Smart wheelchair

1. Team members

Alyaa Ali Abo El Yazeed	Mobile application
Ali Awaad Ahmed	Mobile application
Aliaa Mahmoud Abd El hamid	Arduino code
Ezz Eldien Kamal Kamel	Arduino code
Asjad Mustafa Hussein	Circuit connections
Azzam Khaled Mohammed	Circuit connections
Alyaa Mahmoud El Sayed	Wheelchair construction
Ola Obour Salah	Wheelchair construction

Under supervision of Eng. Ahmed El Baz

2. Abstract

Smart wheelchair is a wheelchair that can be controlled by a mobile application; the mobile application allows two control systems, the first is controlling the wheelchair motion by a joystick to make the chair move in any desired direction. The second control system is clicking the microphone button in the application and saying the desired motion. This mobile application is made by Flutter (a framework made by Google that uses Dart programming language). The mobile application connects to the arduino by bluetooth, so that the application sends any implementation to the arduino ,and this is done by a bluetooth module.

3. Literature review

Wheelchair prototype was developed by using a commercially available manual wheelchair to assist people with both upper and lower limb disabilities. The disabled had to depend on others to move from one place to another, this also consumed large physical effort from families and relatives of the patients. Many attempts were made to solve these problems and facilitate the control of wheelchairs, one of these attempts is a line follower wheelchair that helps the visually impaired people to get to specific places, another one is controlling the wheelchair by a remote control and many other projects like that. We are

introducing a smart wheelchair that can be voice controlled by a mobile application (which doesn't have to be connected to the internet). Arduino microcontroller processes the voice command with the help of a package that uses AI to recognize speeches and controls the motor movement of the wheelchair. There's also an optional joystick in the application that allows the user to move in the four main directions. This minimizes the user's personal effort and force required to move the wheelchair. Furthermore, it allows physically handicapped people to go from one location to another easily.

4. Market research and customer feedback

Multiple questions were asked to the customers interested in the market of the wheelchairs genuinely. Some customers were asked about their opinion whether the smart wheelchair is better or the manual wheelchair and the answers were as follows:

- Nearly 96% voted for the smart wheelchair.
- Only 4% voted for the manual wheelchair.

As you see the majority voted for the smart wheelchair which is expected to make high market profits by the next few years and by these percentages, it is also expected for the manual wheelchair to disappear by the next few years. In the same survey, the customers were asked if the smart wheelchair is easy to use or not. And the results were as follows:

- 95.7% voted for yes.
- The rest (4.3%) voted for no.

From these results we see that the majority of customers (nearly all of them) see that it is easy to deal with the smart wheelchair which makes it an attractive product for comfortability seekers. When the customers were asked to say their opinions, nearly 95% of them said it is a good idea which offers a wide range of comfortability and physical relaxation for the disabled people and that it will be very useful when it widely spreads in the market. Also, the customers were asked if they have any suggestions to improve this project's idea. This question received a few positive responses but the majority said they have no suggestions, the responses were :

- To provide a sensor that produces high noise and stops immediately if the patient is subjected to the danger of falling off the chair.
- To make the project with the highest possible quality to provide the best service with a reasonable price range.
- To control the wheelchair by nerve signals (commands) for the fully disabled people.
- The smart wheelchair is better controlled by a remote control because it is possible for the mobile app to face some issues from time to time.
  And that was the most highlighted customers' suggestions that we will take into consideration. In the same form, The customers were asked to put a price range

for our project based on the data we informed them. And the results were as follows:

-13% from 500 to 1000 EGP.
-41.3% from 1000 to 1500 EGP.
-41.3% from 1500 to 2000 EGP.
- 4.4% for 2000 or above EGP.

The most answers were the intermediate prices, which shows that the wheelchair will not be either expensive or very cheap so that it fits for all society classes and also be profitable. Finally, the customers were asked to rate the project at a scale of 1 to 10 and the results were as follows:

-89.1% voted for 10/10.
-10.9% voted for above 5/10.

That indicates a high range of sales when the product gets released. Here is the link of the survey: Smart wheelchair survey

5. Used components

a. **Arduino nano**
   Input voltage 5V - 9V
   Arduino nano datasheet
b. **L293D Motor Driver**
   Input voltage 4.5V - 36V
   L293D Motor Driver datasheet
c. **HC-05 Bluetooth Module**
   Input voltage (3.3-6)v
   HC-05 Bluetooth Module datasheet
d. **9V battery**
e. **Battery holder**
f. **2 DC geared motors**
   Input voltage 3V - 12V
   Power consumption:190 mA (max. 250 mA)
   Gearbox: 48:1
   Speed: 90 ± 10 rpm
   Torque: (0.078 N.m)
   DC geared motor datasheet
g. **2 DC motor wheels**
h. **Jumper wires**
i. **Breadboard**
j. **Caster wheel**
k. **Slotex wood**

6. Background information

a. PWM (Pulse Width Modulation)

The speed of a DC motor can be controlled by changing its input voltage. A common technique to do this is to use PWM (Pulse Width Modulation). PWM is a technique where the average value of the input voltage is adjusted by sending a series of ON-OFF pulses. The average voltage is proportional to the width of the pulses. The higher the width of pulses, the higher the average voltage applied to the DC motor (resulting in higher speed) and the shorter the width of the pulses, the lower the average voltage applied to the DC motor (resulting in lower speed).The image below shows PWM technique with different cycles and average voltages. PWM datasheet

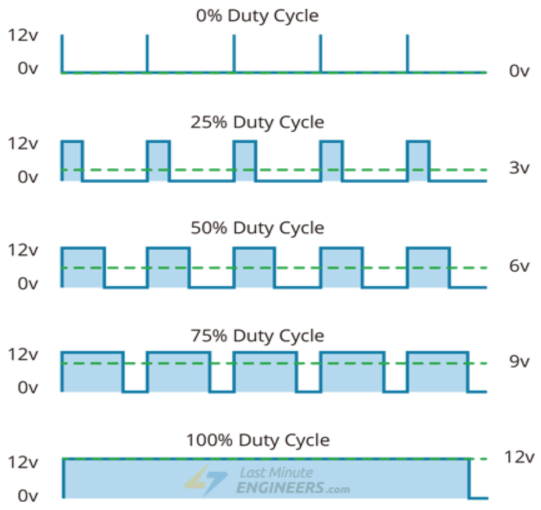

Figure 1 : PWM duty cycles

b. H-Bridge

The spinning direction of a DC motor can be controlled by changing the polarity of its input voltage. A common technique for doing this is to use an H-bridge. An H-bridge circuit consists of four switches with the motor in the center forming an H-like arrangement. Closing two specific switches at a time reverses the polarity of the voltage applied to the motor. This causes a change in the spinning direction of the motor. The H-Bridge also prevents the damage of the circuit when the motors are de-energized. H-Bridge datasheet

The L293D is a dual-channel H-Bridge motor driver capable of driving a pair of DC motors. This means it can drive up to two motors individually which makes it ideal for building a two-wheeled robotic platform. The L293D IC has a supply range of 4.5V to 36V and is capable of 1.2A peak output current per channel, so it works very well with motors. The IC also includes built-in kick-back diodes to prevent damage when the motor is de-energized. The image below shows L293D pinout:

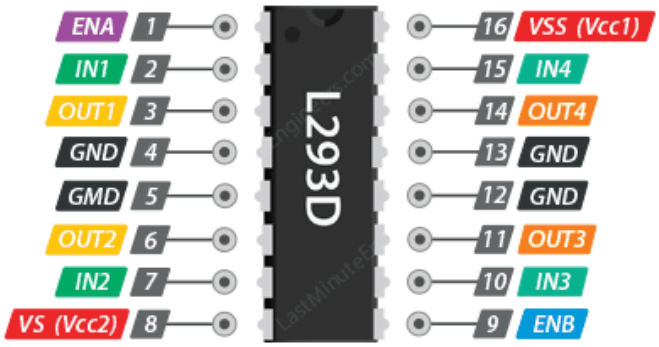

Figure 2 : L293D pinout

7. Design

The figure below shows the block diagram of the project,

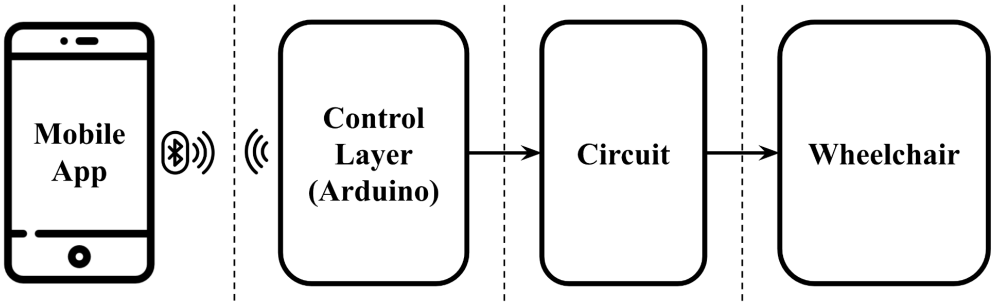

Figure 3 : Project block diagram

The project has four main milestones:

a. **The mobile application**

The mobile application is made by Flutter. Flutter is an open source framework by Google which uses Dart language for building multi-platform applications from a single codebase. The mobile application includes three pages, the first page is to connect the application with the circuit via bluetooth. This is achieved by making an interface for serial port protocol (HC-05 module), it can monitor the status of connection, discover devices, list bonded devices and pair new ones. This is done through a package called "flutter_bluetooth_serail".

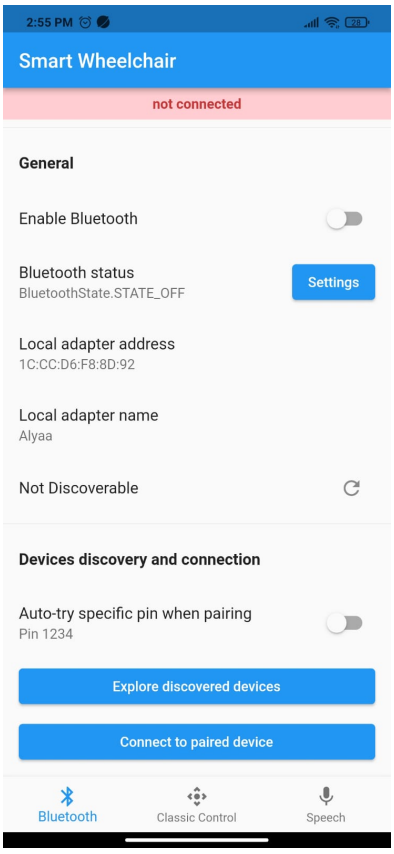

Figure 4 : Bluetooth page

The second page is controlling the motors by a joystick, as when the user moves the joystick to any of the four directions (forward, backwards, right, left), the wheelchair will go in that direction and once the user releases the joystick, the wheelchair stops.

Figure 5 : Joystick page

The third page is controlling the motors by speech, the user will click the mic button, which is in the middle of the page, and say any of the four directions so the wheelchair will move in the desired direction. This is achieved by a package called “speech_to_text” that is supported by artificial intelligence which recognizes the speech with the help of a library that exposes device specific speech recognition capability and converts speech into text, so that we can take the text and process it to make a specific implementation when listening to a specific word or sentence. The user also

can say “stop” to make the wheelchair stop or click the emergency stop button which is below the mic button.

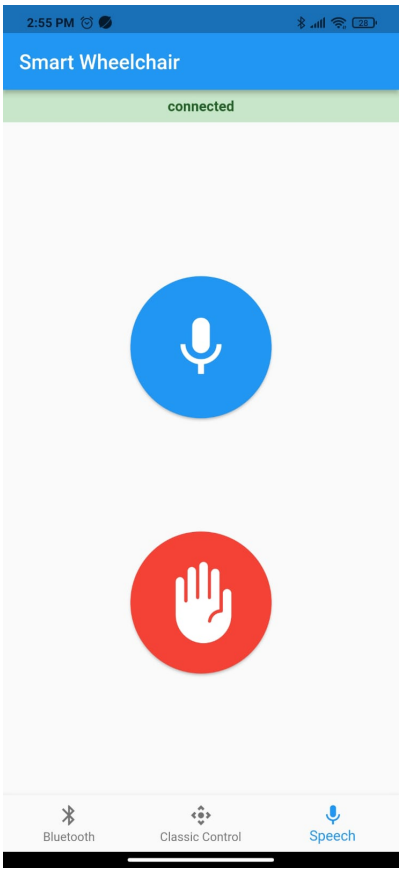

Figure 6 : Speech page

The mode of motion is sent to the arduino through bluetooth, for forward it sends “1”, for backwards it sends “2”, for right it sends “3”, for left it sends “4” and for stopping the motor it sends “0”.

b. Arduino

The arduino has 3 outputs for each motor, analog one (by using PWM) for the speed control (enA, enB), and the others are digital outputs for the spinning direction (inA1, inA2, inB1, inB2). The arduino receives the mode of motion from the mobile application through the bluetooth module, the following table shows how the arduino code handles each mode.

Mode	inA1	inA2	inB1	inB2
Forward (1)	High	Low	High	Low
Backwards (2)	Low	High	Low	High
Right (3)	Low	High	High	Low
Left (4)	High	Low	Low	High
Stop (0)	Low	Low	Low	Low

c. Connecting the circuit

The figure below shows the project circuit:

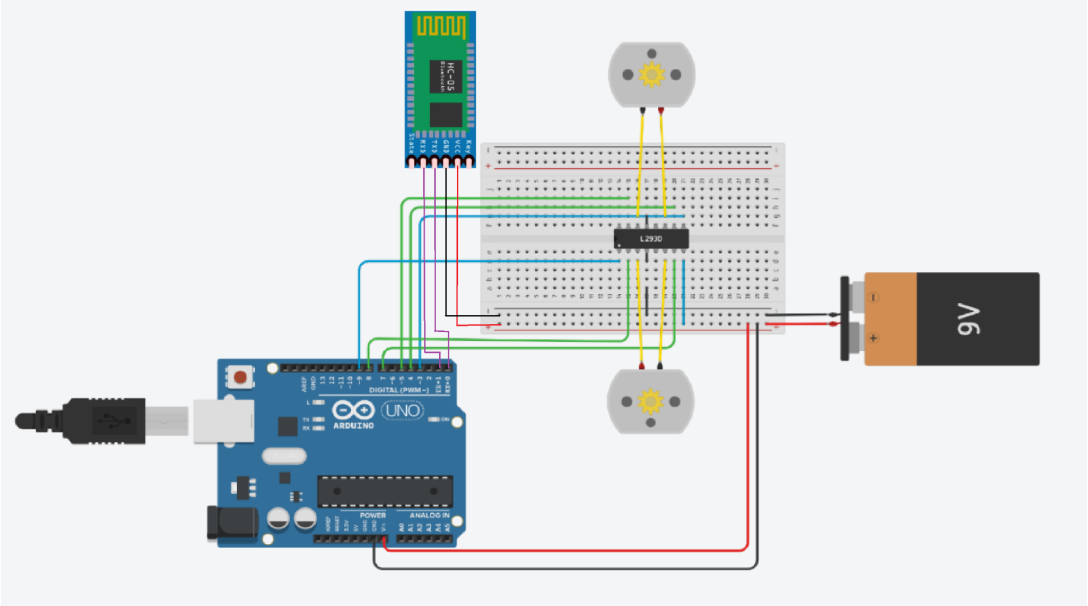

Figure 7 : Project's circuit

- Arduino connections

  The arduino is connected to a 9V battery through VIN pin.

  Arduino has 4 digital outputs, 2 analog outputs and 1 input.

  Pin 0 (RXD) (input) is connected to TXD in HC-05.

  Pins 3 and 9 (analog outputs) are connected to pins 1 and 9 (ENA, ENB) in L293D to control the speed of the two motors.

  Pins 4 and 5 (digital outputs) are connected to pins 2 and 7(INA1, INA2) in L293D to control the spinning direction of the first motor.

Pins 7 and 8 (digital outputs) are connected to pins 10 and 15 (INB1, INB2) in L293D to control the spinning direction of the second motor.

- L293D connections

  The L293D Motor Driver has two power pins and four GND, the first power pin (VS) is connected to a 9v battery to give power to the internal H-Bridge to drive the motors. It also has 4 output pins, OUT1 and OUT2 are connected to the first motor, while OUT3 and OUT4 are connected to the second motor. There are also two input pins for each motor where IN1 and IN2 are for the spinning direction of the first motor, while IN3 and IN4 are for the spinning direction of the second motor and all of them are connected to the arduino.The speed control pins ENA and ENB are used to turn on/off the motors and control their speed.

8. Future Work

In order to make a real wheelchair that can carry a real person, the person's weight could be from 60 kg - 110 kg, the wheelchair's weight is about 10 kg, so we will need a 12.24 N.m torque DC motor which can carry up to 120 kg.

9. Conclusion

To conclude, smart wheelchair is a wheelchair that is controlled by a mobile application. It can be controlled by a joystick or by recognizing the speech, the target audience for this project are people with both upper and lower limb disabilities. The aim of the project is to help the disabled and their families and relatives.

10. Source code

a. Arduino code

```
#include <SoftwareSerial.h>

int enA  = 3;
```

```
int inA1 = 4;
int inA2 = 5;
int enB  = 9;
int inB1 = 19;
int inB2 = 20;
int Tx   = 11;
int Rx   = 12;

SoftwareSerial bluetooth(Rx, Tx);

void setup() {
  pinMode(enA,  OUTPUT);
  pinMode(inA1, OUTPUT);
  pinMode(inA2, OUTPUT);
  pinMode(enB,  OUTPUT);
  pinMode(inB1, OUTPUT);
  pinMode(inB2, OUTPUT);

  digitalWrite(enA, HIGH);
  digitalWrite(enB, HIGH);

   analogWrite(enA,128);
  analogWrite(enB,128);

  bluetooth.begin(38400);
}

void loop() {
  while (!bluetooth.available());
  byte mode = bluetooth.read();

  if (mode >= '0' && mode <= '4') {
    bool F = (mode == '1');
    bool B = (mode == '2');
    bool R = (mode == '3');
    bool L = (mode == '4');

    digitalWrite(inA1, F | L);
    digitalWrite(inA2, B);
    digitalWrite(inB1, F | R);
```

```
    digitalWrite(inB2, B);
  }
}
```

b. Speech code

```
import 'package:flutter/material.dart';
import 'package:flutter_speech/flutter_speech.dart';
import
'package:smartwheelchair/bluetooth_screens/bluetooth-settings.
dart';
import 'dart:convert';
import 'dart:typed_data';

class Speech extends StatefulWidget {
  final Function() notifyParent;
  Speech({required this.notifyParent});

  @override
  _SpeechState createState() => _SpeechState();
}

class _SpeechState extends State<Speech> {
  late SpeechRecognition _speech;

  bool _speechRecognitionAvailable = false;
  bool _isListening = false;

  String transcription = '';

  @override
  initState() {
    super.initState();
    activateSpeechRecognizer();
  }

  void activateSpeechRecognizer() {
    _speech = SpeechRecognition();
    _speech.setAvailabilityHandler(onSpeechAvailability);
```

```
_speech.setRecognitionStartedHandler(onRecognitionStarted);

_speech.setRecognitionCompleteHandler(onRecognitionComplete);
    _speech.setErrorHandler(errorHandler);
    _speech.activate('en_US').then((res) {
      setState(() => _speechRecognitionAvailable = res);
    });
  }

  @override
  Widget build(BuildContext context) {
    return Padding(
            padding: EdgeInsets.all(8.0),
            child: Center(
              child: Column(
                mainAxisSize: MainAxisSize.min,
                crossAxisAlignment: CrossAxisAlignment.center,
                children: [
                      Container(
                          padding: const EdgeInsets.all(8.0),
                          child: Text(transcription)
                          ),
                      Spacer(),

                      ElevatedButton(
                        style: ElevatedButton.styleFrom(shape:
CircleBorder(), padding: EdgeInsets.all(35.0),
backgroundColor: _isListening? Colors.blueGrey : Colors.blue),
                        onPressed: _speechRecognitionAvailable
&& !_isListening
                        ? () => start()
                        : _isListening ? () => cancel() :
null,
                        child: Icon(
                          Icons.mic,
                          size: 70,
                        ),
                      ),
```

```
                    Spacer(),

                    ElevatedButton(
                      style: ElevatedButton.styleFrom(shape:
CircleBorder(), padding: EdgeInsets.all(30.0),
backgroundColor: Colors.red),
                      onPressed: () => {
                        setState(() => {
                          transcription =
setTranscription('stop'),
                            cancel(),
                        })
                      },
                      child: Icon(
                        Icons.front_hand_rounded,
                        size: 80,
                      ),
                    ),

                    Spacer(),
                ],
              ),
            ));
  }

  void start() => _speech.activate("en_US").then((_) {
        return _speech.listen().then((result) {
          setState(() {
            _isListening = result;
          });
        });
      });

  void cancel() => _speech.cancel().then((_) => setState(() =>
_isListening = false));

  void onSpeechAvailability(bool result) =>
      setState(() => _speechRecognitionAvailable = result);

  void onRecognitionStarted() {
```

```
    setState(() => _isListening = true);
  }

  void onRecognitionComplete(String text) {
    setState(() => transcription = setTranscription(text));
    setState(() => _isListening = false);
  }

  void errorHandler() => activateSpeechRecognizer();

  String setTranscription(String text) {
    String mode;

    if(text.contains("stop") && connectionState == 2){
      mode = '0';
      bluetoothSend(mode);
      return "The wheelchair stopped";
    }
    else if(text.contains("forward") && connectionState == 2){
      mode = '1';
      bluetoothSend(mode);
      return "The wheelchair is moving forward";
    }
    else if(text.contains("backward") && connectionState ==
2){
      mode = '2';
      bluetoothSend(mode);
      return "The wheelchair is moving backwards";
    }
    else if(text.contains("right") && connectionState == 2){
      mode = '3';
      bluetoothSend(mode);
      return "The wheelchair is turning right";
    }
    else if(text.contains("left") && connectionState == 2){
      mode = '4';
      bluetoothSend(mode);
      return "The wheelchair is turning left";
    }
    else if (connectionState != 2){
```

```
      return "You are not connected";
    }

    return "Unrecognized";
  }

  void bluetoothSend(String mode) async {
    try {

connection!.output.add(Uint8List.fromList(utf8.encode(mode)));
      await connection!.output.allSent;
    } catch (e) {
      connectionState = 0;
//terminate connection
      widget.notifyParent();
//notify parent
      setState(() {});
    }
  }
}
```

c. Joystick code

```
import 'package:flutter/material.dart';
import
'package:smartwheelchair/bluetooth_screens/bluetooth-settings.dart';
import 'package:flutter_joystick/flutter_joystick.dart';
import 'dart:convert';
import 'dart:typed_data';
import 'dart:async';

class ClassicControl extends StatefulWidget {
  final Function() notifyParent;
  ClassicControl({required this.notifyParent});

  @override
  _ClassicControlState createState() => _ClassicControlState();
}

class _ClassicControlState extends State<ClassicControl> {
```

```
  bool stop = true;

  @override
  Widget build(BuildContext context) {
    final ButtonStyle style = ElevatedButton.styleFrom(padding:
EdgeInsets.all(8.0));

    return Align(
      child: Joystick(
        mode: JoystickMode.horizontalAndVertical,
        listener: (details) {
          setState(() {
            stop = false;

            Timer(Duration(milliseconds: 100), () {
              if (stop) {
                bluetoothSend('0');
              }
              stop = true;
            });

            if (details.y < -0.3) {
              bluetoothSend('1');
            } else if (details.y > 0.3) {
              bluetoothSend('2');
            } else if (details.x > 0.3) {
              bluetoothSend('3');
            } else if (details.x < -0.3) {
              bluetoothSend('4');
            } else {
              bluetoothSend('0');
            }
          });
        },
      ),
    );
  }

  void bluetoothSend(String mode) async {
    try {
```

```
      connection!.output.add(Uint8List.fromList(utf8.encode(mode)));
      await connection!.output.allSent;
    } catch (e) {
      connectionState = 0;
      widget.notifyParent();
      setState(() {});
    }
  }
}
```