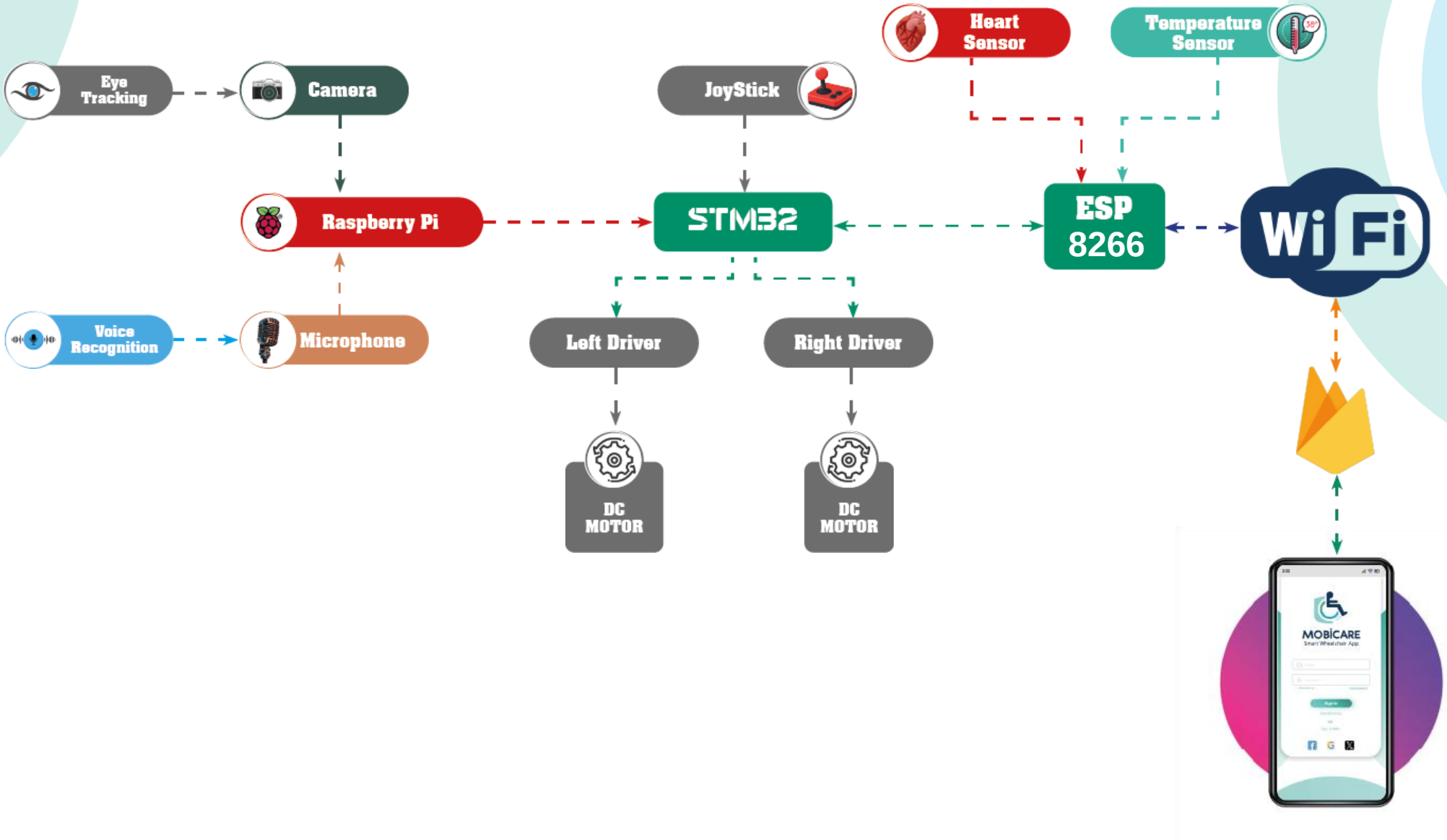
Simplifies integrating Google Sign-In for authenticating users.

FIREBASE CORE

The core library that connects your Flutter app to Firebase services. Required for any Firebase functionality.

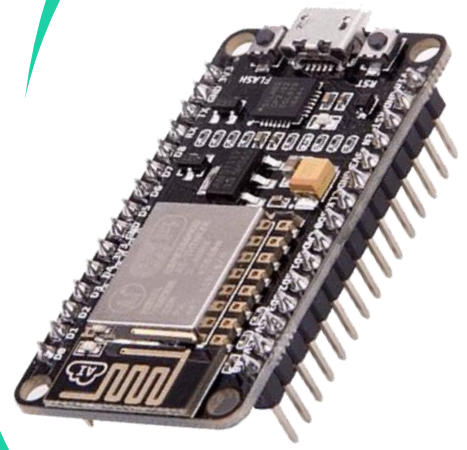
MOBILE
&
EMBEDDED LINK

Block Diagram



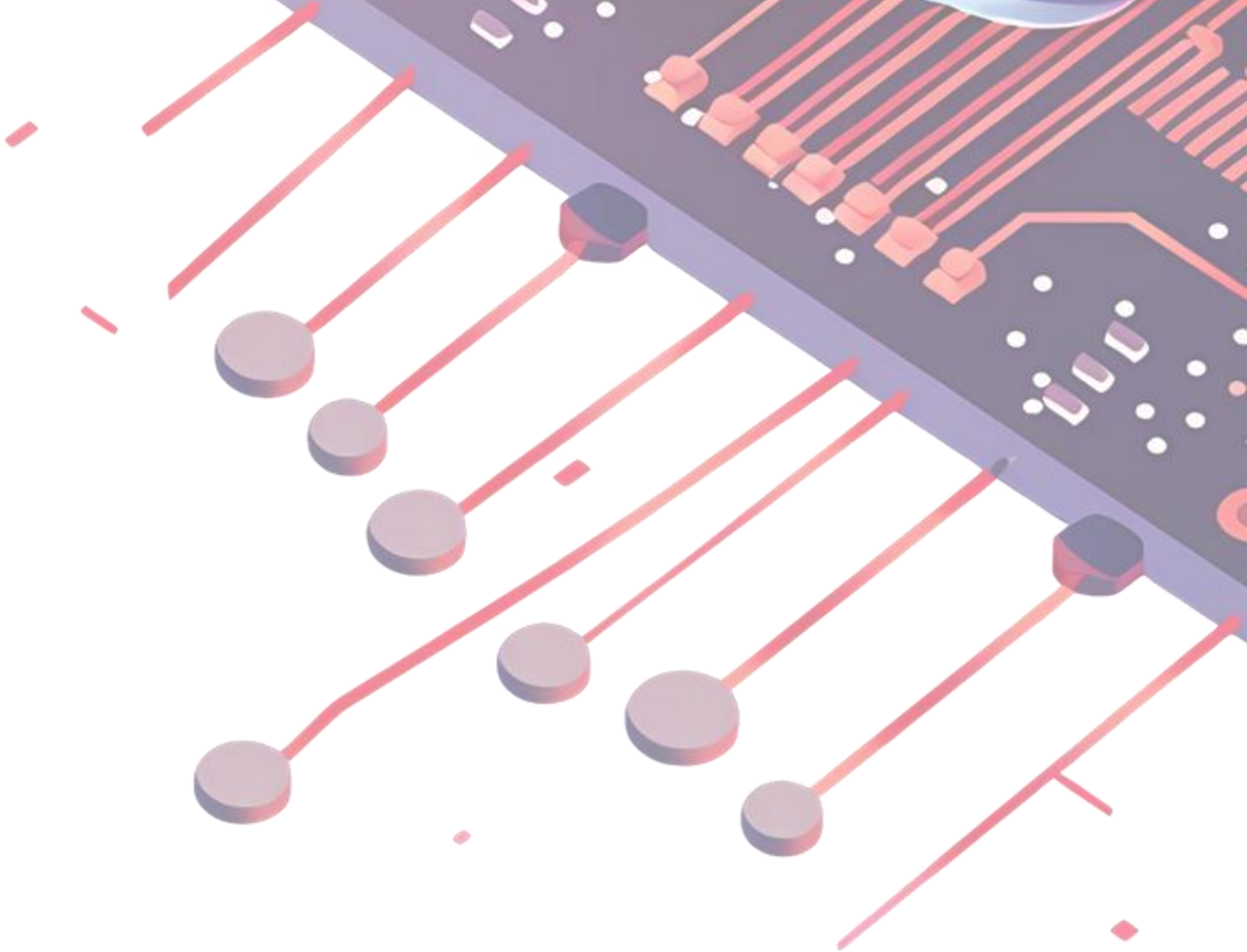
ESP 8266

The ESP8266 is a low-cost Wi-Fi microcontroller module developed by Espressif Systems, widely used in IoT applications. It can connect to Wi-Fi networks and run standalone programs, making it ideal for smart devices and remote sensing.

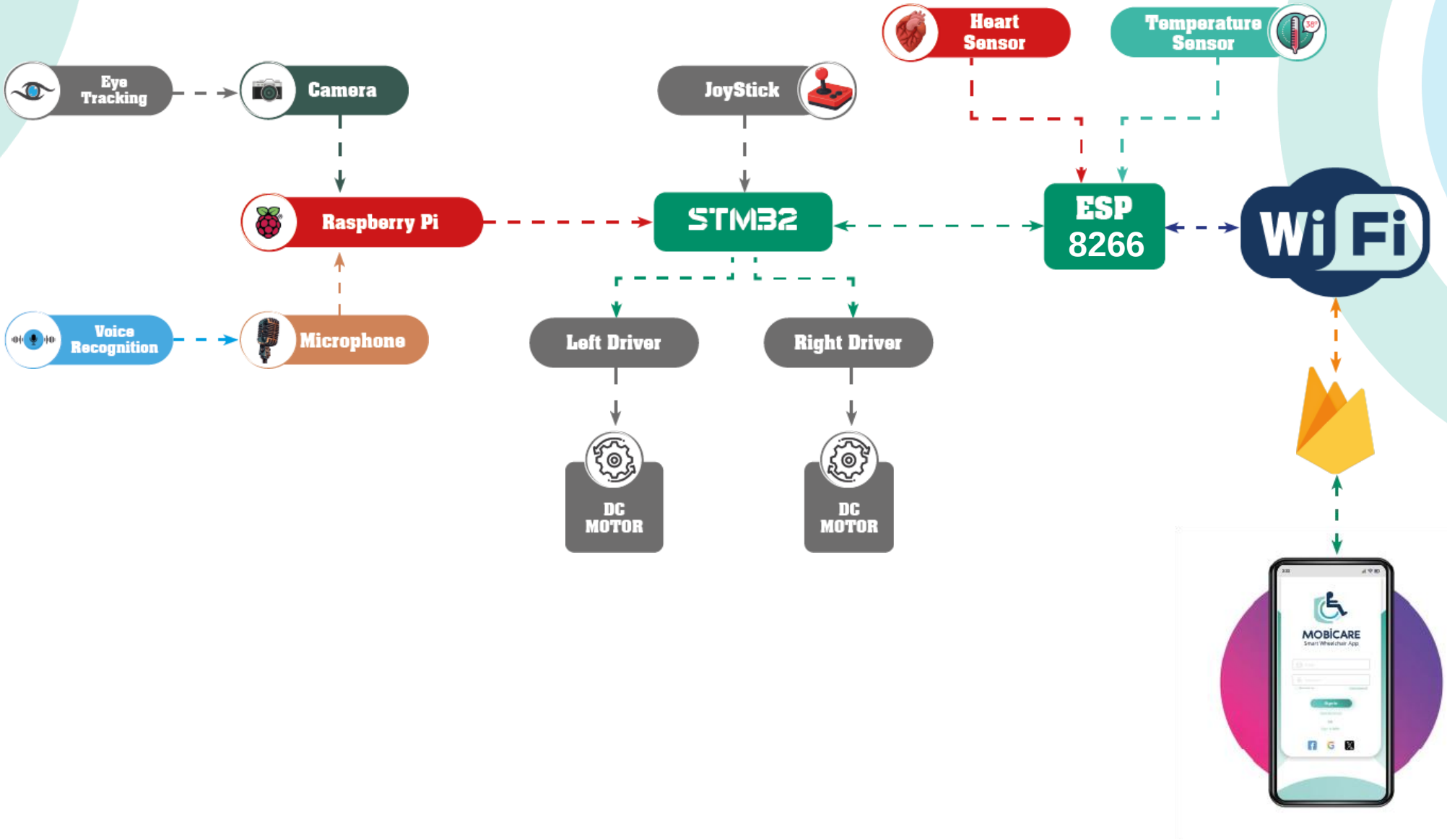


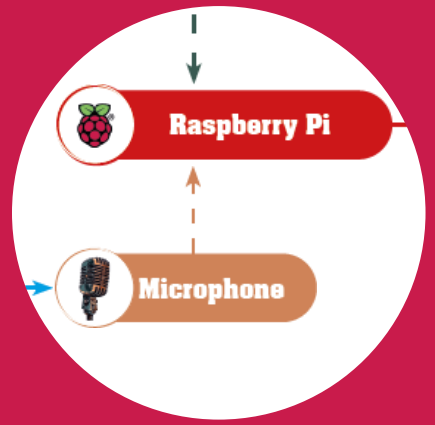
Key Features:

- **Multiple GPIO Pins:** Easy to interface with sensors, motors, and other hardware.
- **Arduino IDE Support:** Simple to program using Arduino libraries and tools**.**
- **Networking Protocols:** Supports TCP/IP, UDP, HTTP, MQTT for internet communication.



Block Diagram

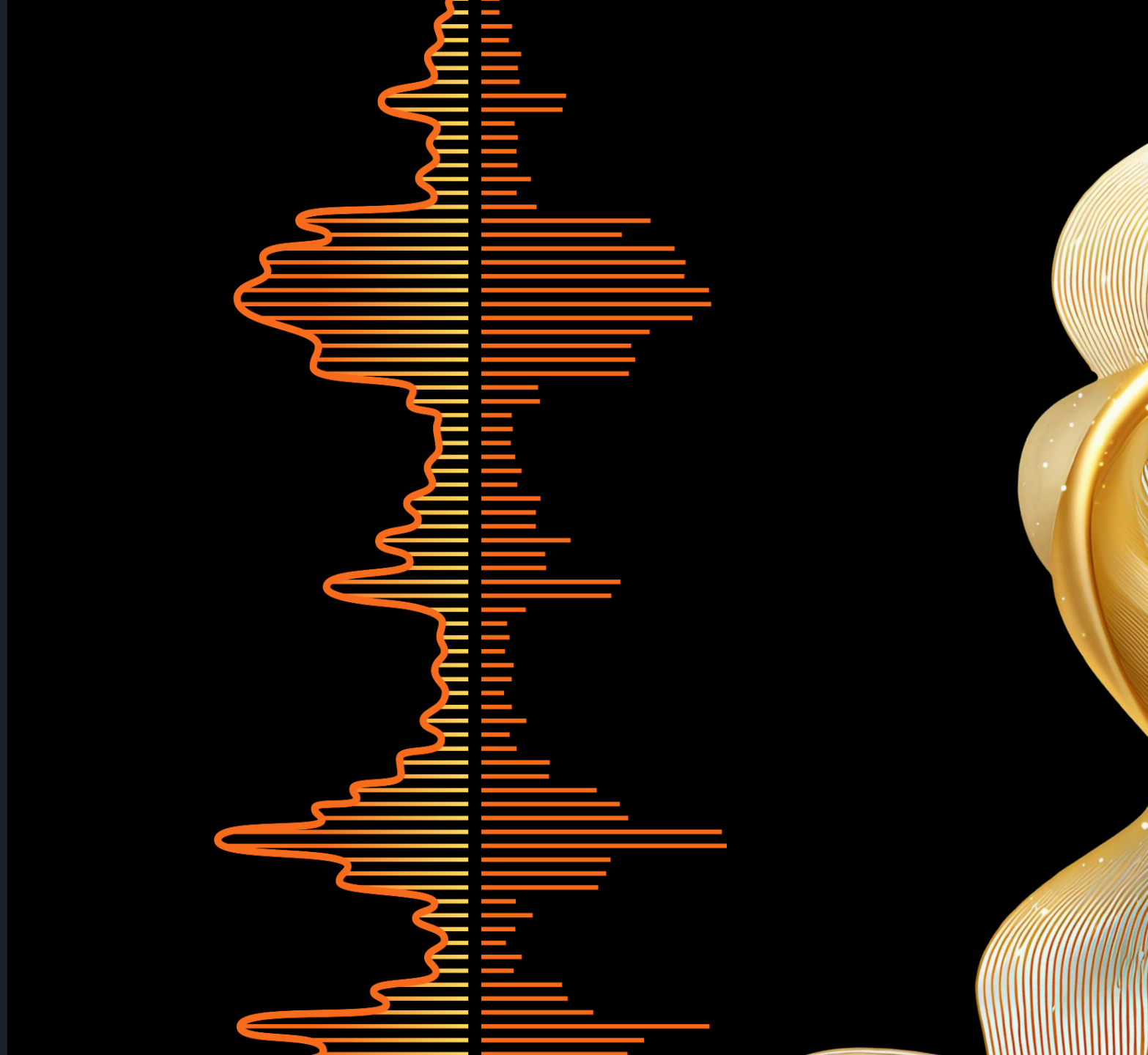




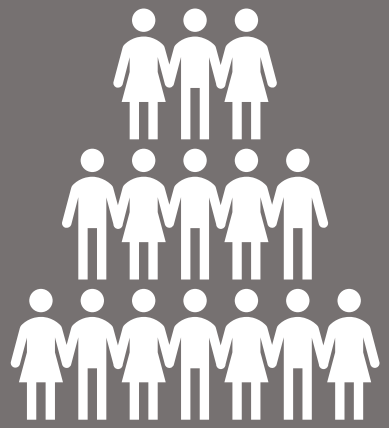
Speech Recognition Model

Speech Recognition Model
Using Deep Learning

- We record the user's voice using a microphone .
- The audio is preprocessed and passed into a deep neural network to classify the spoken word.
- Once the command is recognized with prediction is longer than threshold ,it is sent to the STM32 microcontroller.

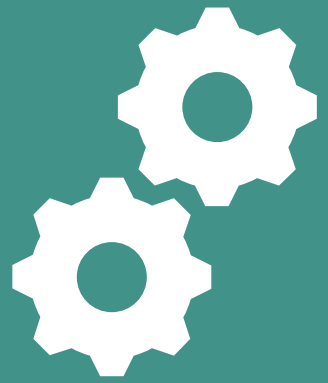


STEPS FOR BUILDING THE MODEL



1

Data Collection

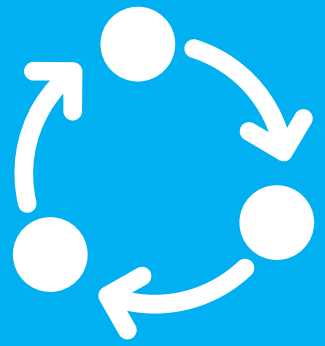


2

Data
Preprocessing

3

Building
Architecture model

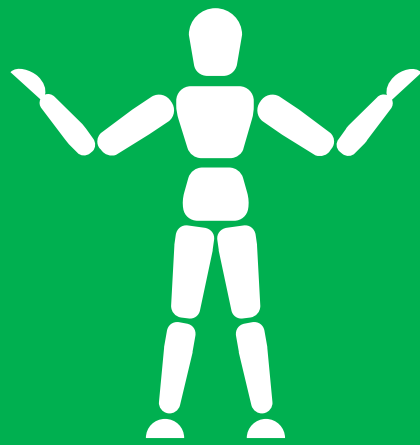


4

Compilation process

5

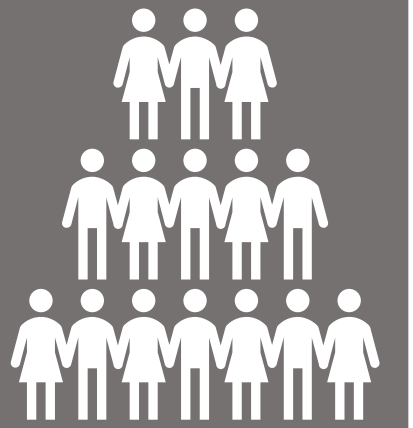
Data training



6

Evaluation model

STEPS FOR BUILDING THE MODEL



1

Data Collection

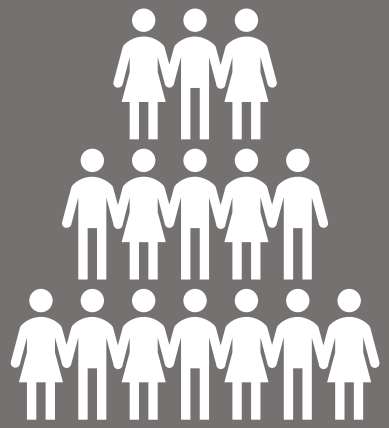
Collect a high-quality dataset of audio samples by:

1- sourcing publicly available speech datasets
2- creating your own recordings

The labels are "forward , backward , left , right , and stop ". The audio files are in WAV format.

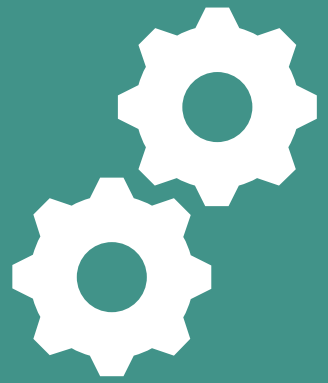
https://www.kaggle.com/competitions/tensorflow-speech-recognition-challenge/data

STEPS FOR BUILDING THE MODEL



1

Data Collection

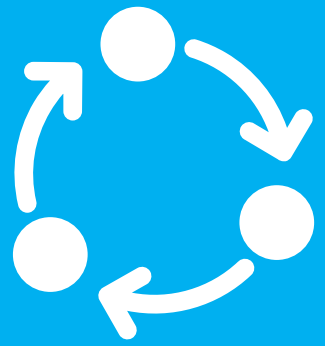


2

Data
Preprocessing

3

Building
Architecture model

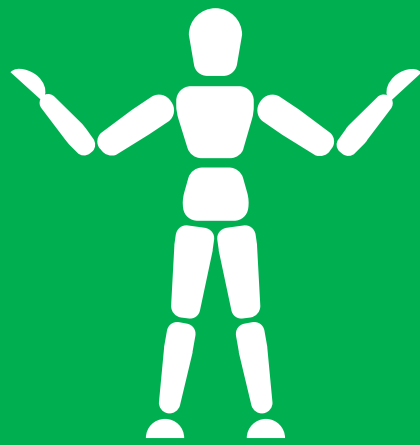


4

Compilation process

5

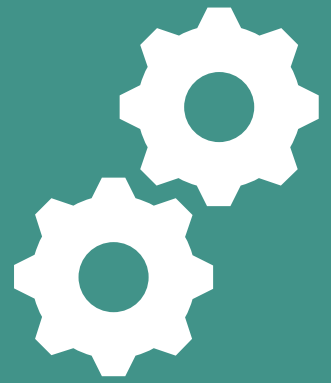
Data training



6

Evaluation model

STEPS FOR BUILDING THE MODEL

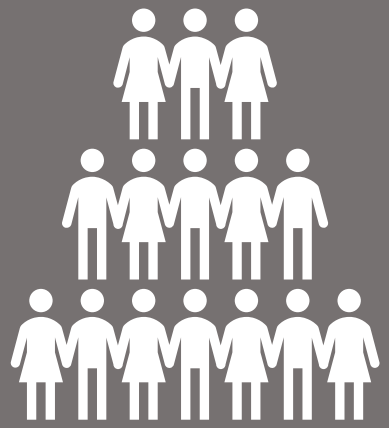


2

Data Preprocessing

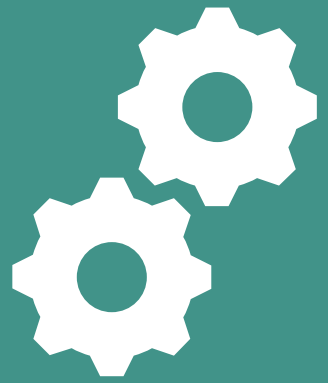
1) Sampling with Librosa Pad or truncate audio Exactly 1 second (8000 samples).
2) Data Augmentation: (Add Noise - Time Shift - Time Stretch - Change Volume - Pitch Shift).
3) Mel-Spectrogram Extraction:
   (64-band mel-spectrogram- log-scaled dB spectrogram)
4) Normalization:
   (zero mean - unit variance - accelerates training).
5) Data Preparation:
   Features stored
   Labels stored→ one-hot encoded (5 classes)

STEPS FOR BUILDING THE MODEL



1

Data Collection

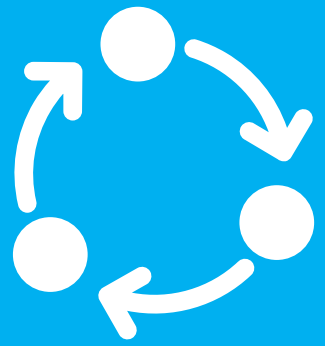


2

Data
Preprocessing

3

Building
Architecture model

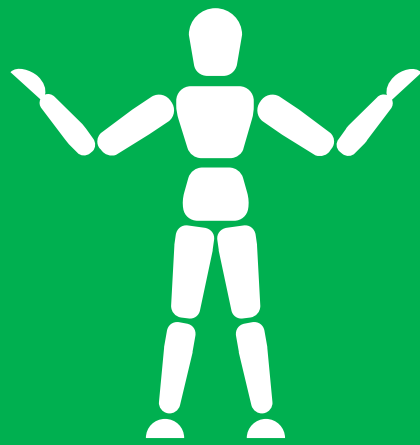


4

Compilation process

5

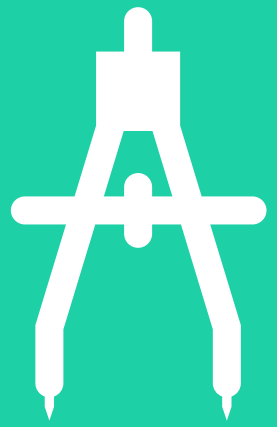
Data training



6

Evaluation model

STEPS FOR BUILDING THE MODEL



3

Building
Architecture
model

1) Input Layer.

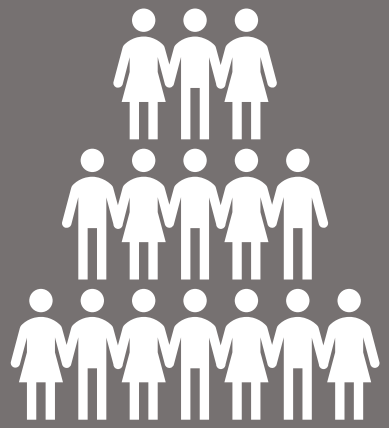
2) 1d Convolutions.

3) Batch Normalization &&Pooling && Dropout Layers.

4) LSTM Layer.

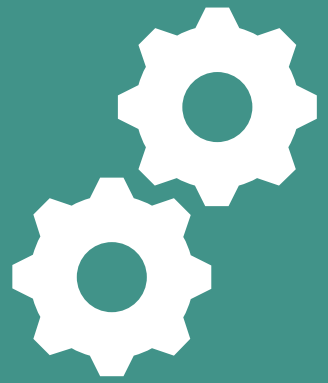
5) Dense Layers.

STEPS FOR BUILDING THE MODEL



1

Data Collection

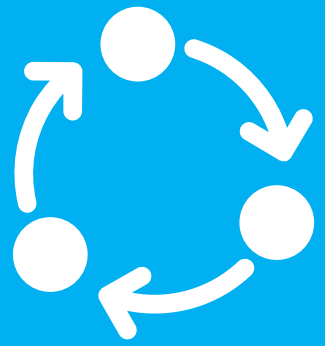


2

Data
Preprocessing

3

Building
Architecture model

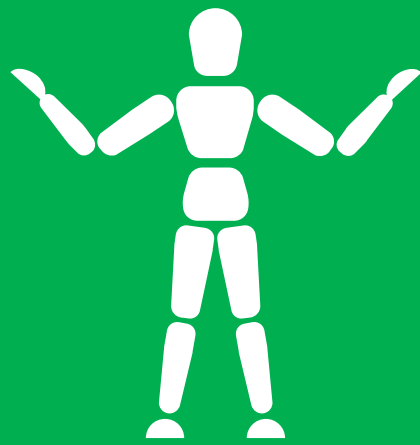


4

Compilation process

5

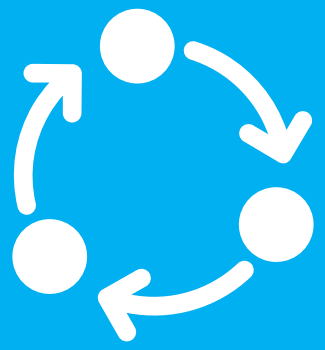
Data training



6

Evaluation model

STEPS FOR BUILDING THE MODEL



4

Compilation process

1) Loss Function:

Choose an appropriate loss function, such as categorical cross-entropy, to measure the model's performance during training.

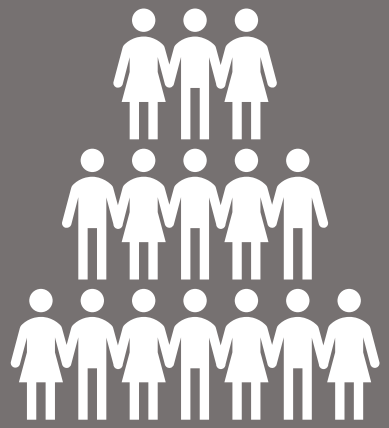
2) Optimizer:

Select an optimizer, like Adam or RMSprop, to update the model's weights and minimize the loss function.

3) Metrics:

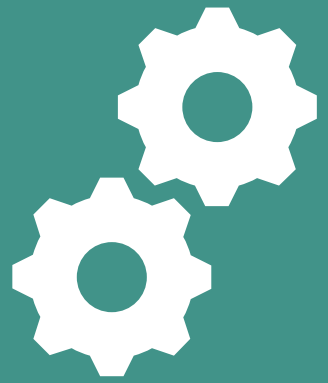
Define relevant metrics, such as accuracy, to track the model's performance on the validation and test sets.

STEPS FOR BUILDING THE MODEL



1

Data Collection

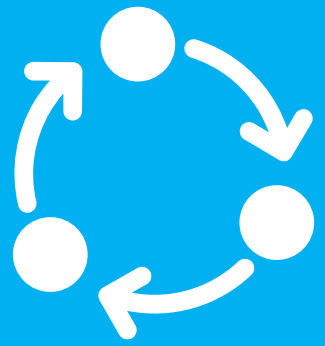


2

Data
Preprocessing

3

Building
Architecture model

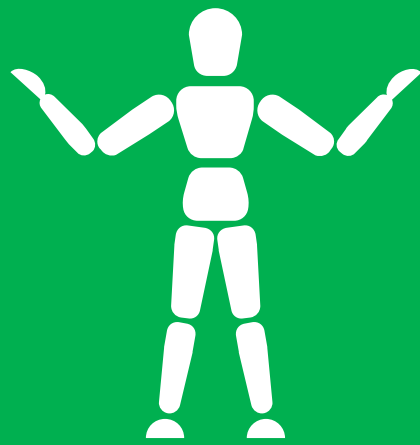


4

Compilation process

5

Data training



6

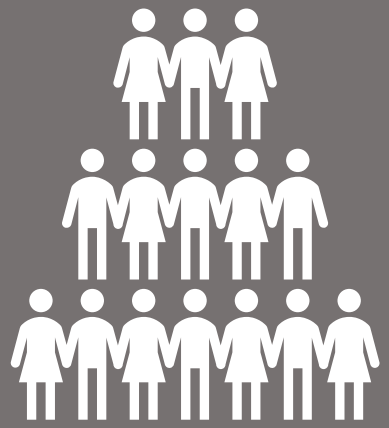
Evaluation model

STEPS FOR BUILDING THE MODEL

5

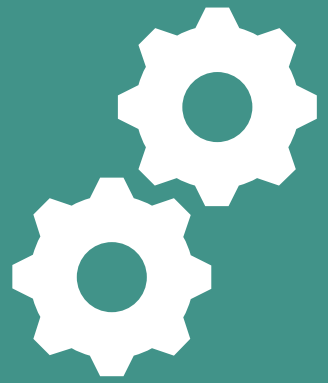
Data training

STEPS FOR BUILDING THE MODEL



1

Data Collection

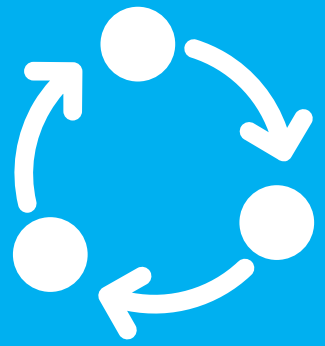


2

Data
Preprocessing

3

Building
Architecture model

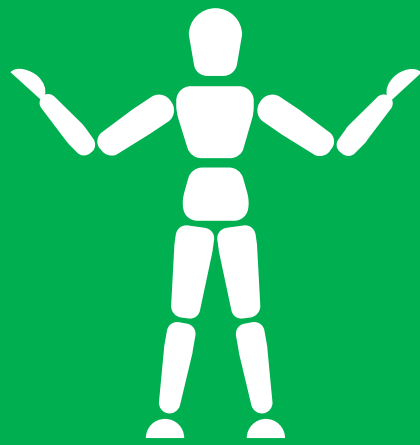


4

Compilation process

5

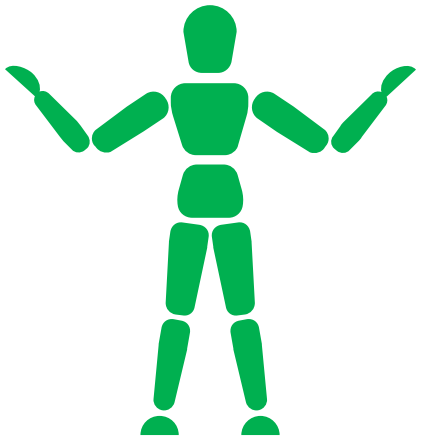
Data training



6

Evaluation model

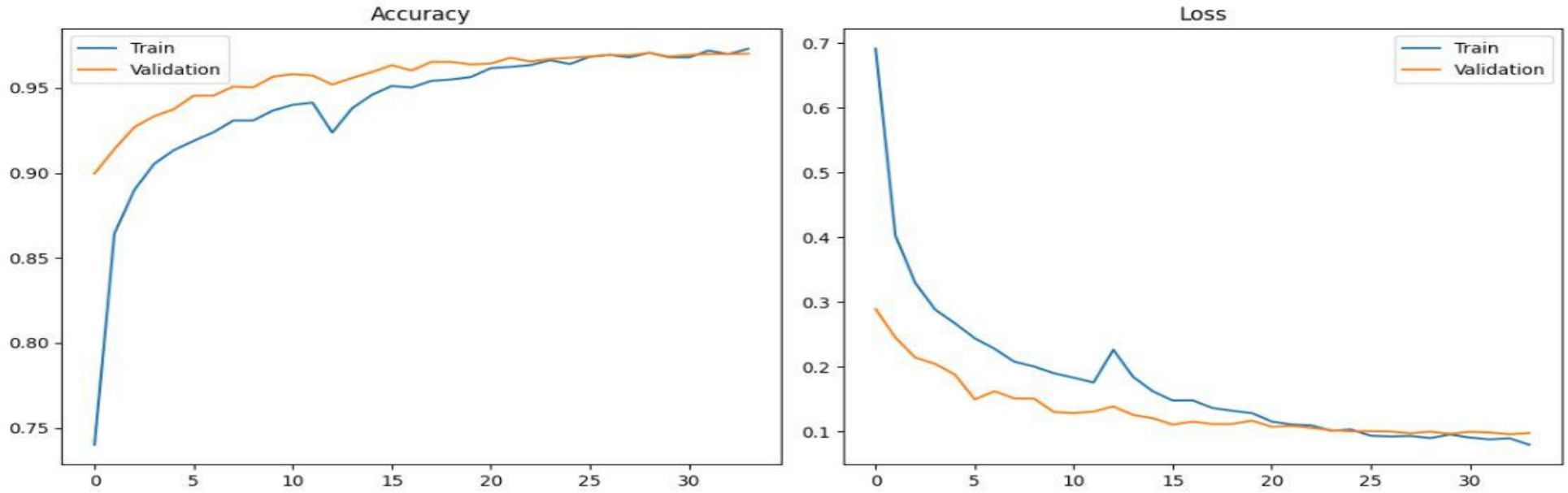
STEPS FOR BUILDING THE MODEL



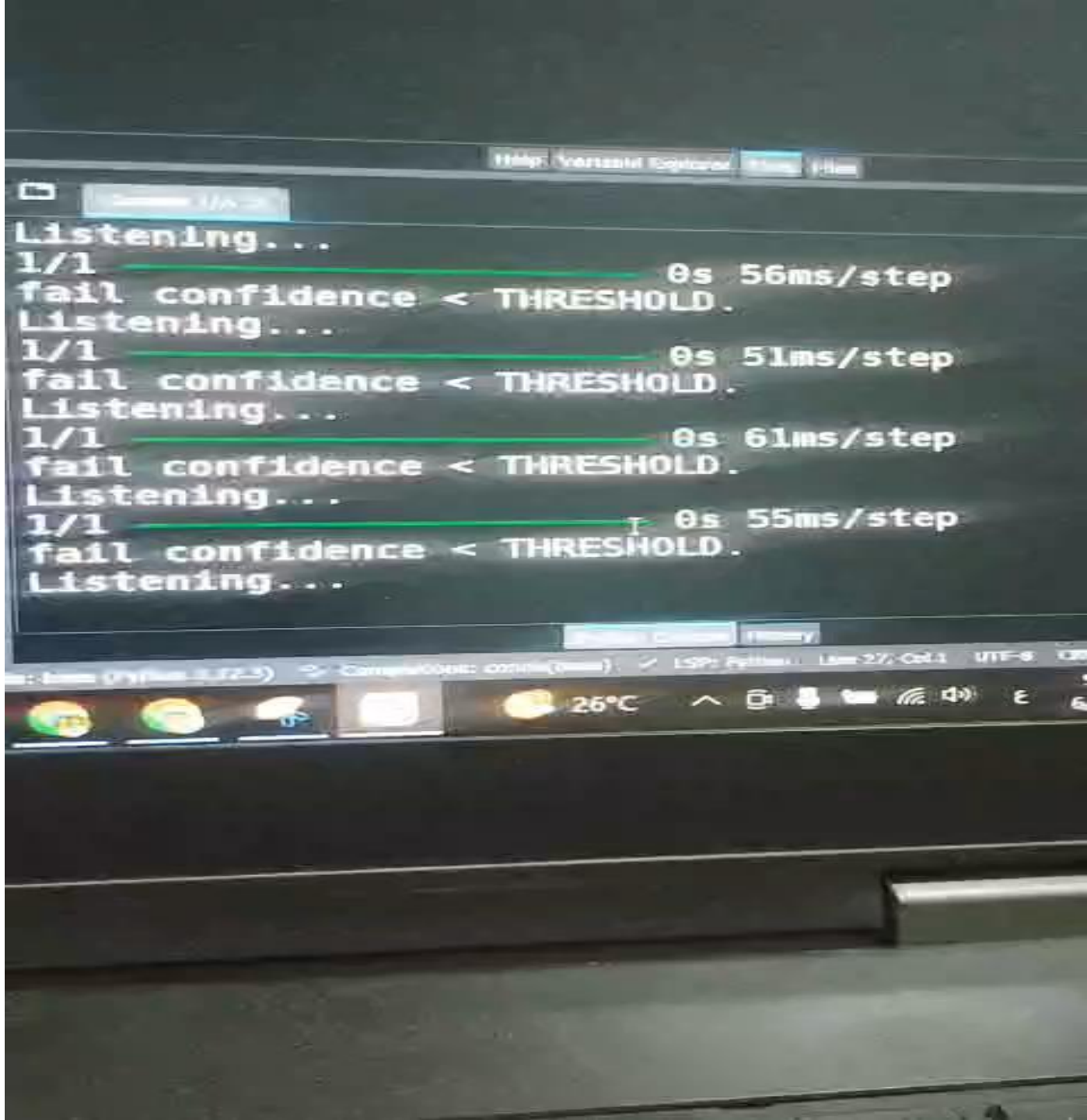
6

Evaluation model

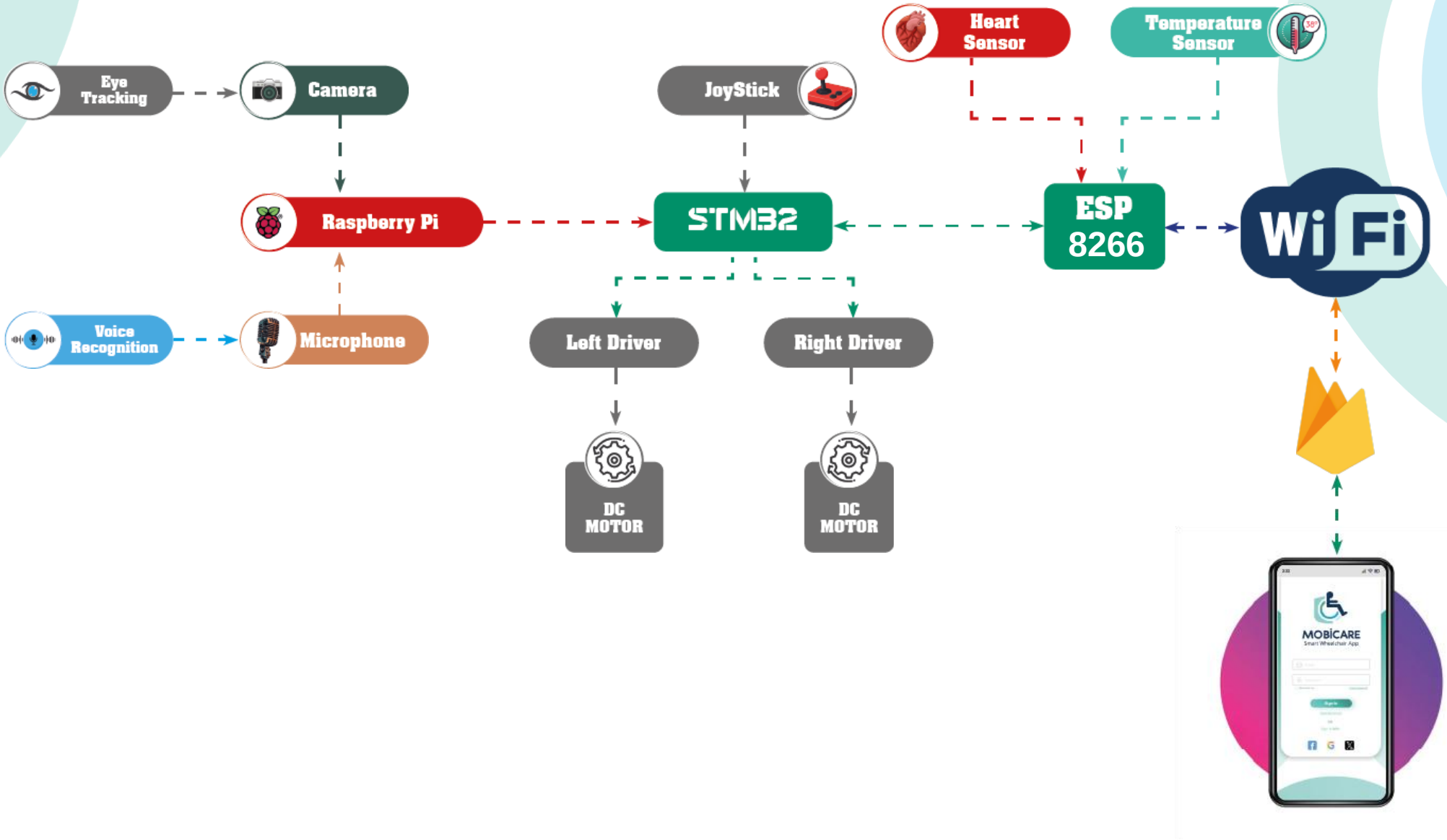
```
Epoch 30/50
517/517 ━━━━━━━━━━━━━━━━━━━━ 6s 11ms/step - accuracy: 0.9698 - loss: 0.0920 - val_accuracy: 0.9685 - val_loss: 0.0963 - learning_rate: 1.2500e-04
Epoch 31/50
517/517 ━━━━━━━━━━━━━━━━━━━━ 12s 14ms/step - accuracy: 0.9657 - loss: 0.0962 - val_accuracy: 0.9695 - val_loss: 0.0995 - learning_rate: 1.2500e-04
Epoch 32/50
517/517 ━━━━━━━━━━━━━━━━━━━━ 9s 13ms/step - accuracy: 0.9713 - loss: 0.0865 - val_accuracy: 0.9700 - val_loss: 0.0985 - learning_rate: 1.2500e-04
Epoch 33/50
517/517 ━━━━━━━━━━━━━━━━━━━━ 6s 12ms/step - accuracy: 0.9714 - loss: 0.0878 - val_accuracy: 0.9700 - val_loss: 0.0959 - learning_rate: 6.2500e-05
Epoch 34/50
517/517 ━━━━━━━━━━━━━━━━━━━━ 11s 14ms/step - accuracy: 0.9730 - loss: 0.0823 - val_accuracy: 0.9702 - val_loss: 0.0977 - learning_rate: 6.2500e-05
130/130 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9713 - loss: 0.0999

✅ Final Accuracy: 97.07% | Loss: 0.0997
```

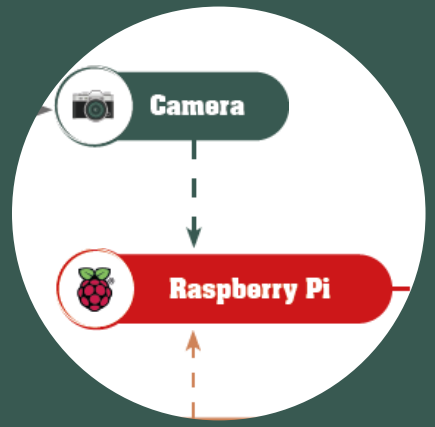


TESTING


Listening...
1/1 0s 56ms/step
fail confidence < THRESHOLD.
Listening...
1/1 0s 51ms/step
fail confidence < THRESHOLD.
Listening...
1/1 0s 61ms/step
fail confidence < THRESHOLD.
Listening...
1/1 0s 55ms/step
fail confidence < THRESHOLD.
Listening...
26°C

Block Diagram





FACE AND EYE DIRECTION TRACKING MODEL

Face Direction Tracking

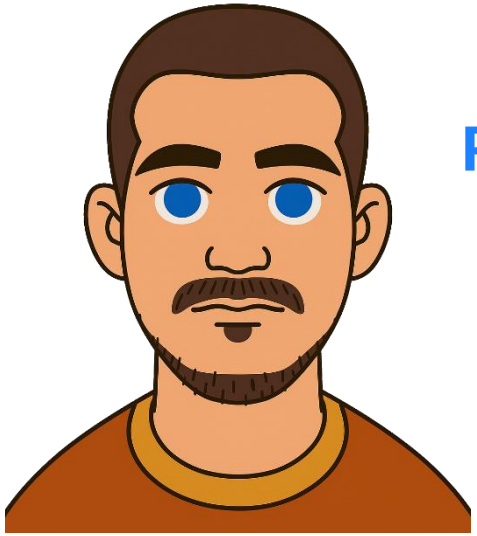
Methodology:

- Determines whether the user is looking forward, left, or right.
- Uses MediaPipe Face Mesh to detect landmarks (nose, eyes).
- Compares the nose’s horizontal position relative to the eyes to assign a direction.

Face Direction Tracking

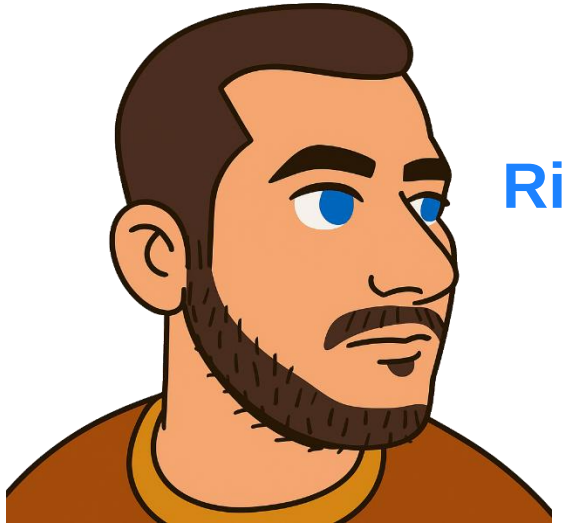
LOGIC

- Nose centered → Forward
- Nose shifted left → Left
- Nose shifted right → Right



Forward

Left

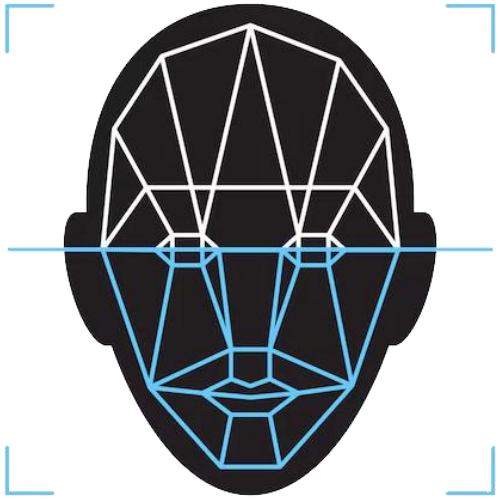


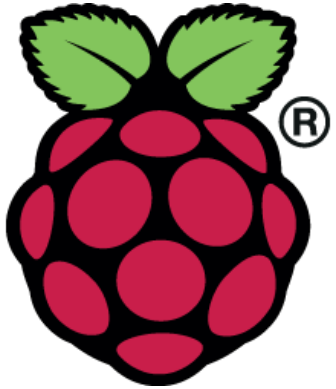
Right

Face Direction Tracking

Technologies Used:

- MediaPipe, OpenCV, Raspberry Pi Camera, Serial Communication





Eye Tracking for Gaze Direction

Methodology:

- Detects iris position inside the eye using image processing.
- **Steps include:**
  - Capture real-time video
  - Extract eye region
  - Apply grayscale, blur, and threshold
  - Use contour detection to find iris center

Face Direction Tracking

LOGIC

- Iris left → Left
- Iris right → Right
- Iris up → Forward
- Iris down → Down
- Iris centered → Stop

MOBICARE

Smart Wheelchair App

Thanks