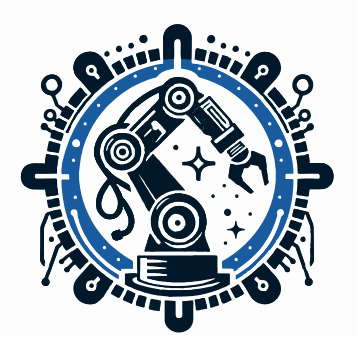
AUTOMATED ROBOTIC ARM DOCUMENTATION

ROBOARM

Supervised by: Dr. Hatem Yousry
: Eng. Eman El-Sayed

جامعة القاهرة الجديدة التكنولوجية

NEW CAIRO TECHNOLOGICAL UNIVERSITY



AUTOMATED ROBOTIC ARM (ROBOARM)

Supervised by:

Dr. Hatem Yousry

Eng. Eman El-Sayed

Team Members

Mohamed Ahmed Al-Saeed

George Samir Yacoub

Mohamed Al-Shabrawy Saber

Walaa Sapry Mahmoud

Ahmed Sayed Abudief

Hassan Khaled Ahmed

July 2024

Index

List of figures

Chapter 1: Introduction.

Section 1.1: Problem definition.

- The labor market faces many challenges related to safety, efficiency, working in hazardous environments, and completing tasks quickly. Humans are limited to lifting relatively light weights, and can only accomplish some tasks. Not all jobs are suitable for everyone, and not all individuals have the right focus; Some people may miscalculate certain calculations, which could lead to disastrous consequences.
- For Example, Ahmed Works in An Automobile Factory Where He Is Responsible for Distributing the Factory's Resources to Their Appropriate Places: Glass to The Glass Corner, Paint to The Paint Corner, Etc., Etc. If He Calculates How Many Workers Are Needed to Accomplish This Task, We Will Need a Large Number of Workers, Some of Whom Will Carry the Glass to The Car Factory. Glass Angle, Some of Them Will Carry the Paint to The Paint Angle, Some of Them Will Need to Lift Heavy Weights. What About the Time It Takes a Worker to Carry Materials? Can One Person Carry a Heavy Weight Alone and Walk with It for A Long Distance? Can All Workers Perform Some Dangerous Tasks Without Fear, And Can

They Work Without Interruption? I Don't Think So. Here We Faced a Problem That Required Us to Think of a Solution.

Section 1.2: Solution approach.

After Thinking Deeply About a Solution to This Problem, We Have Come Up with A Solution That Can Save You Time and Effort. It Can Also Work in Hazardous Environments and Work All Day Long Without Stopping. The Error Rate Is Almost Negligible. We Designed an innovative robotic arm to improve material handling in industrial environments. This state-of-the-art project is designed to move boxes seamlessly between conveyor belts with precision and adaptability. The arm, equipped with an advanced sensor system, detects incoming bins on the belt, quickly stopping the conveyor's movement. Upon detection, the robotic arm maneuvers efficiently, smoothly grab the box, and moves it to the designated belt. The user interface provides multi-faceted control mechanisms. With dedicated desktop software and an easy-to-use mobile app, operators can seamlessly manage arm procedures. It is worth noting that the lever holder, which is regulated by Joysticks, allows fine-tuning and fine adjustments. One great feature is the ability to record and store various arm movements. Users can pre-select and save specific movements, allowing for easy

selection via arm interface buttons, desktop software, or mobile app. Designed primarily for the integration of industrial plants, the arm revolutionizes logistics operations, ensuring efficient automated box transfers between conveyor belts while providing comprehensive control and adaptability for various tasks.

Section 1.3: Tools.

The tools used in the project are divided into:

- Simulation:
  - Make a simulation of cutting the electrical circuit of the robot to ensure the correct connection of the pieces to each other and their compatibility with each other.
  - The simulator used: Proteus. *(Figure –1.3.1)*

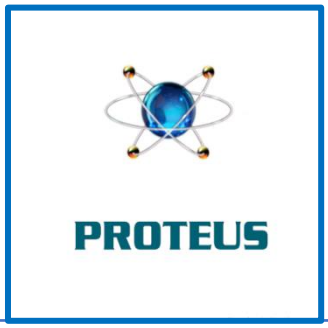


Figure – 1.3.1 - Proteus

- Components:
  - Gear Motor: We used a gear motor to operate the production line and move the products to the robot to be picked up and put in place. *(Figure –1.3.2)*

Figure – 1.3.2- Gear motor

- Connecting wires: Electrical wires to connect all components together.
  (Figure-1.3.4)

Figure – 1.3.4 - Connecting wires

- Robot Arm 6 Dof Kit:(6 Degrees Of Freedom) We used this type of robot because it will give us the best freedom, movement and space in the case of our project *(Figure –1.3.5)*

Figure – 1.3.5- Robot Arm 6

- Power Supply 5v 10 A: we use this power supply to give the power to our robot *(Figure-1.3.6)*

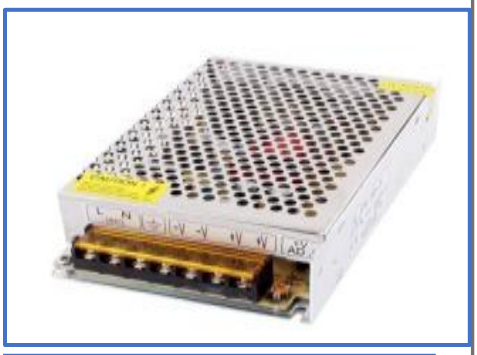


Figure – 1.3.6 - Power Supply

- Servo motor: The function of this motor is that it carries the motor up to 180 degrees, so we can move the arm in any direction we want. *(Figure-1.3.7)*

Figure – 1.3.7 – Servo

- Adaptors (9v 2a) We use this adaptor to give power for our Arduino and form Arduino we give power to other components except (the arm and his servos) *(Figure – 1.3.8)*

Figure –1.3.8 – Adaptor

- The Arduino Mega: Is A Microcontroller Board Based on The Atmega2560. It's Designed for More Complex Projects It's Commonly Used in Robotics, 3D Printers, And Other Projects and we used it in our project *(Figure-1.3.9)*

Figure – 1.3.9 – Arduino Mega

- The PCA9685 Servo Shield: Is A Hardware Component Used to Control mutable Servo Motors together and control it via just 2 pins

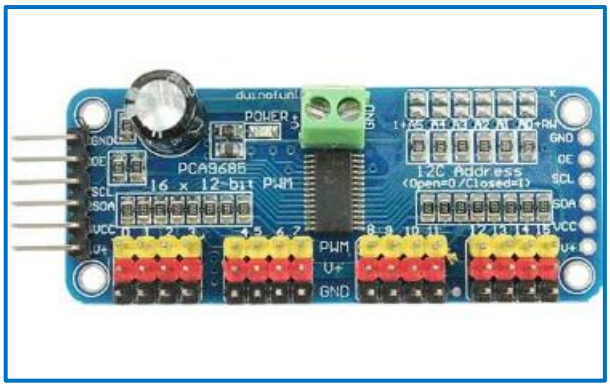


Figure – 1.3.10 – Motor Driver Shield

(SCL, SDA). we use it in our project to control with the arm *(Figure-1.3.10)*

- LCD: Liquid Crystal Display we used lcd 4*20 i2c connection to display the process of our project *(Figure-1.3.11)*

Figure –1.3. 11 –LCD

- Buttons: Are Simple Mechanical or Electronic Devices we Used To control of some function in our robot *(Figure – 1.3.12)*

Figure – 1.3.12 –Button

- A Joystick: Also Known as A Control Column, Is an Input Device we Used For control the servos positions and movement of the robot each joystick can control 2 servo motor in the robot. *(Figure – 1.3.13)*

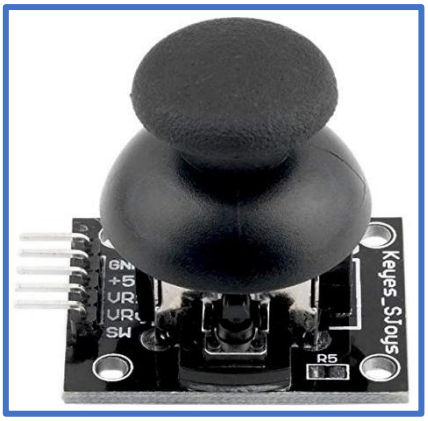


Figure – 1.3.13-Joystick

- IR Sensor: An Infrared Sensor (IR Sensor) we used this sensor to detect the product in the line *(Figure – 1.3.14)*

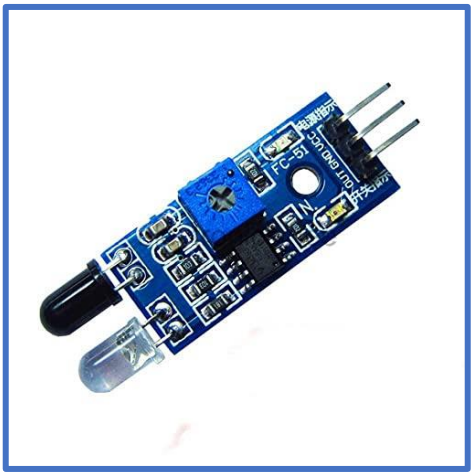


Figure – 1.3.14- IR Sensor

- Breadboard: A Breadboard Is Used for Building Circuits. We used it for connect the buttons to Arduino and also give power to each component form Arduino *(Figure – 1.3.15)*

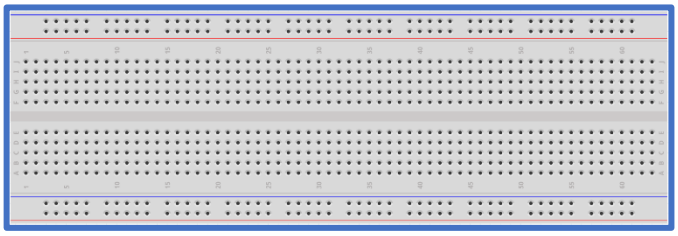


Figure – 1.3.15- Breadboard

- L298d H- Bridge: The L298D Driver Is an Integrated Circuit That Can Drive a Dual-Channel H-Bridge Motor Capable of Driving a Pair of DC Motors Or a Single Motor. We used it for connecting

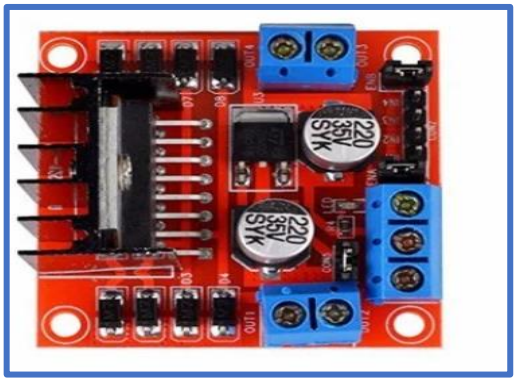


Figure – 1.3.16- L293d H- Bridge

the motor the run the production line *(Figure – 1.3.16)*

- UI/UX:
  - Figma: Is a web-based design and prototyping tool used for creating user interfaces, interactive prototypes, and collaborative design projects. It allows designers to work together in real-time, making it ideal for team-based design workflows. Figma's features include vector graphics editing, design components, version control, and seamless sharing options, making it a popular choice for UI/UX designers and product teams.

    Its cloud-based nature enables accessibility across different devices and platforms without the need for software installation. *(Figure – 1.3.17)*

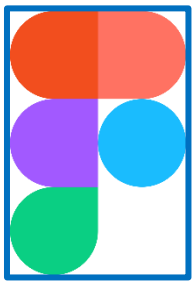


Figure – 1.3.17- Figma

- Programming language:
  - C/C++: We used it to program the Arduino Mega. *(Figure-1.3.18)*

Figure – 1.3.18- C++

- Flutter: We used it so that we could create an application that works on most devices, and we specifically targeted Android and Windows. In Filter, Dart is used as a programming language. *(Figure-1.3.19)*

Figure – 1.3.19- Dart and Flutter

- Kotlin: We used the Kotlin language to interface with Android's Native through Flutter. *(Figure-1.3.20)*

Figure – 1.3.20 – Kotlin

- HTML/CSS/JavaScript: We also used web technologies to confirm new accounts that will be registered in the application, as well as for email templates and for the project website where its details are displayed. *(Figure-1.3.21)*

Figure – 1.3.21 – Web Development Technologies

- Arduino IDE: Used to write and run C++ code on the Arduino. *(Figure-1.3.22)*

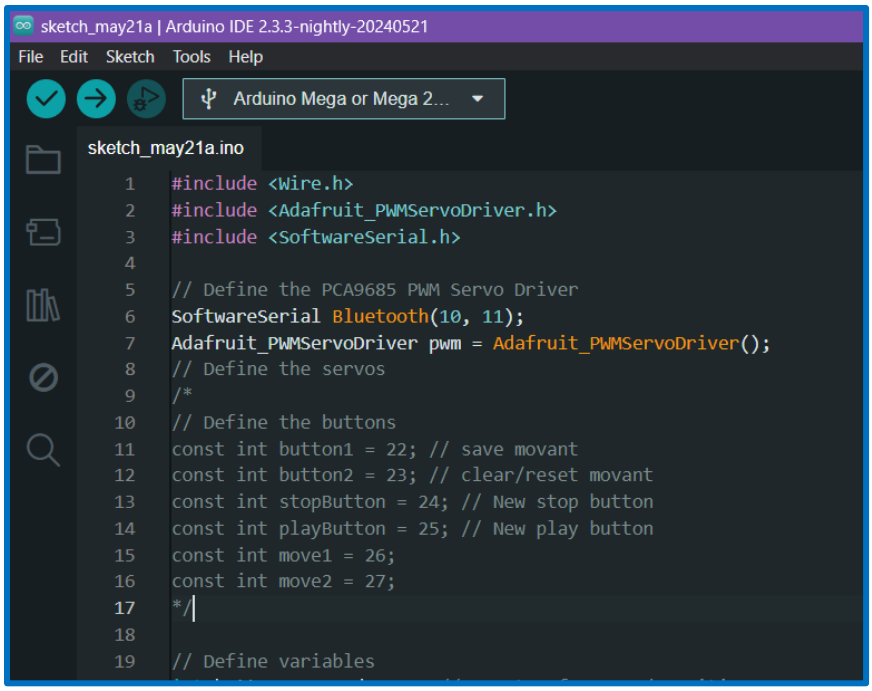


Figure – 1.3.22 – Arduino IDE

- Android Studio: We used Android Studio to program

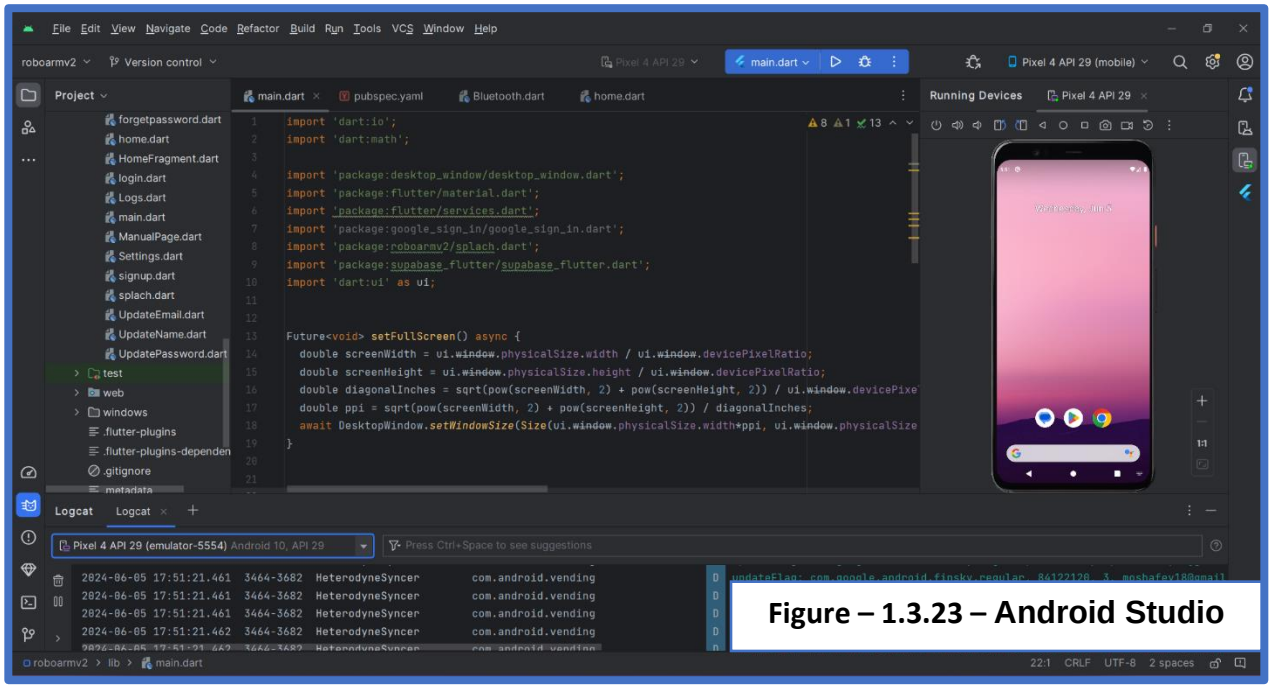


Figure – 1.3.23 – Android Studio

the application, an integrated development

environment that also includes an Android emulator. *(Figure – 1.3.23)*

- Visual Studio: We used Visual Studio only because through it we can run our application on Windows. *(Figure-1.3.24)*

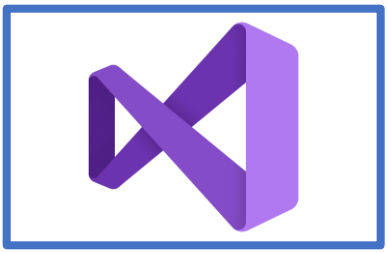


Figure – 1.3.24 – Visual Studio

- Visual Studio Code: We used it to write and program the website for project details, pages, and email templates for confirming new accounts. *(Figure-1.3.25)*

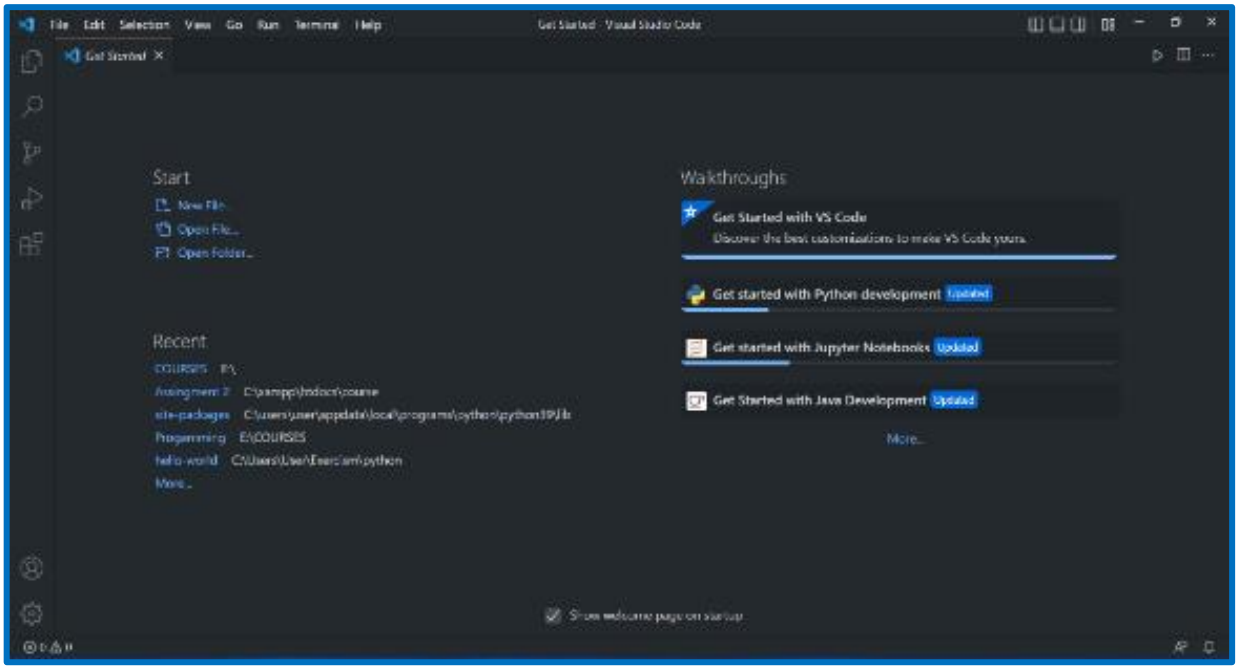


Figure – 1.3.25 – Visual Studio Code

- Supabase: Supabase is an open-source backend as a service (BaaS) platform designed to help developers quickly build secure and scalable applications. It provides a variety of back-end services that are

typically required for web and mobile application development. We have used it to use authentication for users of accounts on the application. *(Figure-1.3.26)*

Figure – 1.3.26 – Supabase

- Firebase: Firebase is a platform developed by Google for creating mobile and web applications. It offers a range of cloud services and tools to help developers build and manage their applications more efficiently. We used it to host the project details website and also browse the web pages for confirming new accounts on the app. *(Figure-1.3.27)*

Figure – 1.3.27 – Firebase

- Google Cloud: Among all its features, we only used Credential to ensure that our app is reliable and any Google user can register through it. *(Figure-1.3.28)*

Figure – 1.3.28 – Google Cloud

- Brevo: It is a comprehensive digital marketing platform that provides tools and services for email marketing, customer relationship management (CRM), marketing automation, and more. It aims to help businesses of

all sizes manage their marketing efforts effectively. It is only used to its SMTP Server in order to send emails through Supabase to the user. *(Figure-1.3.29)*

Figure – 1.3.29 – Brevo

- Word, PowerPoint: We used them to document, write the documentation, and prepare the presentation. *(Figure-1.3.30,31)*

Figure – 1.3.30 – Word

Figure – 1.3.31 –PowerPoint

Chapter 2: Related works.

The industrial sector has seen significant advancements in automation, particularly with the development of robotic arms designed to enhance material handling processes. Several notable examples of such robotic arms currently available in the market include:

- KUKA KR AGILUS Series: The KUKA KR AGILUS series features small, high-speed robotic arms designed for various material handling applications, including picking, packing, and palletizing. *(Figure-2.1)*

Figure – 2.1 – KUKA KR AGILUS Series

  - Defects:
    - Limited Flexibility: The KR AGILUS series, while fast, can be limited in its adaptability to different types of tasks without significant reprogramming.
    - Complex Programming: Requires specialized knowledge for programming and operation, which can be a barrier for

smaller companies or those without extensive technical staff.

- Sensor Integration: Although equipped with sensors, the system may require additional calibration for different environments, impacting efficiency.

- ABB IRB 6700: The ABB IRB 6700 is a robust industrial robot known for its strength and precision in heavy-duty material handling tasks. *(Figure-2.2)*

Figure – 2.2 – ABB IRB 6700-245/3.00

  - Defects:
    - Size and Mobility: The large size of the IRB 6700 can limit its mobility and adaptability in smaller or more dynamic workspaces.
    - Setup and Maintenance: Requires extensive setup and regular maintenance, leading to potential downtime and increased labor costs.

- Fanuc M-20iA/35M: The Fanuc M-20iA/35M is a versatile robotic arm suitable for a range of industrial applications, known for its precision and reliability. *(Figure-2.3)*

Figure – 2.3 – Fanuc M-20iA/35M

    - Defects:
        - User Interface: The interface can be less intuitive, making it harder for operators to manage without significant training.
        - Customization: Limited options for customization can restrict its use in highly specialized tasks.
        - Integration: Integrating with existing systems and conveyors can be challenging, often requiring additional equipment or software.

Transforming Defects into Advantages

Our innovative robotic arm project addresses these deficiencies by introducing several key improvements:

- Enhanced Flexibility and Adaptability

- Advantage: Unlike the KUKA KR AGILUS, our robotic arm features advanced adaptability, allowing it to seamlessly transition between different tasks without extensive reprogramming. The user can easily fine-tune and adjust movements using joysticks and pre-recorded movement selections, facilitating quick adaptation to various tasks and environments.

- Intuitive User Interface
  - Advantage: Our robotic arm incorporates a user-friendly interface accessible via dedicated desktop software and a mobile app. This contrasts with the complex programming required by the ABB IRB 6700 and the less intuitive Fanuc M-20iA/35M interfaces. Operators of all skill levels can manage the arm efficiently, reducing the need for specialized training and minimizing human error
- Sensor System Integration
  - Advantage: Equipped with an advanced sensor system, our robotic arm can detect and respond to incoming bins more effectively. This capability reduces the need for additional calibration and

enhances operational efficiency, addressing the integration and setup challenges faced by competitors like the ABB IRB 6700.

- Compact and Efficient Design
  - Advantage: Our robotic arm is designed to be compact yet powerful, addressing the size and mobility issues of the ABB IRB 6700. Its efficient design allows for better integration into various industrial environments, from large-scale operations to smaller, more dynamic workspaces.

By addressing these deficiencies and transforming them into advantages, our innovative robotic arm not only meets but exceeds the current market standards, providing a superior solution for automated material handling in industrial environments.

Chapter 3: Implementation

Section 3.1: Research.

First, We Did Research on The Components Used in The Robot and Understand How They Work, Their Basic Principles, And How They Communicate With Each Other And Program Them.

Section 3.1.1: Robot Arm 6dof Kit:

- What Is Robot Arm 6dof Kit?
  - A Robot Arm 6 Degrees of Freedom (Dof) Kit Refers to A Robotic Arm Assembly That Provides Six Independent Axes of Motion or Degrees of Freedom. Each Degree of Freedom Corresponds to A Specific Type of Movement That the Robot Arm Can Perform. Here's An Overview of What A 6-Dof Robot Arm Kit Typically Includes and Its Applications: *(Figure-3.1.1.1)*

Figure – 3.1.1.1 – Arm Robot

    - Components: A Typical 6-Dof Robot Arm Kit Includes Mechanical Components Such as Joints, Linkages, Actuators (Motors), Gears, And End-Effectors (Grippers or

Tools). It May Also Include Electronic Components Such as Motor Controllers, Sensors, And Cables.

- Degrees Of Freedom: The Term "6 Degrees of Freedom" Refers to The Ability of The Robot Arm to Move in Six Different Directions Independently. These Directions Are Typically Defined as Follows three Translational Degrees of Freedom (X, Y, Z): The Robot Arm Can Move Horizontally (X And Y Axes) And Vertically (Z Axis) In Three-Dimensional Space.

- Three Rotational Degrees of Freedom (Roll, Pitch, Yaw): The Robot Arm Can Rotate Around Its X, Y, And Z Axes, Allowing for Orientation Adjustments.
    - Applications: 6-Dof Robot Arms Have Various Applications Across Industries, Including:
        - Manufacturing: Assembly Tasks, Pick-And-Place Operations, Material Handling, And Machine Tending.

- Automation: Industrial Automation, Process Automation, And Laboratory Automation.
- Research And Education: Robotics Research, Academic Projects, And Educational Demonstrations.
- Healthcare: Surgical Robotics, Rehabilitation Robotics, And Medical Device Manufacturing.
- Agriculture: Precision Agriculture, Crop Monitoring, And Harvesting.

- Control And Programming: The Robot Arm May Be Controlled Manually Using a Joystick or Other Input Device, Or It May Be Programmed to Execute Predefined Sequences of Movements Autonomously. Programming Can Be Done Using Software Tools, Programming Languages, Or Graphical User Interfaces (Guis).
- Customization And Expansion: Some Robot Arm Kits Allow for Customization and Expansion by Providing Modular Components or Interfaces for Integrating

Additional Sensors, Actuators, Or End-Effectors. This Flexibility Enables Users to Adapt the Robot Arm to Specific Applications or Requirements.

- What Is the Use and Benefit Robot Arm 6 Dof Kit?
    - 6-Axes Robotic Arm Is Useful for Industrial Applications Like Welding, Painting, Assembly, Pick and Place, Product Inspection and Testing with Great Accuracy.
- How Robot Arm 6 Dof Kit Works?
    - The Arm Has Six Axes and Six Degrees Of Freedom. The Six Degrees of Freedom Allows The Robot To Move In All The Three Dimensions. Servo Motors Are Used As Controlling Elements For Providing The Different Degrees Of Freedom To The Robot Arm.

Section 3.1.2: 6 Servos Towr Mg996r:

What Are 6 Servos Tower Pro MG996R?

Figure – 3.1.2.1 – Servos

Servos:

Servos are small motors commonly used in robotics and remote control (RC) applications for precise control of position, speed, and torque. They typically consist of a motor, gearbox, and a feedback mechanism (such as a potentiometer) packaged in a compact housing. *(Figure-3.1.2.1)*

Tower Pro MG996R:

The Tower Pro MG996R is a specific model of servo motor produced by Tower Pro, a manufacturer of electronic components and RC accessories. Known for its high torque output and reliability, the MG996R is popular in robotics, RC vehicles, and other applications requiring precise motion control.

6 Servos:

Using six individual MG996R servos in a setup allows for independent control of each servo. This enables the creation of complex robotic mechanisms or multi-degree-of-freedom (DOF) systems, such as a 6 DOF robotic arm.

Uses and Benefits of 6 Servos Tower Pro MG996R

1. Robotics:

- Control the movement of robotic arms, legs, hexapod robots, or robotic hands. Six servos provide complex and flexible

movements.

2. RC Vehicles:

- Used in remote-controlled cars, boats, or airplanes to control steering, throttle, and other mechanisms.

3. Pan and Tilt Mechanisms:

- Create sophisticated pan and tilt mechanisms for cameras or sensors, allowing precise directional control.

4. Articulated Structures:

- Create realistic and lifelike movements in animatronic figures or model creatures.

5. Educational Purposes:

- Provide learning opportunities in robotics, electronics, and programming. Allow experimentation with different configurations and control algorithms.

6. Custom Applications:

- Versatile for a wide range of custom applications, limited only by creativity and engineering skill.

How 6 Servos Tower Pro MG996R Work

1. Motor:

- Each MG996R contains a DC motor that generates mechanical motion. When voltage is applied, it rotates a gear mechanism.

2. Gear Train:

- The gear train reduces the speed of rotation while increasing torque, allowing the servo to exert greater force.

3. Potentiometer (Feedback Device):

- Attached to the output shaft, the potentiometer provides feedback on the current position. Changes in resistance indicate the shaft's position to the control circuitry.

4. Control Circuitry:

- The control circuitry receives an input signal (usually PWM) specifying the desired position. It compares this with the feedback signal from the potentiometer to determine necessary adjustments.

5. Feedback Loop:

- Continuously adjusts the voltage applied to the DC motor, modulating the pulse width based on the comparison between the desired and actual positions.

6. Position Control:

- The feedback loop ensures the servo accurately maintains the desired position or angle specified by the input signal. This makes the MG996R ideal for applications requiring precise control over movement.

By using multiple MG996R servos, you can independently control each servo motor, allowing for complex and coordinated movements in robotic arms, vehicles, or other mechanisms. Each servo operates independently but can be synchronized and coordinated through control signals.

Detailed Specifications and Features of Tower Pro MG996R

- Dimensions: 40.7 x 19.7 x 42.9 mm
- Weight: 55g
- Stall Torque:
  - 9.4 kg/cm at 4.8V
  - 11 kg/cm at 6.0V
- Operating Speed:
  - 0.19 sec/60 degrees at 4.8V
  - 0.15 sec/60 degrees at 6.0V
- Operating Voltage: 4.8V to 6.6V
- Gear Type: Metal gears for enhanced durability
- Temperature Range: 0°C to 55°C
- Servo Plug: JR (compatible with JR and Futaba)
- Dead Band Width: 1µs
- Servo Wire Length: 30 cm

Features

- High Torque: Provides substantial torque, crucial for heavy-duty applications like robotic arms.
- Precision Improved PCB and IC control system enhances accuracy.
- Durability: Metal gears increase robustness and longevity.
- Shock-Proofing: Upgraded internal components reduce backlash and enhance centering.

Applications

- Robotics: Ideal for robotic arms, including 6 DOF configurations, offering the necessary strength and precision.

- RC Models: Commonly used in RC cars, airplanes, and helicopters due to high torque and reliability.

- DIY Projects: Suitable for various DIY electronics and automation projects.

- Control: Controlled using PWM signals, common in Arduino and other microcontroller platforms.

Section 3.1.3: Arduino Mega:

What Is the Arduino Mega?

- ✓ The Arduino Mega Is A Microcontroller Board Based On The Atmega2560 Chip. It's An Enhanced Version Of The Original Arduino Uno Board, Offering More Digital Input/Output Pins, More Analog Input Pins, More Memory, And More Features Overall. Here's An Overview Of Its Key Features*: (Figure-3.1.3.1)*

Figure – 3.1.3.1 – Arduino Mega

1. Microcontroller: The Arduino Mega Is Powered By The Atmega2560 Microcontroller, Which Has 256 Kb Of Flash Memory For Storing Code, 8 Kb Of Sram For Data Storage, And 4 Kb Of Eeprom For Non-Volatile Storage.

2. Digital I/O Pins: The Mega Has 54 Digital Input/Output Pins, Of Which 15 Can Be Used As Pwm (Pulse Width Modulation) Outputs. These Pins Allow You To Interface With Various Digital Sensors, Actuators, And Other Devices.
3. Analog Inputs: It Features 16 Analog Inputs, Which Can Be Used To Read Analog Signals From Sensors Like Potentiometers, Temperature Sensors, Or Light Sensors.
4. Communication Interfaces: The Mega Supports Multiple Communication Interfaces Including Uart (Serial Communication), Spi (Serial Peripheral Interface), And I2c (Inter-Integrated Circuit). These Interfaces Enable Communication With Other Devices Such As Sensors, Displays, Or Other Microcontrollers.
5. Usb Interface: The Board Includes A Usb Interface For Programming And Serial Communication With A Computer. This Allows You To Upload Code From Your Computer To The Board And Communicate With It For Debugging Or Data Transfer.
6. External Power Options: The Mega Can Be Powered Via The Usb Connection Or An External Power Supply. It Also Has A Vin Pin That Allows You To Power The Board Using A Dc Power Supply Ranging From 7v To 12v.
7. Compatibility: The Arduino Mega Is Compatible With The Arduino Software Development Environment, Making It Easy To Write, Upload, And Debug Code For Your Projects.
8. Expandability: Due To Its Large Number Of Pins And Communication Interfaces, The Mega Is Well-Suited For

Projects That Require A High Level Of Connectivity Or Involve Multiple Sensors And Actuators.

How Arduino Mega Works?

1. Programming: You Write Code For Your Arduino Mega Using The Arduino Integrated Development Environment (Ide), Which Is Based On The Processing Programming Environment. The Code Is Written In The Arduino Programming Language, Which Is A Simplified Version Of C/C++. You Define Variables, Functions, And Logic To Control The Behavior Of The Microcontroller.
2. Compilation: Once You've Written Your Code, You Compile It Using The Arduino Ide. The Ide Translates Your Arduino Code Into Machine-Readable Instructions Called Machine Code. This Machine Code Is Specific To The Microcontroller (Atmega2560) On The Arduino Mega.
3. Upload: After Compilation, You Upload The Compiled Code To The Arduino Mega Board. This Is Typically Done Via A Usb Connection Between Your Computer And The Arduino Mega. The Arduino Ide Communicates With The Microcontroller On The Board And Sends The Compiled Code To Be Stored In The Flash Memory Of The Atmega2560.

4. Execution: Once The Code Is Uploaded, The Microcontroller Begins Executing It. The Program Runs In A Continuous Loop, Executing The Instructions You've Written In Your Code. These Instructions May Involve Reading Sensor Data, Controlling Actuators, Making Decisions Based On Input, And Performing Other Tasks

Defined In Your Program.

Section 3.1.4: Pca9685 Servo Shield:

What Is Pca9685 Servo Shield?

The Pca9685 Servo Shield Is A Popular Breakout Board Designed To Control Multiple Servo Motors Using The Pca9685 Pwm (Pulse Width Modulation) Driver Chip. Here's An Overview Of Its Key Features And Functionality: *(Figure-3.1.4.1)*

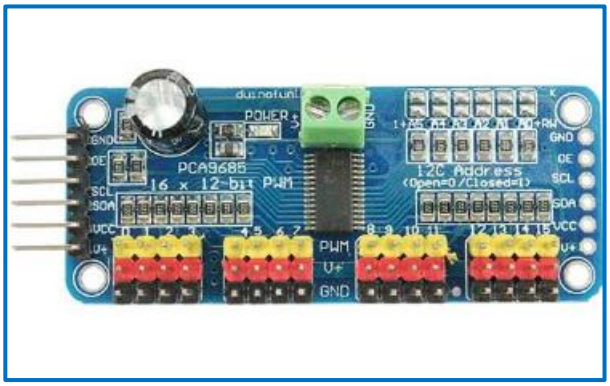


Figure – 3.1.4.1– Pca9685 Servo Shield

1. Pca9685 Pwm Driver:

   The Heart Of The Pca9685 Servo Shield Is The Pca9685 Pwm Driver Chip, Which Is Capable Of Generating 16 Channels Of Pwm Signals. Each Channel Can Be Individually Controlled To Generate Precise Pwm Signals With Adjustable Pulse Width.

2. Multiple Servo Control: The Pca9685 Servo Shield Allows You To Control Up To 16 Servo Motors Simultaneously. This Makes It Ideal For Projects Requiring The Precise Control Of Multiple Servo Motors, Such As Robotic Arms, Drones, Animatronics, And More.

3. I2c Interface: The Pca9685 Servo Shield Communicates With A Microcontroller (Such As An Arduino) Using The I2c (Inter-Integrated Circuit) Protocol. This Allows For Easy Interfacing With A Wide Range Of Microcontrollers, As I2c Is A Common

Communication Protocol Supported By Many Microcontrollers.

4. Stackable Design: The Pca9685 Servo Shield Typically Has A Stackable Design, Allowing You To Easily Stack Multiple Shields On Top Of Each Other To Control Even More Servo Motors If Needed. This Makes It Convenient For Expanding The Servo Control Capabilities Of Your Project Without The Need For Complex Wiring Or Soldering.
5. Built-In Voltage Regulation: Many Pca9685 Servo Shield Boards Include Built-In Voltage Regulation Circuitry, Allowing Them To Accept A Wide Range Of Input Voltages (Usually 5v To 6v) And Provide Stable Power To The Connected Servo Motors.
6. Pwm Frequency Adjustment: The Pca9685 Chip Allows You To Adjust The Pwm Frequency, Which Determines How Often The Pwm Signal Is Updated. This Flexibility Allows You To Optimize The Pwm Frequency For Your Specific Application, Balancing Between Smooth Servo Motion And Cpu Usage.
7. Open-Source Libraries: There Are Various Open-Source Libraries Available For The Pca9685 Servo Shield, Making It Easy To Interface With Popular Development Platforms Such As Arduino. These Libraries Provide Functions To Initialize The Shield, Set The Pwm Duty Cycle For Each Servo, And Control The Servo Motors With Ease.

How Pca9685 Servo Shield Works?

The Pca9685 Servo Shield Works By Using The Pca9685 Pwm (Pulse Width Modulation) Driver Chip To Generate Pwm Signals

That Control The Position Of Servo Motors. Here's A Detailed Explanation Of How It Works:

1. Pwm Generation: The Pca9685 Chip On The Servo Shield Is Capable Of Generating 16 Channels Of Pwm Signals. These Pwm Signals Are Used To Control The Position Of Servo Motors Connected To The Shield.

2. I2c Communication: The Servo Shield Communicates With A Microcontroller (Such As An Arduino) Using The I2c (Inter-Integrated Circuit) Protocol. This Allows The Microcontroller To Send Commands To The Pca9685 Chip To Set The Pwm Duty Cycle For Each Servo Channel.

3. Initialization: To Use The Servo Shield, You First Initialize It By Configuring The Pca9685 Chip And Setting The Pwm Frequency. The Pwm Frequency Determines How Often The Pwm Signals Are Updated, Which Affects The Resolution And Smoothness Of Servo Movement.
4. Servo Control: Once Initialized, You Can Control The Position Of Servo Motors By Setting The Pwm Duty Cycle For Each Servo Channel. The Duty Cycle Determines The Position Of The Servo's Output Shaft. A Duty Cycle Of 0% Corresponds To The Minimum Position, While A Duty Cycle Of 100% Corresponds To The Maximum Position. Intermediate Duty Cycles Correspond To Positions Between The Minimum And Maximum.
5. Smooth Motion: The Pca9685 Chip Generates Smooth Pwm Signals With Adjustable Pulse Widths, Allowing For Precise Control Of Servo Motor Position And Speed. By Adjusting The Pwm Duty Cycle, You Can Smoothly

Transition Between Different Positions And Achieve The Desired Motion In Your Project.

6. Scalability: The Servo Shield Is Stackable, Meaning You Can Easily Stack Multiple Shields On Top Of Each Other To Control More Servo Motors If Needed. Each Shield Communicates Independently With The Microcontroller Via The I2c Bus, Allowing You To Control Multiple Servo Motors With Ease.
7. Power Supply: The Servo Shield Typically Includes Built-In Voltage Regulation Circuitry To Provide Stable Power To The Connected Servo Motors. It Accepts A Wide Range Of Input Voltages (Usually 5v To 6v) And Ensures Consistent Performance Of The Servo Motors.

❖ Connecting:

1.Installing PCA9685 on Arduino Mega:

- Connect the VCC of the PCA9685 to the 5V pin of the Arduino Mega.

- Connect the GND of the PCA9685 to the GND of the Arduino Mega.

- The servo motors should have a separate power supply due to their higher current requirements.

2. I2C Communication:

- Connect the SDA pin of the PCA9685 to the SDA pin (pin 20) on the Arduino Mega.

- Connect the SCL pin of the PCA9685 to the SCL pin (pin

21) on the Arduino Mega.

3. Servo Connections:

- Connect your servos to the PWM output pins of the PCA9685. Each output port has V+, GND, and PWM pins for connecting servos.

Section 3.1.5: Power Supply 5v 10a:

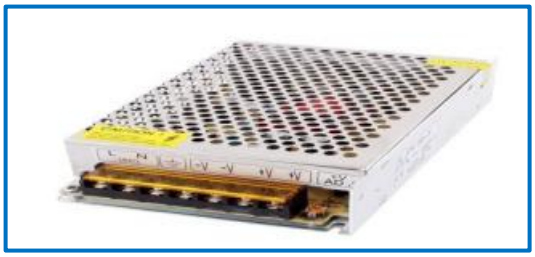


Figure – 3.1.5.1– Power Supply 5v 10a

A power supply rated at 5V 10A provides a constant voltage output of 5 volts and can deliver a maximum current of 10 amperes. *(Figure-3.1.5.1)* Here's what this specification indicates:

Voltage (5V): This specifies the output voltage of the power supply, which is a common voltage used in many electronic devices, microcontrollers, sensors, and other components. This voltage is typically used to power digital circuits, microcontrollers like Arduino, Raspberry Pi, and various sensors and modules.

Current (10A): This indicates the maximum amount of current that the power supply can deliver to connected devices. A current rating of 10A means that the power supply can provide up to 10 amperes of current to connected devices without exceeding its specified limits.

Key Points About a 5V 10A Power Supply:

1.Suitable for High-Power Applications: A 10A current rating makes this power supply suitable for high-power applications that require significant current to operate, such as powering multiple

motors, LED strips, or other power-hungry devices.

2.Overcurrent Protection: Many quality power supplies include overcurrent protection mechanisms to prevent damage to connected devices in case of a short circuit or excessive current draw.

3.Stable Voltage Output: A regulated 5V output ensures that the voltage remains stable even under varying load conditions, which is essential for the proper operation of sensitive electronic components.

4.Multiple Outputs: Some power supplies with a 5V 10A rating may have multiple output channels, allowing you to power several devices simultaneously.

5.Usage: Such power supplies are commonly used in various electronics projects, DIY hobbyist setups, LED lighting installations, and industrial applications where a stable 5V power source with high current capacity is required.

Uses and Benefits of a 5V 10A Power Supply:

1.Powering High-Current Devices: Ideal for powering devices that require higher current than a typical USB port or smaller power supply can provide, including LED strips, high-torque servo motors, stepper motors, DC motors, and other electromechanical devices.

2.Electronics Projects: In electronics projects and DIY setups, it serves as a reliable and stable source of power for microcontrollers (such as Arduino or Raspberry Pi), sensors, displays, communication modules, and other electronic components.

3.LED Lighting: Used in LED lighting installations to ensure consistent brightness and color accuracy, especially in applications where multiple LEDs are arranged in series or parallel configurations.

4.Prototyping and Testing: Useful during the prototyping and testing phases of electronic projects to quickly power up circuits and verify their functionality.

5.Industrial Applications: In industrial settings, it can power control systems, instrumentation, sensors, and other equipment requiring a stable and reliable power source.

6.Charging Devices: Can charge multiple USB-powered devices simultaneously, such as smartphones, tablets, or portable electronic gadgets.

How a 5V 10A Power Supply Works:

1.Input Power: The power supply receives input power from a mains power outlet (AC) or a DC power source.

2.Rectification (AC Input): If the input power is AC, the power supply uses a rectifier circuit to convert AC voltage to DC voltage.

3.Filtering: The power supply includes filtering components to smooth out the pulsating DC waveform and reduce ripple voltage.

4.Voltage Regulation: The power supply includes a voltage regulation circuit to maintain a constant output voltage of 5 volts.

5.Current Limiting: To prevent damage, the power supply typically includes current-limiting mechanisms to restrict the output current to a safe maximum level.

6.Output Protection: The power supply may include overvoltage

protection, overcurrent protection, and short-circuit protection.

7.Output Connection: The stabilized 5V output is provided through output terminals or connectors.

Connecting Components for a 6 DOF Robotic Arm:

1.Power Requirements: Each MG996R servo motor can draw up to 2.5A at stall current. For a 6 DOF arm, a 10A power supply is generally sufficient, but heavy loads can exceed this capacity.

2.Power Supply Choice: A regulated 5V 10A power supply can be directly connected to the PCA9685, which distributes power to all connected servos. Reliable options like Mean Well LRS-50-5 are recommended for their quality and protections.

3.Power Distribution: Using a capacitor (e.g., a 4700uF 16V capacitor) across the power lines can help mitigate voltage drops and protect against inrush currents.

4.Grounding: Ensure all components share a common ground to avoid grounding issues that could lead to erratic behavior.

5.Voltage Regulation: If you experience issues with power stability, consider using additional capacitors on the power rails or a dedicated voltage regulator.

By addressing these factors, you can effectively power your 6 DOF robotic arm with an Arduino Mega and PCA9685 using a 5V 10A power supply, ensuring stable and reliable operation.

Section 3.1.6: Joy Stick:

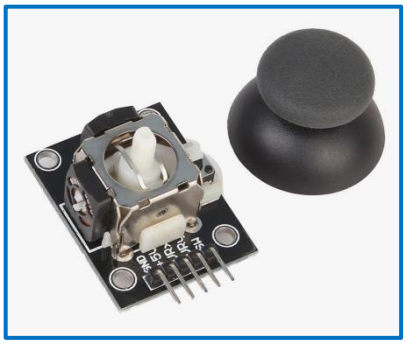


Figure – 3.1.6.1- Joy Stick

Definition and Basic Components:

A joystick is an input device used in gaming and simulations to control the movement of an on-screen object, such as a character or a cursor. It consists of a handheld lever or stick that can be tilted in various directions and often includes one or more buttons for additional functions. *(Figure-3.1.6.1)*

Components:

Lever or Stick: The main component manipulated by the user to control movement.

Base: Provides stability and may include additional buttons or controls.

Buttons: Located on the base or the stick itself, used for various actions.

Connection Interface: Connects to a computer or gaming console via wired (USB or game port) or wireless (Bluetooth) connections.

Force Feedback (Optional): Provides tactile feedback, such as vibrations, to enhance the user experience.

Applications and Benefits:

Gaming:

Control Precision: Enhances gameplay by providing precise control over character movement and camera angles.

Immersion: Improves the gaming experience with more intuitive

control.

Flight Simulation:

Realism: Simulates real aircraft controls for an authentic flying experience.

Training: Used in flight training programs and simulators for pilot training.

Industrial Applications:

Control Systems: Operates machinery, robotic arms, and cranes.

Safety: Allows operators to control hazardous machinery from a safe distance.

Accessibility:

Assistive Technology: Helps individuals with mobility impairments control computers and wheelchairs.

Customization: Adapted with specific grips and switches to accommodate users with disabilities.

Remote Control:

Remote Operation: Used in remote-controlled vehicles, drones, and UAVs for navigation and control.

Aerial Photography: Provides precise camera control for high-quality aerial photos and videos.

How Joysticks Work:

Mechanical Movement: Lever moves in multiple directions using mechanical means such as springs or potentiometers.

Potentiometers: Detect the position of the lever and convert it into electrical signals.

Analog Outputs: Represent the direction and magnitude of the joystick's movement.

Digital Buttons: Perform specific actions when pressed.

Interface: Connects to a computer or console, allowing the device to receive input signals.

Software Integration: Maps joystick inputs to actions within software applications.

Feedback (Optional): Provides tactile feedback in response to in-game events.

Joystick for 6 DOF Robotic Arm Control

Joystick Modules for Arduino:

Basic Analog Joystick Module:

Description: Includes two potentiometers for X and Y axes and a pushbutton.

Connections: VCC (5V), GND, VRx (X-axis), VRy (Y-axis), SW (pushbutton).

Use Case: Suitable for simple, low-cost applications.

PS2 Joystick Module:

Description: Based on the PlayStation 2 controller joystick, offering smooth and responsive control.

Connections: VCC, GND, VRx, VRy, SW.

Use Case: Suitable for projects requiring smoother control.

Connecting Joysticks to Arduino Mega:

Joystick 1:

VRx to A0

VRy to A1

Joystick 2:

VRx to A2

VRy to A3

Joystick 3:

VRx to A4

VRy to A5

connect Joystick to Arduino Mega:

VCC to 5V on Arduino Mega

GND to GND on Arduino Mega

VRx to A0 on Arduino Mega

VRy to A1 on Arduino Mega

The code of joy sticks

```
void joystickMode (){
  int joystick_x_value1 = analogRead(joystick_x1);
  int joystick_y_value1 = analogRead(joystick_y1);
  int joystick_x_value2 = analogRead(joystick_x2);
  int joystick_y_value2 = analogRead(joystick_y2);
  int joystick_x_value3 = analogRead(joystick_x3);
  int joystick_y_value3 = analogRead(joystick_y3);
```

```
  // Update servo positions with speed control, preventing simultaneous movement
  if (abs(joystick_x_value1 - 512) > abs(joystick_y_value1 - 512)) {
    x1_axis_degree = min(199, max(65, x1_axis_degree + (joystick_x_value1 < 340 ?
-x1_speed : (joystick_x_value1 > 680 ? x1_speed : 0))));
  } else {
    y1_axis_degree = min(150, max(90, y1_axis_degree + (joystick_y_value1 < 340 ?
-y1_speed : (joystick_y_value1 > 680 ? y1_speed : 0))));
  }

  if (abs(joystick_x_value2 - 512) > abs(joystick_y_value2 - 512)) {
    x2_axis_degree = min(90, max(0, x2_axis_degree + (joystick_x_value2 < 340 ?
x2_speed : (joystick_x_value2 > 680 ? -x2_speed : 0))));
  } else {
    y2_axis_degree = min(179, max(90, y2_axis_degree + (joystick_y_value2 < 340 ?
y2_speed : (joystick_y_value2 > 680 ? -y2_speed : 0))));
  }

  if (abs(joystick_x_value3 - 512) > abs(joystick_y_value3 - 512)) {
    x3_axis_degree = min(179, max(0, x3_axis_degree + (joystick_x_value3 < 340 ?
-x3_speed : (joystick_x_value3 > 680 ? x3_speed : 0))));
  } else {
    y3_axis_degree = min(179, max(120, y3_axis_degree + (joystick_y_value3 < 340
? y3_speed : (joystick_y_value3 > 680 ? -y3_speed : 0))));
  }
  // Set servo positions
  pwm.setPWM(15, 0, angleToPulse(x1_axis_degree));
  pwm.setPWM(14, 0, angleToPulse(y1_axis_degree));
  pwm.setPWM(13, 0, angleToPulse(x2_axis_degree));
  pwm.setPWM(12, 0, angleToPulse(y2_axis_degree));
  pwm.setPWM(11, 0, angleToPulse(x3_axis_degree));
  pwm.setPWM(10, 0, angleToPulse(y3_axis_degree));

  delay(10); // Increase delay to slow down servo movement
}
```

Section 3.1.7: Adaptor 9v 2a For Arduino:

Figure – 3.1.7.1- Adaptor 9v 2a For Arduino

What Is a 9V 2A Adapter for Arduino? *(Figure-3.1.7.1)*

1.Voltage Compatibility:

- Arduino boards generally operate within a range of voltages, with many models designed to be powered by a 9V input. A 9V power adapter is

suitable for directly powering these boards.

2.Current Capacity:

- The 2A current rating indicates that the power adapter can supply up to 2 amperes of current to connected devices. This should be sufficient for most Arduino projects, but it’s essential to ensure the adapter can meet the power requirements of your specific project.

3.Connector Compatibility:

- The adapter typically comes with a barrel connector or other compatible types that can be plugged into the power input jack on the Arduino board. Ensure that the connector size and polarity match the Arduino's requirements to avoid damage.

4.Stability and Reliability:

- A stable and reliable power supply is crucial for Arduino projects. A quality power adapter with the correct voltage and current rating ensures consistent performance and prevents issues such as erratic behavior or component damage.

Uses and Benefits of a 9V 2A Adapter for Arduino

1.Powering Arduino Boards:

- The primary use of a 9V 2A power adapter is to provide a stable and reliable power source for Arduino boards, ensuring proper operation of the board and its peripherals.

2.Compatibility:

- This adapter is compatible with a wide range of Arduino boards, including the Arduino Uno, Mega, and Nano, making it a versatile

choice for various projects and applications.

3.High Current Capacity:

- With a 2A rating, the adapter can supply sufficient current to power the Arduino board along with attached peripherals like sensors, motors, LEDs, and communication modules. This ensures the adapter can meet most project demands without overheating or voltage drop issues.

4.Reliable Performance:

- The adapter delivers stable and consistent voltage output, essential for the proper operation of Arduino boards and connected devices. This helps prevent issues such as erratic behavior, resets, or damage due to voltage fluctuations or power surges.

5.Convenience:

- Using a dedicated power adapter eliminates the need for batteries or makeshift power supplies, providing a convenient and hassle-free solution for powering Arduino projects.

6.Safety:

- The adapter typically includes built-in safety features like overvoltage, overcurrent, and short-circuit protection to ensure the safety of both connected devices and the user.

How the 9V 2A Adapter for Arduino Works

1.AC to DC Conversion:

- If the power adapter accepts AC input, it first converts the alternating current (AC) from the mains power outlet into direct current (DC) using a rectifier circuit.

2.Voltage Regulation:

- The adapter includes voltage regulation circuitry to ensure the output voltage remains stable at 9V, providing consistent voltage regardless of input power fluctuations.

3.Current Limiting:

- The adapter limits the maximum output current to 2A to prevent overloading and ensure safe operation, protecting both the adapter and connected devices.

4.Output Connection:

- The stabilized 9V output is provided through output terminals or connectors, allowing connection to the Arduino board's power input jack. The output connector is designed to ensure compatibility with the Arduino board.

5.Safety Features:

- Built-in safety features such as overvoltage, overcurrent, and short-circuit protection prevent damage in the event of electrical faults or malfunctions.

6.Usage:

- Once connected, the adapter supplies stable 9V DC to the Arduino board, powering the microcontroller, sensors, communication modules, and other peripherals.

Connecting the 9V 2A Adapter to Arduino Mega

1.Voltage and Current Compatibility:

- The Arduino Mega can be powered by an external source of 7-12V. A 9V 2A adapter fits within this range and will not overload the board.

2.Connector Type:

- The adapter should have a 5.5mm diameter cylindrical plug with a 2.1mm pin hole, providing positive voltage on the inside pin and negative on the outer sleeve.

3.Power Distribution:

- If other components require 9V power, use the same adapter. Ensure that the total current draw does not exceed the adapter's 2A capacity.

Steps to Connect:

1.Connecting the Adapter:

- Plug the 9V 2A adapter into the Arduino Mega's power jack. Ensure the adapter is plugged into a stable power outlet.

2.Additional Components:

- Use a breadboard or distribution block to split the power from the adapter if powering additional components. Connect the positive terminal to the VIN pin and the ground to the GND pin on the Arduino Mega. Ensure all ground connections are common.

Section 3.1.8: Gear Motor:

What Is a Gear Motor?

A gear motor is a combination of a motor and a gearbox, designed to reduce the

Figure – 3.1.8.1-Gear Motor

speed and increase the torque of the motor output. The gearbox is typically composed of a series of gears that effectively reduce the rotational speed of the motor while increasing its torque. *(Figure-3.1.8.1)*

Uses, Benefits, and Applications of Gear Motors

Industrial Automation:

Gear motors are extensively used in industrial automation for driving conveyor belts, robotic arms, and other machinery that require precise control of speed and torque.

Robotics:

In robotics, gear motors are used to control the movement of robotic joints and wheels, providing the necessary torque for heavy lifting and precise movements.

Automotive Applications:

Gear motors are used in automotive systems for powering windshield wipers, seat adjusters, and window regulators, where controlled movement and torque are crucial.

Consumer Electronics:

They are also found in consumer electronics such as electric shavers, where precise control of speed and torque is necessary.

Benefits:

Increased Torque: Gear motors provide high torque output, making them ideal for applications requiring significant force.

Speed Reduction: They allow for precise control of speed, which is essential in applications requiring careful movement control.

Compact Design: Combining the motor and gearbox into a single unit saves space and simplifies installation.

How Gear Motors Work

Gear motors work by using a gearbox to reduce the speed of the motor's output shaft while increasing the torque. The motor provides rotational energy, which is transmitted through the gears in the gearbox. Each gear reduces the speed and increases the torque proportionally. The output shaft of the gearbox then delivers the modified speed and torque to the connected application.

Motor: The electric motor generates rotational movement.

Gearbox: The gears within the gearbox reduce the speed and increase the torque of the motor's output.

Output Shaft: The final shaft connected to the gears delivers the adjusted speed and torque to the application.

Connecting Gear Motors to Arduino Mega via L298D H-Bridge

Components Needed:

Arduino Mega

L298D H-Bridge

Gear Motor

Power Supply

Jumper Wires

Circuit Diagram:

Power Connections:

Connect the 12V power supply to the VCC input of the L298D H-Bridge.

Connect the ground (GND) of the power supply to the GND of the L298D and the Arduino Mega.

Motor Connections:

Connect the output terminals of the L298D (OUT1 and OUT2) to the terminals of the gear motor.

Control Connections:

Connect the IN1 and IN2 pins of the L298D to two digital PWM pins on the Arduino Mega (e.g., pin 8 and pin 9).

Optionally, connect the Enable pin (EN1) to a PWM-capable pin on the Arduino (e.g., pin 10) for speed control.

Simple code

```
const int motorPin1 = 8;  // Pin for IN1
const int motorPin2 = 9;  // Pin for IN2
const int enablePin = 10; // Pin for EN1 (optional for speed control)

void setup() {
  pinMode(motorPin1, OUTPUT);
  pinMode(motorPin2, OUTPUT);
  pinMode(enablePin, OUTPUT);
}

void loop() {
  digitalWrite(motorPin1, HIGH);  // Rotate motor in one direction
  digitalWrite(motorPin2, LOW);
  analogWrite(enablePin, 255);    // Set motor speed (0-255)

  delay(2000);                    // Run for 2 seconds

  digitalWrite(motorPin1, LOW);   // Stop motor
  digitalWrite(motorPin2, LOW);
  analogWrite(enablePin, 0);
```

```
  delay(1000);                    // Pause for 1 second

  digitalWrite(motorPin1, LOW);   // Rotate motor in opposite direction
  digitalWrite(motorPin2, HIGH);
  analogWrite(enablePin, 255);

  delay(2000);                    // Run for 2 seconds

  digitalWrite(motorPin1, LOW);   // Stop motor
  digitalWrite(motorPin2, LOW);
  analogWrite(enablePin, 0);

  delay(1000);                    // Pause for 1 second
}
```

Section 3.1.9: L293d H- Bridge:

What Is L293d H- Bridge?

The L293d Is A Popular Integrated Circuit (Ic) Commonly Used As An H-Bridge Motor Driver. An H-Bridge Is A Circuit That Allows You To Control The Direction And Speed Of A Dc Motor By Controlling The Polarity And Magnitude Of The Voltage Applied To The Motor Terminals. *(Figure-3.1.9.1)* Here's An Overview Of The L293d H-Bridge:

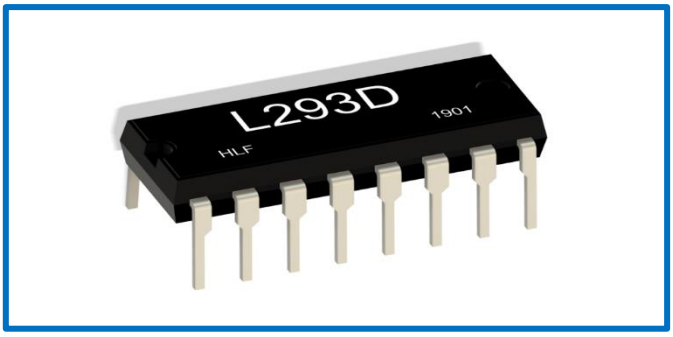


Figure – 3.1.9.1- L293d H-Bridge

1. Functionality: The L293d Ic Contains Two H-Bridge Circuits, Allowing It To Control The Movement Of Two Dc Motors Independently. Each H-Bridge Consists Of Four Transistors Arranged In A Configuration That Allows Bidirectional Current Flow Through The Motor.

2. Motor Control: By Controlling The Input Signals To The L293d Ic, You Can Control The Direction And Speed Of The Connected Dc Motors. The Ic Accepts Two Input Signals Per Motor: One For Controlling The Direction (Forward Or Reverse) And Another For Controlling The Speed (Pwm Signal).

3. Input Pins: The L293d Has Multiple Input Pins For Each Motor Channel. For Each Motor, There Are Two Control Pins For Direction Control (E.G., In1 And In2) And One Control Pin For Speed Control (E.G., Ena). By Varying The Signals On These Input Pins, You Can Achieve Different Motor Behaviors, Such As Forward, Reverse, Braking, Or Coasting.

4. Output Pins: The L293d Provides Output Pins That Connect To The Terminals Of The Dc Motors. These Output Pins (E.G., Out1, Out2 For One Motor) Deliver The Voltage And Current Necessary To Drive The Motors Based On The Input Signals.

5.Protection Features: The L293d Includes Built-In Protection Features To Prevent Damage To The Ic And Connected Components. These Features May Include Thermal Shutdown, Overcurrent Protection, And Diodes For Reverse Voltage Protection.

6. Versatility: The L293d H-Bridge Is Commonly Used In Various Robotics, Automation, And Hobbyist Projects Where Precise Control Of Dc Motors Is Required. It Offers A Simple And Cost-Effective Solution For Driving Dc Motors Bidirectionally With Speed Control.

7. Compatibility: The L293d Ic Is Compatible With A Wide Range

Of Microcontrollers, Such As Arduino, Raspberry Pi, And Other Popular Development Platforms. It Can Be Easily Interfaced With These Microcontrollers To Integrate Motor Control Into Electronic Projects And Prototypes.

What Is The Use And Benefit L293d H- Bridge?

The L293d H-Bridge Is A Versatile Integrated Circuit Commonly Used For Controlling The Movement Of Dc Motors In Various Applications. Here Are Some Of Its Key Uses And Benefits:

1. Motor Control: The Primary Use Of The L293d H-Bridge Is To Control The Direction And Speed Of Dc Motors. It Allows Bidirectional Control, Enabling Motors To Move Forward, Reverse, Brake, Or Coast Depending On The Input Signals Provided To The Ic.
2. Versatility: The L293d H-Bridge Is Compatible With A Wide Range Of Dc Motors, Including Brushed Dc Motors Commonly Used In Robotics, Automation, And Hobbyist Projects. It Can Handle Motor Voltages Up To 36v And Continuous Current Up To 600ma Per Channel (1.2a Peak), Making It Suitable For A Variety Of Motor Sizes And Types.
3. Ease Of Use: The L293d Ic Is Relatively Easy To Use And Requires Minimal External Components For Operation. It Comes In A Convenient Dip (Dual Inline Package) Or Soic (Small Outline Integrated Circuit) Package, Making It Easy To Integrate Into Electronic Circuits And Projects.
4. Compact Design: The L293d H-Bridge Integrates Two Independent H-Bridge Circuits Into A Single Ic Package, Allowing Users To Control The Movement Of Two Dc Motors

Simultaneously With A Single Component. This Compact Design Saves Space On Circuit Boards And Reduces The Overall Complexity Of Motor Control Systems.

5. Compatibility: The L293d H-Bridge Is Compatible With A Wide Range Of Microcontrollers And Development Platforms, Including Arduino, Raspberry Pi, And Other Popular Embedded Systems. It Can Be Easily Interfaced With These Platforms Using Digital Output Pins To Control Motor Direction And Speed.
6. Protection Features: The L293d H-Bridge Includes Built-In Protection Features Such As Thermal Shutdown And Overcurrent Protection To Prevent Damage To The Ic And Connected Components. These Protection Features Help Ensure The Reliable Operation Of Motor Control Systems In Various Environments.
7. Cost-Effective Solution: The L293d H-Bridge Is A Cost-Effective Solution For Motor Control Applications Compared To Other Motor Driver Options. Its Affordability, Combined With Its Versatility And Ease Of Use, Makes It A Popular Choice For Hobbyists, Students, And Professionals Alike.

How L293d H- Bridge Works?

The L293d H-Bridge Works By Controlling The Direction And Speed Of Dc Motors Using A Combination Of Internal Transistors And External Control Signals. Here's A Simplified Explanation Of How It Operates:

1. Internal Transistors: The L293d Ic Contains Four High-Current Half-H-Bridge Drivers, Allowing It To Control The

Direction Of Two Dc Motors Independently. Each Half-H-Bridge Consists Of A Pair Of Transistors (Usually Bipolar Junction Transistors Or Bjts) Arranged In An H-Bridge Configuration.

2. Control Signals: To Control The Movement Of The Motors, The L293d Requires Input Control Signals From A Microcontroller Or Other Control Circuitry. These Control Signals Determine The Direction (Forward Or Reverse) And Speed (Pwm Duty Cycle) Of Each Motor.

3. Direction Control: For Each Motor, The L293d Accepts Two Digital Control Signals: One For Controlling The Direction Of Rotation And Another For Enabling Or Disabling The Motor. These Control Signals Are Typically Referred To As In1, In2, And Ena For One Motor, And In3, In4, And Enb For The Other Motor.

4. Pwm Speed Control: The L293d Also Supports Pulse-Width Modulation (Pwm) Speed Control, Allowing You To Vary The Speed Of The Motors By Adjusting The Duty Cycle Of The Pwm Signal. The Ena And Enb Pins Accept Pwm Signals, Which Modulate The Effective Voltage Applied To The Motors And Control Their Speed.

5. Output Drivers: Based On The Input Control Signals, The L293d Activates The Appropriate Transistors To Drive Current Through The Connected Motors. By Activating The Transistors In Different Combinations, The L293d Can Control The Direction Of Rotation And Speed Of The Motors.

6. Motor Connections: The Dc Motors Are Connected To The Output Pins Of The L293d, Which Deliver The Voltage And

Current Necessary To Drive The Motors. The Output Pins Are Typically Labeled As Out1, Out2 For One Motor, And Out3, Out4 For The Other Motor.

7. Diodes And Protection: The L293d Includes Built-In Protection Diodes Across The Output Pins To Protect The Ic From Voltage Spikes Generated By The Motors During Operation. Additionally, The Ic May Include Other Protection Features Such As Thermal Shutdown And Overcurrent Protection.

Connecting

- Connecting the motor to L293D:
  - ✓ Connect the positive (+) drive terminal to OUTPUT1 on the L293D.
  - ✓ Connect the drive negative (-) terminal to the OUTPUT2 terminal on the L293D.
- Connecting the power adapter to the L293D:
  - ✓ Connect the positive (+) terminal of the power adapter to VCC2 on the L293D.
  - ✓ Connect the negative (-) terminal of the power adapter to the GND terminal on the L293D.
- Connecting L293D to Arduino:
  - ✓ Connect INPUT1 on the L293D to a control pin on the Arduino (such as D8).
  - ✓ Connect the INPUT2 on the L293D to another control pin on the Arduino (eg D9).
  - ✓ Connect ENABLE1 on the L293D to the PWM pin on the Arduino (eg D10).
- Logical feed connection:
  - ✓ Connect VCC1 on the L293D to the 5V logic power of the Arduino.

- ✓ Connect the GND on the L293D to ground (GND) of the Arduino.

Section 3.1.10: Bread Board:

What Is a Breadboard?

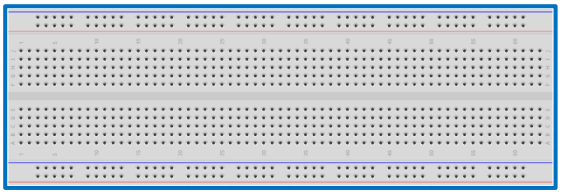


Figure – 3.1.10.1- Bread Board

A breadboard is a fundamental tool used in electronics prototyping and circuit design. It provides a platform for quickly assembling and testing electronic circuits without the need for soldering. It consists of a plastic board with a grid of holes into which electronic components and wires can be inserted to create temporary circuits. *(Figure-3.1.10.1)*

What Is the Use and Benefit of Breadboards?

Prototyping and Testing: Breadboards are primarily used for prototyping circuits rapidly. They allow engineers and hobbyists to experiment with different circuit configurations quickly and without permanent assembly.

No Soldering Required: Components are connected using spring clips or tie points inside the breadboard holes, eliminating the need for soldering. This makes it ideal for beginners and educational purposes.

Reusable: Components can be easily inserted and removed, allowing for iterative testing and modification of circuits.

Fault Finding: Breadboards facilitate easy troubleshooting and debugging of circuits by allowing quick component replacement

and circuit modification.

Applications

Educational Purposes: Breadboards are extensively used in electronics education to teach circuit design and troubleshooting.

Prototyping New Designs: Engineers use breadboards to prototype and test new circuit designs before committing to a final, soldered circuit board.

Quick Circuit Testing: Ideal for testing simple to moderately complex electronic circuits before integrating them into larger projects.

How Breadboards Work

Internal Connections: Breadboards have rows and columns of interconnected metal clips or tie points beneath the surface of the board. These points are typically organized in a grid pattern.

Connection Mechanism: Components such as resistors, capacitors, LEDs, and integrated circuits (ICs) are inserted into the breadboard holes. Each hole typically connects to others in its row or column, facilitating circuit connections.

Power Distribution: Breadboards often feature power rails along the edges (typically labeled as + and -). These rails provide convenient access to power and ground connections for the circuit.

Connecting Components on a Breadboard

Inserting Components:

Insert the leads of electronic components (resistors, LEDs, ICs, etc.) into the appropriate holes on the breadboard. Each component lead should be inserted into a separate row to avoid short circuits.

Creating Connections:

Use jumper wires to create connections between components by inserting one end of the wire into one hole and the other end into another hole where connectivity is needed.

Power Rails Usage:

Connect the power and ground rails to the appropriate power sources (such as a battery or power supply) to provide voltage and ground connections to the circuit.

Testing and Iteration:

Test the circuit by applying power and observing component behavior. Modify the circuit as needed by rearranging components and connections.

Section 3.1.11: Buttons:

What Are Buttons?

Buttons, also known as push-button switches, are mechanical or electrical devices used to make or break electrical connections. They are typically designed as momentary switches, meaning they are only active (closed) when pressed and return to their original state (open) when released. *(Figure-3.1.11.1)*

Figure – 3.1.11.1- Buttons

What Is the Use and Benefit of Buttons?

User Input Control: Buttons provide a tactile interface for users to interact with electronic devices by pressing them to initiate specific actions or functions.

Versatility: They can be used in various applications to trigger events, select modes, start/stop operations, or navigate menus in electronic systems.

Reliability: Mechanical buttons are durable and reliable, capable of enduring numerous press cycles without significant wear or malfunction.

Cost-Effectiveness: Buttons are inexpensive components, making them suitable for integrating into consumer electronics, industrial control systems, and DIY projects.

Applications

Consumer Electronics: Used in remote controls, gaming controllers, appliances, and gadgets for user interaction.

Industrial Control: Employed in control panels, machinery interfaces, and automated systems for manual input and control.

Embedded Systems: Integrated into microcontroller-based projects like Arduino for initiating program actions or responding to user inputs.

How Buttons Work

Mechanical Operation: Buttons typically consist of a plunger or actuator that, when pressed, moves to make contact between two electrical terminals inside the button housing.

Contact Closure: Pressing the button completes an electrical circuit, allowing current to flow through the connected circuitry or microcontroller input.

Debounce Mechanism: To prevent multiple false triggers from a single press (caused by mechanical bouncing), software or hardware debouncing techniques are often used to ensure reliable operation.

Connecting Buttons

Simple Connection:

Connect one terminal of the button to a digital input pin on the Arduino Mega.

Connect the other terminal of the button to ground (GND) to complete the circuit.

Pull-up or Pull-down Resistors:

Use a pull-up resistor (typically 10kΩ) connected between the digital input pin and the Arduino's 5V supply if the button is connected between the input pin and ground.

Alternatively, use a pull-down resistor if the button is connected between the input pin and 5V.

Programming Arduino Mega:

Configure the digital input pin connected to the button as an input with internal pull-up enabled in the Arduino sketch.

Use digitalRead() function to monitor the state of the button (HIGH or LOW) and trigger actions based on its state changes.

Example code

```
const int buttonPin = 2;     // Pin connected to the button
bool buttonState = false;    // Variable to store the state of the button

void setup() {
  pinMode(buttonPin, INPUT_PULLUP);  // Set buttonPin as input with internal
pull-up resistor
  Serial.begin(9600);                // Initialize serial communication
}

void loop() {
  buttonState = digitalRead(buttonPin);  // Read the state of the button

  if (buttonState == LOW) {       // Check if button is pressed (active LOW)
    Serial.println("Button pressed!");
    // Add your desired action or function call here
    delay(200);                   // Debounce delay to avoid multiple triggers
  }
}
```

Section 3.1.12: Ir Sensor:

What Is an IR Sensor?

An IR (Infrared) sensor is a device that emits and detects infrared radiation to sense objects or changes in its surroundings. It uses infrared light, which is not visible to the human eye but can be detected by specialized sensors. *(Figure-3.1.12.1)*

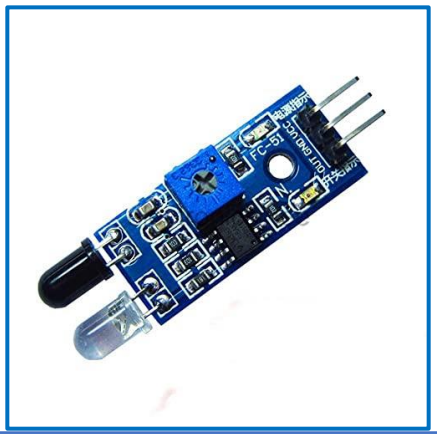


Figure – 3.1.12.1- IR Sensor

What Is the Use and Benefit of IR Sensors?

Object Detection: IR sensors are commonly used for proximity sensing and object detection without physical contact. They detect the presence or absence of objects based on their reflection or

emission of infrared light.

Non-Contact Sensing: They provide a non-contact method of sensing, making them suitable for applications where physical contact is impractical or undesirable.

Versatility: IR sensors can be used in various environmental conditions, including darkness and adverse weather, where visible light sensors may be ineffective.

Cost-Effectiveness: They are generally affordable and offer reliable performance, making them widely used in automation, robotics, security systems, and consumer electronics.

Applications

Automation: Used in industrial automation for detecting objects on conveyor belts, triggering sorting mechanisms, and monitoring production lines.

Proximity Sensing: Employed in automatic door systems, hand dryers, and sanitary equipment for touchless operation.

Security Systems: Integrated into burglar alarms, motion detection systems, and surveillance cameras for detecting intruders or unauthorized movements.

How IR Sensors Work

Emitter and Receiver: IR sensors typically consist of an IR LED (emitter) and an IR photodiode or phototransistor (receiver) placed adjacent to each other.

Operation Principle: The IR LED emits infrared light, which is reflected off nearby objects. The receiver detects the reflected light. The presence or absence of the reflected light indicates the

presence or absence of an object.

Detection Range: The detection range depends on factors such as the power of the IR LED, sensitivity of the receiver, and ambient light conditions.

Connecting IR Sensors

Basic Connections:

Connect the VCC pin of the IR sensor to a 5V power supply.

Connect the GND pin of the IR sensor to the ground (GND) of the Arduino Mega.

Output Connection:

Connect the OUT pin of the IR sensor to a digital input pin on the Arduino Mega.

Power Connections: Connect the VCC and GND pins of the IR sensor to the 5V and GND of the Arduino Mega, respectively.

Signal Connection: Connect the OUT pin of the IR sensor to a digital input pin (e.g., pin 2) on the Arduino Mega.

Programming Arduino Mega:

Configure the digital input pin connected to the IR sensor as an input in the Arduino sketch.

Use digitalRead() function to monitor the state of the IR sensor (HIGH or LOW) to detect the presence or absence of objects.

Section 3.1.13: Lcd:

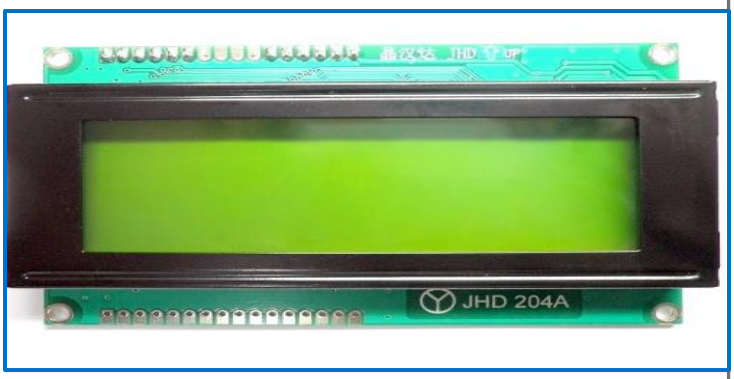


Figure – 3.1.13.1- LCD

What Is Lcd?

What Is LCD 4x20 and What Is I2C Connection?

LCD 4x20: The LCD 4x20 (Liquid Crystal Display 4 lines by 20 characters) is a type of alphanumeric display module that can display up to 4 lines of text with 20 characters each. It utilizes liquid crystal technology to display information such as text, numbers, and symbols.

I2C Connection: I2C (Inter-Integrated Circuit) is a communication protocol that allows multiple devices to communicate with each other using a two-wire serial interface. It simplifies the wiring between microcontrollers and peripherals, reducing the number of pins required for communication. *(Figure-3.1.13.1)*

What Is the Use and Benefit of LCD 4x20 and Applications?

Use and Benefits:

Information Display: LCD 4x20 modules provide a clear and readable display for presenting information in various projects, including status updates, sensor readings, and messages.

Space Efficiency: Compared to smaller displays, the 4x20 format offers more space for displaying longer messages or multiple lines of information without scrolling.

Versatility: They are widely used in educational projects, DIY electronics, instrumentation panels, and consumer electronics for displaying essential information to users.

Ease of Integration: With the I2C interface, connecting the LCD module to microcontrollers like Arduino Mega is simplified, requiring fewer GPIO pins and allowing more devices to be connected on the same bus.

How LCD 4x20 and I2C Connection Works

LCD Operation:

Display Structure: The LCD 4x20 consists of a grid of pixels controlled by a controller chip that interprets commands to display characters and symbols.

Liquid Crystal Display: Liquid crystals change their orientation in response to an electric current, allowing light to pass through or be blocked, forming characters or graphics on the screen.

I2C Communication:

Serial Communication: I2C uses two signal lines, SDA (data line) and SCL (clock line), to transmit data between the Arduino Mega and the LCD module.

Addressing: Each device connected to the I2C bus has a unique address, allowing the Arduino Mega to communicate specifically with the LCD 4x20 module.

Connecting LCD 4x20 with I2C to Arduino Mega

To connect an LCD 4x20 module with I2C to an Arduino Mega:

Hardware Connections:

VCC and GND: Connect VCC and GND of the LCD module to 5V and GND of the Arduino Mega, respectively.

I2C Interface: Connect the SDA pin of the LCD module to the

SDA pin (usually A4) and the SCL pin to the SCL pin (usually A5) on the Arduino Mega.

Example Circuit Diagram:

Power Connections: Ensure proper voltage levels (5V for VCC and GND for ground) are maintained.

I2C Connections: Connect the SDA and SCL lines to the corresponding pins on the Arduino Mega.

Arduino Sketch Configuration:

Include the appropriate I2C library for the LCD module in your Arduino sketch.

Initialize the LCD module and specify its I2C address in the setup() function.

Example code

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
// Set the LCD address to 0x27 for a 20 chars and 4 lines display
LiquidCrystal_I2C lcd(0x27, 20, 4);
void setup() {
  // Initialize the LCD with the I2C address and dimensions
  lcd.begin();
  lcd.backlight();  // Turn on the backlight
  lcd.setCursor(0, 0);  // Set cursor to the first position
  lcd.print("Hello, World!");  // Display initial message
}

void loop() {
  // Example loop code, modify as needed}
```

Section 3.2: Simulation

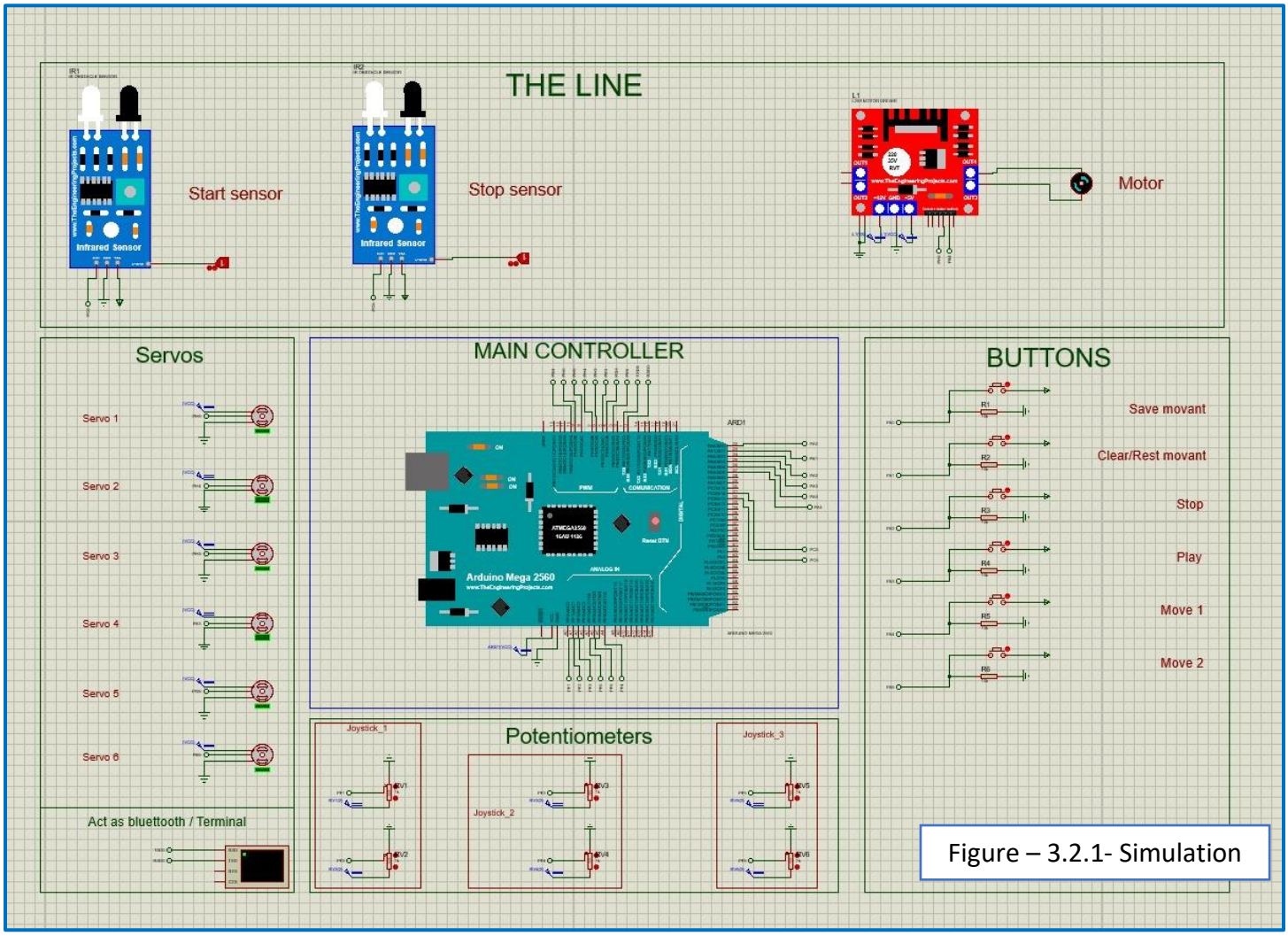


Figure – 3.2.1- Simulation

Make a simulation of the circuit to organize it and understand how each piece is connected to the other in order to avoid any failure or disorganization of the wires and pieces when the actual installation of the hardware components. We did the simulation on Proteus. *(Figure-3.2.1)*

Section 3.3: Application

First, we used Figma to design the user interface *(Figure-3.3.1:3)*:

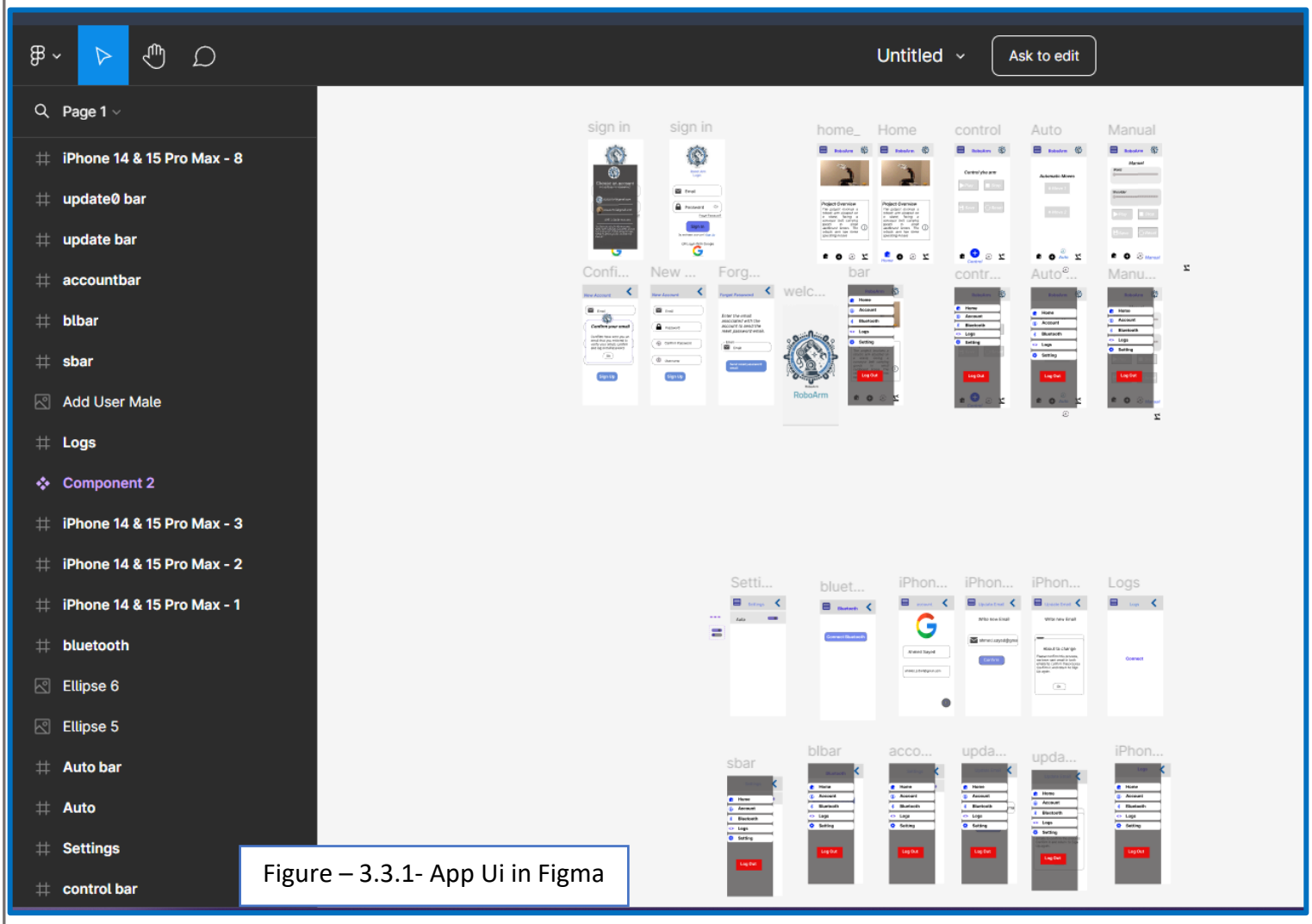


Figure – 3.3.1- App Ui in Figma

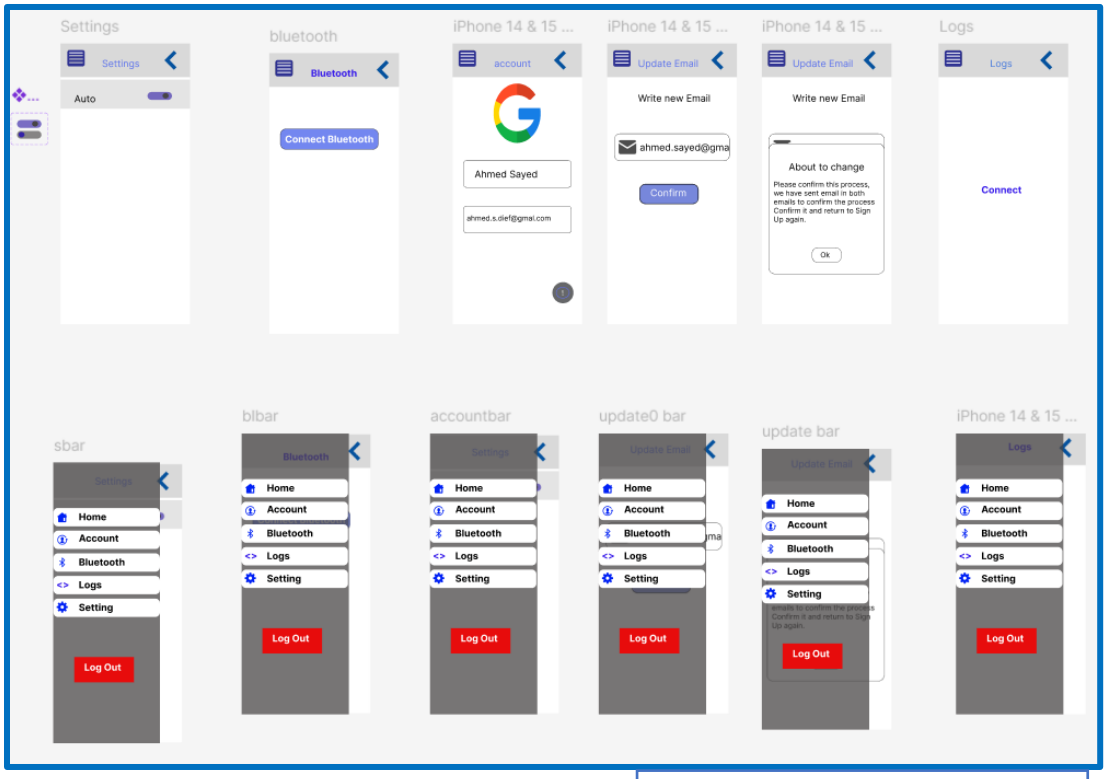


Figure – 3.3.2- App Ui in Figma

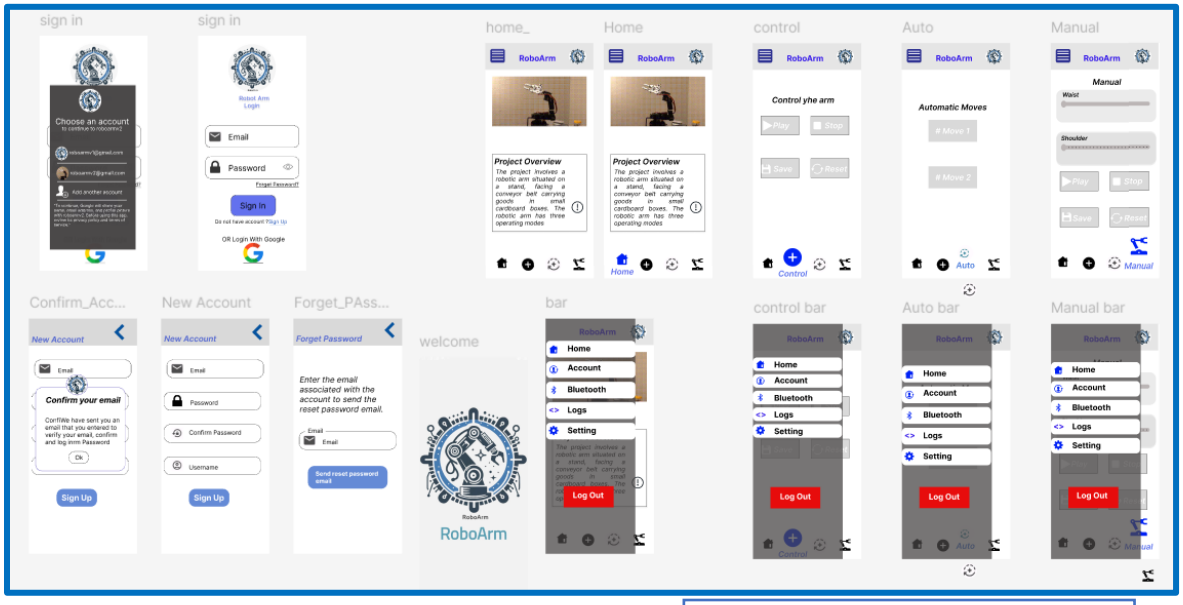


Figure – 3.3.3- App Ui in Figma

Then we made the application using Flutter because through them we can run the application on several platforms, but we only ran it completely on Android and partially on Windows.

Requirements that must be done first before starting the application programming:

- Flutter packages: There are some packages that we will use in the project. We download the packages through the Pub.dev website using (flutter pub add package-name) command. The packages are:
  - Flutter Launcher Icons: A command-line tool that simplifies the task of updating your Flutter app's launcher icon. Fully flexible, allowing you to choose what platform you wish to update the launcher icon for and if you want, the option to keep your old launcher icon in case you want to

revert sometime in the future, its dependences written in pubspec.ymal file:

```yaml
flutter_launcher_icons:
  android: "roboarmlogo"
  ios: false
  image_path: "images/roboarmlogo.png"
  min_sdk_android: 21 # android min sdk min:16, default 21
  web:
    generate: true
    image_path: "images/roboarmlogo.png"
    #background_color: "#hexcode"
    #theme_color: "#hexcode"
  windows:
    generate: true
    image_path: "images/roboarmlogo.png"
    icon_size: 48 # min:48, max:256, default: 48
```

- Supabase Flutter: Flutter integration for Supabase. This package makes it simple for developers to build secure and scalable products.
- Shared Preferences: Wraps platform-specific persistent storage for simple data (NSUserDefaults on iOS and macOS, SharedPreferences on Android, etc.). Data may be persisted to disk asynchronously, and no guarantee that writes will be persisted to disk after returning, so this plugin must not be used for storing critical data.

- Google Sign In: Flutter plugin for Google Sign-In, a secure authentication system for signing in with a Google account.
- Desktop Window: Flutter plugin for Flutter desktop(macOS/Linux/Windows) to change window size.
- Permission Handler: On most operating systems, permissions aren't just granted to apps at install time. Rather, developers have to ask the user for permission while the app is running. This plugin provides a cross-platform (iOS, Android) API to request permissions and check their status. You can also open the device's app settings so users can grant permission. On Android, you can show a rationale for requesting permission.

- Supabase: After we register normally on the site, we create a new project. After that:

- We take the Project URL and API Key and add it to the flutter project in the main file. *(Figure-3.3.4)*

Figure – 3.3.4-Project API

- With this, we have linked the flutter project with the supabase project.
- After that, we only want to use authentication, so we will go to its specific window, which is in the Dashboard, then activate it, then in its CONFIGURATION window, we go to Providers and enable the Email provider so that the user can log in using an email and a password, and we also activate the Sign in with Google also so

that the user can log in using a Google account directly. *(Figure-3.3.5,6)*

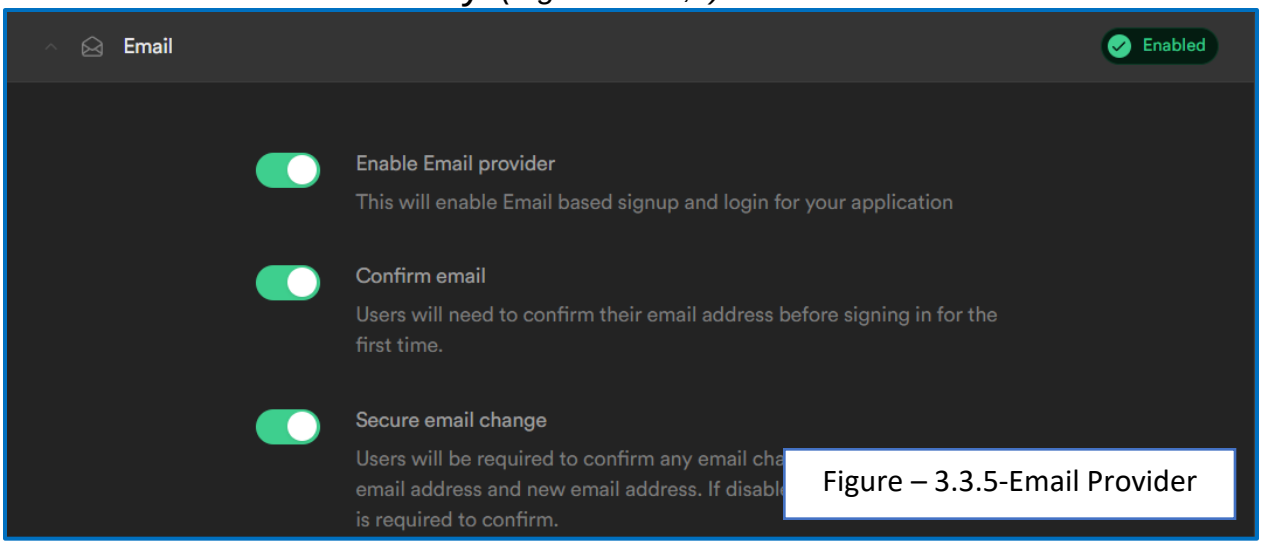


Figure – 3.3.5-Email Provider

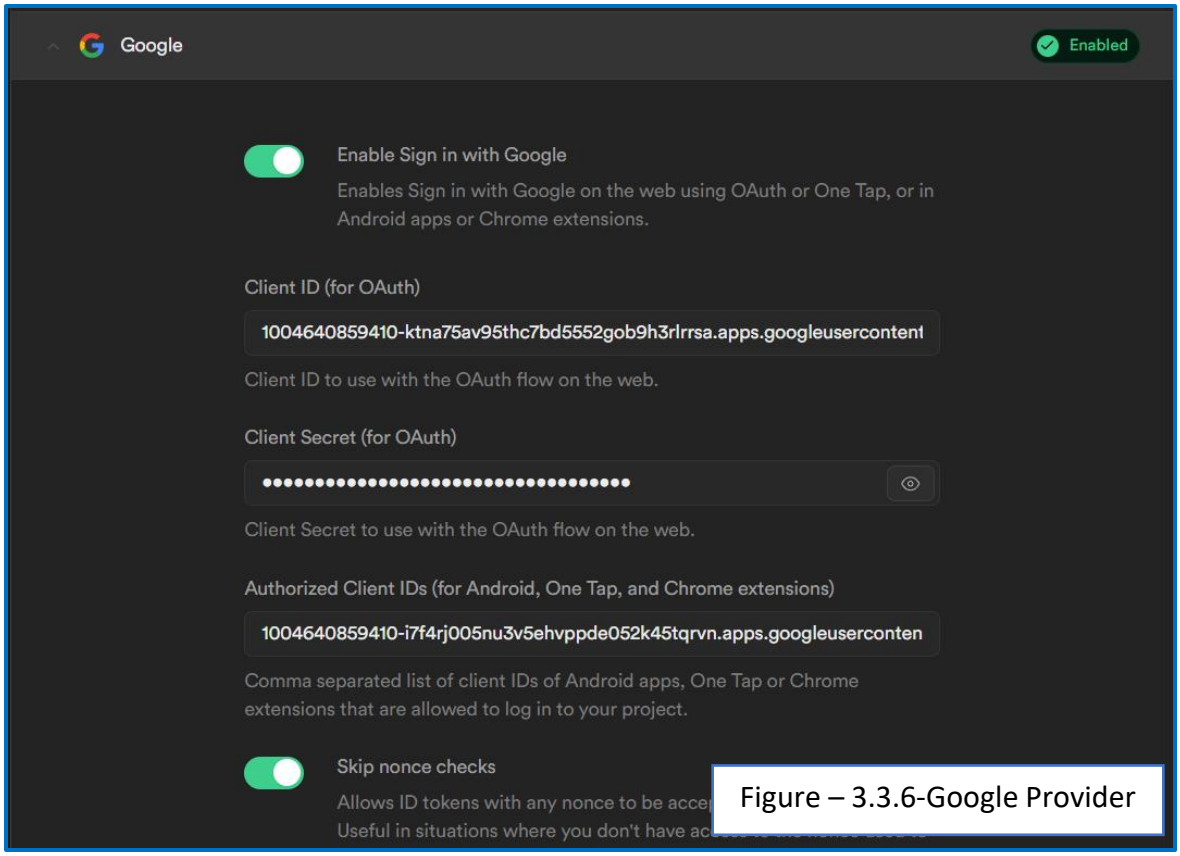


Figure – 3.3.6-Google Provider

- Google Cloud: In Google Cloud we only used credentials to make sure that our app is trusted and any Google user can register through it.

- We create a new project, then search for the credentials, add them to the project, then create three new credentials through the OAuth 2.0 Client.*(Figure-3.3.7)*

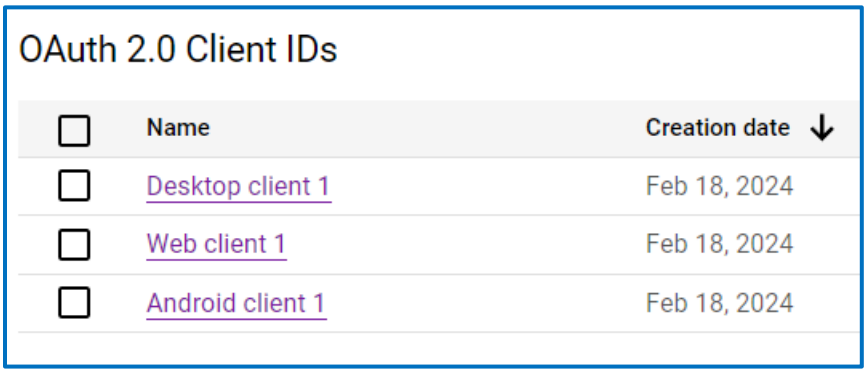


Figure – 3.3.7- OAuth2.0 Client

- The first will be for the application on Windows, in which we will add the Client ID (for OAuth) and Client Secret (for OAuth) that were present in Sign in with Google provider in the supabase. *(Figure-3.3.8)*

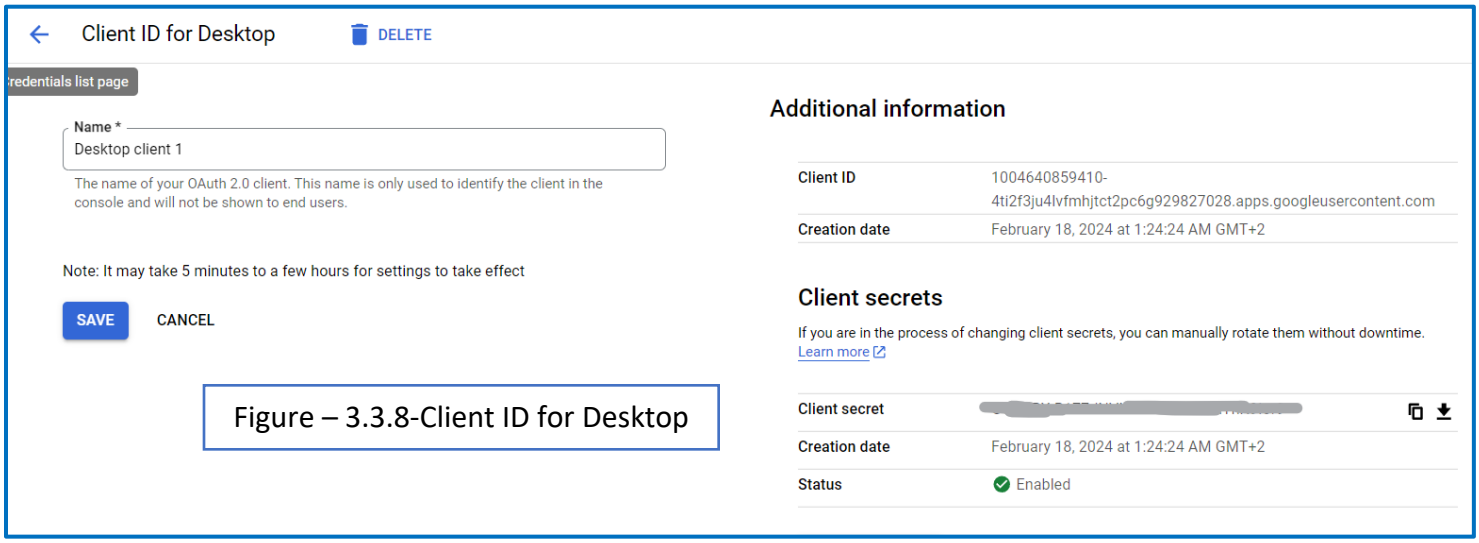


Figure – 3.3.8-Client ID for Desktop

- The second is for Android, in which we enter the Package name of the project and the SHA-1 certificate fingerprint. You will find the Package name in roboarmv2/android/app/src/build.gradle, and you can come up with the SHA-1 certificate fingerprint of the project through the command (keytool - keystore path-to-debug-or-production-keystore -list -v). *(Figure-3.3.9)*

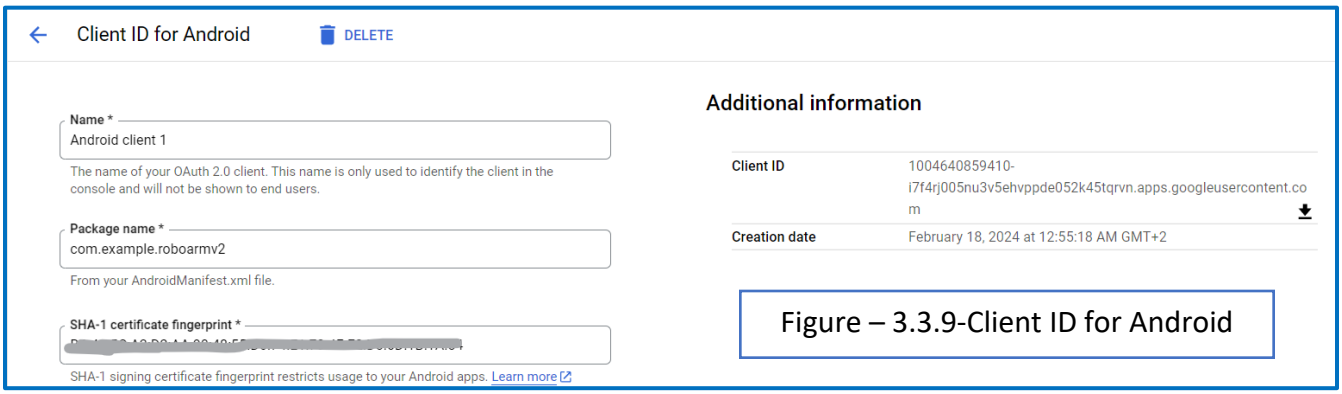


Figure – 3.3.9-Client ID for Android

- The Third for web it required also for work in android, with make the same thing in windows desktop. *(Figure-3.3.10)*

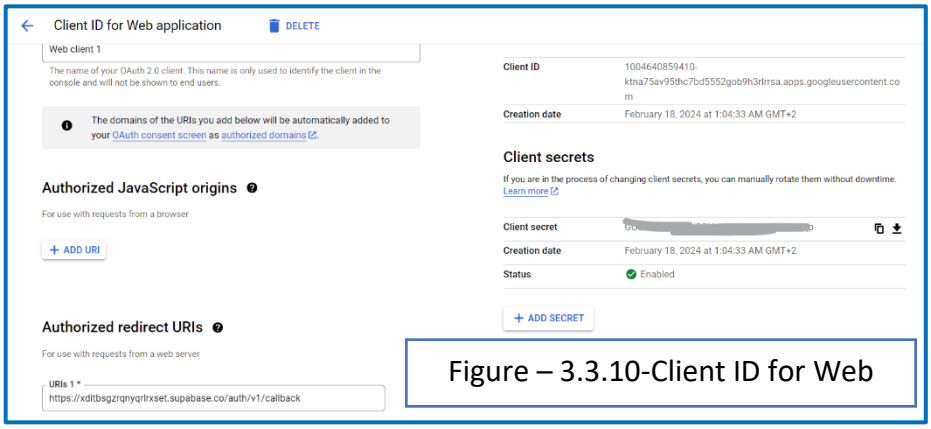


Figure – 3.3.10-Client ID for Web

- Brevo: We register normally on the site, then we go to Settings, then SMTP & API:
    - We create a new SMTP key, and keep (SMTP Server, Port, Login, SMTP key value). *(Figure-3.3.11)*

Figure – 3.3.11-Brevo SMTP

- After that, we return to Supabase and go to the project settings, then Auth settings. We go to SMTP Settings, activate it, add the data for the previous SMTP Server, and then finally save all of that. *(Figure-3.3.12)*

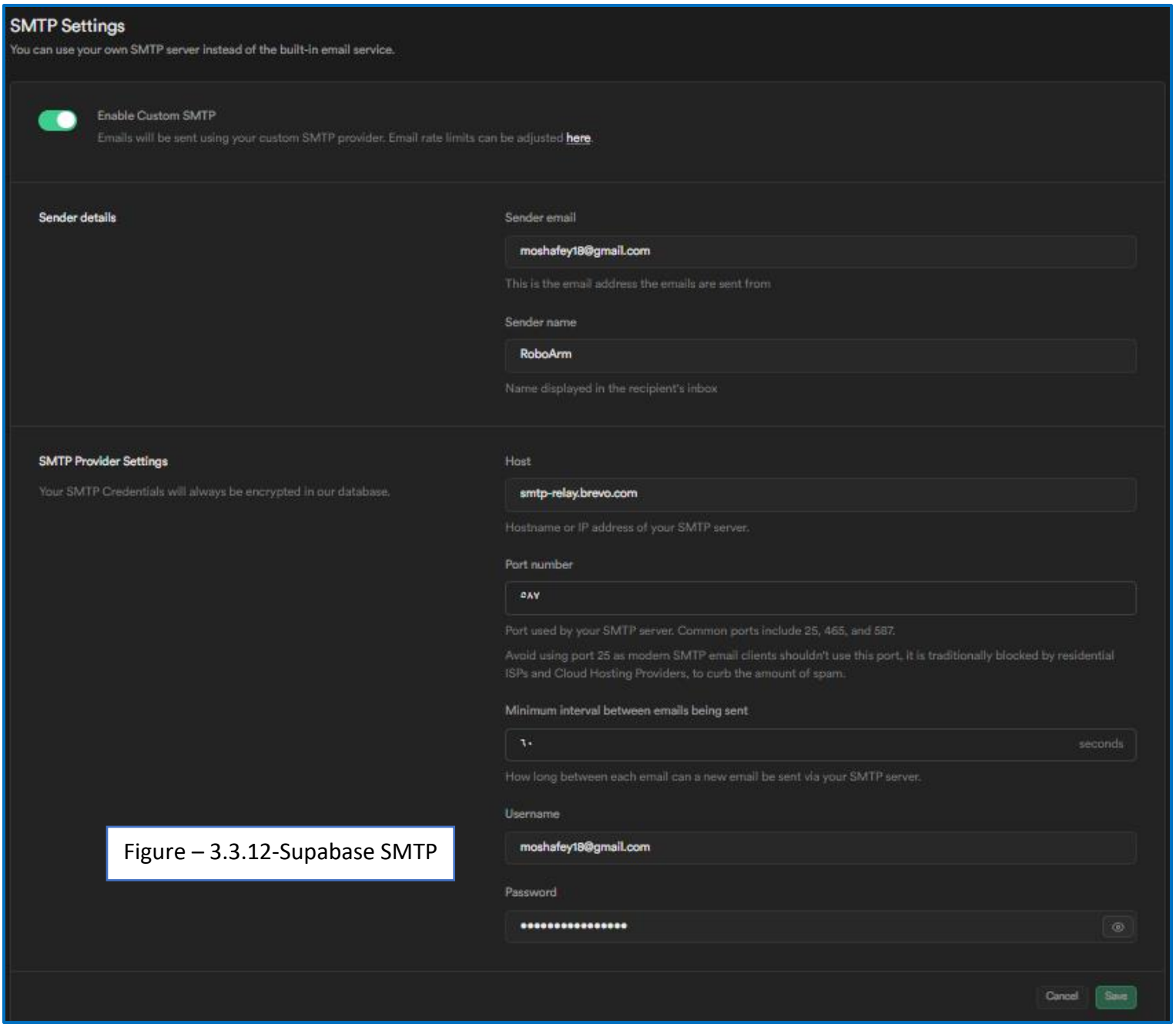


Figure – 3.3.12-Supabase SMTP

- Firebase: I previously mentioned that we used Firebase for two things. The first is to host a website for the project details, and the second is to host the pages related to two-factor authentication. Currently, we will only create a new project on Firebase in order to host the authentication web pages, then we will install the hosting on it and upload the files for the pages. *(Figure-3.3.13)*

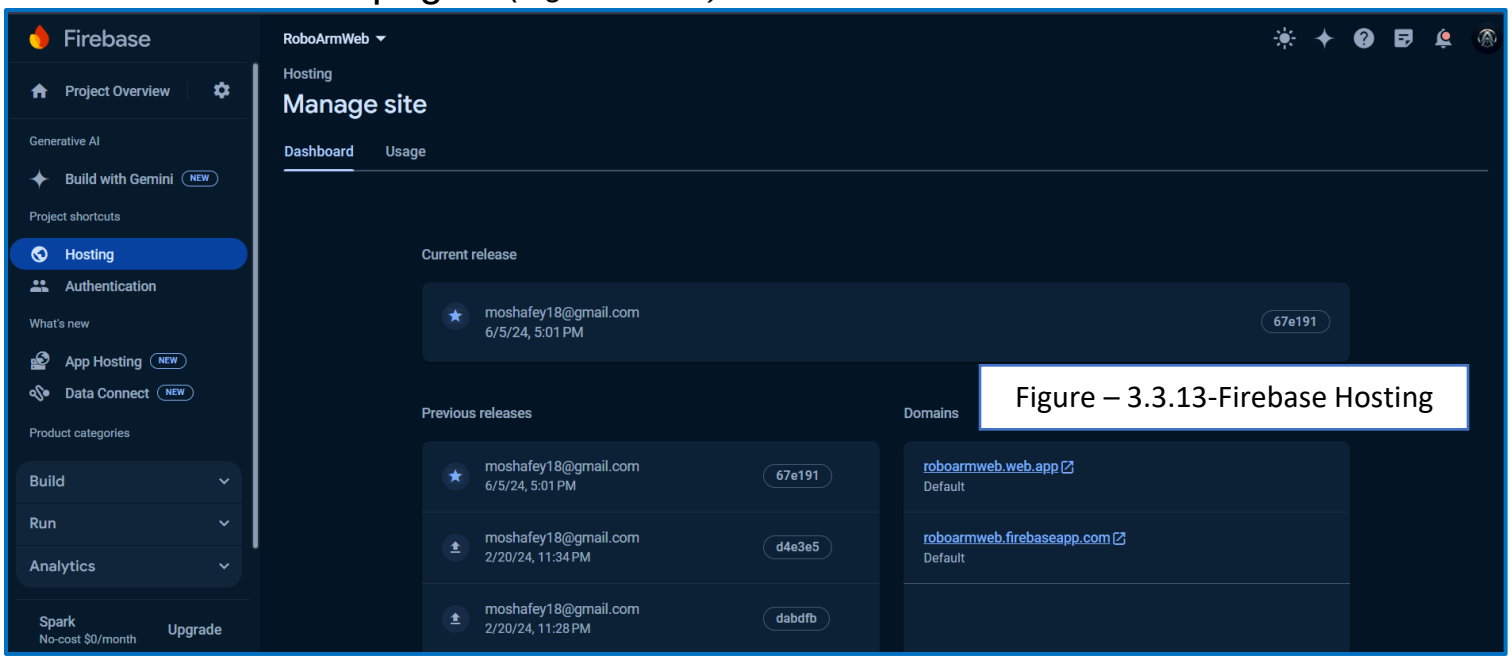


Figure – 3.3.13-Firebase Hosting

Now how does the application work (a special explanation for each file in the flutter project) *(The codes are in Section 3.4.2 and 3.4.3)*:

- Colors: The colors.dart file defines custom colors used throughout the Flutter application. These constants

provide a consistent color scheme throughout the application, enhancing visual coherence and user experience. By centralizing color definitions in a separate file, it becomes easier to manage and update the color scheme of the application.

- main: It serves as the entry point for the Flutter application. It begins by importing necessary Dart and Flutter packages, including ones for desktop window management and Supabase backend services. Within the file, there's a function to adjust window size for Windows platforms, ensuring optimal display. Following this, Supabase is initialized with specific project details for backend operations like database interactions and authentication. Finally, the main function runs the app by instantiating the MyApp widget, which sets up the application's structure, including the initial screen (a splash screen) and a custom theme. Overall, main.dart orchestrates the initialization of essential components and the launch of the Flutter application, laying the foundation for its functionality and user interface.
- Splach: The splach.dart file defines a StatefulWidget called Splash, which represents the splash screen of

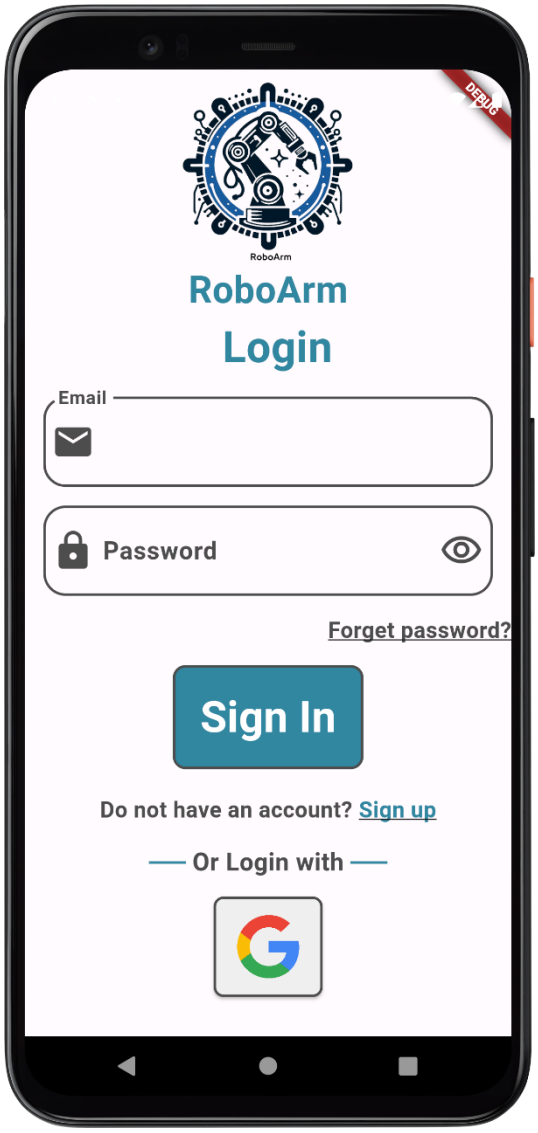
our application. When the Splash widget is created, it initializes a state _SplashState. The initState method is overridden to call the checkSession function, which checks if a user session exists by retrieving a session token from shared preferences. If a session token is found, the user is redirected to the home screen (home()), and if not, the user is redirected to the login screen (Login()). This redirection is done using Navigator.pushReplacement, which replaces the current route with the new route, preventing the user from returning to the splash screen using the back button. The build method returns a Scaffold widget with a body containing a Container widget. Inside the container, there's a Center widget with a Column containing an image of the RoboArm logo and the text "RoboArm" styled with primary colors a nd a bold font. This creates the visual layout of the splash screen, presenting

Figure – 3.3.14- App Ui/UX

branding elements to the user while the app initializes in the background. Overall, splash.dart orchestrates the display of the splash screen, checks for an existing user session, and navigates the user to the appropriate screen based on the session status. It also provides a visually appealing introduction to the application's branding and identity. *(Figure-3.3.14)*

- login: The `login.dart` file defines a `Login` StatefulWidget, representing the login screen of our application. Within this widget, the `_LoginState` class manages the state of the login screen. The screen allows users to sign in using email and password or with a Google account. For email and password authentication, the user's input is validated, and if valid, the user is signed in using Supabase's authentication service. Upon successful authentication, the user is redirected to the home screen (`home()`). In case of invalid credentials, an error dialog is displayed. For Google sign-in, the `_googleSignInAndNavigate` function handles sign-in for Android platforms, while the `signInWithGoogleOnDesktop` function handles sign-in for Windows platforms. After signing in with Google,

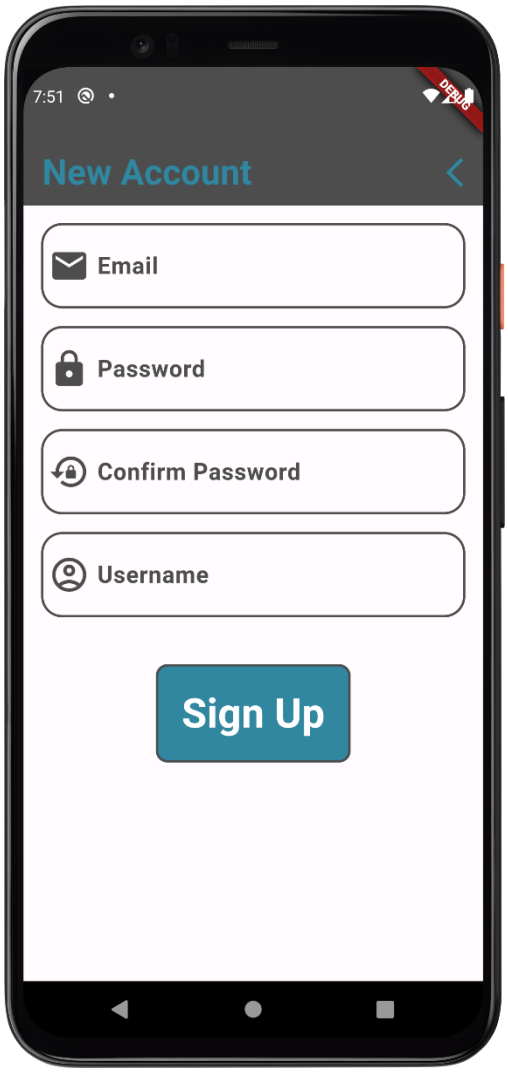
the user's session token is saved to shared preferences, and the user is redirected to the home screen. The visual layout of the login screen includes branding elements such as the RoboArm logo and text. Text input fields for email and password are provided, with validation and visibility toggling for the password field. Additionally, options for password recovery and account registration (via the sign-up screen) are provided below the login form. To facilitate Google sign-in, the login screen also includes a button with the Googale logo, allowing users to sign in using their Google accounts. This button is platform-aware, displaying different behaviors based on the platform (Android or Windows). Overall, `login.dart` manages user authentication through email/password and Google sign-in, providing a



visually appealing and functional login in terface for the application.

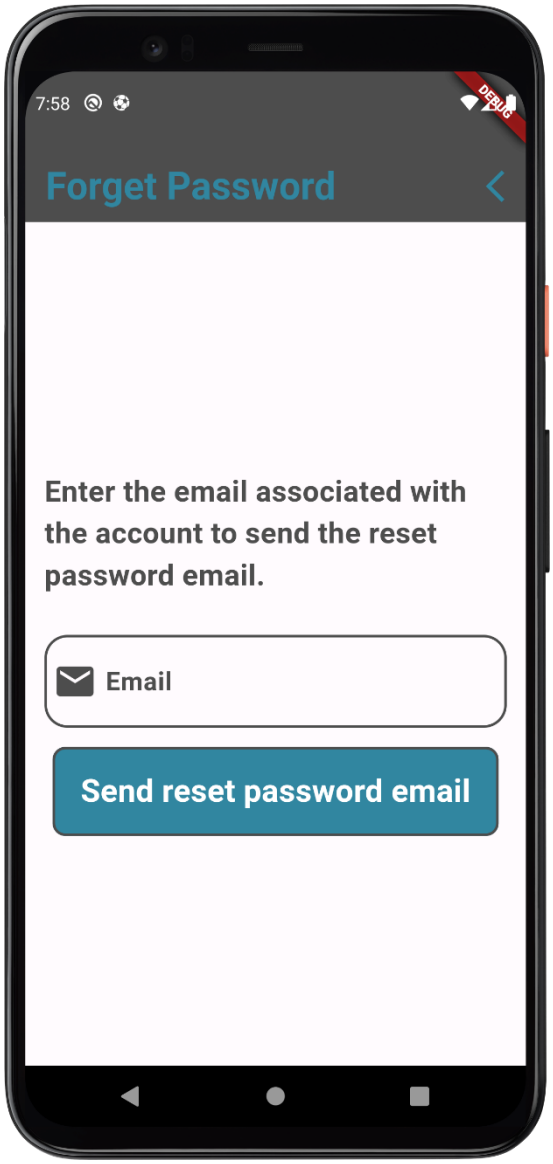
- Signup: The `signup.dart` file defines a `SignUp` StatefulWidget, representing the sign-up screen of our application. Within this widget, the `_SignUpState` class manages the state of the sign-up screen. The screen allows users to create a new account by providing their email, password, confirm password, and username.The visual layout of the sign-up screen includes an app bar with a back button for navigation. Below the app bar, there's a form with text input fields for email, password, confirm password, and username. Each field has its validation logic to ensure that the user provides valid input. For example, the email field checks if the email format is valid, the password field checks if the password is at least 8 characters long and contains at least one uppercase letter, one lowercase letter, one digit, and one special character. Once the user fills in the required information and taps the "Sign Up" button, the entered data is validated. If the password and confirm password match, the `supabase.auth.signUp` method is called to create a new account with Supabase. If

successful, the user is notified to confirm their email address through a dialog and then redirected to the login screen (`Login()`). If an error occurs during the sign-up process, an error dialog is displayed with details of the error. Overall, `signup.dart` provides a user-friendly interface for creating a new account within the application and seamlessly integrates with Supabase for user authentication and data management.



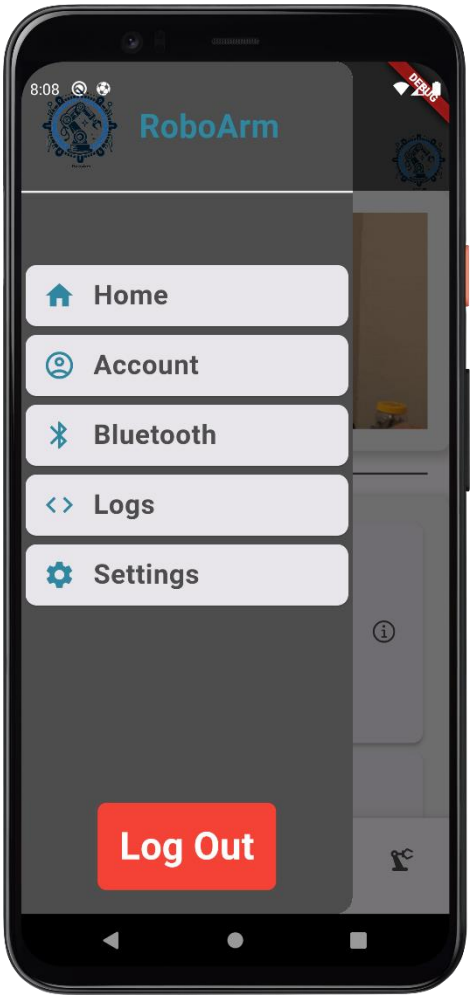
- forgetpassword: The `forgetpassword.dart` file defines a `forgetpassword` StatefulWidget, representing the password reset screen of our application. Within this widget, the `_ForgetPasswordState` class manages the state of the password reset screen. The screen allows users to request a password reset email by providing their email address. The visual layout of the password reset

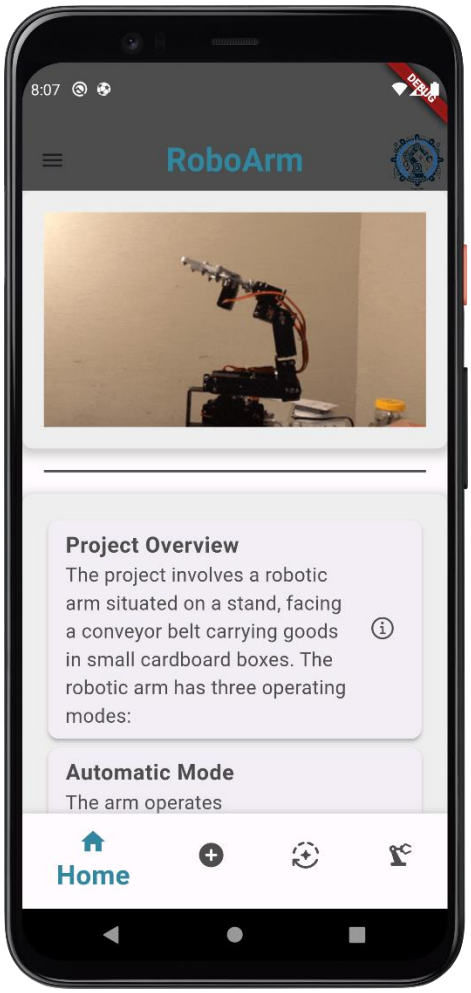
screen includes an app bar with a back button for navigation. Below the app bar, there's a message prompting the user to enter the email associated with their account to receive a reset password email. A text input field for the email address and a button to submit the password reset request are provided. When the user taps the "Send reset password email" button, the entered email address is validated. If the email field is not empty, a password reset email is sent using the `supabase.auth.resetPasswordForEmail` method. Additionally, if a session token exists in shared preferences (indicating the user is logged in), it is removed, and the user is signed out to ensure a secure reset process. Finally, a dialog informs the user that the reset email has been sent, and they are



prompted to return to the login screen. If the email field is empty, an error dialog is displayed prompting the user to enter their email address before submitting the password reset request. Overall, `forgetpassword.dart` provides a user-friendly interface for initiating the password reset process within the application, seamlessly integrating with Supabase for user authentication and email communication.

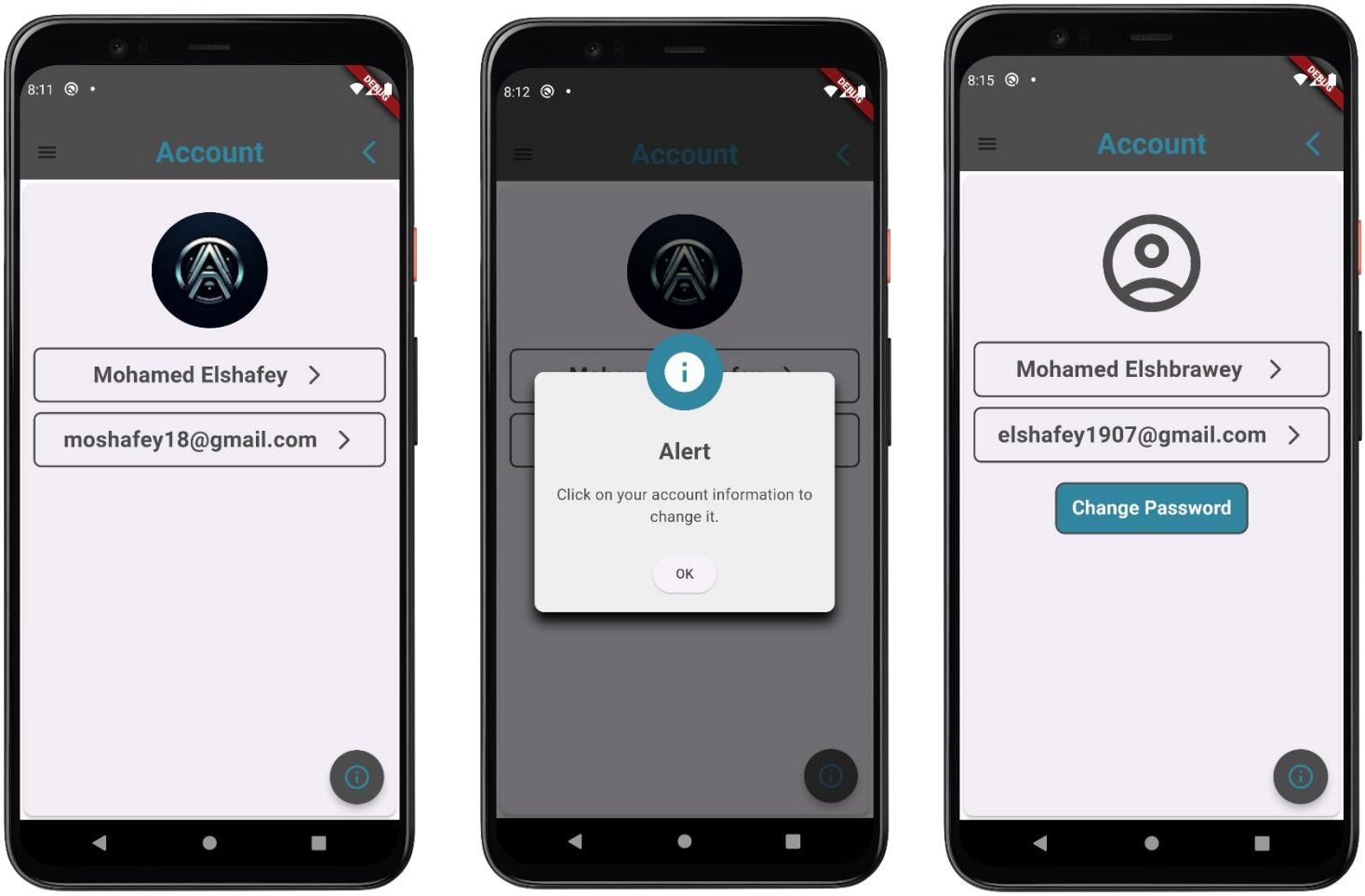
- Home: The `home.dart` file defines the main screen of the application, encompassing an app bar with a logo and title, a side drawer facilitating navigation to different sections like "Home", "Account", "Bluetooth", "Logs", and "Settings", and a bottom navigation bar offering quick access to "Home", "Control", "Auto", and "Manual" sections. The body dynamically switches content based on the selected navigation item, displaying appropriate fragments for each section. Additionally, it requests necessary permissions related to Bluetooth and location access, ensuring the app's functionality. A "Log Out" button in the drawer facilitates user logout, redirecting them to the splash screen upon logout.





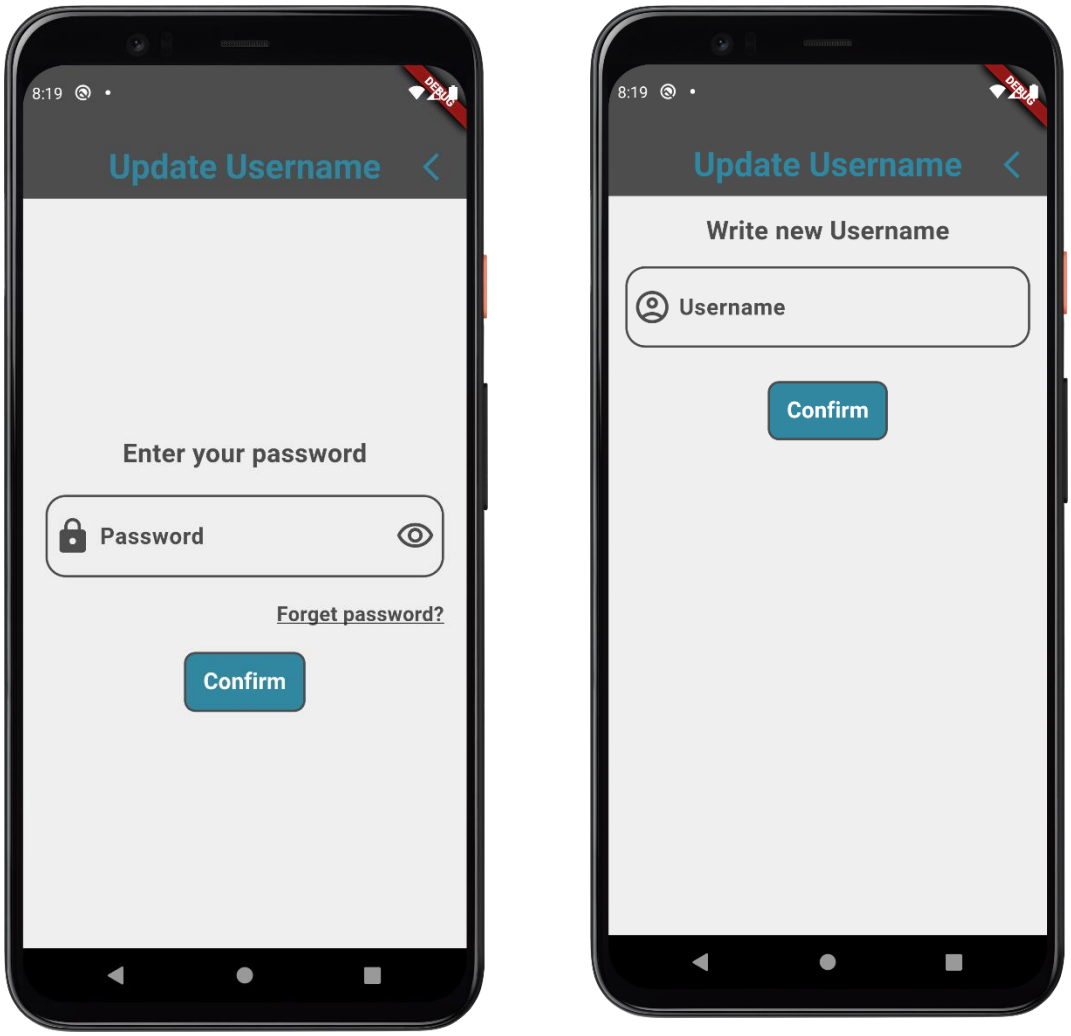
- Account: In the `account.dart` file, a Flutter widget is implemented to display and manage user account settings. The interface includes options for updating profile information such as name and email address. It also provides a button to change the password if the user is logged in using email authentication. The user's profile picture is displayed if they have signed in with Google, otherwise, a default profile icon is shown. The layout is organized using cards and

containers with appropriate padding and styling. Interaction with authentication services like Supabase is handled for signing out and updating user information securely. Additionally, a dialog box is displayed when the floating action button is pressed to provide information about changing account details.



- UpdateName: In the `updatename.dart` file, two Flutter widgets are implemented: `UpdateName` and `NewWidget`. The `UpdateName` widget manages

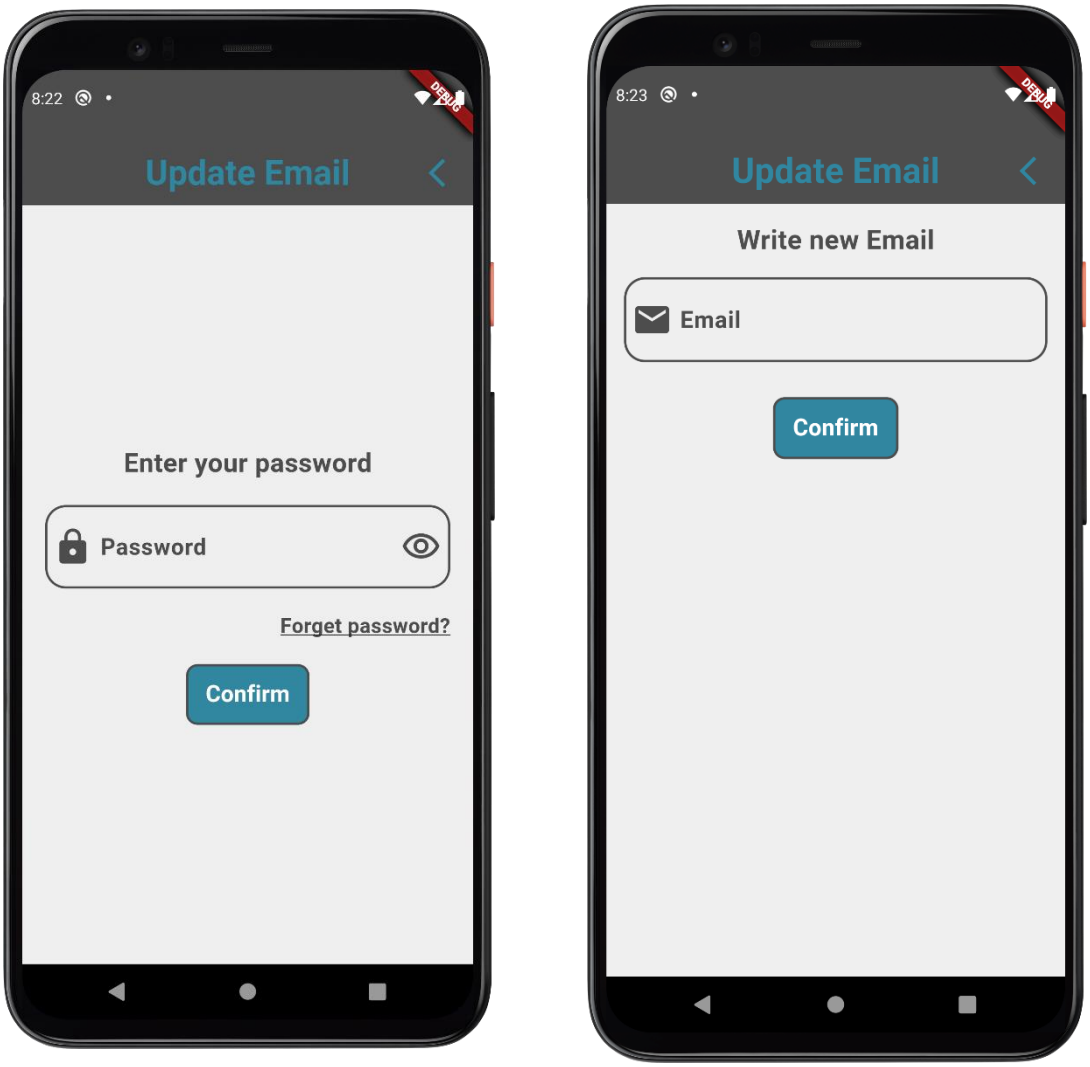
the process of updating the user's username. Initially, it displays a form where the user needs to enter their password for authentication. Once the password is confirmed, the widget switches to display the `NewWidget`, where the user can input their new username. Upon confirming the new username, the user's account details are updated, and a dialog is shown to notify the user about the successful update or any errors that may occur during the process. The `NewWidget` widget presents a form for entering the new username. Once the user submits the new username, it is sent to the backend for updating the user's account information. If the update is successful, a dialog is shown to inform the user. Otherwise, if an error occurs, another dialog is displayed to notify the user about the issue. Both widgets utilize the Supabase Flutter library for authentication and updating user information securely. They also incorporate custom dialog widgets for providing feedback to the user during the update process. Additionally, appropriate styling and validation are applied to the form fields to enhance the user experience.



- UpdateEmail: The `updateemail.dart` file contains two Flutter widgets: `UpdateEmail` and `NewWidget`. The `UpdateEmail` widget manages the process of updating the user's email address. Initially, it checks if the current user's authentication provider is Google. If it is, it sets the `pageVisibility` variable to true, indicating that the user's email cannot be updated. Otherwise, it displays a form where the user needs to

enter their password for authentication. Once the password is confirmed, the widget switches to display the `NewWidget`, where the user can input their new email address. Upon confirming the new email address, the user's account details are updated, and a dialog is shown to inform the user about the successful update or any errors that may occur during the process. The `NewWidget` widget presents a form for entering the new email address. It performs validation to ensure that the entered email address is in the correct format. Once the user submits the new email address, it is sent to the backend for updating the user's account information. If the update is successful, a dialog is shown to inform the user. Otherwise, if an error occurs, another dialog is displayed to notify the user about the issue. Both widgets utilize the Supabase Flutter library for authentication and updating user information securely. They also incorporate custom dialog widgets for providing feedback to the user during the update process. Additionally, appropriate styling and

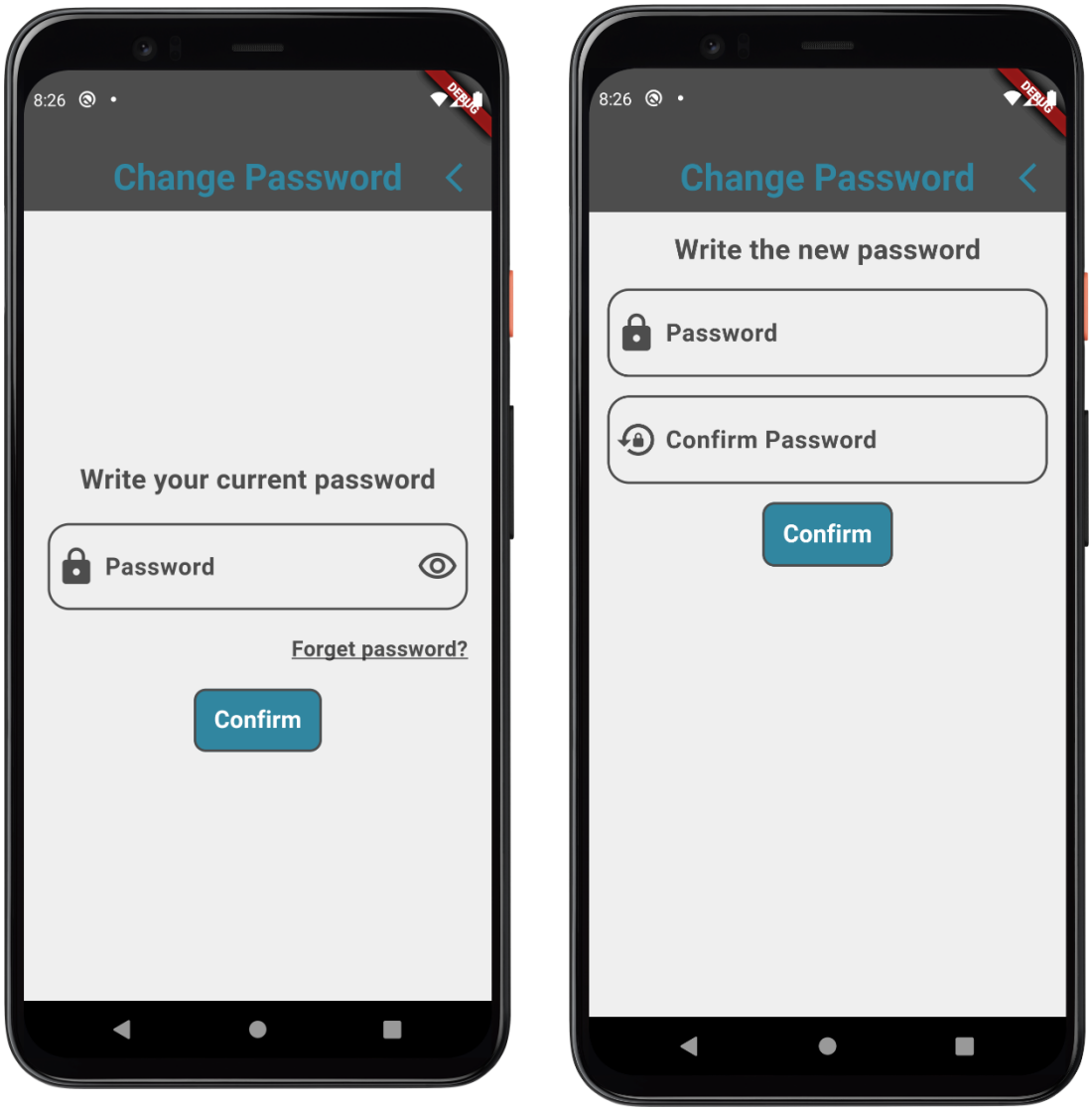
validation are applied to the form fields to enhance the user experience.



- UpdatePassword: The `updatepassword.dart` file contains two Flutter widgets: `UpdatePassword` and `NewWidget`. The `UpdatePassword` widget manages the process of updating the user's password. Initially, it displays a form where the user needs to enter their current password for

authentication. Once the password is confirmed, the widget switches to display the `NewWidget`, where the user can input their new password and confirm it. Upon confirming the new password, the user's account password is updated, and a dialog is shown to inform the user about the successful update or any errors that may occur during the process. The `NewWidget` widget presents a form for entering the new password and confirming it. It performs validation to ensure that the entered passwords meet certain criteria, such as being at least 8 characters long and containing at least one uppercase letter, one lowercase letter, one digit, and one special character. Once the user submits the new password and confirms it, it is sent to the backend for updating the user's account information. If the update is successful, a dialog is shown to inform the user. Otherwise, if an error occurs, another dialog is displayed to notify the user about the issue. Both widgets utilize the Supabase Flutter library for authentication and updating user information securely. They also incorporate custom dialog widgets for providing feedback to the user during the update process.

Additionally, appropriate styling and validation are applied to the form fields to enhance the user experience.



- MainActivity(Kotlin side): In `MainActivity.kt`, the `configureFlutterEngine` function sets up communication channels between Flutter and native Android code using MethodChannels and EventChannels. The MethodChannel

"bluetooth_channel" handles method calls from Flutter, enabling functions like connecting to Bluetooth, writing data to a connected device, and reading data from it. The EventChannel "bluetooth_event_channel" facilitates event streaming from native Android code to Flutter, allowing real-time data updates. When Flutter starts listening to this channel, it triggers the `onListen` method, establishing the EventSink for event transmission. When Flutter stops listening, the `onCancel` method is invoked, removing the EventSink. These mechanisms enable seamless interaction between Flutter UI and native Android functionality, particularly for Bluetooth operations in the `MyBluetooth` class.

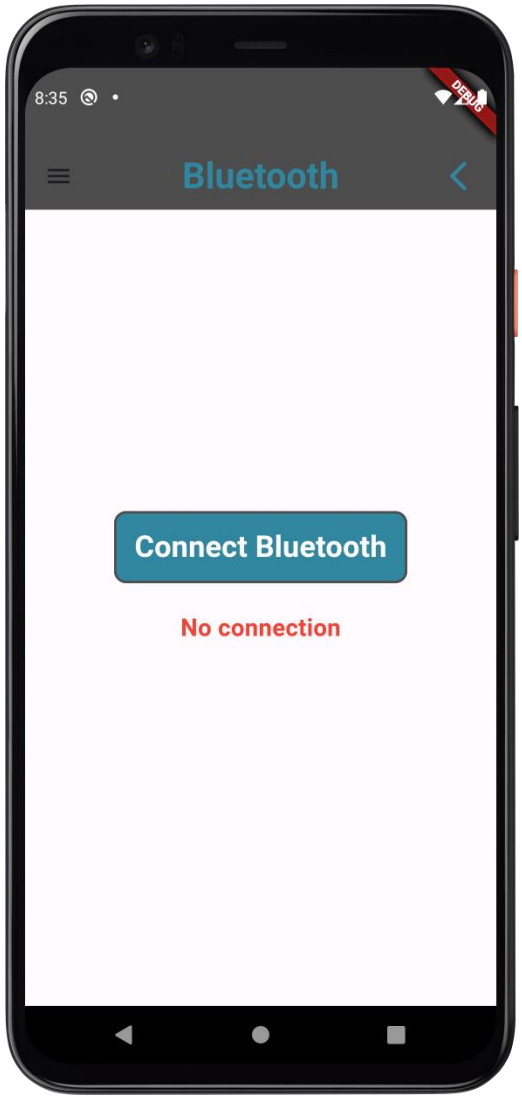
- MyBluetooth: In `MyBluetooth.kt`, a class named `MyBluetooth` is defined to handle Bluetooth operations in the Android app. It includes methods for connecting to a Bluetooth device, reading and writing data, and setting up event streaming for real-time data updates. The `connect()` function establishes a Bluetooth connection with a specified device using its MAC address, while the `startListeningForData()` method continuously listens for incoming data from

the connected device. Data is read in a separate thread to prevent blocking the UI. The `writeData()` method sends data to the connected Bluetooth device, and `readData()` reads data from the device. Additionally, the class manages the Bluetooth socket, event sink for event streaming, and handles exceptions gracefully.

- Bluetooth: In `bluetooth.dart`, the `Bluetooth` class is implemented as a stateful widget responsible for managing Bluetooth connectivity in the app. It includes methods for connecting to a Bluetooth device, writing data to the device, and handling received data through event streaming. The widget also provides a user interface with a button to initiate the Bluetooth connection, display connection status, and log out functionality. Additionally, the widget integrates navigation to other app screens through a drawer menu, allowing users to navigate between different sections of the app seamlessly.
    - Also: In the "bluetooth.dart" file, two types of channels are used: "MethodChannel" and "EventChannel", both from the Flutter framework's Channels API.

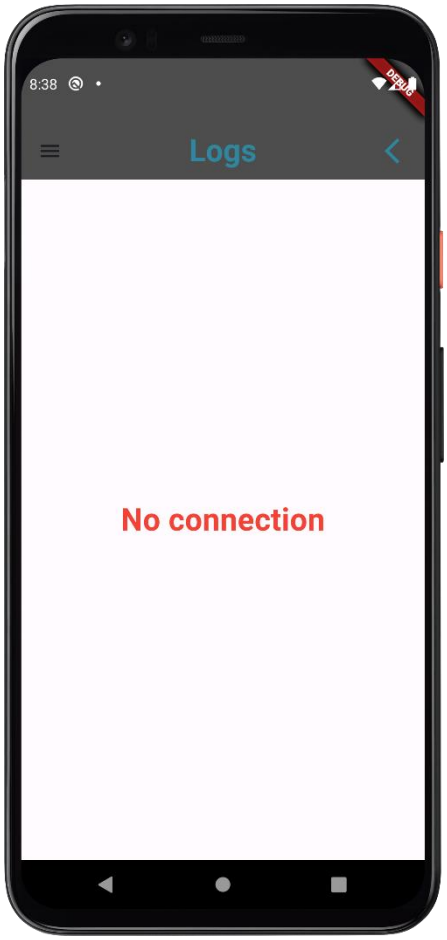
- MethodChannel: This channel, called "bluetooth_channel", is used to call platform-specific methods. In this case, it is used to initiate Bluetooth actions such as connecting to a device (“connectBluetooth”) and writing data to a Bluetooth device (“writeData”). These methods are implemented on the native platform side (Android or iOS) and are called from Dart code.
- EventChannel: The event channel, called "bluetooth_event_channel", is used to broadcast events, allowing communication from the platform back to the Dart side of the application. In this implementation, it is used to receive data from a Bluetooth device asynchronously (`_eventChannel.receiveBroadcastStream()`). When data is received from a Bluetooth device, it is streamed back to the Dart side, where it can be processed and displayed to the user.

- Overall, these channels facilitate two-way communication between the Dart and the native platform, allowing the application to interact with the Bluetooth functionality on the device.

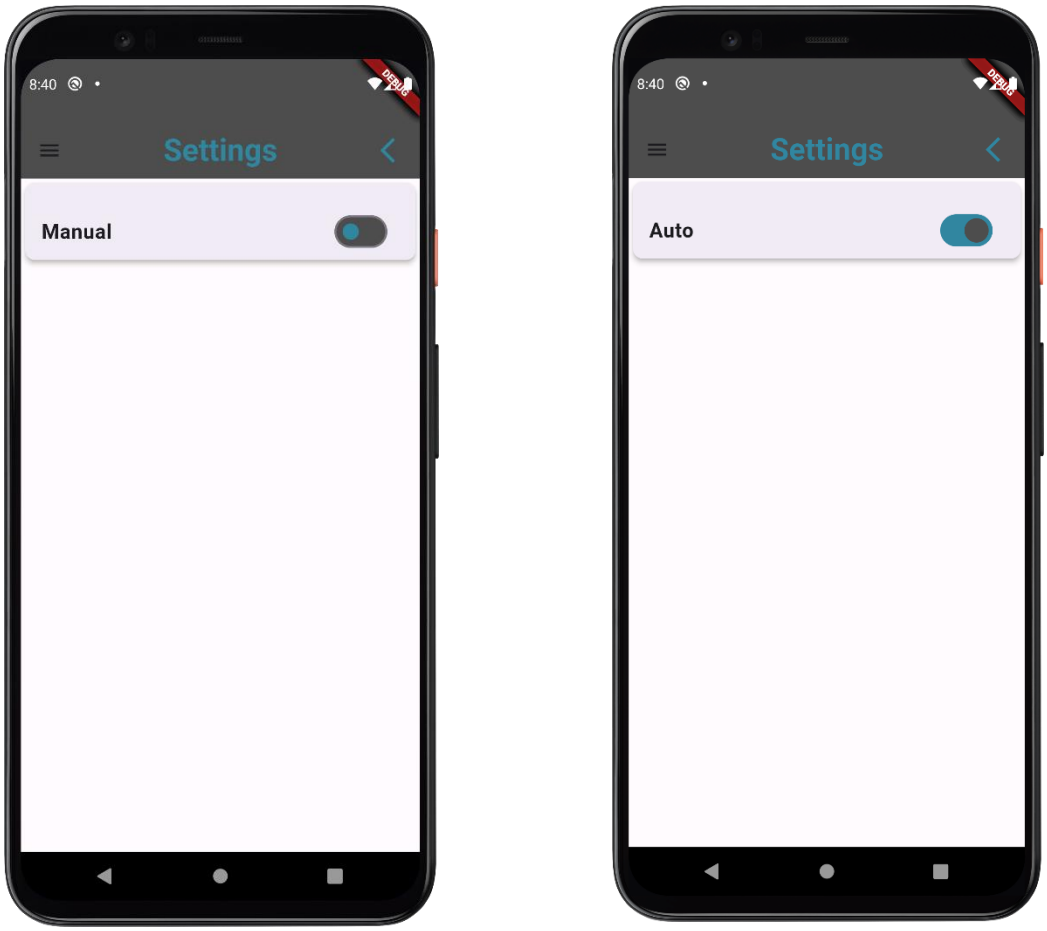


- Logs: In `logs.dart`, the `MethodChannel` named "bluetooth_channel" is utilized to trigger platform-specific methods for Bluetooth functionality like

connecting to a device and writing data, while the `EventChannel` named "bluetooth_event_channel" streams data received from the Bluetooth device back to the Dart side of the app. Upon receiving data, it's appended to `_receivedDataList` along with a timestamp for tracking, and the UI is updated using `setState` to display the data in real-time. This bidirectional communication enables the app to interact with Bluetooth devices and showcase received data to the user dynamically.

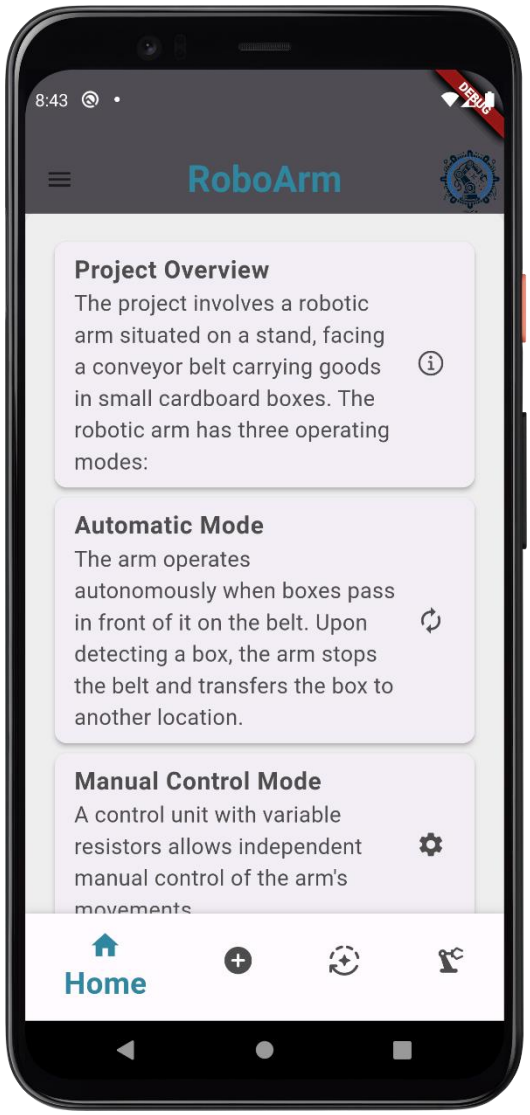


- Settings: The `settings.dart` file contains the implementation of a settings page in a Flutter application. It defines a `Settings` widget, which is a stateful widget representing the settings screen. Within this widget, there is a `SwitchListTile` that allows users to toggle between auto and manual modes. The state of the switch is stored in the `_switchCondition` variable, and changes to this state are persisted using the `setSwitch` function, which utilizes `SharedPreferences`. Additionally, the widget includes a navigation drawer with options to navigate to other sections of the app and a log out button for user authentication management. Overall, this file contributes to the user interface and functionality of the settings section within the Flutter app.



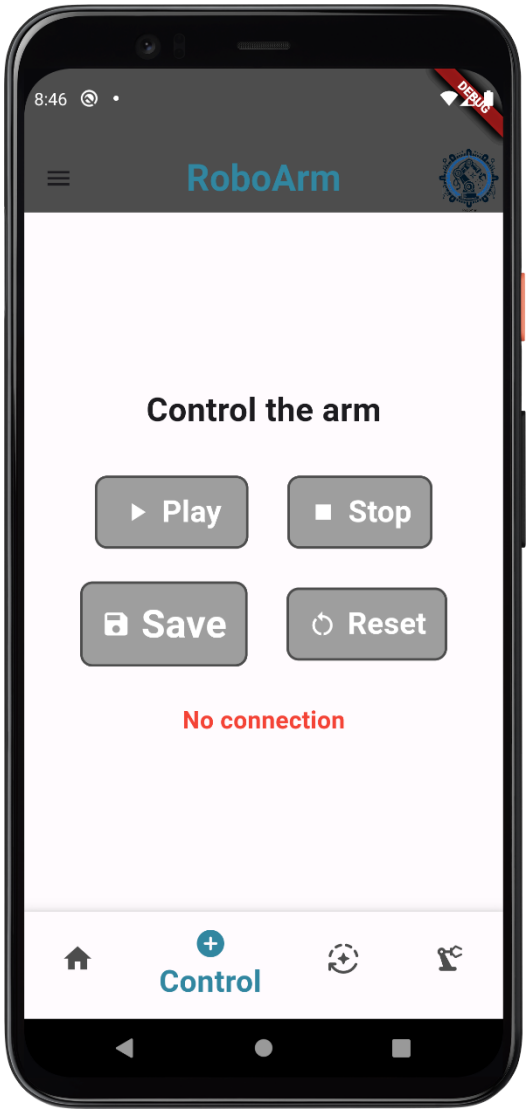
- HomeFragment: The `HomeFragment` class represents a fragment in a Flutter application's user interface, specifically for the home screen. This fragment displays various cards containing project overview information. Each card contains a title, subtitle, and a trailing icon for additional actions or information. The information displayed on each card includes details about the project overview, different operating modes of a robotic arm, and features such

as automatic mode, manual control mode, programmable mode, LCD screen, and applications. The UI design incorporates images, text, and icons to provide a visually appealing and informative user experience. Additionally, the fragment is scrollable to accommodate different screen sizes. Overall, this fragment contributes to presenting project-related information effectively on the home screen of the Flutter application.



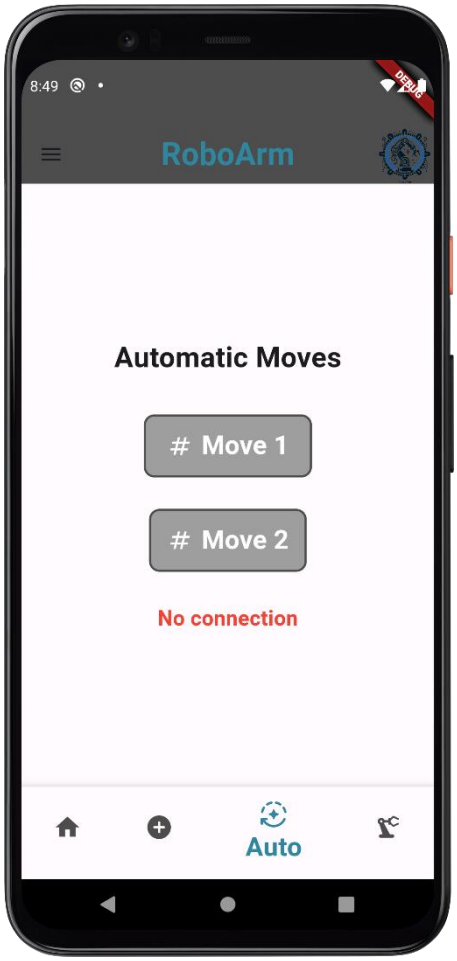
- ControlPage: The `ControlPage` class represents a page in a Flutter application responsible for controlling a robotic arm. This page consists of several buttons for controlling different actions of the arm, such as play, stop, save, and reset. The availability of these buttons is determined by the switch condition and the success of the Bluetooth connection. The switch

condition determines whether the buttons should be enabled or disabled, and it is loaded asynchronously from shared preferences. The page also displays feedback messages indicating the status of the connection and any received data from the arm. The UI design includes text labels, icons, and buttons arranged in rows and columns to provide an intuitive interface for controlling the robotic arm. Additionally, the page utilizes method channels and event channels for communication with native code to handle Bluetooth operations asynchronously.



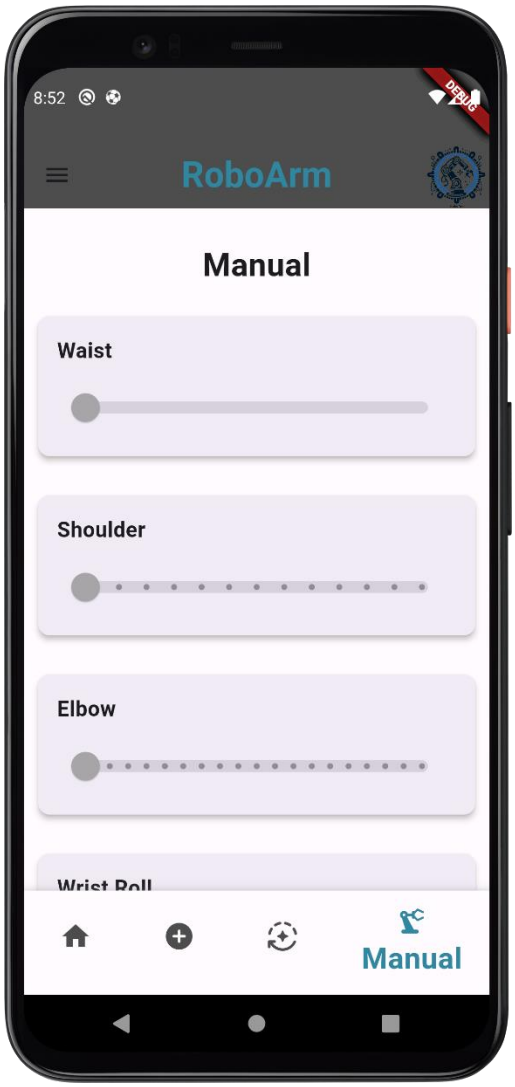
- AutoPage: The `AutoPage` class represents a page in a Flutter application designed for controlling automatic moves of a robotic arm. This page contains buttons to trigger predefined movements of the arm, labeled as "Move 1" and "Move 2." These buttons are enabled or disabled based on the success of the Bluetooth connection. When a button is pressed, the

corresponding movement command is sent to the robotic arm via Bluetooth. Feedback messages indicating the status of the connection and any received data from the arm are displayed below the buttons. The UI design includes text labels, icons, and buttons arranged in a column to provide a straightforward interface for triggering automatic movements of the robotic arm. Additionally, the page utilizes method channels and event channels for communication with native code to handle Bluetooth operations asynchronously.



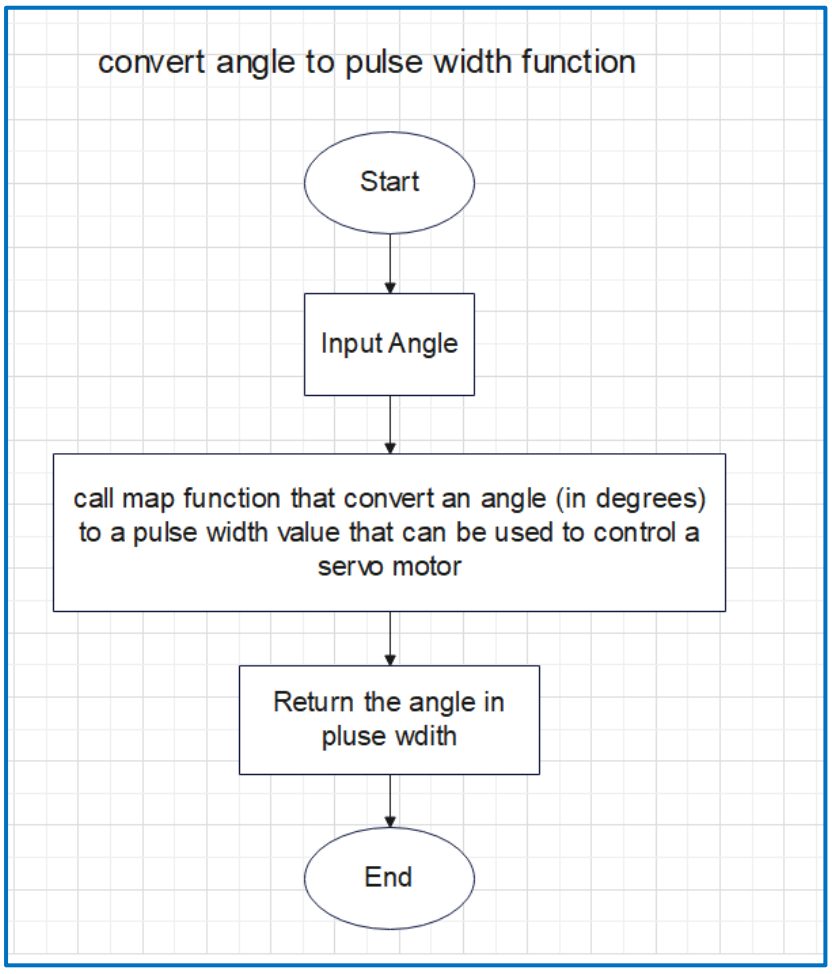
- ManualPage: The `ManualPage` class represents a page in a Flutter application designed for manually controlling the movements of a robotic arm. This page contains sliders for adjusting the angles of different joints of the robotic arm, labeled as "Waist," "Shoulder," "Elbow," "Wrist Roll," "Wrist Pitch," and "Grip." The sliders allow users to select values within predefined ranges, and the selected values are sent to the robotic arm via Bluetooth when they are

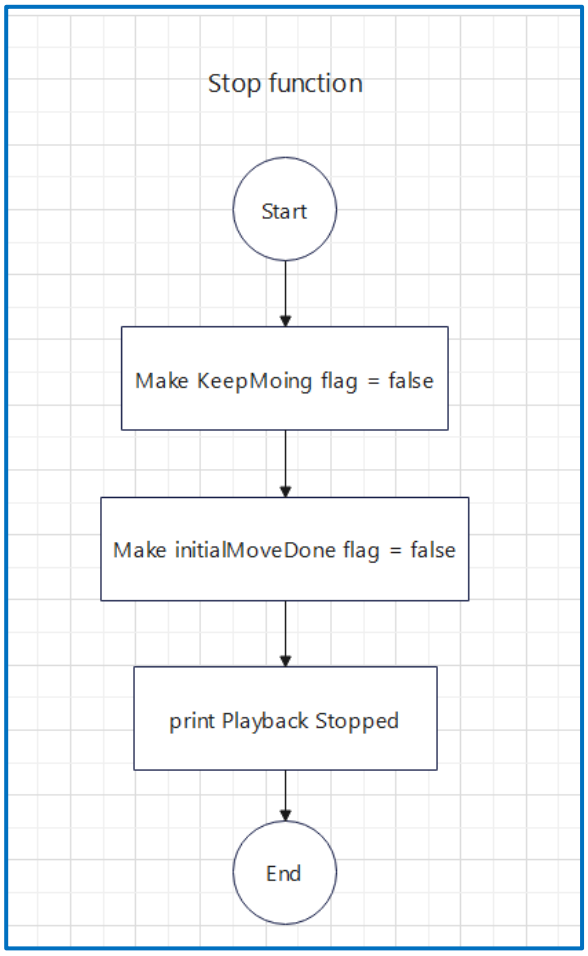
adjusted. Additionally, there are action buttons labeled as "Play," "Stop," "Save," and "Reset," which trigger corresponding commands sent to the arm when pressed. The UI design includes text labels, sliders, and buttons arranged in a column to provide an intuitive interface for controlling the robotic arm manually. The page also displays feedback messages indicating the status of the connection and any received data from the arm. Method channels and event channels are utilized for communication with native code to handle Bluetooth operations asynchronously.

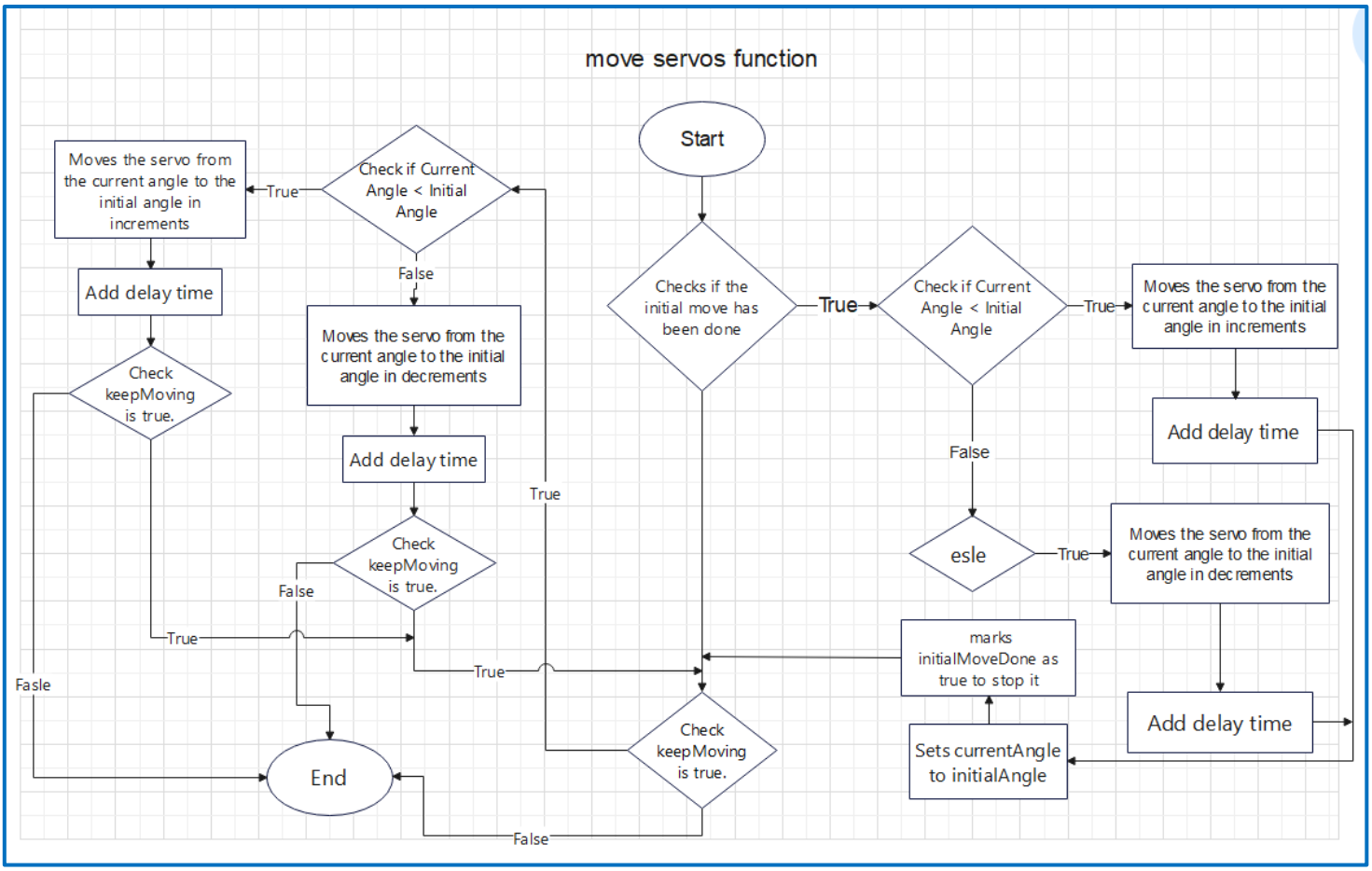

8:52
DEBUG
RoboArm
Manual
Waist
Shoulder
Elbow
Wrist Roll
Manual

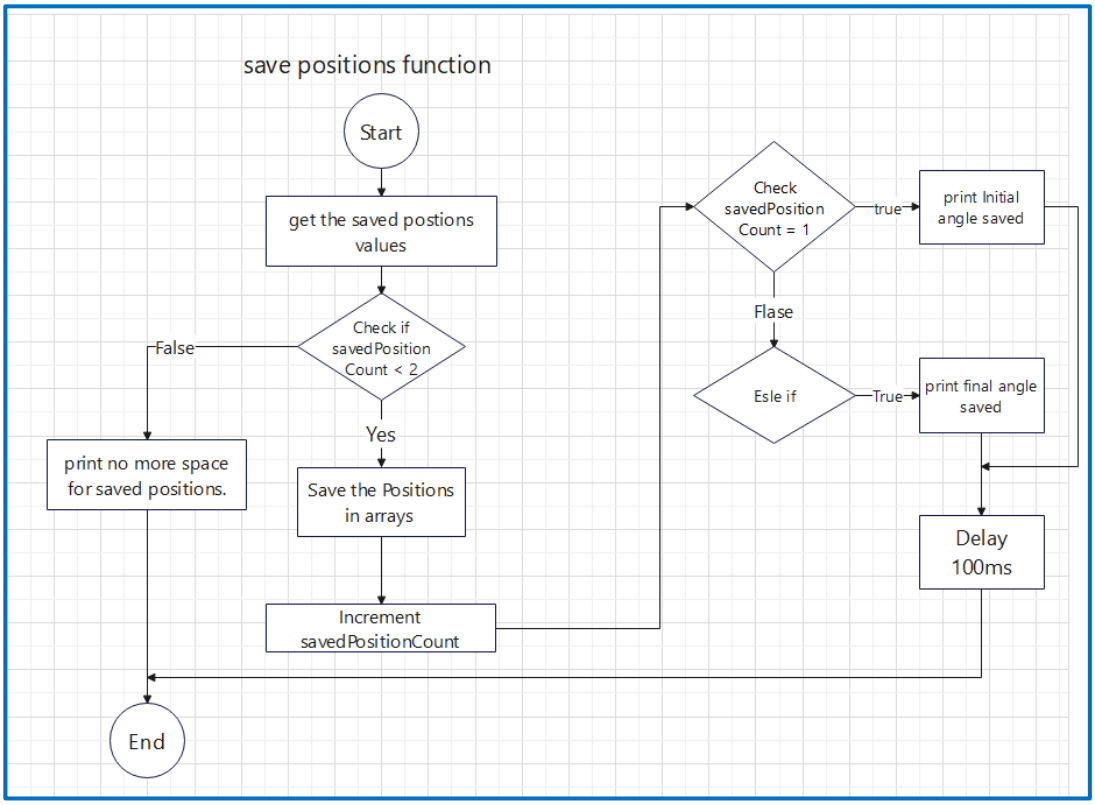

8:53
DEBUG
RoboArm
Wrist Pitch
Grip
Play
Stop
Save
Reset
No connection
Manual

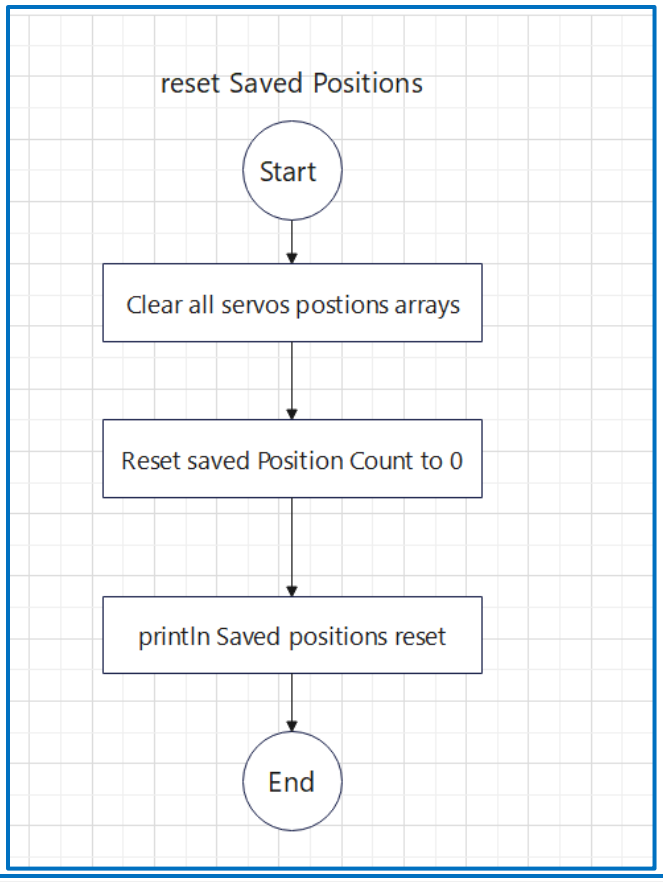
Section 3.4: Arduino Flowcharts

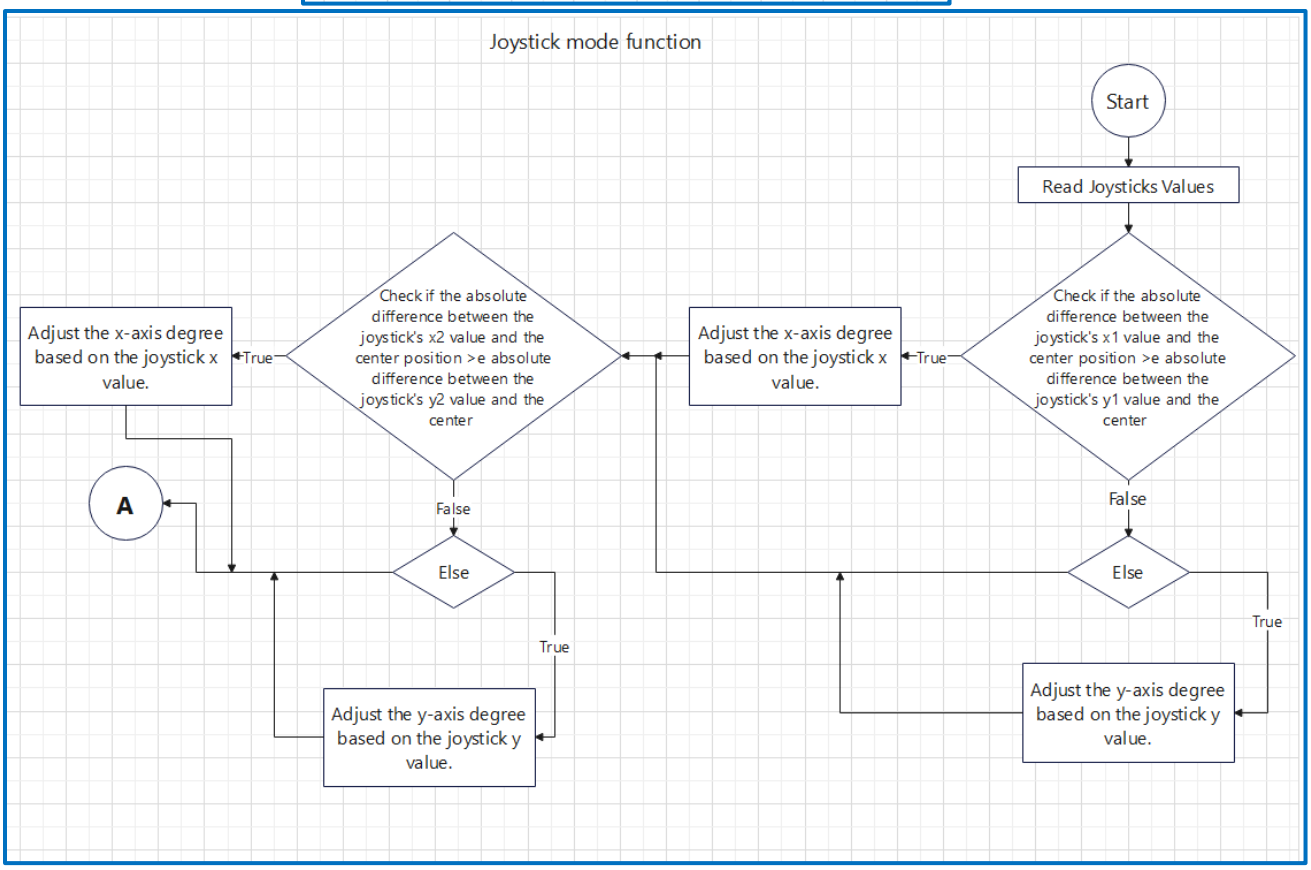


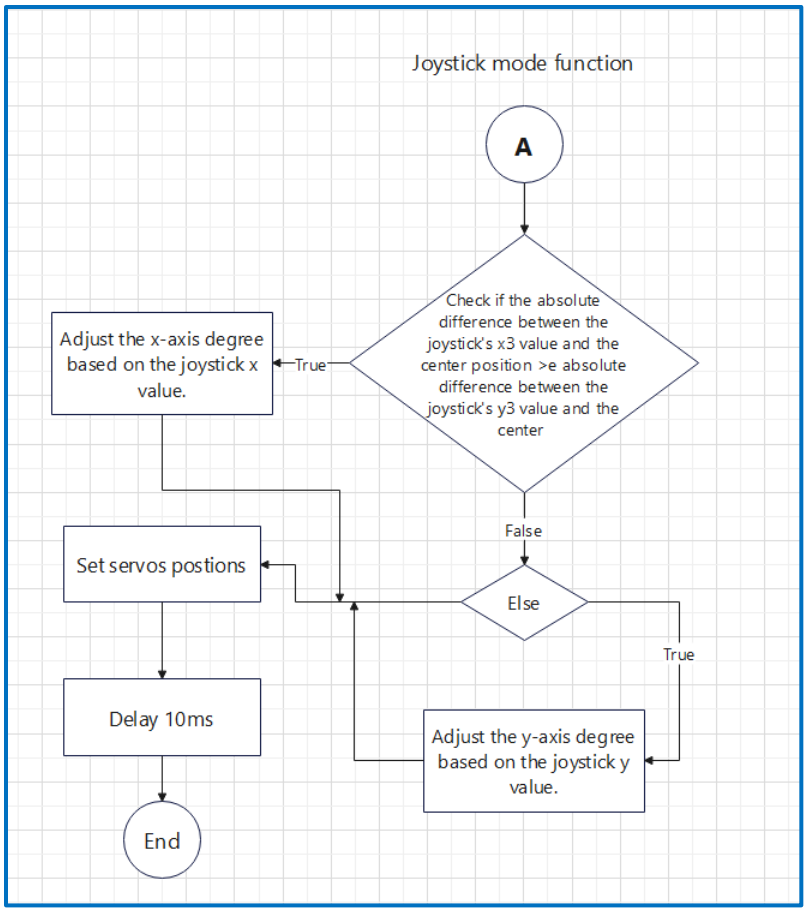


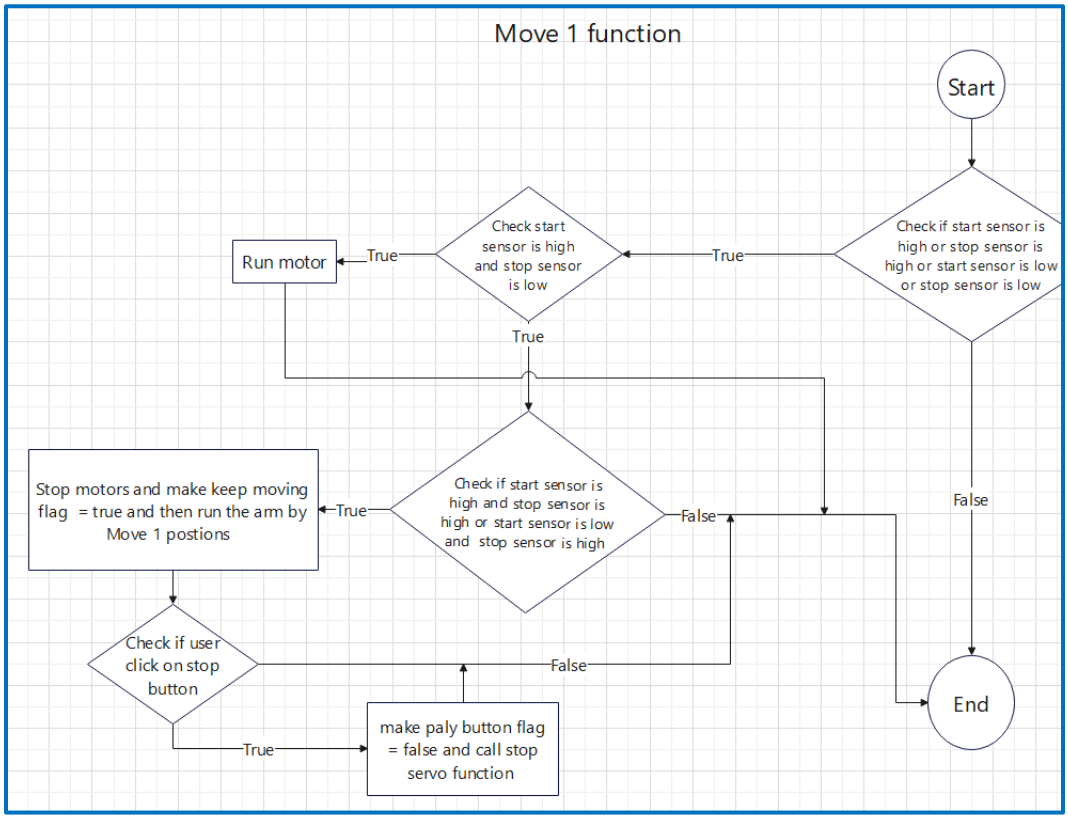

move servos function
Start
Checks if the initial move has been done
True
Check if Current Angle < Initial Angle
True
Moves the servo from the current angle to the initial angle in increments
Add delay time
False
esle
True
Moves the servo from the current angle to the initial angle in decrements
Add delay time
marks initialMoveDone as true to stop it
Sets currentAngle to initialAngle
Check keepMoving is true.
True
Check if Current Angle < Initial Angle
True
Moves the servo from the current angle to the initial angle in decrements
Add delay time
False
Check keepMoving is true.
True
False
Check keepMoving is true.
Fasle
End
False

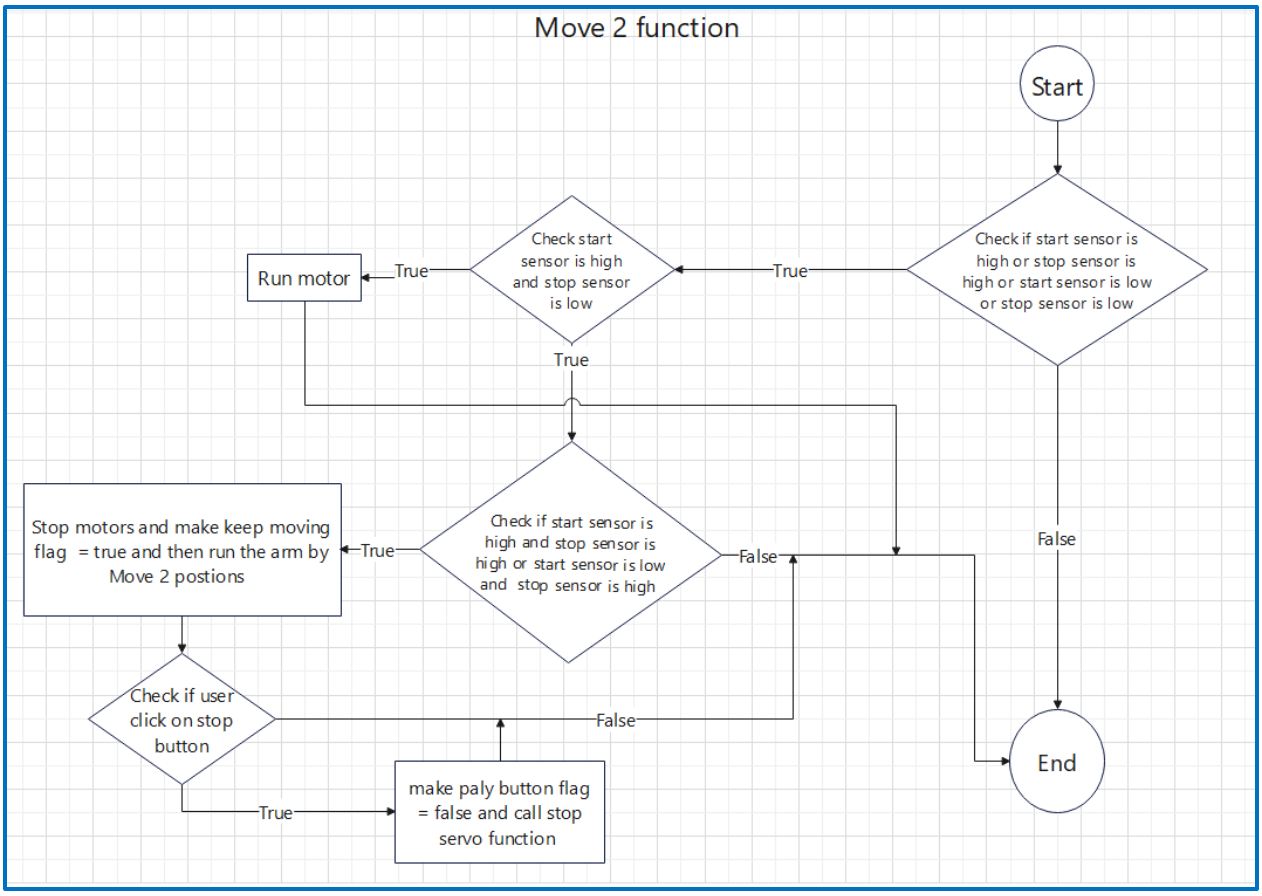

save positions function
Start
get the saved postions values
Check if savedPosition Count < 2
False
Yes
print no more space for saved positions.
Save the Positions in arrays
Increment savedPositionCount
Check savedPosition Count = 1
true
print Initial angle saved
Flase
Esle if
True
print final angle saved
Delay 100ms
End

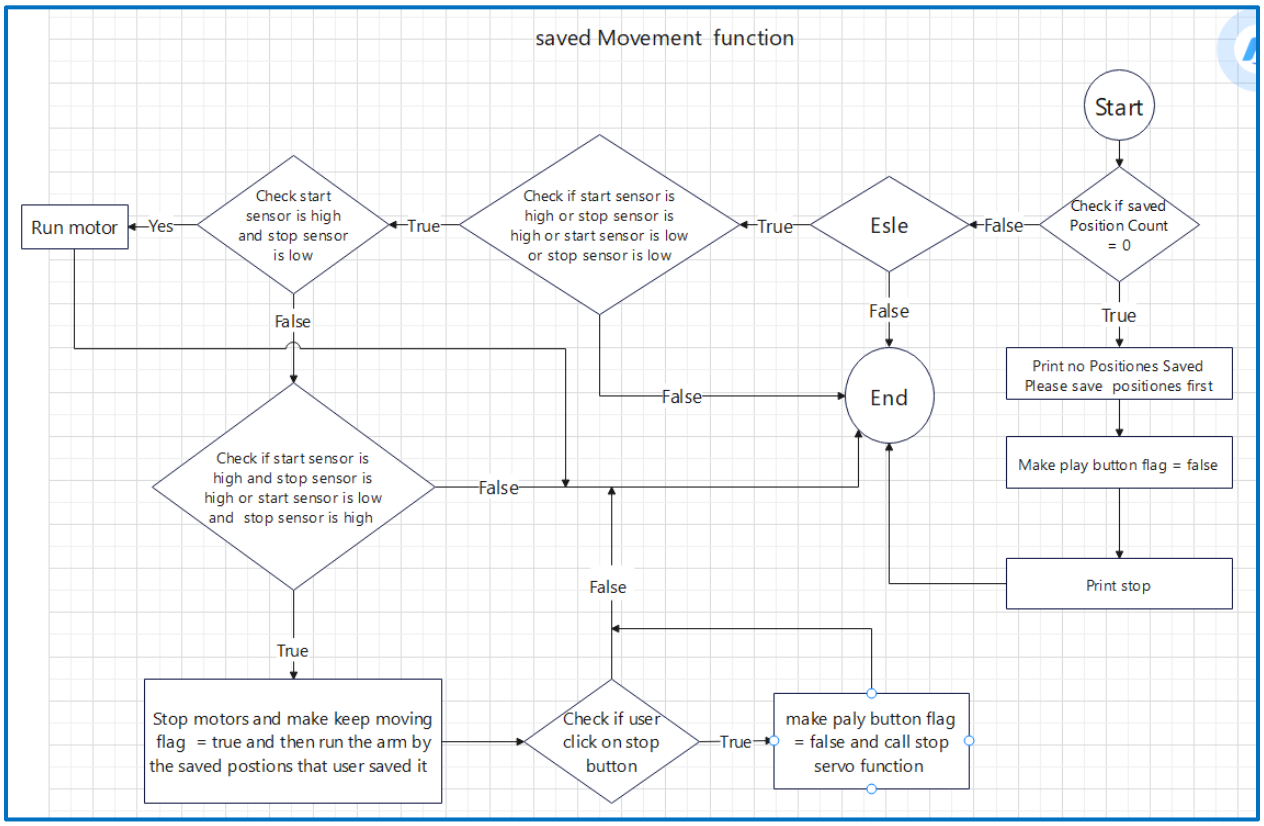

reset Saved Positions
Start
Clear all servos postions arrays
Reset saved Position Count to 0
println Saved positions reset
End

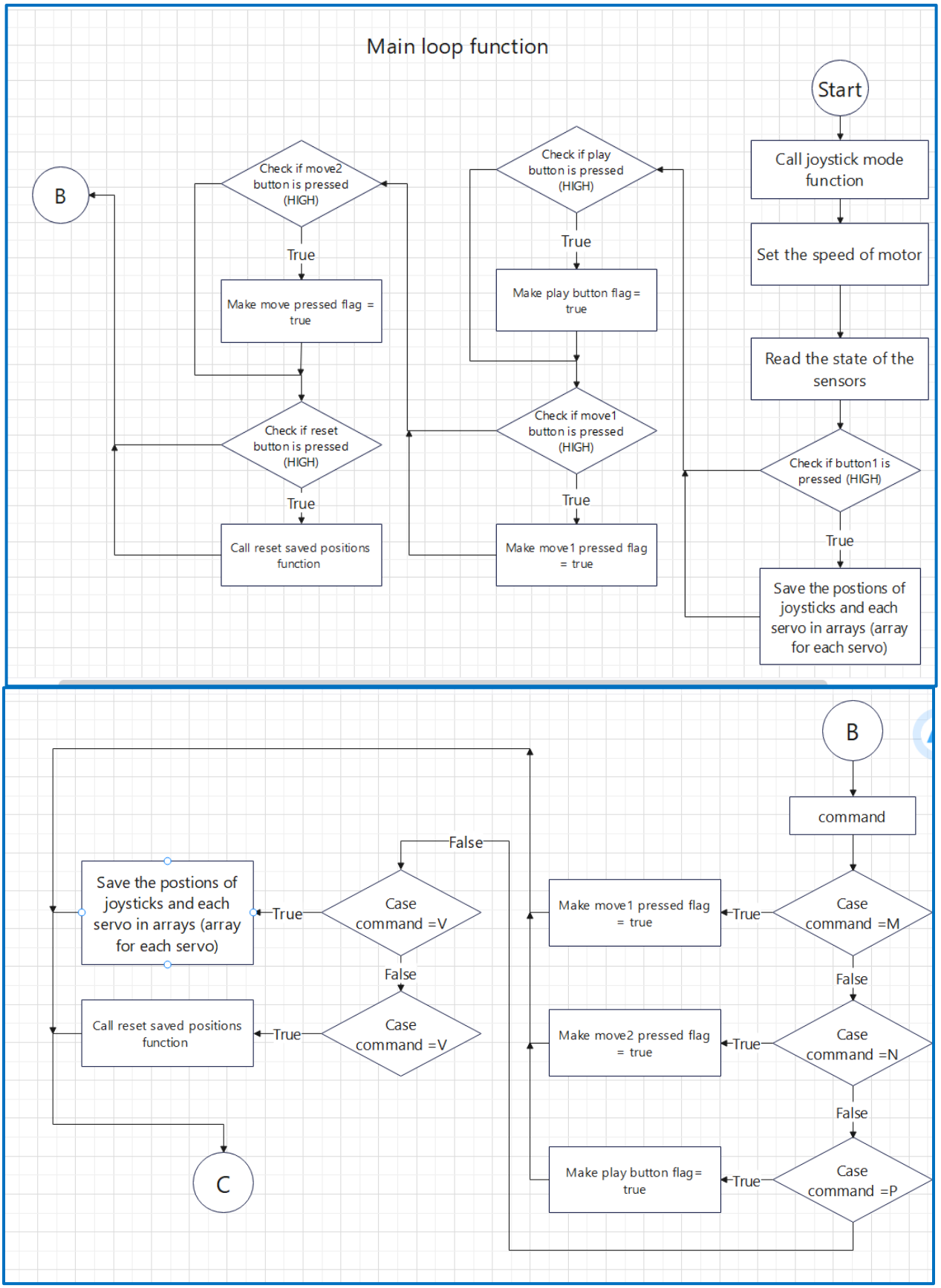

Joystick mode function
Start
Read Joysticks Values
Check if the absolute difference between the joystick's x1 value and the center position >e absolute difference between the joystick's y1 value and the center
True
Adjust the x-axis degree based on the joystick x value.
Check if the absolute difference between the joystick's x2 value and the center position >e absolute difference between the joystick's y2 value and the center
True
Adjust the x-axis degree based on the joystick x value.
False
Else
True
Adjust the y-axis degree based on the joystick y value.
A
False
Else
True
Adjust the y-axis degree based on the joystick y value.

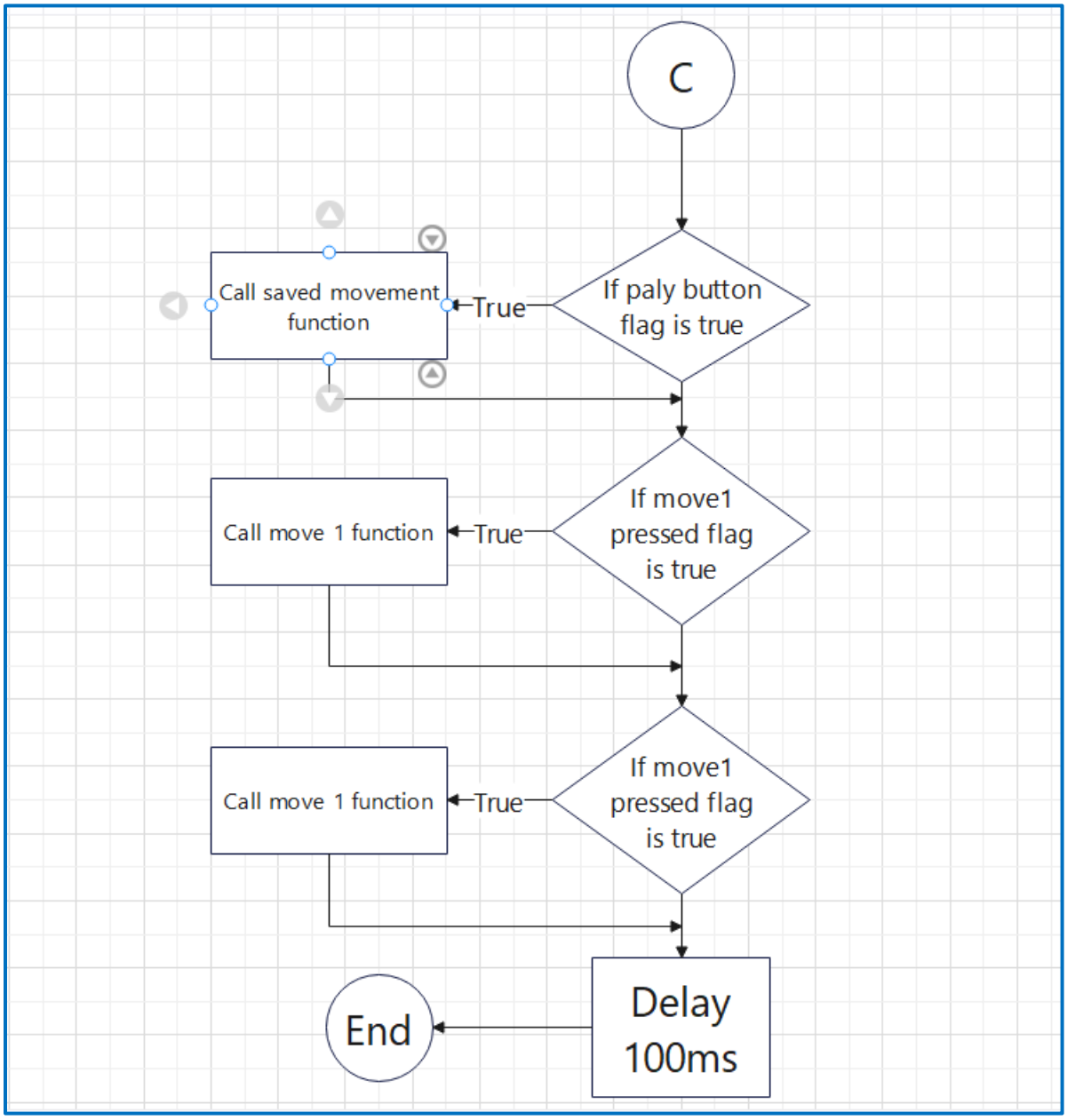

Joystick mode function
A
Check if the absolute difference between the joystick's x3 value and the center position >e absolute difference between the joystick's y3 value and the center
True
Adjust the x-axis degree based on the joystick x value.
False
Set servos postions
Else
True
Delay 10ms
Adjust the y-axis degree based on the joystick y value.
End


Move 1 function
Start
Check start sensor is high and stop sensor is low
True
Run motor
True
Check if start sensor is high or stop sensor is high or start sensor is low or stop sensor is low
True
Stop motors and make keep moving flag = true and then run the arm by Move 1 postions
True
Check if start sensor is high and stop sensor is high or start sensor is low and stop sensor is high
False
False
Check if user click on stop button
False
End
True
make paly button flag = false and call stop servo function


Move 2 function
Start
Check if start sensor is high or stop sensor is high or start sensor is low or stop sensor is low
True
Check start sensor is high and stop sensor is low
True
Run motor
True
True
False
Check if start sensor is high and stop sensor is high or start sensor is low and stop sensor is high
True
False
Stop motors and make keep moving flag = true and then run the arm by Move 2 postions
Check if user click on stop button
False
True
make paly button flag = false and call stop servo function
End


saved Movement function
Start
Check if saved Position Count = 0
False
True
Esle
True
Check if start sensor is high or stop sensor is high or start sensor is low or stop sensor is low
True
Check start sensor is high and stop sensor is low
Yes
Run motor
False
False
End
False
Print no Positiones Saved Please save positiones first
Make play button flag = false
Print stop
Check if start sensor is high and stop sensor is high or start sensor is low and stop sensor is high
False
False
True
Stop motors and make keep moving flag = true and then run the arm by the saved postions that user saved it
Check if user click on stop button
True
make paly button flag = false and call stop servo function


Main loop function
Start
Call joystick mode function
Set the speed of motor
Read the state of the sensors
Check if button1 is pressed (HIGH)
True
Save the postions of joysticks and each servo in arrays (array for each servo)
Check if play button is pressed (HIGH)
True
Make play button flag= true
Check if move1 button is pressed (HIGH)
True
Make move1 pressed flag = true
Check if move2 button is pressed (HIGH)
True
Make move pressed flag = true
Check if reset button is pressed (HIGH)
True
Call reset saved positions function
B
B
command
Case command =M
True
Make move1 pressed flag = true
False
Case command =N
True
Make move2 pressed flag = true
False
Case command =P
True
Make play button flag= true
False
Case command =V
True
Save the postions of joysticks and each servo in arrays (array for each servo)
False
Case command =V
True
Call reset saved positions function
C


C
If paly button flag is true
True
Call saved movement function
If move1 pressed flag is true
True
Call move 1 function
If move1 pressed flag is true
True
Call move 1 function
Delay 100ms
End

Chapter 4: Result

As a result of all the work we did, from researching and preparing the components, simulating the circuit, installing the components, and programming them, we reached the goal of making the robotic arm and the application work as we intended.

1- Application: The design of the user interface (UI) for both the desktop software and mobile application was guided by principles of simplicity and functionality:

1) Mobile App: The application begins with a splash-screen display interface in which the logo of the application is displayed along with its name, then another interface in which it is a login page. You can log in using email and password or log in using Google directly. If you forget your password, you can recover it via the (Forgot Password?) button if you press It will take you to another page with a field to enter the email to which the account is linked, then it will send you a verification link via email in order to reset your password. There is also a button on the login page that takes you to another page, which is the page for creating a new account in which you enter your email and password and confirm. The password and user name, and after you click on Sign Up, he will send you an email to verify the validity of that email that

you entered, that it already exists and that it is with you, and he will not log in with that email until you confirm that via the email that was sent to you. After that, if you By logging in, you will go to the main page of the program. It is a simple page with brief information about the project idea. At the bottom of it is a navigation bar between three other pages. They are: The first page is a control whose function is only. It has four buttons to control the robotic arm. A button to start, a button to stop, a button that saves a movement, and a button that performs a reset. It also shows the state the arm is in now. The second page is Auto. Its function is that it has two movements pre-recorded on the arm. We can activate any of them by pressing its button. The third and final page is the Manual page, and with it we can control the arm manually, such as a joystick for each. The servo on the arm has a slider that we move through, but the application is on Auto by default. If you want to operate the arm manually, you will be taken to the settings page from the Drawer, which you can drag from the left, or by clicking on its icon above on the left, you will be taken to the settings page and By activating manual mode, auto mode will be canceled and manual mode will be

activated, and you will be able to control the arm manually. There is a special page for the account with account information, in which you will find a picture of your computer (if you are logged in with Google, the picture of your Google account will appear, but if you are registered with an email and password, it will not appear). The page also displays the user's name and his email, and you can change them by clicking on it. You will enter your password and then he will make you enter the updated statement. He will then make you log in again to ensure your validity. The same also applies to changing the password. He will send an email to verify and change the password via email. In the event that the email is also changed, he will send an email via email to the old email. An email will also be sent to the new email to verify the authenticity of the two. There is then a Bluetooth page through which you can connect to the arm. At the end, there is a special page for Logs to display all the details of what is happening in the arm now. At the end of the Drawer there is a button to log out.

2) Desktop App: It is the same thing as the mobile phone application, but we encountered some problems with it in connecting to the arm because there are packages to deal with Bluetooth. It cannot be found with special packages for the Subabase due to different versions. We solved that problem in the mobile phone application by writing the code to connect to Bluetooth and dealing with it from During the native side with Kotlin, instead of using ready-made packages, we could have done the same thing with the Windows application by accessing its side, but due to the limited time to implement the project and the lack of assistance with it (until we found a solution for it), we did not implement it, but the program works normally in recording cases. New accounts, logging in and out, and everything related to accounts, and here is the application *(Figure – 4.1)*:

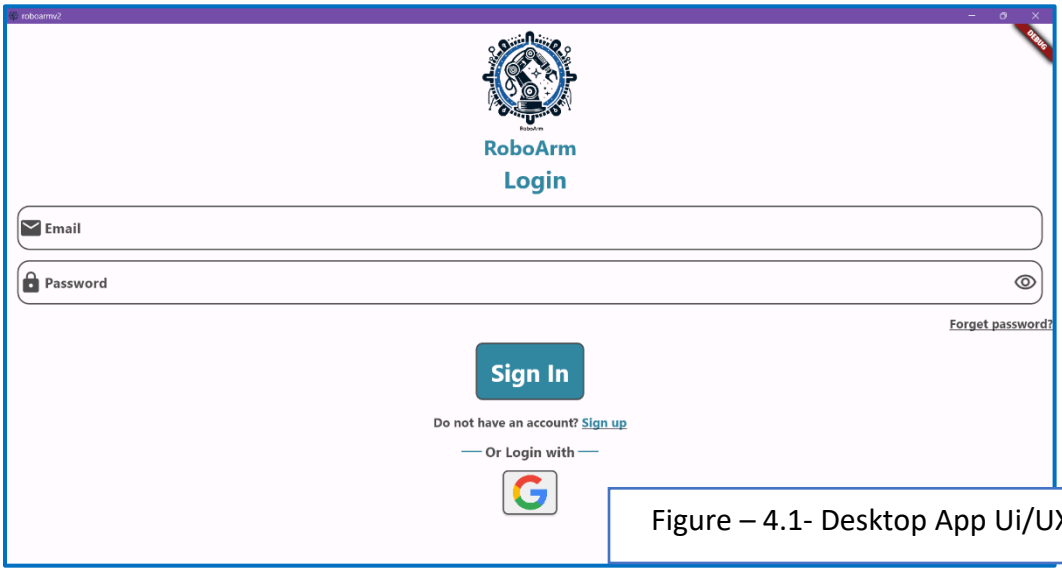
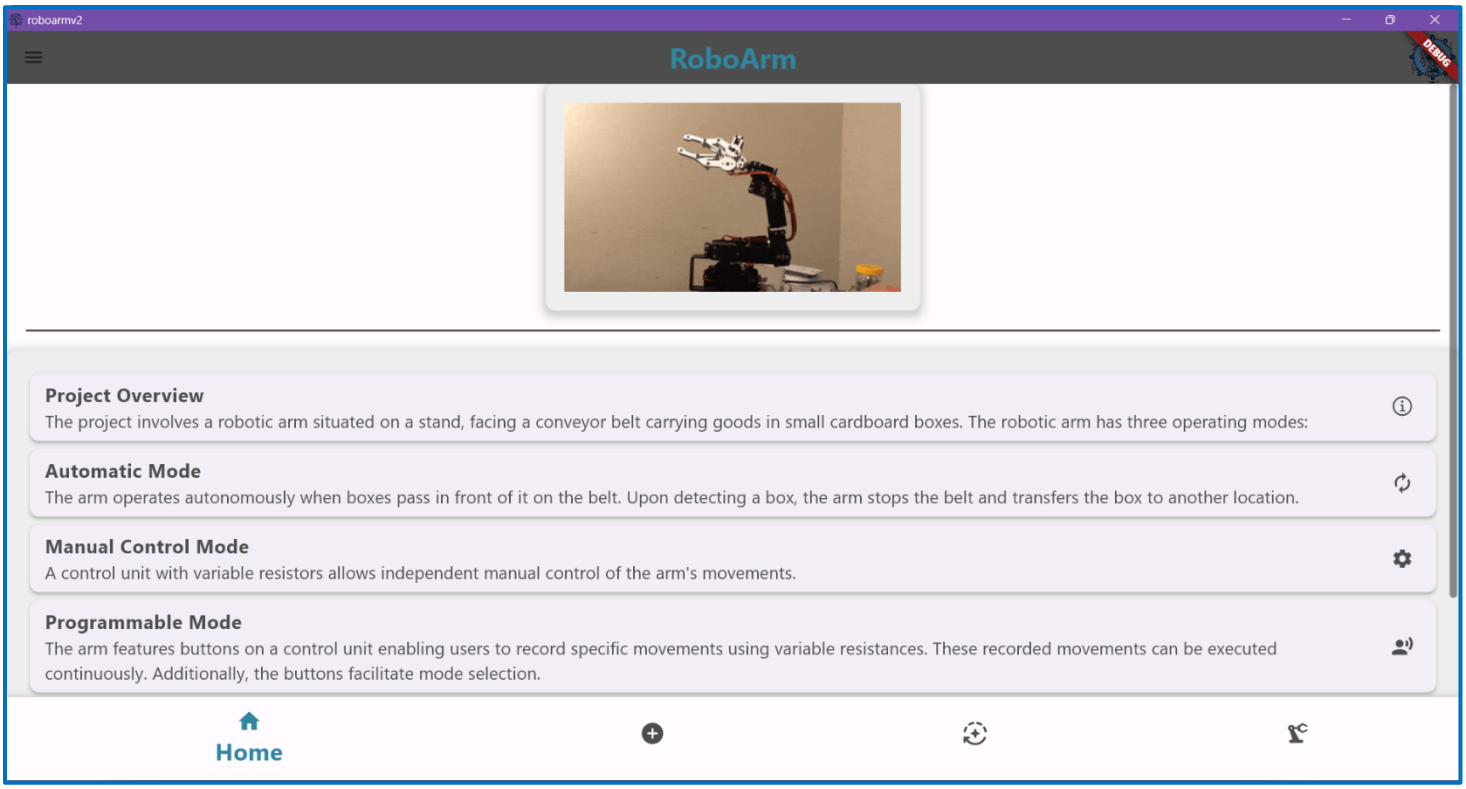
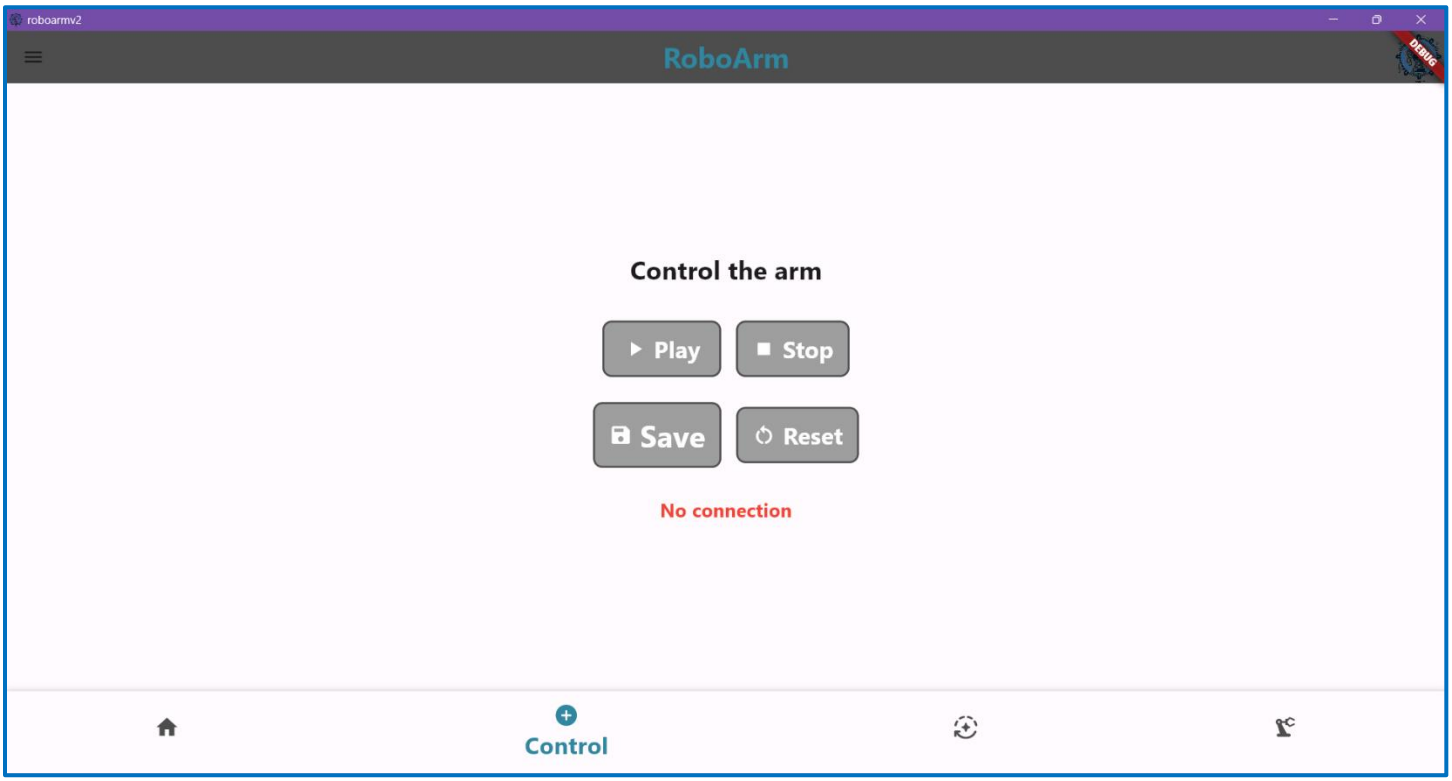
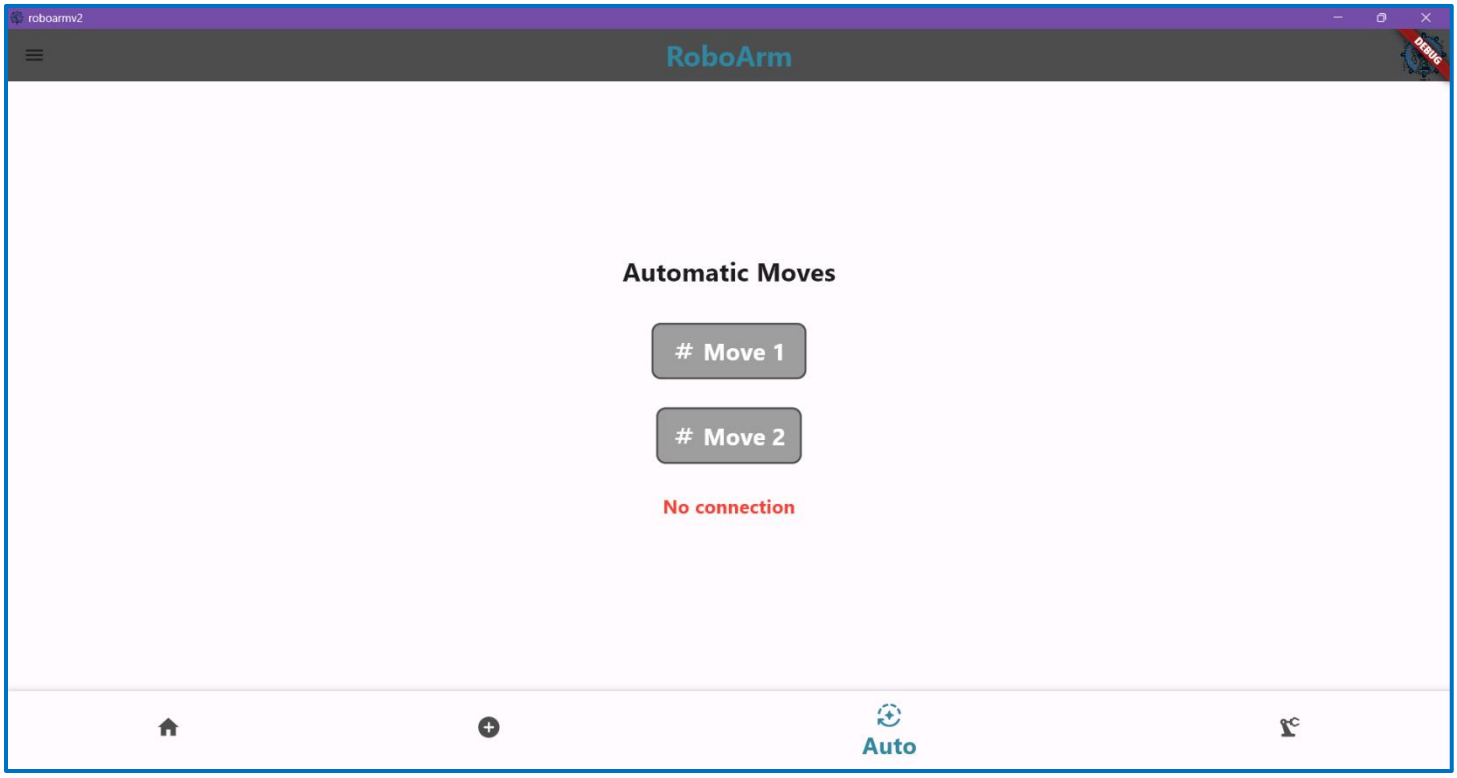
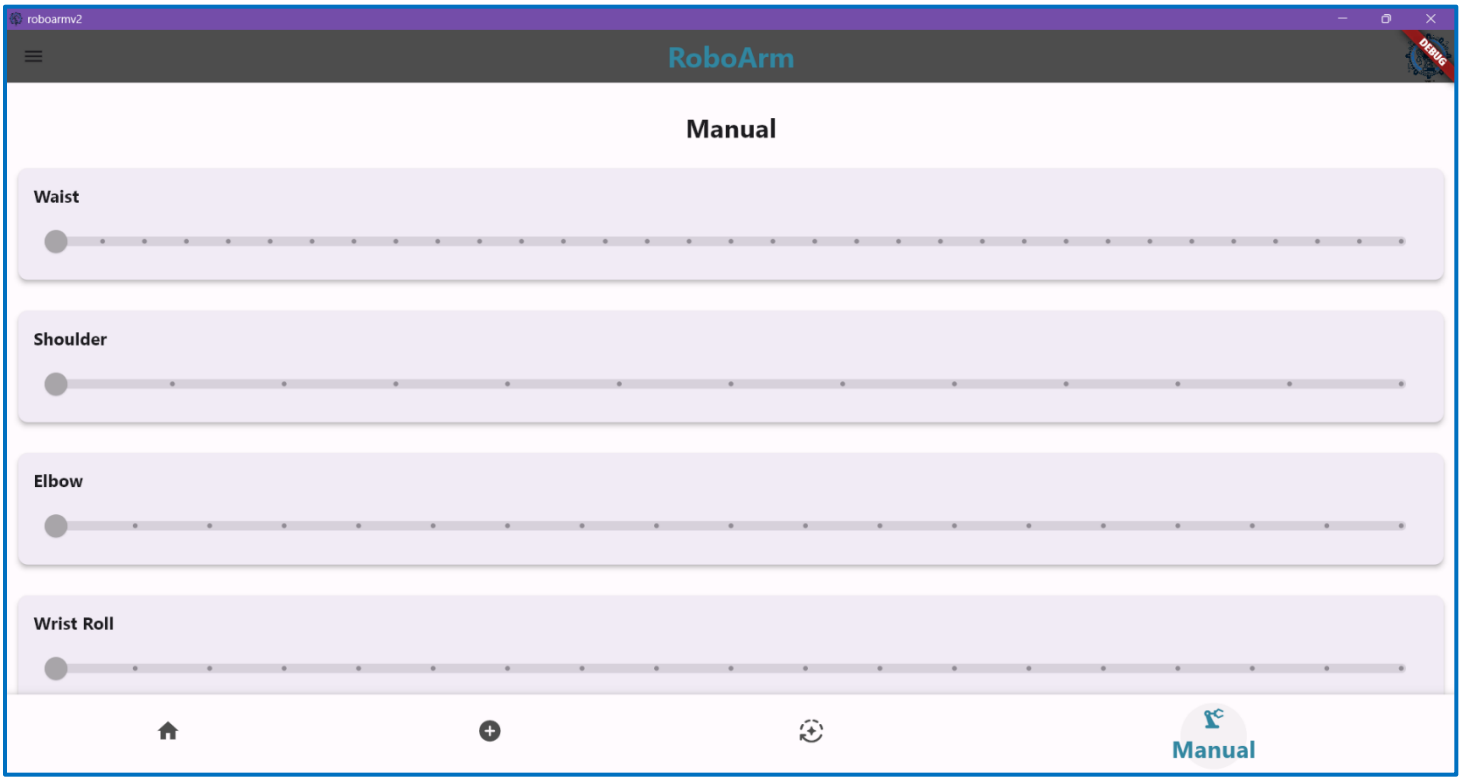


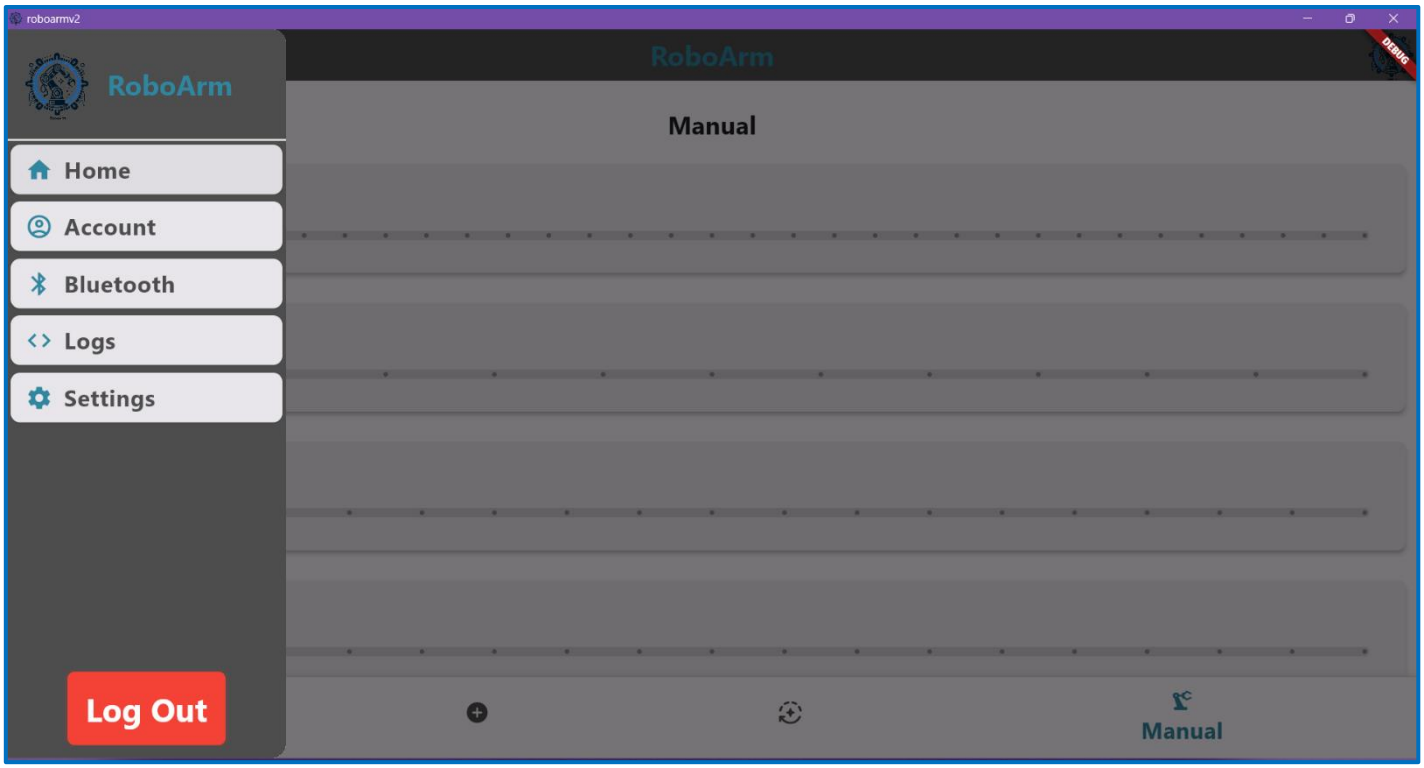
Figure – 4.1- Desktop App Ui/UX

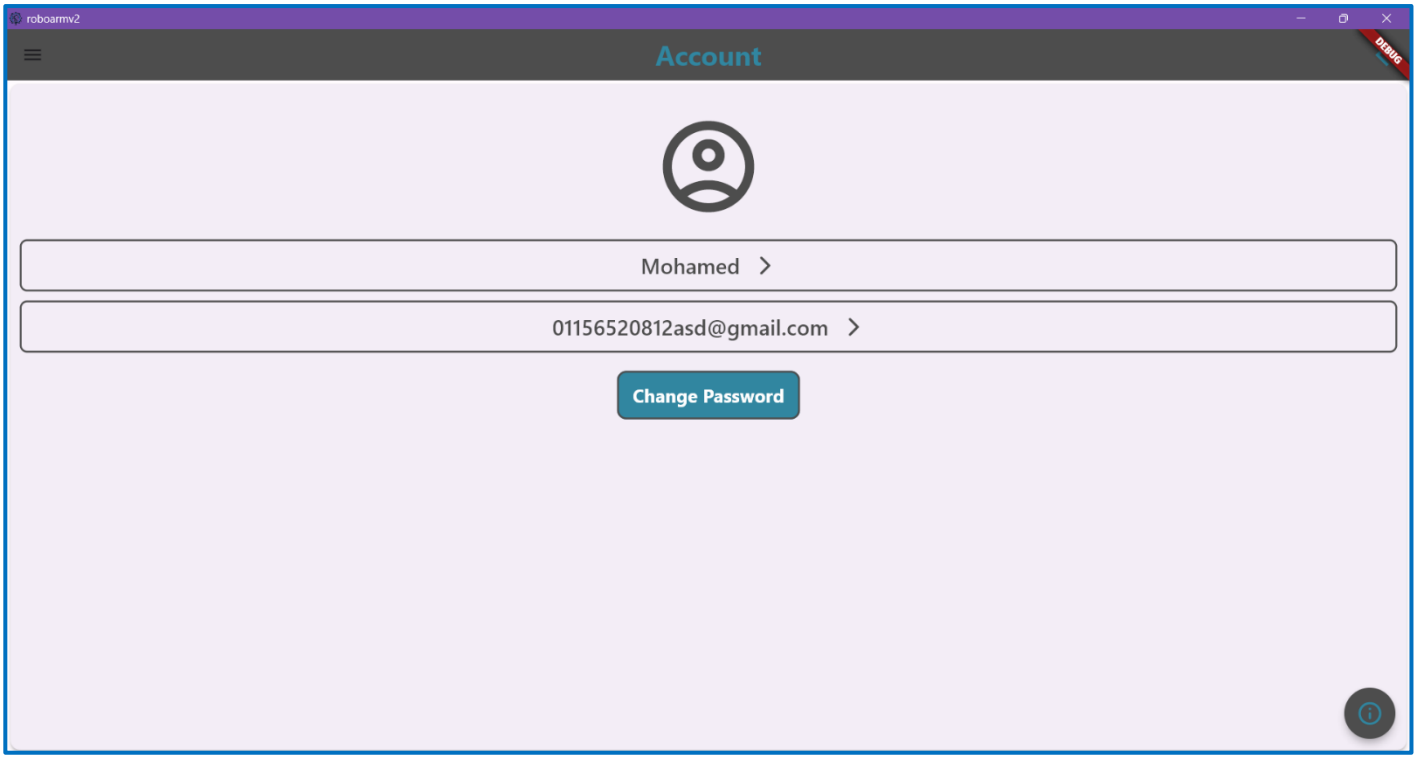

roboarmv2
RoboArm
DEBUG
Project Overview
The project involves a robotic arm situated on a stand, facing a conveyor belt carrying goods in small cardboard boxes. The robotic arm has three operating modes:
Automatic Mode
The arm operates autonomously when boxes pass in front of it on the belt. Upon detecting a box, the arm stops the belt and transfers the box to another location.
Manual Control Mode
A control unit with variable resistors allows independent manual control of the arm's movements.
Programmable Mode
The arm features buttons on a control unit enabling users to record specific movements using variable resistances. These recorded movements can be executed continuously. Additionally, the buttons facilitate mode selection.
Home

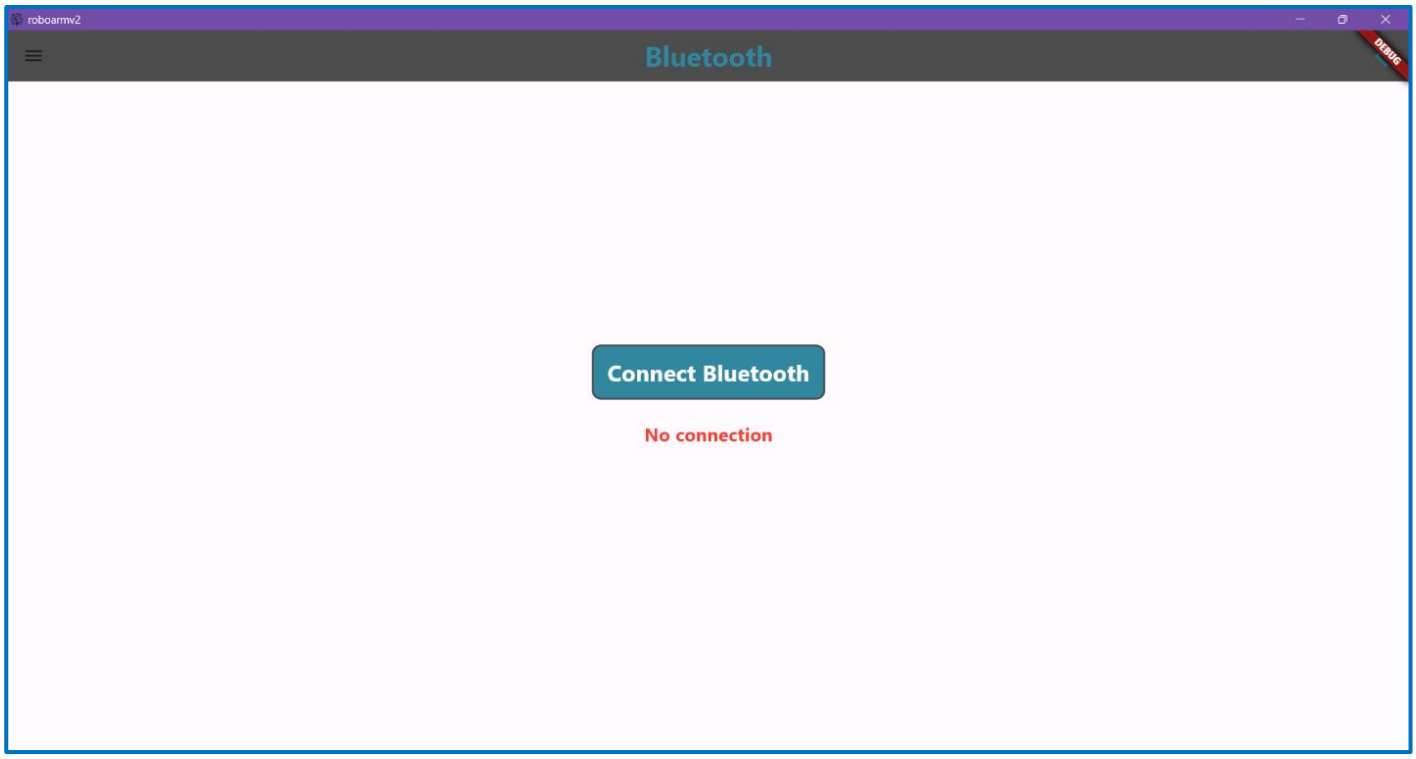

roboarmv2
RoboArm
DEBUG
Control the arm
Play
Stop
Save
Reset
No connection
Control

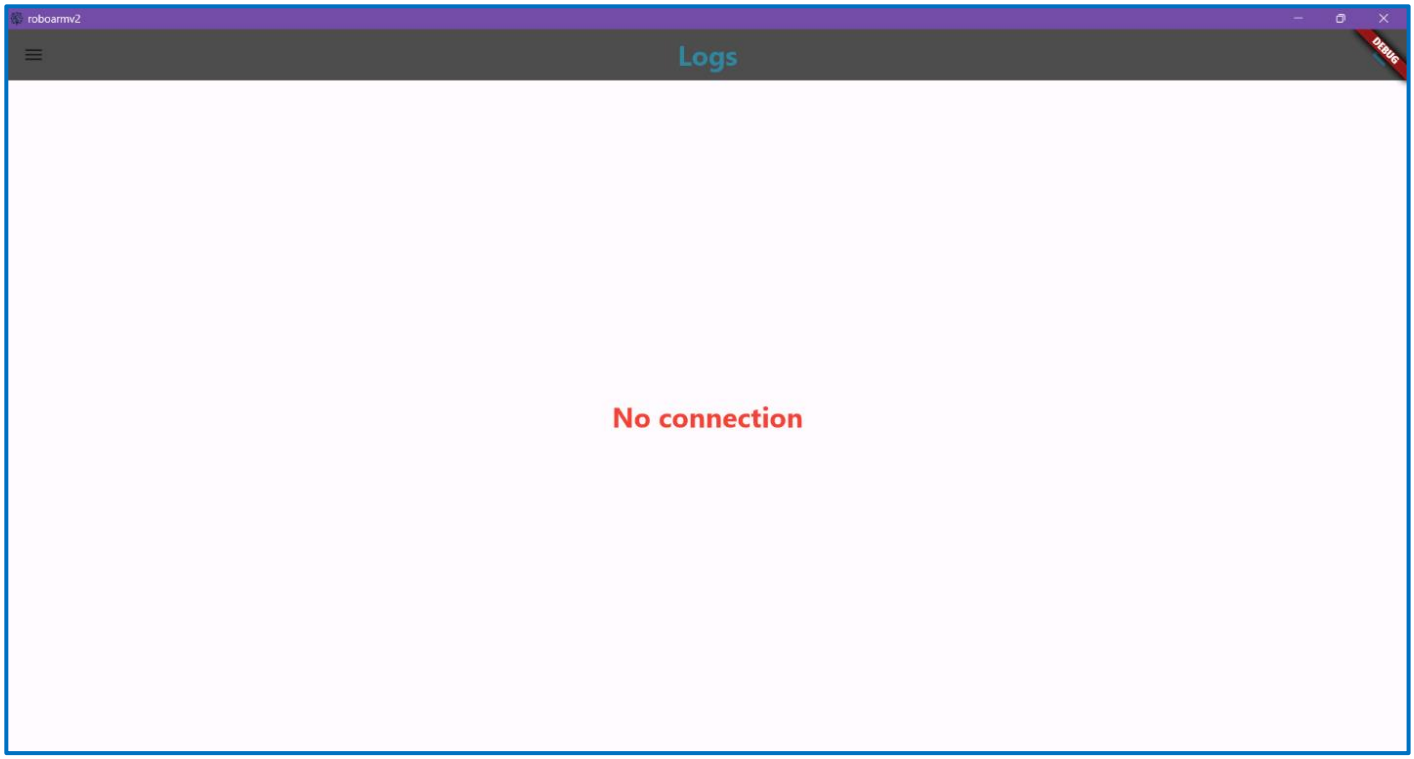

roboarmv2
RoboArm
DEBUG
Automatic Moves
Move 1
Move 2
No connection
Auto

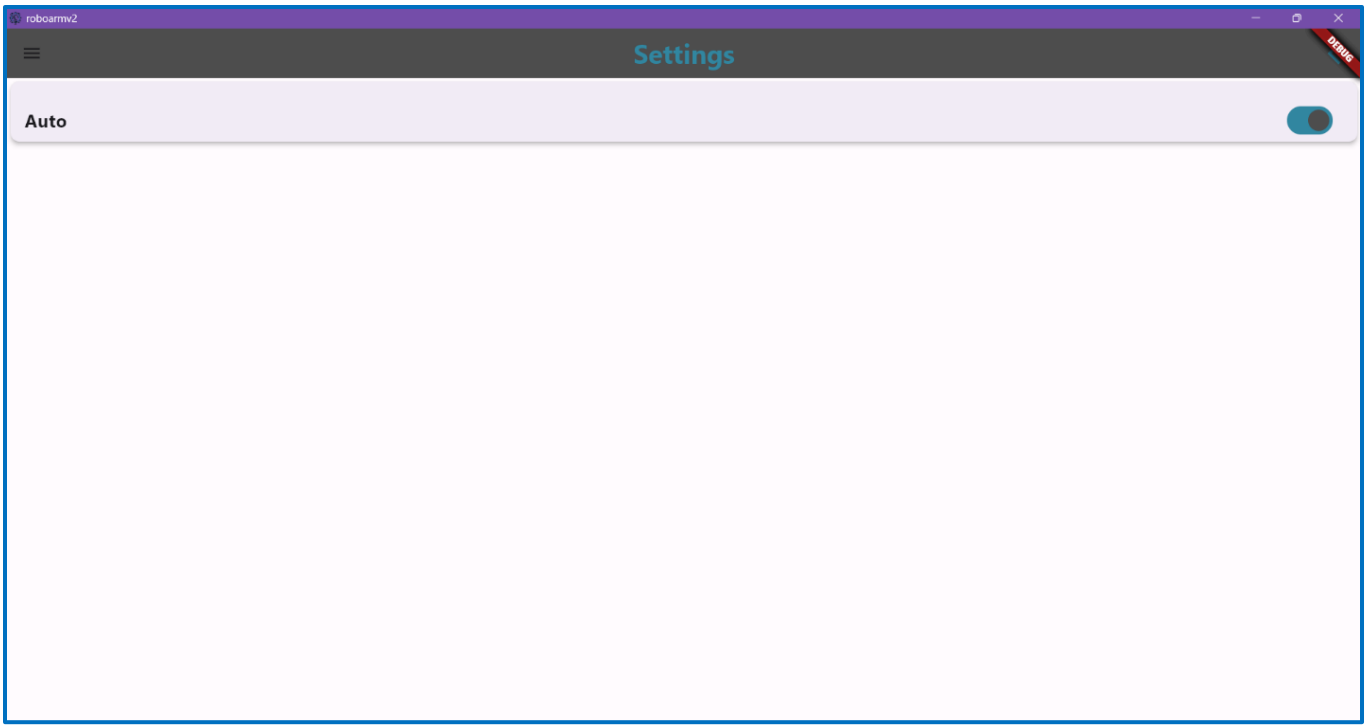

roboarmv2
RoboArm
DEBUG
Manual
Waist
Shoulder
Elbow
Wrist Roll
Manual


roboarmv2
RoboArm
RoboArm
DEBUG
Manual
Home
Account
Bluetooth
Logs
Settings
Log Out
Manual


roboarmv2
Account
DEBUG
Mohamed
01156520812asd@gmail.com
Change Password


roboarmv2
Bluetooth
DEBUG
Connect Bluetooth
No connection


roboarmv2
Logs
DEBUG
No connection



2- Robotic Arm system: The system has a belt on which the products (for example, we will call it a box) move from a starting point to an end point. That end point has a sensor to know the location of the box, and at that moment the belt stops working, then the arm starts working and goes to the box and carries it. This is all in the event that the arm It works with the automatic system, which is with pre-recorded movements that it repeats. There is also a manual system, with which I can control the arm through joysticks, and also, through those joysticks, I can record a new movement so that the arm can repeat it. *(Figure – 4.11)*:

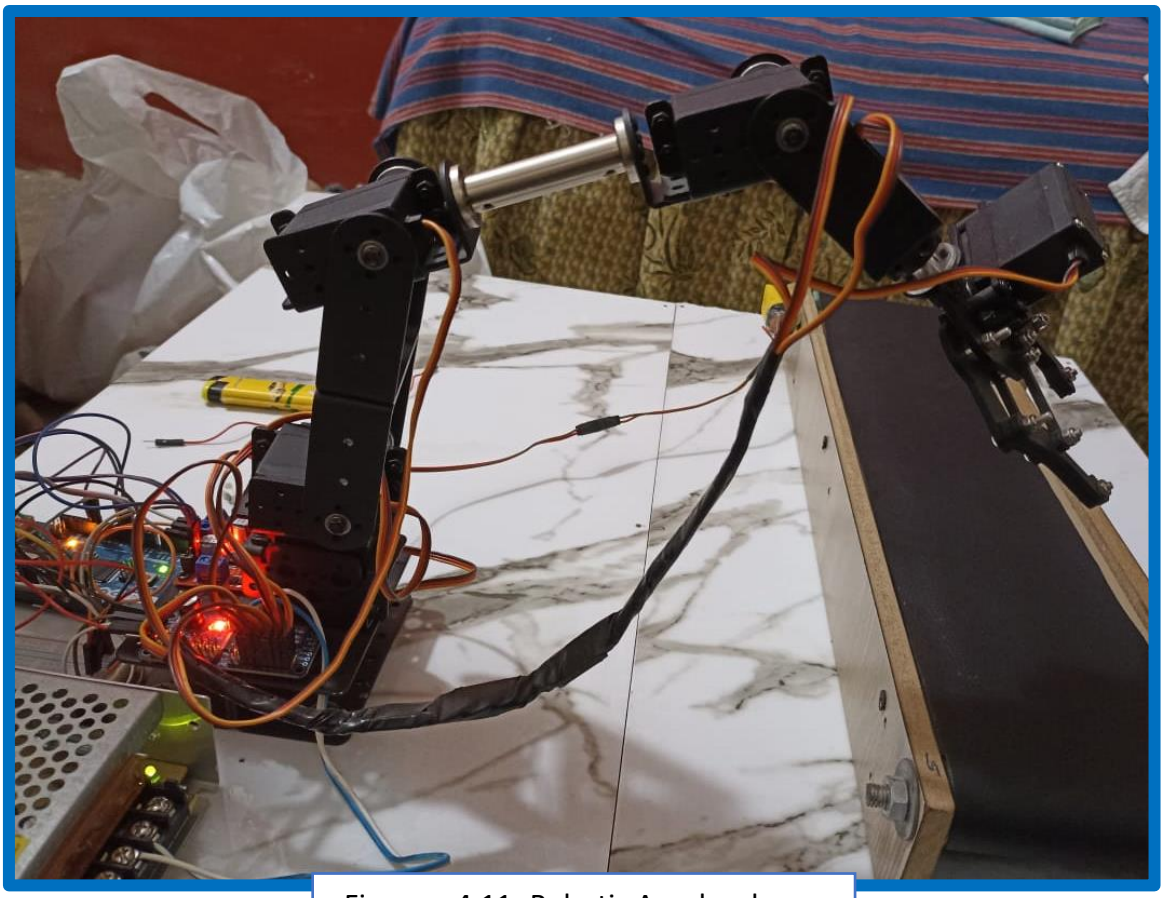
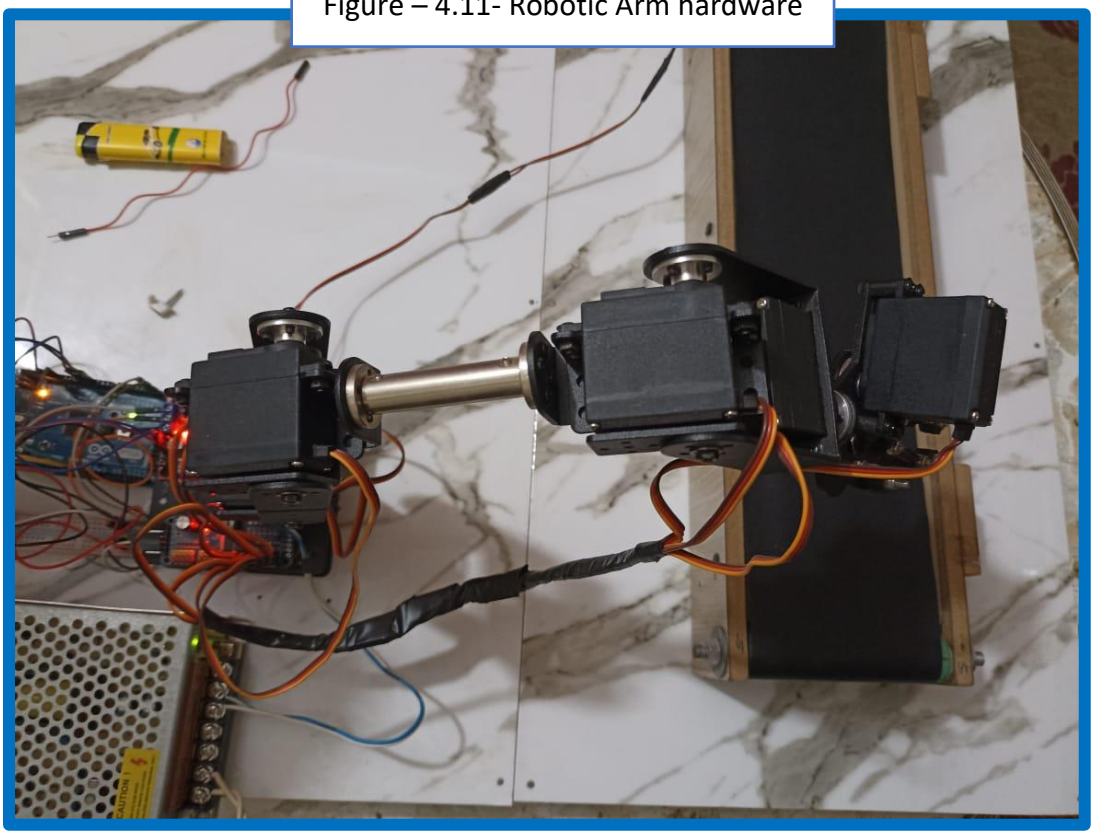
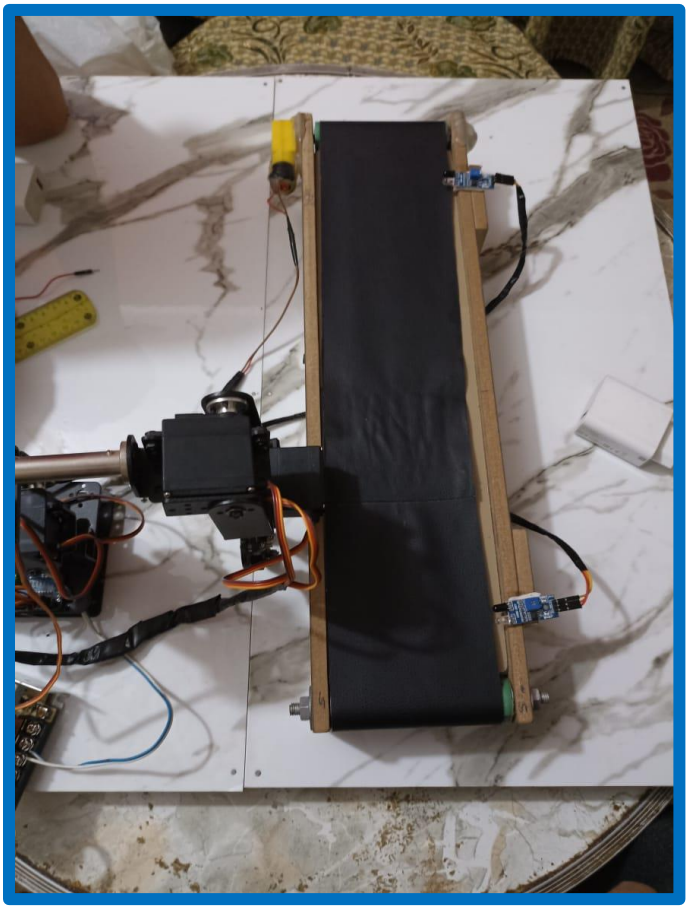


Figure – 4.11- Robotic Arm hardware





Chapter 5: Conclusion

The ROBOARM project represents a significant milestone in the advancement of automated robotic systems for industrial applications. Our dedicated team has developed a robotic arm that not only meets but exceeds the stringent demands of modern industrial environments. The journey from conceptualization to implementation has been marked by meticulous planning, rigorous testing, and continuous improvement, culminating in a system that stands out in terms of performance, accuracy, and usability.

Section:5.1 Summary of Achievements

The development and deployment of ROBOARM have yielded several noteworthy achievements:

- Enhanced Performance: The robotic arm demonstrated exceptional speed and load-handling capabilities, completing tasks significantly faster than manual processes while maintaining stability and reliability under various load conditions.
- High Precision: The arm's advanced sensor system and precise control mechanisms ensured minimal positional and repetition errors, making it highly suitable for tasks requiring exact placement and alignment.

- User-Friendly Design: The intuitive user interface and control mechanisms, including the desktop software, mobile application, and joystick controls, facilitate ease of operation and quick adoption by users. The ability to automate repetitive tasks further enhanced operational efficiency.

Section:5.2 Impact on Industry

The successful implementation of ROBOARM has the potential to revolutionize material handling processes in various industries. By automating repetitive and physically demanding tasks, the robotic arm can significantly enhance productivity, reduce human error, and improve workplace safety. The precision and reliability of ROBOARM make it an invaluable asset in environments where accuracy and efficiency are paramount.

Section:5.3 Future Prospects

Building on the success of this project, several avenues for future development and enhancement have been identified:

- Increased Load Capacity: Future iterations of ROBOARM could focus on increasing the load capacity to handle heavier objects, broadening its applicability across more industrial sectors.

- Advanced Sensor Integration: Incorporating more advanced sensor technologies could further enhance the arm's ability to detect and respond to its environment, improving accuracy and operational efficiency.
- Expanded Software Capabilities: Enhancing the software to include more complex automation tasks and integration with other industrial systems could unlock new possibilities for ROBOARM, making it even more versatile and powerful.

Section:5.4 Concluding Remarks

The ROBOARM project has been a resounding success, showcasing the potential of robotics to transform industrial processes. The collaborative efforts of our team have resulted in a sophisticated, reliable, and user-friendly robotic arm that addresses key challenges in material handling. As we look to the future, the foundation laid by this project provides a solid platform for ongoing innovation and development in the field of industrial automation.

In conclusion, the ROBOARM stands as a testament to the power of technological innovation and teamwork. Its successful deployment marks a significant step forward in our journey towards more efficient, safe, and productive industrial environments. We are confident that the advancements made through this project will inspire further research and development, driving the continued evolution of robotic systems in industry.

Thank you

Appendix

Arduino Code

```
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>
#include <SoftwareSerial.h>

// Define the PCA9685 PWM Servo Driver
SoftwareSerial Bluetooth(10, 11);
Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();

// Servo parameters
#define SERVOMIN  125 // Minimum pulse length
#define SERVOMAX  575 // Maximum pulse length
/*
// Define the buttons
const int button1 = 22; // save movant
const int button2 = 23; // clear/reset movant
const int stopButton = 24; // New stop button
const int playButton = 25; // New play button
const int move1 = 26;
const int move2 = 27;
*/
// Define variables
const int startSensorPin = 2;  // Digital pin for the "Start" IR sensor
const int stopSensorPin = 3;   // Digital pin for the "Stop" IR sensor
const int motorPin1 = 4;       // Motor control pin 1
const int motorPin2 = 5;       // Motor control pin 2
const int enablePin = 6 ; // Connect to enable pin (ENA) of L298N
int button1Pressed = 0;  // counter for saved positions
bool playButton = false; // Flag for the play button
bool stopButtonPressed = false; // Flag for the stop button
bool move1Pressed = false; // Flag for the move1 button
bool move2Pressed = false; // Flag for the move2 button
bool startSensorState; //Flag for "Start" IR sensor
bool stopSensorState; //Flag for "Stop" IR sensor
bool initialMoveDone = false; // Flag to track whether the initial movement has
been completed
bool keepMoving = true;  // Flag to control the loop
int savedPositionCount = 0; // Define a counter for saved positions
String dataIn = "";  // to control each servo
int speedDelay = 30;
```

```
// Define JoySticks
int joystick_x1 = A0;
int joystick_y1 = A1;
int joystick_x2 = A2;
int joystick_y2 = A3;
int joystick_x3 = A4;
int joystick_y3 = A5;

// Define variables for angles of the JoySticks
int x1_axis_degree = 121;
int y1_axis_degree = 90;
int x2_axis_degree = 21;
int y2_axis_degree = 179;
int x3_axis_degree = 10;
int y3_axis_degree = 179;

// Define speed control variables for each servo
int x1_speed = 4;
int y1_speed = 4;
int x2_speed = 4;
int y2_speed = 6;
int x3_speed = 6;
int y3_speed = 6;

// Define arrays to store servo positions
int servo01SP[2];
int servo02SP[2];
int servo03SP[2];
int servo04SP[2];
int servo05SP[2];
int servo06SP[2];

// Function to convert angle to pulse width
uint16_t angleToPulse(uint16_t angle) {
  return map(angle, 0, 180, SERVOMIN, SERVOMAX);
}

// Function to move servo's to their postions
void moveServo(int servoNum, int currentAngle, int initialAngle, int finalAngle,
int delayTime) {
  if (!initialMoveDone) {
    // Move from current position to initial angle
    if (currentAngle < initialAngle) {
      for (int angle = currentAngle; angle <= initialAngle; angle++) {
        pwm.setPWM(servoNum, 0, angleToPulse(angle));
```

```
        delay(delayTime);
      }
    } else {
      for (int angle = currentAngle; angle >= initialAngle; angle--) {
        pwm.setPWM(servoNum, 0, angleToPulse(angle));
        delay(delayTime);
      }
    }
    currentAngle = initialAngle;
    initialMoveDone = true;  // Set the flag to true after the initial move
  }

  // Move repeatedly between initialAngle and finalAngle
  while (keepMoving) {
    if(currentAngle < initialAngle){
    for (int angle = initialAngle; angle <= finalAngle; angle++) {
      pwm.setPWM(servoNum, 0, angleToPulse(angle));
      delay(delayTime);
      if (!keepMoving) return;  // Exit if the stop condition is met
    }
    }
    else{
    for (int angle = finalAngle; angle >= initialAngle; angle--) {
      pwm.setPWM(servoNum, 0, angleToPulse(angle));
      delay(delayTime);
      if (!keepMoving) return;  // Exit if the stop condition is met
    }
    }
  }
}

// Function to handle stopping the servo motion
void stopServo() {
  keepMoving = false;  // Set the flag to false to stop the loop
  initialMoveDone = false;  // Reset the flag to allow initial movement next time
  Bluetooth.println("Playback Stopped");
}

// function to control with servos via Joystick
void joystickMode (){
  int joystick_x_value1 = analogRead(joystick_x1);
  int joystick_y_value1 = analogRead(joystick_y1);
  int joystick_x_value2 = analogRead(joystick_x2);
  int joystick_y_value2 = analogRead(joystick_y2);
  int joystick_x_value3 = analogRead(joystick_x3);
```

```
  int joystick_y_value3 = analogRead(joystick_y3);

  // Update servo positions with speed control, preventing simultaneous movement
  if (abs(joystick_x_value1 - 512) > abs(joystick_y_value1 - 512)) {
    x1_axis_degree = min(199, max(65, x1_axis_degree + (joystick_x_value1 < 340 ?
-x1_speed : (joystick_x_value1 > 680 ? x1_speed : 0))));
  } else {
    y1_axis_degree = min(150, max(90, y1_axis_degree + (joystick_y_value1 < 340 ?
-y1_speed : (joystick_y_value1 > 680 ? y1_speed : 0))));
  }

  if (abs(joystick_x_value2 - 512) > abs(joystick_y_value2 - 512)) {
    x2_axis_degree = min(90, max(0, x2_axis_degree + (joystick_x_value2 < 340 ?
x2_speed : (joystick_x_value2 > 680 ? -x2_speed : 0))));
  } else {
    y2_axis_degree = min(179, max(90, y2_axis_degree + (joystick_y_value2 < 340 ?
y2_speed : (joystick_y_value2 > 680 ? -y2_speed : 0))));
  }

  if (abs(joystick_x_value3 - 512) > abs(joystick_y_value3 - 512)) {
    x3_axis_degree = min(179, max(0, x3_axis_degree + (joystick_x_value3 < 340 ?
-x3_speed : (joystick_x_value3 > 680 ? x3_speed : 0))));
  } else {
    y3_axis_degree = min(179, max(120, y3_axis_degree + (joystick_y_value3 < 340
? y3_speed : (joystick_y_value3 > 680 ? -y3_speed : 0))));
  }
  // Set servo positions
  pwm.setPWM(15, 0, angleToPulse(x1_axis_degree));
  pwm.setPWM(14, 0, angleToPulse(y1_axis_degree));
  pwm.setPWM(13, 0, angleToPulse(x2_axis_degree));
  pwm.setPWM(12, 0, angleToPulse(y2_axis_degree));
  pwm.setPWM(11, 0, angleToPulse(x3_axis_degree));
  pwm.setPWM(10, 0, angleToPulse(y3_axis_degree));

  delay(10); // Increase delay to slow down servo movement
}

// function to save position
void savePosition(int save1 , int save2 , int save3 , int save4 ,int save5 ,int
save6) {
  if (savedPositionCount < 2) { // Check if there is space in the array
    servo01SP[savedPositionCount] = save1;
    servo02SP[savedPositionCount] = save2;
    servo03SP[savedPositionCount] = save3;
    servo04SP[savedPositionCount] = save4;
```

```
    servo05SP[savedPositionCount] = save5;
    servo06SP[savedPositionCount] = save6;
    savedPositionCount++; // Increment the counter
    if (savedPositionCount == 1)
    {
      Bluetooth.println("Initial angle saved");
    }
    else if(savedPositionCount == 2)
    {
      Bluetooth.println("Final angle saved");
    }
    delay(1000);
  } else {
    Bluetooth.println("No more space for saved positions.");
  }
}

// Function to reset saved positions
void resetSavedPositions() {
  memset(servo01SP, 0, sizeof(servo01SP)); // Clear the array data to 0
  memset(servo02SP, 0, sizeof(servo02SP));
  memset(servo03SP, 0, sizeof(servo03SP));
  memset(servo04SP, 0, sizeof(servo04SP));
  memset(servo05SP, 0, sizeof(servo05SP));
  memset(servo06SP, 0, sizeof(servo06SP));
  savedPositionCount = 0; // Reset the counter
  Serial.println("Saved positions reset");
  Bluetooth.println("Saved positions reset");
}

void savedMovement(){
     if ( savedPositionCount == 0)
    {
      Bluetooth.println("No Positiones Saved Please save  positiones first");
      playButton = false;
      Bluetooth.println("Stop");
    }
    else{
       // Motor control logic
    if (!startSensorState || !stopSensorState || startSensorState ||
stopSensorState) {
    // Start sensor is HIGH
    if(!startSensorState && stopSensorState)
    {
      digitalWrite(motorPin1, HIGH); // Set motor direction forward
```

```
      digitalWrite(motorPin2, LOW);

    }
    else if (!startSensorState && !stopSensorState || startSensorState &&
!stopSensorState){
      digitalWrite(motorPin1, LOW); // Set motor direction forward
      digitalWrite(motorPin2, LOW);

      // paly the movement that user save it
      keepMoving = true; // Set the flag to true to start the loop
      moveServo(15, x1_axis_degree, servo01SP[0], servo01SP[1], 100);
      moveServo(14, y1_axis_degree, servo02SP[0], servo02SP[1], 100);
      moveServo(13, x2_axis_degree, servo03SP[0], servo03SP[1], 100);
      moveServo(12, y2_axis_degree, servo04SP[0], servo04SP[1], 100);
      moveServo(11, x3_axis_degree, servo05SP[0], servo05SP[1], 100);
      moveServo(10, y3_axis_degree, servo06SP[0], servo06SP[1], 100);
      if (Bluetooth.available() > 0 && Bluetooth.readString() == 'S') {
        playButton = false;
        stopServo();
    }
    }
  }
  }
}

void move1(){
      // Motor control logic
  if (!startSensorState || !stopSensorState || startSensorState ||
stopSensorState) {
    // Start sensor is HIGH
    if(!startSensorState && stopSensorState)
    {
      digitalWrite(motorPin1, HIGH); // Set motor direction forward
      digitalWrite(motorPin2, LOW);
    }
    else if (!startSensorState && !stopSensorState || startSensorState &&
!stopSensorState){
      digitalWrite(motorPin1, LOW); // Set motor direction forward
      digitalWrite(motorPin2, LOW);

      // paly the movement that user save it
      keepMoving = true; // Set the flag to true to start the loop
      moveServo(15, x1_axis_degree, 40, 120, 100);
      moveServo(14, y1_axis_degree, 40, 120, 100);
      moveServo(13, x2_axis_degree, 40, 120, 100);
```

```
      moveServo(12, y2_axis_degree, 40, 120, 100);
      moveServo(11, x3_axis_degree, 40, 120, 100);
      moveServo(10, y3_axis_degree, 40, 120, 100);
      if (Bluetooth.available() > 0 && Bluetooth.readString() == 'S') {
        move1Pressed = false;
        stopServo();
    }

    }
  }
}

void move2(){
  // Motor control logic
  if (!startSensorState || !stopSensorState || startSensorState ||
stopSensorState) {
    // Start sensor is HIGH
    if(!startSensorState && stopSensorState)
    {
      digitalWrite(motorPin1, HIGH); // Set motor direction forward
      digitalWrite(motorPin2, LOW);
    }
    else if (!startSensorState && !stopSensorState || startSensorState &&
!stopSensorState){
      digitalWrite(motorPin1, LOW); // Set motor direction forward
      digitalWrite(motorPin2, LOW);

      // paly the movement that user save it
      keepMoving = true; // Set the flag to true to start the loop
      moveServo(15, x1_axis_degree, 40, 120, 100);
      moveServo(14, y1_axis_degree, 40, 120, 100);
      moveServo(13, x2_axis_degree, 40, 120, 100);
      moveServo(12, y2_axis_degree, 40, 120, 100);
      moveServo(11, x3_axis_degree, 40, 120, 100);
      moveServo(10, y3_axis_degree, 40, 120, 100);
      if (Bluetooth.available() > 0 && Bluetooth.readString() == 'S') {
        move2Pressed = false;
        stopServo();
    }
    }
  }
}


void setup() {
```

```
  Serial.begin(9600); // Serial monitor
  Bluetooth.begin(9600); // Default baud rate of the Bluetooth module
  pwm.begin();
  pwm.setPWMFreq(60);  // Analog servos run at ~60 Hz updates
  Wire.begin();
  /*
  // Define buttons as input units
  pinMode(button1, INPUT); // counter for save postions
  pinMode(button2, INPUT); // reset button
  pinMode(stopButton, INPUT); // New stop button
  pinMode(playButton, INPUT); // New play button
  pinMode(move1,INPUT);
  pinMode(move2,INPUT);
  */
  pinMode(startSensorPin, INPUT);
  pinMode(stopSensorPin, INPUT);
  pinMode(motorPin1, OUTPUT);
  pinMode(motorPin2, OUTPUT);
  pinMode(enablePin, OUTPUT);

  // Ensure motor is off at startup
  digitalWrite(motorPin1, LOW);
  digitalWrite(motorPin2, LOW);
  analogWrite(enablePin, 0);

}

void loop() {
  joystickMode();
  int motorSpeed = 255; // Adjust the speed as needed
  // Control the speed of the motor using PWM
  analogWrite(enablePin, motorSpeed);
  // Read the state of the sensors
  startSensorState = digitalRead(startSensorPin);
  stopSensorState = digitalRead(stopSensorPin);

  // Check if button1 is pressed (HIGH), save the potentiometers' position
  /*if (digitalRead(button1) == HIGH) {
    // Save positions as long as button1 is pressed
      savePosition(x1_axis_degree, y1_axis_degree, x2_axis_degree,
y2_axis_degree, x3_axis_degree, y3_axis_degree);
  }

  // Check if the play button is pressed (HIGH)
  if (digitalRead(playButton) == HIGH) {
```

```
    playButton = true; // Resume servo movement
  }

  // Check if the move1 button is pressed (HIGH)
  if (digitalRead(move1) == HIGH) {
    move1Pressed = true; // Resume servo movement
  }

  // Check if the move2 button is pressed (HIGH)
   if (digitalRead(move2) == HIGH) {
    move2Pressed = true; // Resume servo movement
  }

    // Function to reset positions that saved
  if (digitalRead(button2) == HIGH) {
    resetSavedPositions();
  }
*/
  // check if convert to bluetooth control
  if (Bluetooth.available() > 0) {
    char command = Bluetooth.read();
    switch (command) {
      case 'M':
        move1Pressed = true;
        Serial.println("Move 1");
        Bluetooth.println("Move 1");
        break;
      case 'N':
        move2Pressed = true;
        Serial.println("Move 2");
        Bluetooth.println("Move 2");
        break;
      case 'P':
        playButton = true;
        Serial.println("Play mode");
        Bluetooth.println("1-Play mode");
        break;
      case 'V':
          savePosition(x1_axis_degree, y1_axis_degree, x2_axis_degree,
y2_axis_degree, x3_axis_degree, y3_axis_degree);
        break;
      case 'R':
        resetSavedPositions();
        break;
    }
```

```
  }
  // Functio to play the custom move
  if (playButton) {
    savedMovement();
  }

  // Functio to play the move 1
  if (move1Pressed) {
    move1();
  }

  // Functio to play the move 2
  if (move2Pressed) {
    move2();
  }
  delay(100);
}
```

Application

- colors.dart:

```
import 'package:flutter/material.dart';

const Color primaryColor = Color(0xFF3186A0);
const Color backgroundColor = Color(0xFFEFEFEF);
const Color secondryColor = Color(0xFF4D4D4D);
```

- main.dart:

```
import 'dart:io';
import 'dart:math';
import 'package:desktop_window/desktop_window.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:roboarmv2/splach.dart';
import 'package:supabase_flutter/supabase_flutter.dart';
import 'dart:ui' as ui;

Future<void> setFullScreen() async {
  double screenWidth = ui.window.physicalSize.width /
ui.window.devicePixelRatio;
```

```
  double screenHeight = ui.window.physicalSize.height /
ui.window.devicePixelRatio;
  double diagonalInches = sqrt(pow(screenWidth, 2) + pow(screenHeight,
2)) / ui.window.devicePixelRatio;
  double ppi = sqrt(pow(screenWidth, 2) + pow(screenHeight, 2)) /
diagonalInches;
  await
DesktopWindow.setWindowSize(Size(ui.window.physicalSize.width*ppi,
ui.window.physicalSize.height*ppi - 40));
}

Future<void> main() async {
  if(Platform.isWindows){
    WidgetsFlutterBinding.ensureInitialized();
    await setFullScreen();
  }
  await Supabase.initialize(
    url: 'https://xditbsgzrqnyqrlrxset.supabase.co',
    anonKey:
'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6
InhkaXRic2d6cnFueXFybHJ4c2V0Iiwicm9sZSI6ImFub24iLCJpYXQiOjE3MDgxOTI3Nj
csImV4cCI6MjAyMzc2ODc2N30.uwQA5-
Bt4T7Jv5QtjSdkvnOiCIesePoumYEGU3bUv64',
  );
  runApp(MyApp());
}

final supabase = Supabase.instance.client;

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: const Splash(),
      theme: ThemeData(
        fontFamily: "cairo",
      ),
    );
  }
}
```

- splach.dart:

```
import 'package:flutter/material.dart';
import 'package:roboarmv2/login.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'colors.dart';
import 'home.dart';

class Splash extends StatefulWidget {
```

```
  const Splash({Key? key});
  @override
  State<Splash> createState() => _SplashState();
}

class _SplashState extends State<Splash> {
  @override
  void initState() {
    super.initState();
    checkSession();
  }

  checkSession() async {
    final SharedPreferences prefs = await
SharedPreferences.getInstance();
    final String? supabaseSessionToken =
prefs.getString('supabaseSessionToken');

    await Future.delayed(const Duration(milliseconds: 2000));

    if (supabaseSessionToken != null) {
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (context) => home()),
      );
    } else {
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (context) => Login()),
      );
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Container(
        color: backgroundColor,
        child: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            crossAxisAlignment: CrossAxisAlignment.center,
            children: [
              Image.asset(
                "images/roboarmlogo.png",
              ),
              const Text(
                "RoboArm",
                style: TextStyle(
                  color: primaryColor,
                  fontWeight: FontWeight.bold,
                  fontSize: 50,
```

```
                ),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

- login.dart:

```
import 'dart:io';
import 'package:flutter/material.dart';
import 'package:google_sign_in/google_sign_in.dart';
import 'package:roboarmv2/signup.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:supabase_flutter/supabase_flutter.dart';
import 'CustomDialog.dart';
import 'colors.dart';
import 'forgetpassword.dart';
import 'home.dart';
import 'main.dart';

class Login extends StatefulWidget {
  const Login({Key? key}) : super(key: key);
  @override
  State<Login> createState() => _LoginState();
}

Future<void> signInWithGoogleOnDesktop(BuildContext context) async {
  try {
    await supabase.auth.signInWithOAuth(OAuthProvider.google);
    final Session? session = supabase.auth.currentSession;
    final SharedPreferences prefs = await
SharedPreferences.getInstance();
    prefs.setString('supabaseSessionToken', session?.accessToken ??
'');
  } catch (error) {
    showDialog(
      context: context,
      builder: (BuildContext context) {
        return CustomDialog(
          title: 'Google Sign-In Error',
          content: 'An error occurred while signing in with Google.',
        );
      },
    );
  }
}

Future<AuthResponse> _googleSignIn() async {
```

```
  const webClientId = '1004640859410-
ktna75av95thc7bd5552gob9h3rlrrsa.apps.googleusercontent.com';
  final GoogleSignIn googleSignIn = GoogleSignIn(
    serverClientId: webClientId,
  );
  final googleUser = await googleSignIn.signIn();
  final googleAuth = await googleUser!.authentication;
  final accessToken = googleAuth.accessToken;
  final idToken = googleAuth.idToken;
  if (accessToken == null) {
    throw 'No Access Token found.';
  }
  if (idToken == null) {
    throw 'No ID Token found.';
  }
  return supabase.auth.signInWithIdToken(
    provider: OAuthProvider.google,
    idToken: idToken,
    accessToken: accessToken,
  );
}

Future<void> _saveSessionToSharedPreferences(String accessToken) async
{
  final SharedPreferences prefs = await
SharedPreferences.getInstance();
  prefs.setString('supabaseSessionToken', accessToken);
}

Future<void> _googleSignInAndNavigate(BuildContext context) async {
  try {
    final AuthResponse res = await _googleSignIn();
    final Session? session = res.session;
    final User? user = res.user;
    if (session != null) {
      await _saveSessionToSharedPreferences(session.accessToken!);
      Navigator.of(context).pushAndRemoveUntil(
        MaterialPageRoute(builder: (context) => home()),
            (Route<dynamic> route) => false,
      );
    }
  } catch (error, stackTrace) {
    print("Google Sign-In Error: $error");
    showDialog(
      context: context,
      builder: (BuildContext context) {
        return CustomDialog(
          title: 'Google Sign-In Error',
          content: 'An error occurred while signing in with Google.',
        );
      },
    );
```

```
  }
}

class _LoginState extends State<Login> {
  TextEditingController email = TextEditingController();
  TextEditingController password = TextEditingController();
  GlobalKey<FormState> formState = GlobalKey();
  bool visibility = false;

  @override
  Widget build(BuildContext context) {
    BuildContext scaffoldContext;
    return MaterialApp(
      color: backgroundColor,
      home: Scaffold(
        body: Builder(
          builder: (BuildContext context) {
            scaffoldContext = context;
            return SingleChildScrollView(
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  Container(
                    alignment: Alignment.center,
                    child: Column(
                      children: [
                        Image.asset(
                          "images/roboarmlogo.png",
                          height: MediaQuery.of(context).size.height /
5,
                        ),
                        const Text(
                          "RoboArm",
                          style: TextStyle(
                            color: primaryColor,
                            fontWeight: FontWeight.bold,
                            fontSize: 30,
                          ),
                        ),
                      ],
                    ),
                  ),
                  const Padding(
                    padding: EdgeInsets.only(left: 15),
                    child: Text(
                      "Login",
                      style: TextStyle(
                        fontSize: 35,
                        fontWeight: FontWeight.bold,
                        color: primaryColor,
                      ),
                    ),
```

```
                    ),
                    Padding(
                      padding: const EdgeInsets.all(15),
                      child: Form(
                        autovalidateMode:
AutovalidateMode.onUserInteraction,
                        child: Column(
                          crossAxisAlignment:
CrossAxisAlignment.stretch,
                          children: [
                            TextFormField(
                              validator: (emailValue) {
                                if (emailValue!.isEmpty) {
                                  return "Field is empty!";
                                }
                                return null;
                              },
                              decoration: InputDecoration(
                                border: const OutlineInputBorder(),
                                labelText: "Email",
                                labelStyle: const TextStyle(
                                  color: secondryColor,
                                  fontSize: 20,
                                  fontWeight: FontWeight.bold,
                                ),
                                prefixIcon: const Icon(
                                  Icons.email,
                                  color: secondryColor,
                                  size: 35,
                                ),
                                focusedBorder: OutlineInputBorder(
                                  borderSide: const BorderSide(color:
secondryColor, width: 2),

                                  borderRadius:
BorderRadius.circular(18.0),
                                ),
                                enabledBorder: OutlineInputBorder(
                                  borderSide: const BorderSide(color:
secondryColor, width: 2),

                                  borderRadius:
BorderRadius.circular(18.0),
                                ),
                              ),
                              style: const TextStyle(
                                fontSize: 22,
                                fontWeight: FontWeight.bold,
                                color: secondryColor,
                              ),
                              controller: email,
                            ),
                            const SizedBox(height: 15),
                            TextFormField(
```

```
                                validator: (passwordValue) {
                                  if (passwordValue!.isEmpty) {
                                    return "Field is empty!";
                                  }
                                  return null;
                                },
                                obscureText: !visibility,
                                decoration: InputDecoration(
                                  border: const OutlineInputBorder(),
                                  labelText: "Password",
                                  labelStyle: const TextStyle(
                                    color: secondryColor,
                                    fontSize: 20,
                                    fontWeight: FontWeight.bold,
                                  ),
                                  prefixIcon: const Icon(
                                    Icons.lock,
                                    color: secondryColor,
                                    size: 35,
                                  ),
                                  suffixIcon: IconButton(
                                    icon: Icon(
                                      visibility
                                          ? Icons.visibility_off_outlined
                                          : Icons.visibility_outlined,
                                      color: secondryColor,
                                      size: 35,
                                    ),
                                    onPressed: () {
                                      setState(() {
                                        visibility = !visibility;
                                      });
                                    },
                                  ),
                                  focusedBorder: OutlineInputBorder(
                                    borderSide: const BorderSide(color:
secondryColor, width: 2),
                                    borderRadius:
BorderRadius.circular(18.0),
                                  ),
                                  enabledBorder: OutlineInputBorder(
                                    borderSide: const BorderSide(color:
secondryColor, width: 2),
                                    borderRadius:
BorderRadius.circular(18.0),
                                  ),
                                ),
                                style: const TextStyle(
                                  fontSize: 22,
                                  fontWeight: FontWeight.bold,
                                  color: secondryColor,
                                ),
```

```
                            controller: password,
                          ),
                        ],
                      ),
                    ),
                  ),
                  Row(
                    mainAxisAlignment: MainAxisAlignment.end,
                    children: [
                      GestureDetector(
                        onTap: () {
                          Navigator.push(
                            context,
                            MaterialPageRoute(builder: (context) =>
const forgetpassword()),
                          );
                        },
                        child: const Text(
                          "Forget password?",
                          style: TextStyle(
                            fontSize: 18,
                            fontWeight: FontWeight.w700,
                            color: secondryColor,
                            decoration: TextDecoration.underline,
                          ),
                        ),
                      ),
                    ],
                  ),
                  const SizedBox(height: 15),
                  MaterialButton(
                    onPressed: () async {
                      try {
                        final AuthResponse res = await
supabase.auth.signInWithPassword(
                          email: email.text,
                          password: password.text,
                        );
                        final Session? session = res.session;
                        final User? user = res.user;

                        final SharedPreferences prefs = await
SharedPreferences.getInstance();
                        prefs.setString('supabaseSessionToken',
session?.accessToken ?? '');

                        Navigator.of(context).pushAndRemoveUntil(
                          MaterialPageRoute(builder: (context) =>
home()),
                              (Route<dynamic> route) => false,
                        );
                      } catch (error) {
```

```
                        showDialog(
                          context: context,
                          builder: (BuildContext context) {
                            return CustomDialog(
                              title: 'Wrong entry!',
                              content: 'Check the email or password.',
                            );
                          },
                        );
                      }

                    },

                    child: Container(
                      decoration: BoxDecoration(
                        color: primaryColor,
                        borderRadius: BorderRadius.circular(10.0),
                        border: Border.all(color: secondryColor,
width: 2.0),
                      ),
                      padding: const EdgeInsets.symmetric(horizontal:
20, vertical: 15),
                      child: const Text(
                        "Sign In",
                        style: TextStyle(fontSize: 35, fontWeight:
FontWeight.bold, color: Colors.white),
                      ),
                    ),
                  ),
                  const SizedBox(height: 20),
                  Row(
                    mainAxisAlignment: MainAxisAlignment.center,
                    children: [
                      const Text(
                        "Do not have an account?",
                        style: TextStyle(
                          fontSize: 18,
                          fontWeight: FontWeight.w700,
                          color: secondryColor,
                        ),
                      ),
                      GestureDetector(
                        onTap: () {
                          Navigator.push(context,
                              MaterialPageRoute(builder: (context) =>
const SignUp()));
                        },
                        child: const Padding(
                          padding: EdgeInsets.only(left: 5),
                          child: Text(
                            "Sign up",
                            style: TextStyle(
```

```
                                fontSize: 18,
                                fontWeight: FontWeight.bold,
                                color: primaryColor,
                                decoration: TextDecoration.underline,
                              ),
                            ),
                          ),
                        ),
                      ],
                    ),
                    const SizedBox(height: 15),
                    Center(
                      child: Column(
                        mainAxisAlignment: MainAxisAlignment.center,
                        crossAxisAlignment: CrossAxisAlignment.center,
                        children: [
                          Row(
                            mainAxisAlignment: MainAxisAlignment.center,
                            children: [
                              Container(
                                margin: const EdgeInsets.only(right: 5),
                                width: 30,
                                color: primaryColor,
                                height: 2,
                              ),
                              const Text(
                                "Or Login with",
                                style: TextStyle(
                                  color: secondryColor,
                                  fontWeight: FontWeight.w700,
                                  fontSize: 20,
                                ),
                              ),
                              Container(
                                margin: const EdgeInsets.only(left: 5),
                                width: 30,
                                color: primaryColor,
                                height: 2,
                              ),
                            ],
                          ),
                          const SizedBox(height: 13),
                          MaterialButton(
                            onPressed: () async {
                              if(Platform.isAndroid){
                                await _googleSignInAndNavigate(context);
                              } else if (Platform.isWindows){
                                await
signInWithGoogleOnDesktop(context);
                              }
                            },
                            color: backgroundColor,
```

```
                            padding: const EdgeInsets.all(15),
                            shape: RoundedRectangleBorder(
                              side: const BorderSide(
                                color: secondryColor,
                                width: 2.0,
                              ),
                              borderRadius: BorderRadius.circular(8.0),
                            ),
                            child: Image.asset(
                              "images/google_logo.png",
                              width: 50,
                            ),
                          ),
                        ],
                      ),
                    ],
                  ),
                ),
              );
            },
          ),
        ),
      );
  }
}
```

- signup.dart:

```
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:supabase_flutter/supabase_flutter.dart';
import 'CustomDialog.dart';
import 'colors.dart';
import 'login.dart';
import 'main.dart';

void main() {
  runApp(const SignUp());
}

class SignUp extends StatefulWidget {
  const SignUp({Key? key}) : super(key: key);
  @override
  State<SignUp> createState() => _SignUpState();
}

class _SignUpState extends State<SignUp> {

  TextEditingController email = TextEditingController();
  TextEditingController password = TextEditingController();
  TextEditingController confirmPassword = TextEditingController();
```

```
  TextEditingController username = TextEditingController();
  GlobalKey<FormState> formState = GlobalKey<FormState>();

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: const Text(
            "New Account",
            style: TextStyle(
              color: primaryColor,
              fontWeight: FontWeight.bold,
              fontSize: 30,
            ),
          ),
          backgroundColor: secondryColor,
          actions: [
            IconButton(
              onPressed: () {
                Navigator.pop(context);
              },
              icon: const Icon(
                Icons.arrow_back_ios_new,
                color: primaryColor,
                size: 30,
              ),
            ),
          ],
        ),
        body: SingleChildScrollView(
          child: Center(
            child: Column(
              mainAxisAlignment: MainAxisAlignment.start,
              children: [
                Padding(
                  padding: const EdgeInsets.all(15),
                  child: Form(
                    key: formState,
                    autovalidateMode:
AutovalidateMode.onUserInteraction,
                    child: Column(
                      crossAxisAlignment: CrossAxisAlignment.stretch,
                      children: [
                        TextFormField(
                          validator: (emailValue) {
                            if (emailValue!.isEmpty) {
                              return "Field is empty!";
                            } else if (!RegExp(r"^[a-zA-Z0-9_.+-]+@[a-
zA-Z0-9-]+\.[a-zA-Z0-9-.]+$").hasMatch(emailValue)) {
                              return "Enter a valid email
address!\nexample@mail.com";
```

```
                        }
                        return null;
                      },
                      decoration: InputDecoration(
                        border: const OutlineInputBorder(),
                        labelText: "Email",
                        labelStyle: const TextStyle(
                          color: secondryColor,
                          fontSize: 20,
                          fontWeight: FontWeight.bold,
                        ),
                        prefixIcon: const Icon(
                          Icons.email,
                          color: secondryColor,
                          size: 35,
                        ),
                        focusedBorder: OutlineInputBorder(
                          borderSide: const BorderSide(color:
secondryColor, width: 2),
                          borderRadius:
BorderRadius.circular(18.0),
                        ),
                        enabledBorder: OutlineInputBorder(
                          borderSide: const BorderSide(color:
secondryColor, width: 2),
                          borderRadius:
BorderRadius.circular(18.0),
                        ),
                      ),
                      style: const TextStyle(
                        fontSize: 22,
                        fontWeight: FontWeight.bold,
                        color: secondryColor,
                      ),
                      controller: email,
                    ),
                    const SizedBox(height: 15),
                    TextFormField(
                      validator: (passwordValue) {
                        if (passwordValue!.isEmpty) {
                          return "Field is empty!";
                        } else if (passwordValue.length < 8) {
                          return "Password must be at least 8
characters long!";
                        } else if (!RegExp(r'^(?=.*\d)(?=.*[a-
z])(?=.*[A-Z])(?=.*[a-zA-Z]).{8,}$').hasMatch(passwordValue)) {
                          return "Password must contain at least
one uppercase \nletter (A, Z), one lowercase letter (a, z), \none
digit (0, 9), and one special character (#, @, ., &)!";
                        }
                        return null;
                      },
```

```
                            decoration: InputDecoration(
                              border: const OutlineInputBorder(),
                              labelText: "Password",
                              labelStyle: const TextStyle(
                                color: secondryColor,
                                fontSize: 20,
                                fontWeight: FontWeight.bold,
                              ),
                              prefixIcon: const Icon(
                                Icons.lock,
                                color: secondryColor,
                                size: 35,
                              ),
                              focusedBorder: OutlineInputBorder(
                                borderSide: const BorderSide(color:
secondryColor, width: 2),
                                borderRadius:
BorderRadius.circular(18.0),
                              ),
                              enabledBorder: OutlineInputBorder(
                                borderSide: const BorderSide(color:
secondryColor, width: 2),
                                borderRadius:
BorderRadius.circular(18.0),
                              ),
                            ),
                            style: const TextStyle(
                              fontSize: 22,
                              fontWeight: FontWeight.bold,
                              color: secondryColor,
                            ),
                            controller: password,
                            obscureText: true,
                          ),
                          const SizedBox(height: 15),
                          TextFormField(
                            validator: (confirmPasswordValue) {
                              if (confirmPasswordValue!.isEmpty) {
                                return "Field is empty!";
                              }
                              return null;
                            },
                            decoration: InputDecoration(
                              border: const OutlineInputBorder(),
                              labelText: "Confirm Password",
                              labelStyle: const TextStyle(
                                color: secondryColor,
                                fontSize: 20,
                                fontWeight: FontWeight.bold,
                              ),
                              prefixIcon: const Icon(
                                Icons.lock_reset,
```

```
                                color: secondryColor,
                                size: 35,
                              ),
                              focusedBorder: OutlineInputBorder(
                                borderSide: const BorderSide(color:
secondryColor, width: 2),
                                borderRadius:
BorderRadius.circular(18.0),
                              ),
                              enabledBorder: OutlineInputBorder(
                                borderSide: const BorderSide(color:
secondryColor, width: 2),
                                borderRadius:
BorderRadius.circular(18.0),
                              ),
                            ),
                            style: const TextStyle(
                              fontSize: 22,
                              fontWeight: FontWeight.bold,
                              color: secondryColor,
                            ),
                            controller: confirmPassword,
                            obscureText: true,
                          ),
                          const SizedBox(height: 15),
                          TextFormField(
                            validator: (usernameValue) {
                              if (usernameValue!.isEmpty) {
                                return "Field is empty!";
                              }
                              return null;
                            },
                            decoration: InputDecoration(
                              border: const OutlineInputBorder(),
                              labelText: "Username",
                              labelStyle: const TextStyle(
                                color: secondryColor,
                                fontSize: 20,
                                fontWeight: FontWeight.bold,
                              ),
                              prefixIcon: const Icon(
                                Icons.account_circle_outlined,
                                color: secondryColor,
                                size: 35,
                              ),
                              focusedBorder: OutlineInputBorder(
                                borderSide: const BorderSide(color:
secondryColor, width: 2),
                                borderRadius:
BorderRadius.circular(18.0),
                              ),
                              enabledBorder: OutlineInputBorder(
```

```
                                borderSide: const BorderSide(color:
secondryColor, width: 2),
                                borderRadius:
BorderRadius.circular(18.0),
                              ),
                            ),
                            style: const TextStyle(
                              fontSize: 22,
                              fontWeight: FontWeight.bold,
                              color: secondryColor,
                            ),
                            controller: username,
                          ),
                        ],
                      ),
                    ),
                  ),
                  const SizedBox(height: 25),
                  MaterialButton(
                    onPressed: () async {
                      if(password.text == confirmPassword.text){
                        try {
                          final AuthResponse res = await
supabase.auth.signUp(
                            email: email.text,
                            password: password.text,
                            data: {'username': username.text},
                          );
                          final Session? session = res.session;
                          final User? user = res.user;
                          final SharedPreferences prefs = await
SharedPreferences.getInstance();
                          prefs.setString('supabaseSessionToken',
session?.accessToken ?? '');
                          showDialog(
                            context: context,
                            builder: (BuildContext context) {
                              return CustomDialog(
                                title: 'Confirm your email',
                                content: 'We have sent you an email that
you entered to verify your email, confirm and log in.',
                                onOkPressed: (BuildContext context) {

Navigator.of(context).pushAndRemoveUntil(
                                    MaterialPageRoute(builder: (context)
=> Login()),

                                        (Route<dynamic> route) => false,
                                  );
                                },
                              );
                            },
                          );
```

```
                        } catch (error) {
                          showDialog(
                            context: context,
                            builder: (BuildContext context) {
                              return CustomDialog(
                                title: 'Error!',
                                content: 'An error occurred: $error',
                              );
                            },
                          );
                        }
                      }else {
                        showDialog(
                          context: context,
                          builder: (BuildContext context) {
                            return CustomDialog(
                              title: 'Entry is wrong!',
                              content: 'The password and confirm
password are not matches!\nPlease try again.',
                            );
                          },
                        );
                      }
                    },

                    child: Container(
                      decoration: BoxDecoration(
                        color: primaryColor,
                        borderRadius: BorderRadius.circular(10.0),
                        border: Border.all(color: secondryColor, width:
2.0),
                      ),
                      padding: const EdgeInsets.symmetric(horizontal:
20, vertical: 15),
                      child: const Text(
                        "Sign Up",
                        style: TextStyle(fontSize: 35, fontWeight:
FontWeight.bold, color: Colors.white),
                      ),
                    ),
                  ),
                ],
              ),
            ),
          ),
        ),
      );
  }
}
```

- forgetpassword.dart:

```dart
import 'package:flutter/material.dart';
import 'package:google_sign_in/google_sign_in.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:supabase_flutter/supabase_flutter.dart';
import 'CustomDialog.dart';
import 'colors.dart';
import 'login.dart';
import 'main.dart';

void main() {
  runApp(const forgetpassword());
}

class forgetpassword extends StatefulWidget {
  const forgetpassword({Key? key}) : super(key: key);
  @override
  State<forgetpassword> createState() => _ForgetPasswordState();
}

class _ForgetPasswordState extends State<forgetpassword> {
  TextEditingController email = TextEditingController();
  GlobalKey<FormState> fieldState = GlobalKey();

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: const Text(
            "Forget Password",
            style: TextStyle(
              color: primaryColor,
              fontWeight: FontWeight.bold,
              fontSize: 30,
            ),
          ),
          backgroundColor: secondryColor,
          actions: [
            IconButton(
              onPressed: () {
                Navigator.pop(context);
              },
              icon: const Icon(
                Icons.arrow_back_ios_new,
                color: primaryColor,
                size: 30,
              ),
            ),
          ],
        ),
        body: Builder(
          builder: (scaffoldContext) => Center(
```

```
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Container(
                  alignment: Alignment.center,
                  padding: const EdgeInsets.all(15.0),
                  child: const Text(
                    "Enter the email associated with the account to
send the reset password email.",
                    style: TextStyle(
                      color: secondryColor,
                      fontWeight: FontWeight.bold,
                      fontSize: 23,
                    ),
                  ),
                ),
                Padding(
                  padding: const EdgeInsets.all(15),
                  child: Container(
                    child: TextFormField(
                      validator: (emailValue) {
                        if (emailValue!.isEmpty) {
                          return "Field is empty!";
                        }
                        return null;
                      },
                      decoration: InputDecoration(
                        border: const OutlineInputBorder(),
                        labelText: "Email",
                        labelStyle: const TextStyle(
                          color: secondryColor,
                          fontSize: 20,
                          fontWeight: FontWeight.bold,
                        ),
                        prefixIcon: const Icon(
                          Icons.email,
                          color: secondryColor,
                          size: 35,
                        ),
                        focusedBorder: OutlineInputBorder(
                          borderSide: const BorderSide(color:
secondryColor, width: 2),
                          borderRadius: BorderRadius.circular(18.0),
                        ),
                        enabledBorder: OutlineInputBorder(
                          borderSide: const BorderSide(color:
secondryColor, width: 2),
                          borderRadius: BorderRadius.circular(18.0),
                        ),
                      ),
                      style: const TextStyle(
                        fontSize: 22,
```

```
                        fontWeight: FontWeight.bold,
                        color: secondryColor,
                      ),
                      controller: email,
                    ),
                  ),
                ),
                MaterialButton(
                  onPressed: ()  async {
                    if (!(email.text!.isEmpty)) {
                      supabase.auth.resetPasswordForEmail(email.text);
                      final SharedPreferences prefs = await
SharedPreferences.getInstance();
                      bool containsToken =
prefs.containsKey('supabaseSessionToken');
                      if (containsToken) {
                        await prefs.remove('supabaseSessionToken');
                        final User? user = supabase.auth.currentUser;
                        if(user!.appMetadata?["provider"] ==
"google"){
                          GoogleSignIn().signOut();
                        }
                        await supabase.auth.signOut();
                      }
                      showDialog(
                        context: context,
                        builder: (BuildContext context) {
                          return CustomDialog(
                            title: 'Sending reset email complete',
                            content: 'We have sent you an email to
reset your password\nReset and return to log in.',
                            onOkPressed: (BuildContext context) {

Navigator.of(context).pushAndRemoveUntil(
                                MaterialPageRoute(builder: (context)
=> Login()),
                                    (Route<dynamic> route) => false,
                              );
                            },
                          );
                        },
                      );
                    } else {
                      showDialog(
                        context: context,
                        builder: (BuildContext context) {
                          return CustomDialog(
                            title: 'Wrong Entry!',
                            content: 'Please Write your email.',
                          );
                        },
                      );
```

```
                    }
                  },
                  child: Container(
                    decoration: BoxDecoration(
                      color: primaryColor,
                      borderRadius: BorderRadius.circular(10.0),
                      border: Border.all(color: secondryColor, width:
2.0),
                    ),
                    padding: const EdgeInsets.symmetric(horizontal:
20, vertical: 15),
                    child: const Text(
                      "Send reset password email",
                      style: TextStyle(fontSize: 25, fontWeight:
FontWeight.bold, color: Colors.white),
                    ),
                  ),
                ),
              ],
            ),
          ),
        ),
      ),
    );
  }
}
```

- home.dart:

```
import 'dart:io';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:google_sign_in/google_sign_in.dart';
import 'package:permission_handler/permission_handler.dart';
import 'package:roboarmv2/ControlPage.dart';
import 'package:roboarmv2/AutoPage.dart';
import 'package:roboarmv2/HomeFragment.dart';
import 'package:roboarmv2/ManualPage.dart';
import 'package:roboarmv2/splach.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:supabase_flutter/supabase_flutter.dart';
import 'Account.dart';
import 'Bluetooth.dart';
import 'Logs.dart';
import 'Settings.dart';
import 'colors.dart';
import 'main.dart';

void main() {
  runApp(const home());
}
```

```
class home extends StatefulWidget {
  const home({Key? key}) : super(key: key);

  @override
  State<home> createState() => _HomeState();
}

class _HomeState extends State<home> {
  Future<void> _requestPermissions() async {
    Map<Permission, PermissionStatus> statuses = await [
      Permission.bluetooth,
      Permission.bluetoothScan,
      Permission.locationWhenInUse,
      Permission.bluetoothConnect,
    ].request();
    statuses.forEach((permission, status) {
      print('$permission: $status');
    });
    if (statuses[Permission.bluetooth] == PermissionStatus.granted &&
        statuses[Permission.bluetoothScan] == PermissionStatus.granted
&&
        statuses[Permission.locationWhenInUse] ==
PermissionStatus.granted &&
        statuses[Permission.bluetoothConnect] ==
PermissionStatus.granted) {
      print("All permissions granted!");
    } else {
      print("Permissions not granted");
    }
  }

  @override
  void initState() {
    super.initState();
    _requestPermissions();
  }

  int _currentIndex = 0;

  final List<String> itemName = ["Home", "Account",
"Bluetooth","Logs", "Settings"];

  final List<IconData> icons = [
    Icons.home,
    Icons.account_circle_outlined,
    Icons.bluetooth,
    Icons.code,
    Icons.settings,
  ];

  final List<Widget> pages = [
    home(),
```

```
    Account(),
    Bluetooth(),
    Logs(),
    Settings(),
  ];

  int _selectedIndex = 0;

  void _onItemTapped(int index) {
    setState(() {
      _selectedIndex = index;
    });
  }

  static final List<Widget> _widgetOptions = <Widget>[
    HomeFragment(),
    ControlPage(),
    AutoPage(),
    ManualPage(),
  ];

  final User? user = supabase.auth.currentUser;

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: const Text(
            "RoboArm",
            style: TextStyle(
              color: primaryColor,
              fontWeight: FontWeight.bold,
              fontSize: 30,
            ),
          ),
          backgroundColor: secondryColor,
          centerTitle: true,
          actions: [
            Image.asset(
              "images/roboarmlogo.png",
              height: MediaQuery.of(context).size.height / 10,
            ),
          ],
        ),
        drawer: Drawer(
          child: Container(
            color: secondryColor,
            child: Column(
              children: [
                Container(
                  padding: EdgeInsets.only(left: 15, top: 20, right:
```

```
15, bottom: 20),
                    child: Row(
                      children: [
                        Image.asset(
                          "images/roboarmlogo.png",
                          height: MediaQuery.of(context).size.height /
10,
                        ),
                        SizedBox(width: 15),
                        Text(
                          "RoboArm",
                          style: TextStyle(
                            fontWeight: FontWeight.bold,
                            fontSize: 30,
                            color: primaryColor,
                          ),
                        ),
                      ],
                    ),
                  ),
                  Container(color: backgroundColor, height: 2),
                  Expanded(
                    child: ListView.builder(
                      itemCount: itemName.length,
                      itemBuilder: (context, i) {
                        return Card(
                          shape: RoundedRectangleBorder(
                            borderRadius: BorderRadius.circular(10),
                          ),
                          color: backgroundColor,
                          child: ListTile(
                            title: Text(
                              itemName[i],
                              style: TextStyle(
                                fontWeight: FontWeight.bold,
                                fontSize: 25,
                                color: secondryColor,
                              ),
                            ),
                            leading: Icon(
                              icons[i],
                              color: primaryColor,
                              size: 30,
                            ),
                            onTap: () {
                              if (_currentIndex == i) {

Navigator.of(context).pushNamedAndRemoveUntil(
                                    '/',
                                        (Route<dynamic> route) => false,
                                  );
                                } else {
```

```
                                Navigator.push(
                                  context,
                                  MaterialPageRoute(builder: (context)
=> pages[i]),
                                );
                              }
                            },
                          ),
                          elevation: 1,
                        );
                      },
                    ),
                  ),
                  MaterialButton(
                    padding: EdgeInsets.only(bottom: 20),
                    onPressed: () async {
                      if(Platform.isAndroid){
                        if(user!.appMetadata?["provider"] == "google"){
                          GoogleSignIn().signOut();
                        }
                      }
                      await supabase.auth.signOut();
                      final SharedPreferences prefs = await
SharedPreferences.getInstance();
                      await prefs.remove('supabaseSessionToken');
                      Navigator.of(context).pushAndRemoveUntil(
                        MaterialPageRoute(builder: (context) =>
Splash()),
                            (Route<dynamic> route) => false,
                      );
                    },
                    child: Container(
                      decoration: BoxDecoration(
                        color: Colors.red,
                        borderRadius: BorderRadius.circular(10.0),
                        border: Border.all(color: secondryColor, width:
2.0),
                      ),
                      padding: const EdgeInsets.symmetric(horizontal:
20, vertical: 15),
                      child: const Text(
                        "Log Out",
                        style: TextStyle(fontSize: 35, fontWeight:
FontWeight.bold, color: Colors.white),
                      ),
                    ),
                  ),
                ],
              ),
            ),
          ),
          bottomNavigationBar: Container(
```

```
            decoration: BoxDecoration(
              color: secondryColor,
              boxShadow: [
                BoxShadow(
                  color: Colors.grey.withOpacity(0.5),
                  spreadRadius: 2,
                  blurRadius: 7,
                  offset: Offset(0, 3),
                ),
              ],
            ),
            child: BottomNavigationBar(
              backgroundColor: secondryColor,
              items: <BottomNavigationBarItem>[
                BottomNavigationBarItem(
                  icon: Icon(Icons.home, size: 27,),
                  label: 'Home',
                ),
                BottomNavigationBarItem(
                  icon: Icon(Icons.add_circle_rounded, size: 27,),
                  label: 'Control',
                ),
                BottomNavigationBarItem(
                  icon: Icon(Icons.auto_mode_rounded, size: 27,),
                  label: 'Auto',
                ),
                BottomNavigationBarItem(
                  icon: Icon(Icons.precision_manufacturing_rounded,
size: 27,),
                  label: 'Manual',
                ),
              ],
              currentIndex: _selectedIndex,
              selectedItemColor: primaryColor,
              selectedFontSize: 25,
              unselectedFontSize: 23,
              unselectedItemColor: secondryColor,
              selectedLabelStyle: TextStyle(fontWeight:
FontWeight.bold),
              unselectedLabelStyle: TextStyle(fontWeight:
FontWeight.normal),
              onTap: _onItemTapped,
            ),
          ),
          body: Center(
            child: _widgetOptions.elementAt(_selectedIndex),
          ),
        ),
      );
    }
}
```

- Account.dart:

```
import 'dart:io';
import 'package:flutter/material.dart';
import 'package:google_sign_in/google_sign_in.dart';
import 'package:roboarmv2/CustomDialog.dart';
import 'package:roboarmv2/UpdateName.dart';
import 'package:roboarmv2/UpdatePassword.dart';
import 'package:roboarmv2/splach.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:supabase_flutter/supabase_flutter.dart';
import 'Bluetooth.dart';
import 'Logs.dart';
import 'Settings.dart';
import 'UpdateEmail.dart';
import 'colors.dart';
import 'home.dart';
import 'main.dart';

void main() {
  runApp(Account());
}

class Account extends StatefulWidget {
  @override
  State<Account> createState() => _MyAppState();
}

class _MyAppState extends State<Account> {
  int _currentIndex = 1;

  final List<String> itemName = ["Home", "Account",
"Bluetooth","Logs", "Settings"];

  final List<IconData> icons = [
    Icons.home,
    Icons.account_circle_outlined,
    Icons.bluetooth,
    Icons.code,
    Icons.settings,
  ];

  final List<Widget> pages = [
    home(),
    Account(),
    Bluetooth(),
    Logs(),
    Settings(),
  ];
```

```
  final User? user = supabase.auth.currentUser;

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: const Text(
            "Account",
            style: TextStyle(
              color: primaryColor,
              fontWeight: FontWeight.bold,
              fontSize: 30,
            ),
          ),
          backgroundColor: secondryColor,
          centerTitle: true,
          actions: [
            IconButton(
              onPressed: () {
                Navigator.of(context).pushAndRemoveUntil(
                  MaterialPageRoute(builder: (context) => home()),
                      (Route<dynamic> route) => false,
                );
              },
              icon: Icon(
                Icons.arrow_back_ios,
                color: primaryColor,
                size: 30,
              ),
            )
          ],
        ),
        drawer: Drawer(
          child: Container(
            color: secondryColor,
            child: Column(
              children: [
                Container(
                  padding: EdgeInsets.only(left: 15, top: 20, right:
15, bottom: 20),
                  child: Row(
                    children: [
                      Image.asset(
                        "images/roboarmlogo.png",
                        height: MediaQuery.of(context).size.height /
10,
                      ),
                      SizedBox(width: 15),
                      Text(
                        "RoboArm",
                        style: TextStyle(
```

```
                            fontWeight: FontWeight.bold,
                            fontSize: 30,
                            color: primaryColor,
                          ),
                        ),
                      ],
                    ),
                  ),
                  Container(color: backgroundColor, height: 2),
                  Expanded(
                    child: ListView.builder(
                      itemCount: itemName.length,
                      itemBuilder: (context, i) {
                        return Card(
                          shape: RoundedRectangleBorder(
                            borderRadius: BorderRadius.circular(10),
                          ),
                          color: backgroundColor,
                          child: ListTile(
                            title: Text(
                              itemName[i],
                              style: TextStyle(
                                fontWeight: FontWeight.bold,
                                fontSize: 25,
                                color: secondryColor,
                              ),
                            ),
                            leading: Icon(
                              icons[i],
                              color: primaryColor,
                              size: 30,
                            ),
                            onTap: () {
                              if (_currentIndex == i) {

Navigator.of(context).pushNamedAndRemoveUntil(
                                    '/',
                                        (Route<dynamic> route) => false,
                                  );
                                } else {
                                  Navigator.push(
                                    context,
                                    MaterialPageRoute(builder: (context)
=> pages[i]),

                                  );
                                }
                              },
                            ),
                            elevation: 1,
                          );
                        },
                      ),
```

```
                ),
                MaterialButton(
                  padding: EdgeInsets.only(bottom: 20),
                  onPressed: () async {
                    if(Platform.isAndroid){
                      if(user!.appMetadata?["provider"] == "google"){
                        GoogleSignIn().signOut();
                      }
                    }
                    await supabase.auth.signOut();
                    final SharedPreferences prefs = await
SharedPreferences.getInstance();
                    await prefs.remove('supabaseSessionToken');
                    Navigator.of(context).pushAndRemoveUntil(
                      MaterialPageRoute(builder: (context) =>
Splash()),
                           (Route<dynamic> route) => false,
                    );
                  },
                  child: Container(
                    decoration: BoxDecoration(
                      color: Colors.red,
                      borderRadius: BorderRadius.circular(10.0),
                      border: Border.all(color: secondryColor, width:
2.0),
                    ),
                    padding: const EdgeInsets.symmetric(horizontal:
20, vertical: 15),
                    child: const Text(
                      "Log Out",
                      style: TextStyle(fontSize: 35, fontWeight:
FontWeight.bold, color: Colors.white),
                    ),
                  ),
                ),
              ],
            ),
          ),
        ),
        body: Center(
          child: Container(
            child: Card(
              elevation: 3,
              shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(10),
              ),
              child: Padding(
                padding: const EdgeInsets.all(10),
                child: Column(
                  children: [
                    Padding(
                      padding: EdgeInsets.all(20),
```

```
                        child: user!.appMetadata?["provider"] ==
"google"
                            ? ClipRRect(
                          borderRadius: BorderRadius.circular(60),
                          child: Image.network(
                            user!.userMetadata?["picture"],
                            width: 120,
                            height: 120,
                            fit: BoxFit.cover,
                          ),
                        )
                            : Icon(
                          Icons.account_circle_outlined,
                          size: 120,
                          color: secondryColor,
                        ),
                      ),
                      Container(
                        padding: EdgeInsets.symmetric(vertical: 10,
horizontal: 5),
                        decoration: BoxDecoration(
                          border: Border.all(
                            color: secondryColor,
                            width: 2,
                          ),
                          borderRadius: BorderRadius.circular(8.0),
                        ),
                        child: InkWell(
                          child: Container(
                            child: Row(
                              mainAxisAlignment:
MainAxisAlignment.center,
                              children: [
                                Text(
                                  user!.appMetadata?["provider"] ==
"email"
                                      ? (user!.userMetadata != null ?
user!.userMetadata!["username"] ?? "" : "")
                                      :
user!.userMetadata!["full_name"],
                                  style: TextStyle(
                                    fontSize: 23,
                                    fontWeight: FontWeight.w600,
                                    color: secondryColor,
                                    overflow: TextOverflow.ellipsis,
                                  ),
                                ),
                                SizedBox(width: 15,),
                                Icon(Icons.arrow_forward_ios_rounded,
size: 25, color: secondryColor,),
                              ],
                            ),
```

```
                        ),
                        onTap: () {
                          if(user!.appMetadata?["provider"] ==
"email"){
                            Navigator.of(context).push(
                              MaterialPageRoute(builder: (context) =>
UpdateName()),
                            );
                          }
                        },
                      ),
                    ),

                    SizedBox(height: 10),
                    Container(
                      padding: EdgeInsets.symmetric(vertical: 10,
horizontal: 5),
                      decoration: BoxDecoration(
                        border: Border.all(
                          color: secondryColor,
                          width: 2,
                        ),
                        borderRadius: BorderRadius.circular(8.0),
                      ),
                      child: InkWell(
                        child: Row(
                          mainAxisAlignment: MainAxisAlignment.center,
                          children: [
                            Flexible(
                              child: Text(
                                user!.email.toString(),
                                style: TextStyle(
                                  fontSize: 23,
                                  fontWeight: FontWeight.w600,
                                  color: secondryColor,
                                ),
                                overflow: TextOverflow.ellipsis,
                              ),
                            ),
                            SizedBox(width: 15,),
                            Icon(Icons.arrow_forward_ios_rounded,
size: 25, color: secondryColor,),
                          ],
                        ),
                        onTap: () {
                          // if(user!.appMetadata?["provider"] ==
"email"){
                          //   Navigator.of(context).push(
                          //     MaterialPageRoute(builder: (context)
=> UpdateEmail()),
                          //   );
                          // }
```

```
                            Navigator.of(context).push(
                              MaterialPageRoute(builder: (context) =>
UpdateEmail()),
                            );

                        },
                      ),
                    ),

                    SizedBox(height: 20),
                    user!.appMetadata?["provider"] == "email"
                        ? MaterialButton(
                      onPressed: () {
                        Navigator.of(context).push(
                          MaterialPageRoute(builder: (context) =>
UpdatePassword()),
                        );
                      },
                      child: Container(
                        decoration: BoxDecoration(
                          color: primaryColor,
                          borderRadius: BorderRadius.circular(10.0),
                          border: Border.all(color: secondryColor,
width: 2.0),
                        ),
                        padding: const
EdgeInsets.symmetric(horizontal: 15, vertical: 10),
                        child: const Text(
                          "Change Password",
                          style: TextStyle(fontSize: 20, fontWeight:
FontWeight.bold, color: Colors.white),
                        ),
                      ),
                    )
                        : Container(),
                    SizedBox(height: 20),

                  ],
                ),
              ),
            ),
          ),
        ),
        floatingActionButton: FloatingActionButton(
          onPressed: () {
            showDialog(
              context: context,
              builder: (BuildContext context) {
                return CustomDialog(title: "Alert", content: "Click on
your account information to change it.");
              },
            );
```

```
          },
          child: Icon(Icons.info_outline_rounded, color: primaryColor,
size: 30,),
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(50),
          ),
          backgroundColor: secondryColor,
        ),
        floatingActionButtonLocation:
FloatingActionButtonLocation.endFloat,
      ),
    );
  }
}
```

- UpdateName.dart:

```
import 'package:flutter/material.dart';
import 'package:supabase_flutter/supabase_flutter.dart';
import 'Account.dart';
import 'CustomDialog.dart';
import 'colors.dart';
import 'forgetpassword.dart';
import 'main.dart';

void main() {
  runApp(UpdateName());
}

class UpdateName extends StatefulWidget {
  @override
  State<UpdateName> createState() => _MyAppState();
}

class _MyAppState extends State<UpdateName> {
  TextEditingController password = TextEditingController();
  bool visibility = false;
  final User? user = supabase.auth.currentUser;
  bool pageVisibility = false;

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        backgroundColor: backgroundColor,
        appBar: AppBar(
          title: const Text(
            "Update Username",
            style: TextStyle(
              color: primaryColor,
              fontWeight: FontWeight.bold,
              fontSize: 30,
```

```
            ),
          ),
          backgroundColor: secondryColor,
          centerTitle: true,
          actions: [
            IconButton(
              onPressed: () {
                Navigator.of(context).pop();
              },
              icon: Icon(
                Icons.arrow_back_ios,
                color: primaryColor,
                size: 30,
              ),
            )
          ],
        ),
        body: pageVisibility
            ? NewWidget()
            : Center(
          child: Container(
            padding: EdgeInsets.all(20),
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Text(
                  "Enter your password",
                  style: TextStyle(color: secondryColor, fontSize: 23,
fontWeight: FontWeight.w700),
                ),
                SizedBox(height: 20),
                Container(
                  child: TextFormField(
                    validator: (passwordValue) {
                      if (passwordValue!.isEmpty) {
                        return "Field is empty!";
                      }
                      return null;
                    },
                    obscureText: !visibility,
                    decoration: InputDecoration(
                      border: const OutlineInputBorder(),
                      labelText: "Password",
                      labelStyle: const TextStyle(
                        color: secondryColor,
                        fontSize: 20,
                        fontWeight: FontWeight.bold,
                      ),
                      prefixIcon: const Icon(
                        Icons.lock,
                        color: secondryColor,
                        size: 35,
```

```
                    ),
                    suffixIcon: IconButton(
                      icon: Icon(
                        visibility ? Icons.visibility_off_outlined :
Icons.visibility_outlined,
                        color: secondryColor,
                        size: 35,
                      ),
                      onPressed: () {
                        setState(() {
                          visibility = !visibility;
                        });
                      },
                    ),
                    focusedBorder: OutlineInputBorder(
                      borderSide: const BorderSide(color:
secondryColor, width: 2),
                      borderRadius: BorderRadius.circular(18.0),
                    ),
                    enabledBorder: OutlineInputBorder(
                      borderSide: const BorderSide(color:
secondryColor, width: 2),
                      borderRadius: BorderRadius.circular(18.0),
                    ),
                  ),
                  style: const TextStyle(
                    fontSize: 22,
                    fontWeight: FontWeight.bold,
                    color: secondryColor,
                  ),
                  controller: password,
                ),
              ),
              SizedBox(height: 20),
              Row(
                mainAxisAlignment: MainAxisAlignment.end,
                children: [
                  GestureDetector(
                    onTap: () {
                      Navigator.push(
                        context,
                        MaterialPageRoute(builder: (context) =>
const forgetpassword()),
                      );
                    },
                    child: const Text(
                      "Forget password?",
                      style: TextStyle(
                        fontSize: 18,
                        fontWeight: FontWeight.w700,
                        color: secondryColor,
                        decoration: TextDecoration.underline,
```

```
                        ),
                      ),
                    ),
                  ],
                ),
                SizedBox(height: 20),
                MaterialButton(
                  onPressed: () async {
                    try {
                      final AuthResponse res = await
supabase.auth.signInWithPassword(
                        email: user?.email,
                        password: password.text,
                      );
                      setState(() {
                        pageVisibility = true;
                      });
                    } catch (error) {
                      showDialog(
                        context: context,
                        builder: (BuildContext context) {
                          return CustomDialog(
                            title: 'Wrong Password!',
                            content: 'You have entered wrong
password!\nPlease try again.',
                          );
                        },
                      );
                    }
                  },
                  child: Container(
                    decoration: BoxDecoration(
                      color: primaryColor,
                      borderRadius: BorderRadius.circular(10.0),
                      border: Border.all(color: secondryColor, width:
2.0),
                    ),
                    padding: const EdgeInsets.symmetric(horizontal:
15, vertical: 10),
                    child: const Text(
                      "Confirm",
                      style: TextStyle(fontSize: 20, fontWeight:
FontWeight.bold, color: Colors.white),
                    ),
                  ),
                ),
              ],
            ),
          ),
        ),
      ),
    );
```

```
  }
}

class NewWidget extends StatelessWidget {
  TextEditingController username = TextEditingController();
  GlobalKey<FormState> formStateP = GlobalKey<FormState>();

  @override
  Widget build(BuildContext context) {
    return SingleChildScrollView(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Padding(
            padding: const EdgeInsets.all(15),
            child: Text(
              "Write new Username",
              style: TextStyle(color: secondryColor, fontSize: 23,
fontWeight: FontWeight.w700),
            ),
          ),
          Padding(
            padding: const EdgeInsets.only(left: 15, right: 15),
            child: Form(
              //key: formStateP,
              autovalidateMode: AutovalidateMode.onUserInteraction,
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.stretch,
                children: [
                  TextFormField(
                    validator: (usernameValue) {
                      if (usernameValue!.isEmpty) {
                        return "Field is empty!";
                      }
                      return null;
                    },
                    decoration: InputDecoration(
                      border: const OutlineInputBorder(),
                      labelText: "Username",
                      labelStyle: const TextStyle(
                        color: secondryColor,
                        fontSize: 20,
                        fontWeight: FontWeight.bold,
                      ),
                      prefixIcon: const Icon(
                        Icons.account_circle_outlined,
                        color: secondryColor,
                        size: 35,
                      ),
                      focusedBorder: OutlineInputBorder(
                        borderSide: const BorderSide(color:
secondryColor, width: 2),
```

```
                        borderRadius: BorderRadius.circular(18.0),
                      ),
                      enabledBorder: OutlineInputBorder(
                        borderSide: const BorderSide(color:
secondryColor, width: 2),
                        borderRadius: BorderRadius.circular(18.0),
                      ),
                    ),
                    style: const TextStyle(
                      fontSize: 22,
                      fontWeight: FontWeight.bold,
                      color: secondryColor,
                    ),
                    controller: username,
                  ),
                  const SizedBox(height: 15),
                ],
              ),
            ),
          ),
          Padding(
            padding: const EdgeInsets.all(15),
            child: MaterialButton(
              onPressed: () async {
                if (username.text != "" ) {
                  try {

                    final UserResponse res = await
supabase.auth.updateUser(
                      UserAttributes(
                        data: { 'username': username.text },
                      ),
                    );
                    showDialog(
                      context: context,
                      builder: (BuildContext context) {
                        return CustomDialog(
                          title: 'Username has changed',
                          content: 'Your username has been changed.',
                          onOkPressed: (BuildContext context) {
                            Navigator.of(context).pushAndRemoveUntil(
                              MaterialPageRoute(builder: (context) =>
Account()),
                                  (Route<dynamic> route) => false,
                            );
                          },
                        );
                      },
                    );
                  } catch (error) {
                    showDialog(
                      context: context,
```

```
                    builder: (BuildContext context) {
                      return CustomDialog(
                        title: 'Error!',
                        content: 'Sorry something is wrong!\nPlease
try again.',
                      );
                    },
                  );
                }
              } else {
                showDialog(
                  context: context,
                  builder: (BuildContext context) {
                    return CustomDialog(
                      title: 'Entry is wrong!',
                      content: 'Filed is empty!\nPlease try again.',
                    );
                  },
                );
              }
            },
            child: Container(
              decoration: BoxDecoration(
                color: primaryColor,
                borderRadius: BorderRadius.circular(10.0),
                border: Border.all(color: secondryColor, width:
2.0),
              ),
              padding: const EdgeInsets.symmetric(horizontal: 15,
vertical: 10),
              child: const Text(
                "Confirm",
                style: TextStyle(fontSize: 20, fontWeight:
FontWeight.bold, color: Colors.white),
              ),
            ),
          ),
        ),
      ],
    ),
  );
 }
}
```

- UpdateEmail.dart:

```
import 'package:flutter/material.dart';
import 'package:google_sign_in/google_sign_in.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:supabase_flutter/supabase_flutter.dart';
import 'CustomDialog.dart';
```

```dart
import 'colors.dart';
import 'forgetpassword.dart';
import 'login.dart';
import 'main.dart';

void main() {
  runApp(UpdateEmail());
}

class UpdateEmail extends StatefulWidget {
  @override
  State<UpdateEmail> createState() => _MyAppState();
}

class _MyAppState extends State<UpdateEmail> {
  TextEditingController password = TextEditingController();
  bool visibility = false;
  bool pageVisibility = false;

  @override
  void initState() {
    super.initState();
    final User? user = supabase.auth.currentUser;
    pageVisibility = user?.appMetadata?["provider"] == "google" ? true
: false;
  }

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        backgroundColor: backgroundColor,
        appBar: AppBar(
          title: const Text(
            "Update Email",
            style: TextStyle(
              color: primaryColor,
              fontWeight: FontWeight.bold,
              fontSize: 30,
            ),
          ),
          backgroundColor: secondryColor,
          centerTitle: true,
          actions: [
            IconButton(
              onPressed: () {
                Navigator.of(context).pop();
              },
              icon: Icon(
                Icons.arrow_back_ios,
                color: primaryColor,
                size: 30,
```

```
              ),
            )
          ],
        ),
        body: pageVisibility
            ? NewWidget()
            : Center(
          child: Container(
            padding: EdgeInsets.all(20),
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Text(
                  "Enter your password",
                  style: TextStyle(color: secondryColor, fontSize: 23,
fontWeight: FontWeight.w700),
                ),
                SizedBox(height: 20),
                Container(
                  child: TextFormField(
                    validator: (passwordValue) {
                      if (passwordValue!.isEmpty) {
                        return "Field is empty!";
                      }
                      return null;
                    },
                    obscureText: !visibility,
                    decoration: InputDecoration(
                      border: const OutlineInputBorder(),
                      labelText: "Password",
                      labelStyle: const TextStyle(
                        color: secondryColor,
                        fontSize: 20,
                        fontWeight: FontWeight.bold,
                      ),
                      prefixIcon: const Icon(
                        Icons.lock,
                        color: secondryColor,
                        size: 35,
                      ),
                      suffixIcon: IconButton(
                        icon: Icon(
                          visibility ? Icons.visibility_off_outlined :
Icons.visibility_outlined,
                          color: secondryColor,
                          size: 35,
                        ),
                        onPressed: () {
                          setState(() {
                            visibility = !visibility;
                          });
                        },
```

```
                    ),
                    focusedBorder: OutlineInputBorder(
                      borderSide: const BorderSide(color:
secondryColor, width: 2),
                      borderRadius: BorderRadius.circular(18.0),
                    ),
                    enabledBorder: OutlineInputBorder(
                      borderSide: const BorderSide(color:
secondryColor, width: 2),
                      borderRadius: BorderRadius.circular(18.0),
                    ),
                  ),
                  style: const TextStyle(
                    fontSize: 22,
                    fontWeight: FontWeight.bold,
                    color: secondryColor,
                  ),
                  controller: password,
                ),
              ),
              SizedBox(height: 20),
              Row(
                mainAxisAlignment: MainAxisAlignment.end,
                children: [
                  GestureDetector(
                    onTap: () {
                      Navigator.push(
                        context,
                        MaterialPageRoute(builder: (context) =>
const forgetpassword()),
                      );
                    },
                    child: const Text(
                      "Forget password?",
                      style: TextStyle(
                        fontSize: 18,
                        fontWeight: FontWeight.w700,
                        color: secondryColor,
                        decoration: TextDecoration.underline,
                      ),
                    ),
                  ),
                ],
              ),
              SizedBox(height: 20),
              MaterialButton(
                onPressed: () async {
                  try {
                    final User? user = supabase.auth.currentUser;
                    final AuthResponse res = await
supabase.auth.signInWithPassword(
                      email: user?.email,
```

```
                        password: password.text,
                      );
                      setState(() {
                        pageVisibility = true;
                      });
                    } catch (error) {
                      showDialog(
                        context: context,
                        builder: (BuildContext context) {
                          return CustomDialog(
                            title: 'Wrong Password!',
                            content: 'You have entered wrong
password!\nPlease try again.',
                          );
                        },
                      );
                    }
                  },
                  child: Container(
                    decoration: BoxDecoration(
                      color: primaryColor,
                      borderRadius: BorderRadius.circular(10.0),
                      border: Border.all(color: secondryColor, width:
2.0),
                    ),
                    padding: const EdgeInsets.symmetric(horizontal:
15, vertical: 10),
                    child: const Text(
                      "Confirm",
                      style: TextStyle(fontSize: 20, fontWeight:
FontWeight.bold, color: Colors.white),
                    ),
                  ),
                ),
              ],
            ),
          ),
        ),
      );
  }
}

class NewWidget extends StatelessWidget {
  TextEditingController email = TextEditingController();
  GlobalKey<FormState> formStateP = GlobalKey<FormState>();

  @override
  Widget build(BuildContext context) {
    return SingleChildScrollView(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
```

```
        children: [
          Padding(
            padding: const EdgeInsets.all(15),
            child: Text(
              "Write new Email",
              style: TextStyle(color: secondryColor, fontSize: 23,
fontWeight: FontWeight.w700),
            ),
          ),
          Padding(
            padding: const EdgeInsets.only(left: 15, right: 15),
            child: Form(
              //key: formStateP,
              autovalidateMode: AutovalidateMode.onUserInteraction,
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.stretch,
                children: [
                  TextFormField(
                    validator: (emailValue) {
                      if (emailValue!.isEmpty) {
                        return "Field is empty!";
                      } else if (!RegExp(r"^[a-zA-Z0-9_.+-]+@[a-zA-Z0-
9-]+\.[a-zA-Z0-9-.]+$").hasMatch(emailValue)) {
                        return "Enter a valid email
address!\nexample@mail.com";
                      }
                      return null;
                    },
                    decoration: InputDecoration(
                      border: const OutlineInputBorder(),
                      labelText: "Email",
                      labelStyle: const TextStyle(
                        color: secondryColor,
                        fontSize: 20,
                        fontWeight: FontWeight.bold,
                      ),
                      prefixIcon: const Icon(
                        Icons.email,
                        color: secondryColor,
                        size: 35,
                      ),
                      focusedBorder: OutlineInputBorder(
                        borderSide: const BorderSide(color:
secondryColor, width: 2),
                        borderRadius: BorderRadius.circular(18.0),
                      ),
                      enabledBorder: OutlineInputBorder(
                        borderSide: const BorderSide(color:
secondryColor, width: 2),
                        borderRadius: BorderRadius.circular(18.0),
                      ),
                    ),
```

```
                    style: const TextStyle(
                      fontSize: 22,
                      fontWeight: FontWeight.bold,
                      color: secondryColor,
                    ),
                    controller: email,
                  ),
                  const SizedBox(height: 15),
                ],
              ),
            ),
          ),
          Padding(
            padding: const EdgeInsets.all(15),
            child: MaterialButton(
              onPressed: () async {
                if (email.text != "" ) {
                  try {
                    final UserResponse res = await
supabase.auth.updateUser(
                      UserAttributes(
                        email: email.text,
                      ),
                    );

                    final User? user = supabase.auth.currentUser;
                    if(user!.appMetadata?["provider"] == "google"){
                      GoogleSignIn().signOut();
                    }

                    await supabase.auth.signOut();
                    final SharedPreferences prefs = await
SharedPreferences.getInstance();
                    await prefs.remove('supabaseSessionToken');
                    showDialog(
                      context: context,
                      builder: (BuildContext context) {
                        return CustomDialog(
                          title: 'About to change',
                          content: '\nPlease confirm this process, we
have sent email in both emails to confirm the process\nConfirm it and
return to Sign Up again.',
                          onOkPressed: (BuildContext context) {
                            Navigator.of(context).pushAndRemoveUntil(
                              MaterialPageRoute(builder: (context) =>
Login()),
                                  (Route<dynamic> route) => false,
                            );
                          },
                        );
                      },
                    );
```

```
                    } catch (error) {
                      showDialog(
                        context: context,
                        builder: (BuildContext context) {
                          return CustomDialog(
                            title: 'Error!',
                            content: 'Sorry something is wrong!\nPlease
try again.',
                          );
                        },
                      );
                    }
                  } else {
                    showDialog(
                      context: context,
                      builder: (BuildContext context) {
                        return CustomDialog(
                          title: 'Entry is wrong!',
                          content: 'Filed is empty!\nPlease try again.',
                        );
                      },
                    );
                  }
                },
                child: Container(
                  decoration: BoxDecoration(
                    color: primaryColor,
                    borderRadius: BorderRadius.circular(10.0),
                    border: Border.all(color: secondryColor, width:
2.0),
                  ),
                  padding: const EdgeInsets.symmetric(horizontal: 15,
vertical: 10),
                  child: const Text(
                    "Confirm",
                    style: TextStyle(fontSize: 20, fontWeight:
FontWeight.bold, color: Colors.white),
                  ),
                ),
              ),
            ),
          ],
        ),
      );
    }
}
```

- UpdatePassword.dart:

```
import 'package:flutter/material.dart';
import 'package:shared_preferences/shared_preferences.dart';
```

```dart
import 'package:supabase_flutter/supabase_flutter.dart';
import 'CustomDialog.dart';
import 'colors.dart';
import 'forgetpassword.dart';
import 'login.dart';
import 'main.dart';

void main() {
  runApp(UpdatePassword());
}

class UpdatePassword extends StatefulWidget {
  @override
  State<UpdatePassword> createState() => _MyAppState();
}

class _MyAppState extends State<UpdatePassword> {
  TextEditingController password = TextEditingController();
  bool visibility = false;
  final User? user = supabase.auth.currentUser;
  bool pageVisibility = false;

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        backgroundColor: backgroundColor,
        appBar: AppBar(
          title: const Text(
            "Change Password",
            style: TextStyle(
              color: primaryColor,
              fontWeight: FontWeight.bold,
              fontSize: 30,
            ),
          ),
          backgroundColor: secondryColor,
          centerTitle: true,
          actions: [
            IconButton(
              onPressed: () {
                Navigator.of(context).pop();
              },
              icon: Icon(
                Icons.arrow_back_ios,
                color: primaryColor,
                size: 30,
              ),
            )
          ],
        ),
        body: pageVisibility
```

```
            ? NewWidget()
            : Center(
          child: Container(
            padding: EdgeInsets.all(20),
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Text(
                  "Write your current password",
                  style: TextStyle(color: secondryColor, fontSize: 23,
fontWeight: FontWeight.w700),
                ),
                SizedBox(height: 20),
                Container(
                  child: TextFormField(
                    validator: (passwordValue) {
                      if (passwordValue!.isEmpty) {
                        return "Field is empty!";
                      }
                      return null;
                    },
                    obscureText: !visibility,
                    decoration: InputDecoration(
                      border: const OutlineInputBorder(),
                      labelText: "Password",
                      labelStyle: const TextStyle(
                        color: secondryColor,
                        fontSize: 20,
                        fontWeight: FontWeight.bold,
                      ),
                      prefixIcon: const Icon(
                        Icons.lock,
                        color: secondryColor,
                        size: 35,
                      ),
                      suffixIcon: IconButton(
                        icon: Icon(
                          visibility ? Icons.visibility_off_outlined :
Icons.visibility_outlined,
                          color: secondryColor,
                          size: 35,
                        ),
                        onPressed: () {
                          setState(() {
                            visibility = !visibility;
                          });
                        },
                      ),
                      focusedBorder: OutlineInputBorder(
                        borderSide: const BorderSide(color:
secondryColor, width: 2),
                        borderRadius: BorderRadius.circular(18.0),
```

```
                    ),
                    enabledBorder: OutlineInputBorder(
                      borderSide: const BorderSide(color:
secondryColor, width: 2),
                      borderRadius: BorderRadius.circular(18.0),
                    ),
                  ),
                  style: const TextStyle(
                    fontSize: 22,
                    fontWeight: FontWeight.bold,
                    color: secondryColor,
                  ),
                  controller: password,
                ),
              ),
              SizedBox(height: 20),
              Row(
                mainAxisAlignment: MainAxisAlignment.end,
                children: [
                  GestureDetector(
                    onTap: () {
                      Navigator.push(
                        context,
                        MaterialPageRoute(builder: (context) =>
const forgetpassword()),
                      );
                    },
                    child: const Text(
                      "Forget password?",
                      style: TextStyle(
                        fontSize: 18,
                        fontWeight: FontWeight.w700,
                        color: secondryColor,
                        decoration: TextDecoration.underline,
                      ),
                    ),
                  ),
                ],
              ),
              SizedBox(height: 20),
              MaterialButton(
                onPressed: () async {
                  try {
                    final AuthResponse res = await
supabase.auth.signInWithPassword(
                      email: user?.email,
                      password: password.text,
                    );
                    setState(() {
                      pageVisibility = true;
                    });
                  } catch (error) {
```

```
                        showDialog(
                          context: context,
                          builder: (BuildContext context) {
                            return CustomDialog(
                              title: 'Wrong Password!',
                              content: 'You have entered wrong
password!\nPlease try again.',
                              );
                          },
                        );
                      }
                    },
                    child: Container(
                      decoration: BoxDecoration(
                        color: primaryColor,
                        borderRadius: BorderRadius.circular(10.0),
                        border: Border.all(color: secondryColor, width:
2.0),
                      ),
                      padding: const EdgeInsets.symmetric(horizontal:
15, vertical: 10),
                      child: const Text(
                        "Confirm",
                        style: TextStyle(fontSize: 20, fontWeight:
FontWeight.bold, color: Colors.white),
                      ),
                    ),
                  ),
                ],
              ),
            ),
          ),
        ),
      );
  }
}

class NewWidget extends StatelessWidget {
  TextEditingController passwordP = TextEditingController();
  TextEditingController confirmPasswordP = TextEditingController();
  GlobalKey<FormState> formStateP = GlobalKey<FormState>();

  @override
  Widget build(BuildContext context) {
    return SingleChildScrollView(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Padding(
            padding: const EdgeInsets.all(15),
            child: Text(
              "Write the new password",
```

```
              style: TextStyle(color: secondryColor, fontSize: 23,
fontWeight: FontWeight.w700),
              ),
            ),
            Padding(
              padding: const EdgeInsets.only(left: 15, right: 15),
              child: Form(
                //key: formStateP,
                autovalidateMode: AutovalidateMode.onUserInteraction,
                child: Column(
                  crossAxisAlignment: CrossAxisAlignment.stretch,
                  children: [
                    TextFormField(
                      validator: (passwordValue) {
                        if (passwordValue!.isEmpty) {
                          return "Field is empty!";
                        } else if (passwordValue.length < 8) {
                          return "Password must be at least 8 characters
long!";
                        } else if (!RegExp(r'^(?=.*\d)(?=.*[a-
z])(?=.*[A-Z])(?=.*[a-zA-Z]).{8,}$').hasMatch(passwordValue)) {
                          return "Password must contain at least one
uppercase \nletter (A, Z), one lowercase letter (a, z), \none digit
(0, 9), and one special character (#, @, ., &)!";
                        }
                        return null;
                      },
                      decoration: InputDecoration(
                        border: const OutlineInputBorder(),
                        labelText: "Password",
                        labelStyle: const TextStyle(
                          color: secondryColor,
                          fontSize: 20,
                          fontWeight: FontWeight.bold,
                        ),
                        prefixIcon: const Icon(
                          Icons.lock,
                          color: secondryColor,
                          size: 35,
                        ),
                        focusedBorder: OutlineInputBorder(
                          borderSide: const BorderSide(color:
secondryColor, width: 2),
                          borderRadius: BorderRadius.circular(18.0),
                        ),
                        enabledBorder: OutlineInputBorder(
                          borderSide: const BorderSide(color:
secondryColor, width: 2),
                          borderRadius: BorderRadius.circular(18.0),
                        ),
                      ),
                      style: const TextStyle(
```

```
                    fontSize: 22,
                    fontWeight: FontWeight.bold,
                    color: secondryColor,
                  ),
                  controller: passwordP,
                  obscureText: true,
                ),
                const SizedBox(height: 15),
                TextFormField(
                  validator: (confirmPasswordValue) {
                    if (confirmPasswordValue!.isEmpty) {
                      return "Field is empty!";
                    }
                    return null;
                  },
                  decoration: InputDecoration(
                    border: const OutlineInputBorder(),
                    labelText: "Confirm Password",
                    labelStyle: const TextStyle(
                      color: secondryColor,
                      fontSize: 20,
                      fontWeight: FontWeight.bold,
                    ),
                    prefixIcon: const Icon(
                      Icons.lock_reset,
                      color: secondryColor,
                      size: 35,
                    ),
                    focusedBorder: OutlineInputBorder(
                      borderSide: const BorderSide(color:
secondryColor, width: 2),
                      borderRadius: BorderRadius.circular(18.0),
                    ),
                    enabledBorder: OutlineInputBorder(
                      borderSide: const BorderSide(color:
secondryColor, width: 2),
                      borderRadius: BorderRadius.circular(18.0),
                    ),
                  ),
                  style: const TextStyle(
                    fontSize: 22,
                    fontWeight: FontWeight.bold,
                    color: secondryColor,
                  ),
                  controller: confirmPasswordP,
                  obscureText: true,
                ),
              ],
            ),
          ),
        ),
        Padding(
```

```
            padding: const EdgeInsets.all(15),
            child: MaterialButton(
              onPressed: () async {
                if (passwordP.text == confirmPasswordP.text) {
                  try {
                    final UserResponse res = await
supabase.auth.updateUser(
                      UserAttributes(
                        password: passwordP.text,
                      ),
                    );
                    await supabase.auth.signOut();
                    final SharedPreferences prefs = await
SharedPreferences.getInstance();
                    await prefs.remove('supabaseSessionToken');
                    showDialog(
                      context: context,
                      builder: (BuildContext context) {
                        return CustomDialog(
                          title: 'Password has changed',
                          content: 'Your Password has been
changed.\nPlease Sign in again.',
                          onOkPressed: (BuildContext context) {
                            Navigator.of(context).pushAndRemoveUntil(
                              MaterialPageRoute(builder: (context) =>
Login()),
                                  (Route<dynamic> route) => false,
                            );
                          },
                        );
                      },
                    );
                  } catch (error) {
                    showDialog(
                      context: context,
                      builder: (BuildContext context) {
                        return CustomDialog(
                          title: 'Error!',
                          content: 'Sorry something is wrong!\nPlease
try again.',
                        );
                      },
                    );
                  }
                } else {
                  showDialog(
                    context: context,
                    builder: (BuildContext context) {
                      return CustomDialog(
                        title: 'Entry is wrong!',
                        content: 'The password and confirm password
are not matches!\nPlease try again.',
```

```
                    );
                  },
                );
              }
            },
            child: Container(
              decoration: BoxDecoration(
                color: primaryColor,
                borderRadius: BorderRadius.circular(10.0),
                border: Border.all(color: secondryColor, width:
2.0),
              ),
              padding: const EdgeInsets.symmetric(horizontal: 15,
vertical: 10),
              child: const Text(
                "Confirm",
                style: TextStyle(fontSize: 20, fontWeight:
FontWeight.bold, color: Colors.white),
              ),
            ),
          ),
        ),
      ],
    ),
  );
 }
}
```

- MainActivity.kt:

```
package com.example.roboarmv2
import android.os.Bundle
import android.widget.Toast
import io.flutter.embedding.android.FlutterActivity
import io.flutter.embedding.engine.FlutterEngine
import io.flutter.plugin.common.MethodChannel
import io.flutter.plugin.common.EventChannel



class MainActivity : FlutterActivity() {
    private val CHANNEL = "bluetooth_channel"
    private val EVENT_CHANNEL = "bluetooth_event_channel"
    var myBluetooth = MyBluetooth(this)
    override fun configureFlutterEngine(flutterEngine: FlutterEngine)
{
        super.configureFlutterEngine(flutterEngine)

        MethodChannel(flutterEngine.dartExecutor.binaryMessenger,
CHANNEL).setMethodCallHandler { call, result ->
            when (call.method) {
```

```
                "connectBluetooth" -> {
                    val isConnected = myBluetooth.connect()

                    result.success(isConnected)
                }
                "writeData" -> {
                    val message = call.argument<String>("message")
                    if (message != null) {
                        val success = myBluetooth.writeData(message)
                        result.success(success)
                    } else {
                        result.error("INVALID_ARGUMENT", "Message is
null", null)
                    }
                }
                "readData" -> {
                    val data = myBluetooth.readData()
                    result.success(data)
                    val message = if (data.isEmpty()) "Connect the HC-
05" else data
                    Toast.makeText(this, message,
Toast.LENGTH_SHORT).show()

                }
                else -> {
                    result.notImplemented()
                }
            }
        }
        EventChannel(flutterEngine.dartExecutor.binaryMessenger,
EVENT_CHANNEL).setStreamHandler(
            object : EventChannel.StreamHandler {
                override fun onListen(arguments: Any?, events:
EventChannel.EventSink?) {
                    myBluetooth.setEventSink(events)
                }

                override fun onCancel(arguments: Any?) {
                    myBluetooth.setEventSink(null)
                }
            }
        )
    }
}
```

- MyBluetooth.kt:

```
package com.example.roboarmv2

import android.bluetooth.BluetoothAdapter
import android.bluetooth.BluetoothSocket
```

```
import android.content.ContentValues.TAG
import android.content.Context
import android.os.Handler
import android.os.Looper
import android.util.Log
import android.widget.Toast
import java.io.IOException
import java.util.*
import io.flutter.plugin.common.MethodChannel
import io.flutter.plugin.common.EventChannel


class MyBluetooth(private val context: Context) {
    private val bluetoothAdapter: BluetoothAdapter? =
BluetoothAdapter.getDefaultAdapter()
    private lateinit var btSocket: BluetoothSocket
    private val _myUUID: UUID = UUID.fromString("00001101-0000-1000-
8000-00805f9b34fb")
    private var _isConnected: Boolean = false
    private var eventSink: EventChannel.EventSink? = null
    private val handler = Handler(Looper.getMainLooper())///////

    val isConnected: Boolean
        get() {
            _isConnected = this::btSocket.isInitialized &&
btSocket.isConnected
            return _isConnected
        }

    fun connect(): Boolean {
        bluetoothAdapter?.let { adapter ->
            val device = adapter.getRemoteDevice("98:D6:32:35:8F:DF")
            Toast.makeText(context, "Connecting...",
Toast.LENGTH_SHORT).show()
            adapter.cancelDiscovery()

            try {
                btSocket =
device.createRfcommSocketToServiceRecord(_myUUID)
                btSocket.connect()
                Toast.makeText(context, "Connection made.",
Toast.LENGTH_SHORT).show()
                startListeningForData()
                return true
            } catch (e: IOException) {
                Log.e(TAG, "Connection error: ${e.message}")
                try {
                    btSocket.close()
                } catch (e2: IOException) {
                    Log.e(TAG, "Error closing socket: ${e2.message}")
                }
                Toast.makeText(context, "Socket creation or connection
```

```
failed.", Toast.LENGTH_SHORT).show()
                }
            } ?: run {
                Toast.makeText(context, "Bluetooth is not available on
this device.", Toast.LENGTH_SHORT).show()
            }
            return false
        }

        private fun startListeningForData() {
            Thread {
                try {
                    val inStream = btSocket.inputStream
                    val buffer = ByteArray(1024)
                    var bytesRead: Int
                    var accumulatedMessage = ""

                    while (true) {
                        bytesRead = inStream.read(buffer)
                        if (bytesRead > 0) {
                            val readMessage = String(buffer, 0, bytesRead)
                            accumulatedMessage += readMessage


                            val endIndex =
accumulatedMessage.indexOf('\n')
                            if (endIndex != -1) {
                                val completeMessage =
accumulatedMessage.substring(0, endIndex).trim()
                                accumulatedMessage =
accumulatedMessage.substring(endIndex + 1)

                                handler.post {
                                    eventSink?.success(completeMessage)
                                }
                            }
                        }
                    }
                } catch (e: IOException) {
                    Log.e(TAG, "Error reading data: ${e.message}")
                }
            }.start()
        }
        fun setEventSink(eventSink: EventChannel.EventSink?) {
            this.eventSink = eventSink
        }

        fun writeData(data: String): Boolean {
            if (!isConnected)
                return false
            var outStream = btSocket.outputStream
            try {
```

```
            outStream = btSocket.outputStream
        } catch (e: IOException) {
            return false
        }
        val msgBuffer = data.toByteArray()

        try {
            outStream.write(msgBuffer)
            print("Send from kotlin")

        } catch (e: IOException) {
            return false
        }
        return true
    }

    fun readData(): String {
        if (!isConnected)
            return ""
        var inStream = btSocket.inputStream
        try {
            inStream = btSocket.inputStream
        } catch (e: IOException) {
        }

        var s = ""

        try {
            while (inStream.available() > 0) {
              s += inStream.read().toChar()
                        }
                    } catch (e: IOException) {
                    } finally {
                        return s
                    }
                }
            }
```

- Bluetooth.dart:

```
import 'dart:async';
import 'dart:io';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:google_sign_in/google_sign_in.dart';
import 'package:roboarmv2/splach.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:supabase_flutter/supabase_flutter.dart';
import 'Account.dart';
import 'Logs.dart';
import 'Settings.dart';
import 'colors.dart';
```

```dart
import 'home.dart';
import 'main.dart';

bool success = false;
void main() {
  runApp(Bluetooth());
}

class Bluetooth extends StatefulWidget {
  Bluetooth({Key? key}) : super(key: key);

  @override
  State<Bluetooth> createState() => _BluetoothState();
}

class _BluetoothState extends State<Bluetooth> {
  int _currentIndex = 2;

  final List<String> itemName = ["Home", "Account",
"Bluetooth","Logs", "Settings"];

  final List<IconData> icons = [
    Icons.home,
    Icons.account_circle_outlined,
    Icons.bluetooth,
    Icons.code,
    Icons.settings,
  ];

  final List<Widget> pages = [
    home(),
    Account(),
    Bluetooth(),
    Logs(),
    Settings(),
  ];
  TextEditingController _messageController = TextEditingController();
  static const MethodChannel _channel =
MethodChannel('bluetooth_channel');
  static const EventChannel _eventChannel =
EventChannel('bluetooth_event_channel');

  StreamSubscription? _streamSubscription;
  String _receivedData = "No Message!";


  String msg = "No Message!";



  Future<void> connectBluetooth() async {
    try {
```

```
      bool connectionSuccess = await
_channel.invokeMethod('connectBluetooth');
      setState(() {
        success = connectionSuccess;
      });
    } on PlatformException catch (e) {
      print("Error: ${e.message}");
      setState(() {
        success = false;
      });
    }
  }


  Future<bool> writeData(String data) async {
    try {
      final bool success = await _channel.invokeMethod('writeData',
{'message': data});
      print("Send from Flutter");
      return success;
    } on PlatformException catch (e) {
      print("Error: ${e.message}");
      return false;
    }
  }

  // Future<String> readData() async {
  //   try {
  //     final String result = await
_channel.invokeMethod('readData');
  //     return result;
  //   } on PlatformException catch (e) {
  //     print("Error: ${e.message}");
  //     return '';
  //   }
  // }

  @override
  void initState() {
    super.initState();
    _streamSubscription =
_eventChannel.receiveBroadcastStream().listen((event) {
      setState(() {
        _receivedData = event;
        print(_receivedData);
      });
    }, onError: (error) {
      print("Error receiving data: $error");
    });
  }

  @override
```

```
  void dispose() {
    _streamSubscription?.cancel();
    _messageController.dispose();
    super.dispose();
  }

  final User? user = supabase.auth.currentUser;

  @override
  Widget build(BuildContext context) {

    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: const Text(
            "Bluetooth",
            style: TextStyle(
              color: primaryColor,
              fontWeight: FontWeight.bold,
              fontSize: 30,
            ),
          ),
          backgroundColor: secondryColor,
          centerTitle: true,
          actions: [
            IconButton(
              onPressed: () {
                Navigator.of(context).pushAndRemoveUntil(
                  MaterialPageRoute(builder: (context) => home()),
                      (Route<dynamic> route) => false,
                );
              },
              icon: Icon(
                Icons.arrow_back_ios,
                color: primaryColor,
                size: 30,
              ),
            )
          ],
        ),
        drawer: Drawer(
          child: Container(
            color: secondryColor,
            child: Column(
              children: [
                Container(
                  padding: const EdgeInsets.only(left: 15, top: 20,
right: 15, bottom: 20),
                  child: Row(
                    children: [
                      Image.asset(
                        "images/roboarmlogo.png",
```

```
                        height: MediaQuery.of(context).size.height /
10,
                      ),
                      const SizedBox(width: 15),
                      const Text(
                        "RoboArm",
                        style: TextStyle(
                          fontWeight: FontWeight.bold,
                          fontSize: 30,
                          color: primaryColor,
                        ),
                      ),
                    ],
                  ),
                ),
                Container(color: backgroundColor, height: 2),
                Expanded(
                  child: ListView.builder(
                    itemCount: itemName.length,
                    itemBuilder: (context, i) {
                      return Card(
                        shape: RoundedRectangleBorder(
                          borderRadius: BorderRadius.circular(10),
                        ),
                        color: backgroundColor,
                        elevation: 1,
                        child: ListTile(
                          title: Text(
                            itemName[i],
                            style: const TextStyle(
                              fontWeight: FontWeight.bold,
                              fontSize: 25,
                              color: secondryColor,
                            ),
                          ),
                          leading: Icon(
                            icons[i],
                            color: primaryColor,
                            size: 30,
                          ),
                          onTap: () {
                            if (_currentIndex == i) {

Navigator.of(context).pushNamedAndRemoveUntil(
                                  '/',
                                      (Route<dynamic> route) => false,
                                );
                              } else {
                                Navigator.push(
                                  context,
                                  MaterialPageRoute(builder: (context)
=> pages[i]),
```

```
                                    );
                                  }
                                },
                              ),
                            );
                          },
                        ),
                      ),
                      MaterialButton(
                        padding: EdgeInsets.only(bottom: 20),
                        onPressed: () async {
                          if(Platform.isAndroid){
                            if(user!.appMetadata?["provider"] == "google"){
                              GoogleSignIn().signOut();
                            }
                          }
                          await supabase.auth.signOut();
                          final SharedPreferences prefs = await
SharedPreferences.getInstance();
                          await prefs.remove('supabaseSessionToken');
                          Navigator.of(context).pushAndRemoveUntil(
                            MaterialPageRoute(builder: (context) =>
Splash()),
                                (Route<dynamic> route) => false,
                          );
                        },
                        child: Container(
                          decoration: BoxDecoration(
                            color: Colors.red,
                            borderRadius: BorderRadius.circular(10.0),
                            border: Border.all(color: secondryColor, width:
2.0),
                          ),
                          padding: const EdgeInsets.symmetric(horizontal:
20, vertical: 15),
                          child: const Text(
                            "Log Out",
                            style: TextStyle(fontSize: 35, fontWeight:
FontWeight.bold, color: Colors.white),
                          ),
                        ),
                      ),
                    ],
                  ),
                ),
              ),
            ),
            body: Center(
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  MaterialButton(
                    onPressed: () {
```

```
                  connectBluetooth();
                },

                child: Container(
                  decoration: BoxDecoration(
                    color: primaryColor,
                    borderRadius: BorderRadius.circular(10.0),
                    border: Border.all(color: secondryColor, width:
2.0),
                  ),
                  padding: const EdgeInsets.symmetric(horizontal: 15,
vertical: 10),
                  child: const Text(
                    "Connect Bluetooth",
                    style: TextStyle(fontSize: 25, fontWeight:
FontWeight.bold, color: Colors.white),
                  ),
                ),
              ),
              const SizedBox(height: 23),
              Text(success ? "Connection made" : "No connection",
style: TextStyle(fontSize: 20, fontWeight: FontWeight.w700, color:
success ? Colors.green :  Colors.red)),
              const SizedBox(height: 20),
              Text(success ? "Connected to HC-05" : "", style:
TextStyle(fontSize: 20, fontWeight: FontWeight.w700),),

            ],
          ),
        ),
      ),
    );
  }
}
```

- Logs.dart:

```
import 'dart:async';
import 'dart:io';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:google_sign_in/google_sign_in.dart';
import 'package:roboarmv2/splach.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:supabase_flutter/supabase_flutter.dart';
import 'Account.dart';
import 'Bluetooth.dart';
import 'Settings.dart';
import 'colors.dart';
```

```dart
import 'home.dart';
import 'main.dart';

void main() {
  runApp(Logs());
}

class Logs extends StatefulWidget {
  @override
  State<Logs> createState() => _MyAppState();
}

class _MyAppState extends State<Logs> {
  int _currentIndex = 3;
  final List<String> itemName = ["Home", "Account",
"Bluetooth","Logs", "Settings"];

  final List<IconData> icons = [
    Icons.home,
    Icons.account_circle_outlined,
    Icons.bluetooth,
    Icons.code,
    Icons.settings,
  ];

  final List<Widget> pages = [
    home(),
    Account(),
    Bluetooth(),
    Logs(),
    Settings(),
  ];
  final User? user = supabase.auth.currentUser;

  static const MethodChannel _channel =
MethodChannel('bluetooth_channel');
  static const EventChannel _eventChannel =
EventChannel('bluetooth_event_channel');

  StreamSubscription? _streamSubscription;
  TextEditingController _messageController = TextEditingController();
  String _receivedData = "No Message!";
  List<Map<String, dynamic>> _receivedDataList = [];

  String msg = "No Message!";

  final ScrollController _scrollController = ScrollController();

  Future<void> connectBluetooth() async {
    try {
      await _channel.invokeMethod('connectBluetooth');
    } on PlatformException catch (e) {
```

```
      print("Error: ${e.message}");
    }
  }

  Future<bool> writeData(String data) async {
    try {
      final bool success = await _channel.invokeMethod('writeData',
{'message': data});
      print("Send from Flutter");
      return success;
    } on PlatformException catch (e) {
      print("Error: ${e.message}");
      return false;
    }
  }

  String _twoDigits(int n) {
    if (n >= 10) return "$n";
    return "0$n";
  }

  @override
  void initState() {
    super.initState();
    _streamSubscription =
_eventChannel.receiveBroadcastStream().listen((event) {
      setState(() {
        // Add received data with timestamp to the list
        _receivedDataList.add({
          'data': event,
          'timestamp': DateTime.now(), // Current timestamp
        });
        print(event);
        _scrollToBottom();
      });
    }, onError: (error) {
      print("Error receiving data: $error");
    });
  }

  void _scrollToBottom() {
    WidgetsBinding.instance?.addPostFrameCallback((_) {
      if (_scrollController.hasClients) {
        _scrollController.animateTo(
          _scrollController.position.maxScrollExtent,
          duration: Duration(milliseconds: 300),
          curve: Curves.easeOut,
        );
      }
    });
  }
```

```
  @override
  void dispose() {
    _streamSubscription?.cancel();
    _messageController.dispose();
    _scrollController.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
          appBar: AppBar(
            title: const Text(
              "Logs",
              style: TextStyle(
                color: primaryColor,
                fontWeight: FontWeight.bold,
                fontSize: 30,
              ),
            ),
            backgroundColor: secondryColor,
            centerTitle: true,
            actions: [
              IconButton(onPressed: (){
                Navigator.of(context).pushAndRemoveUntil(
                  MaterialPageRoute(builder: (context) => home()),
                      (Route<dynamic> route) => false,
                );
              },
                icon: Icon(Icons.arrow_back_ios,
                  color: primaryColor,
                  size: 30,),)
            ],
          ),
          drawer: Drawer(
            child: Container(
              color: secondryColor,
              child: Column(
                children: [
                  Container(
                    padding: EdgeInsets.only(left: 15, top: 20, right:
15, bottom: 20),
                    child: Row(
                      children: [
                        Image.asset(
                          "images/roboarmlogo.png",
                          height: MediaQuery.of(context).size.height /
10,
                        ),
                        SizedBox(width: 15),
                        Text(
```

```
                        "RoboArm",
                        style: TextStyle(
                          fontWeight: FontWeight.bold,
                          fontSize: 30,
                          color: primaryColor,
                        ),
                      ),
                    ],
                  ),
                ),
                Container(color: backgroundColor, height: 2),
                Expanded(
                  child: ListView.builder(
                    itemCount: itemName.length,
                    itemBuilder: (context, i) {
                      return Card(
                        shape: RoundedRectangleBorder(
                          borderRadius: BorderRadius.circular(10),
                        ),
                        color: backgroundColor,
                        child: ListTile(
                          title: Text(
                            itemName[i],
                            style: TextStyle(
                              fontWeight: FontWeight.bold,
                              fontSize: 25,
                              color: secondryColor,
                            ),
                          ),
                          leading: Icon(
                            icons[i],
                            color: primaryColor,
                            size: 30,
                          ),
                          onTap: () {
                            if (_currentIndex == i) {

Navigator.of(context).pushNamedAndRemoveUntil(
                                    '/',
                                        (Route<dynamic> route) => false,
                                  );
                                } else {
                                  Navigator.push(
                                    context,
                                    MaterialPageRoute(builder: (context)
=> pages[i]),

                                  );
                                }
                              },
                            ),
                            elevation: 1,
                          );
```

```
                    },
                  ),
                ),
                MaterialButton(
                  padding: EdgeInsets.only(bottom: 20),
                  onPressed: () async {
                    if(Platform.isAndroid){
                      if(user!.appMetadata?["provider"] ==
"google"){
                        GoogleSignIn().signOut();
                      }
                    }
                    await supabase.auth.signOut();
                    final SharedPreferences prefs = await
SharedPreferences.getInstance();
                    await prefs.remove('supabaseSessionToken');
                    Navigator.of(context).pushAndRemoveUntil(
                      MaterialPageRoute(builder: (context) =>
Splash()),
                          (Route<dynamic> route) => false,
                    );
                  },
                  child: Container(
                    decoration: BoxDecoration(
                      color: Colors.red,
                      borderRadius: BorderRadius.circular(10.0),
                      border: Border.all(color: secondryColor,
width: 2.0),
                    ),
                    padding: const EdgeInsets.symmetric(horizontal:
20, vertical: 15),
                    child: const Text(
                      "Log Out",
                      style: TextStyle(fontSize: 35, fontWeight:
FontWeight.bold, color: Colors.white),
                    ),
                  ),
                ),
              ],
            ),
          ),
        ),
        body: success? ListView.builder(
          controller: _scrollController,
          itemCount: _receivedDataList.length,
          itemBuilder: (context, index) {
            // Extract data and timestamp
            final receivedData = _receivedDataList[index]['data'];
            final timestamp = _receivedDataList[index]['timestamp']
as DateTime;
            // Format timestamp
            final formattedTime = '${timestamp.year}-
```

```
${_twoDigits(timestamp.month)}-${_twoDigits(timestamp.day)}
${_twoDigits(timestamp.hour)}:${_twoDigits(timestamp.minute)}:${_twoDi
gits(timestamp.second)}';
                return Card(
                  elevation: 4.0,
                  shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(10.0),
                  ),
                  child: ListTile(
                    title: Text(receivedData.substring(2), style:
TextStyle(fontWeight: FontWeight.w700)),
                    subtitle: Text('Received at: $formattedTime'),
                  ),
                );
              },
            ) : Center(
              child: Text("No connection",
                style: TextStyle(
                  fontSize: 30,
                  fontWeight: FontWeight.bold,
                  color: Colors.red,
                ),
              ),
            )
        ),
      );
  }
}
```

- Settings.dart:

```
import 'dart:io';
import 'package:flutter/material.dart';
import 'package:google_sign_in/google_sign_in.dart';
import 'package:roboarmv2/splach.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:supabase_flutter/supabase_flutter.dart';
import 'Account.dart';
import 'Bluetooth.dart';
import 'Logs.dart';
import 'colors.dart';
import 'home.dart';
import 'main.dart';

bool _switchCondition = true;
Future<void> setSwitch(bool state) async{
  final SharedPreferences prefs = await
SharedPreferences.getInstance();
  prefs.setBool('_switchCondition', state);
}

void main() {
```

```
  runApp(Settings());
}

class Settings extends StatefulWidget {
  @override
  State<Settings> createState() => _MyAppState();
}

class _MyAppState extends State<Settings> {

  int _currentIndex = 4;

  final List<String> itemName = ["Home", "Account",
"Bluetooth","Logs", "Settings"];

  final List<IconData> icons = [
    Icons.home,
    Icons.account_circle_outlined,
    Icons.bluetooth,
    Icons.code,
    Icons.settings,
  ];

  final List<Widget> pages = [
    home(),
    Account(),
    Bluetooth(),
    Logs(),
    Settings(),
  ];

  final User? user = supabase.auth.currentUser;



  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: const Text(
            "Settings",
            style: TextStyle(
              color: primaryColor,
              fontWeight: FontWeight.bold,
              fontSize: 30,
            ),
          ),
          backgroundColor: secondryColor,
          centerTitle: true,
          actions: [
            IconButton(onPressed: (){
```

```
              Navigator.of(context).pushAndRemoveUntil(
                MaterialPageRoute(builder: (context) => home()),
                    (Route<dynamic> route) => false,
              );
            },
              icon: Icon(Icons.arrow_back_ios,
                color: primaryColor,
                size: 30,),)
          ],
        ),
        drawer: Drawer(
          child: Container(
            color: secondryColor,
            child: Column(
              children: [
                Container(
                  padding: EdgeInsets.only(left: 15, top: 20, right:
15, bottom: 20),
                  child: Row(
                    children: [
                      Image.asset(
                        "images/roboarmlogo.png",
                        height: MediaQuery.of(context).size.height /
10,
                      ),
                      SizedBox(width: 15),
                      Text(
                        "RoboArm",
                        style: TextStyle(
                          fontWeight: FontWeight.bold,
                          fontSize: 30,
                          color: primaryColor,
                        ),
                      ),
                    ],
                  ),
                ),
                Container(color: backgroundColor, height: 2),
                Expanded(
                  child: ListView.builder(
                    itemCount: itemName.length,
                    itemBuilder: (context, i) {
                      return Card(
                        shape: RoundedRectangleBorder(
                          borderRadius: BorderRadius.circular(10),
                        ),
                        color: backgroundColor,
                        child: ListTile(
                          title: Text(
                            itemName[i],
                            style: TextStyle(
                              fontWeight: FontWeight.bold,
```

```
                                fontSize: 25,
                                color: secondryColor,
                              ),
                            ),
                            leading: Icon(
                              icons[i],
                              color: primaryColor,
                              size: 30,
                            ),
                            onTap: () {
                              if (_currentIndex == i) {

Navigator.of(context).pushNamedAndRemoveUntil(
                                    '/',
                                        (Route<dynamic> route) => false,
                                  );
                                } else {
                                  Navigator.push(
                                    context,
                                    MaterialPageRoute(builder: (context)
=> pages[i]),

                                  );
                                }
                              },
                            ),
                            elevation: 1,
                          );
                        },
                      ),
                    ),
                    MaterialButton(
                      padding: EdgeInsets.only(bottom: 20),
                      onPressed: () async {
                        if(Platform.isAndroid){
                          if(user!.appMetadata?["provider"] == "google"){
                            GoogleSignIn().signOut();
                          }
                        }
                        await supabase.auth.signOut();
                        final SharedPreferences prefs = await
SharedPreferences.getInstance();
                        await prefs.remove('supabaseSessionToken');
                        Navigator.of(context).pushAndRemoveUntil(
                          MaterialPageRoute(builder: (context) =>
Splash()),
                              (Route<dynamic> route) => false,
                        );
                      },
                      child: Container(
                        decoration: BoxDecoration(
                          color: Colors.red,
                          borderRadius: BorderRadius.circular(10.0),
```

```
                      border: Border.all(color: secondryColor, width:
2.0),
                    ),
                    padding: const EdgeInsets.symmetric(horizontal:
20, vertical: 15),
                    child: const Text(
                      "Log Out",
                      style: TextStyle(fontSize: 35, fontWeight:
FontWeight.bold, color: Colors.white),
                    ),
                  ),
                ),
              ],
            ),
          ),
        ),
        body: LayoutBuilder(
          builder: (context, constraints) {
            return SingleChildScrollView(
              child: Card(
                elevation: 4.0,
                shape: RoundedRectangleBorder(
                  borderRadius: BorderRadius.circular(10.0),
                ),
                child: ConstrainedBox(
                  constraints: BoxConstraints(
                    maxWidth: constraints.maxWidth,
                  ),
                  child: Column(
                    children: <Widget>[
                      const SizedBox(height: 20),
                      SwitchListTile(
                        title: Text(_switchCondition?'Auto':"Manual",
style: TextStyle(fontWeight: FontWeight.bold, fontSize: 20,),),
                        value: _switchCondition,
                        activeColor: secondryColor,
                        inactiveThumbColor: primaryColor,
                        activeTrackColor: primaryColor,
                        inactiveTrackColor: secondryColor,
                        onChanged: (value) {
                          setState(() {
                            _switchCondition = value;
                            setSwitch(_switchCondition);

                          });
                        },
                      ),
                    ],
                  ),
                ),
              )
            );
```

```
          },
        ),
      ),
    );
  }
}
```

- AutoPage.dart:

```
import 'dart:async';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'Bluetooth.dart';
import 'colors.dart';

class AutoPage extends StatefulWidget {
  @override
  State<AutoPage> createState() => _AutoPageState();
}

class _AutoPageState extends State<AutoPage> {
  static const MethodChannel _channel =
MethodChannel('bluetooth_channel');
  static const EventChannel _eventChannel =
EventChannel('bluetooth_event_channel');

  StreamSubscription? _streamSubscription;
  TextEditingController _messageController = TextEditingController();
  String _receivedData = "No Message!";

  String msg = "No Message!";


  Future<void> connectBluetooth() async {
    try {
      await _channel.invokeMethod('connectBluetooth');
    } on PlatformException catch (e) {
      print("Error: ${e.message}");
    }
  }

  Future<bool> writeData(String data) async {
    try {
      final bool success = await _channel.invokeMethod('writeData',
{'message': data});
      print("Send from Flutter");
      return success;
    } on PlatformException catch (e) {
      print("Error: ${e.message}");
      return false;
    }
```

```
  }

  // Future<String> readData() async {
  //   try {
  //     final String result = await
_channel.invokeMethod('readData');
  //     return result;
  //   } on PlatformException catch (e) {
  //     print("Error: ${e.message}");
  //     return '';
  //   }
  // }

  @override
  void initState() {
    super.initState();
    _streamSubscription =
_eventChannel.receiveBroadcastStream().listen((event) {
      setState(() {
        _receivedData = event;
        print(_receivedData);
      });
    }, onError: (error) {
      print("Error receiving data: $error");
    });
  }

  @override
  void dispose() {
    _streamSubscription?.cancel();
    _messageController.dispose();
    super.dispose();
  }
  @override
  Widget build(BuildContext context) {
    return Container(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Text(
            'Automatic Moves',
            style: TextStyle(fontSize: 27, fontWeight:
FontWeight.bold),
          ),
          const SizedBox(height: 35),
          MaterialButton(
            onPressed: () {

              if(success){
                writeData("M");
              }else{
                null;
```

```
              }

            },
            child: Container(
              decoration: BoxDecoration(
                color: success? primaryColor: Colors.grey,
                borderRadius: BorderRadius.circular(10.0),
                border: Border.all(color: secondryColor, width: 2.0),
              ),
              padding: const EdgeInsets.symmetric(horizontal: 20,
vertical: 10),
              child: Row(
                mainAxisSize: MainAxisSize.min,
                children: [
                  Icon(
                    Icons.numbers_sharp,
                    color: Colors.white,
                    size: 25,
                  ),
                  const SizedBox(width: 8),
                  const Text(
                    "Move 1",
                    style: TextStyle(
                      fontSize: 25,
                      fontWeight: FontWeight.bold,
                      color: Colors.white,
                    ),
                  ),
                ],
              ),
            ),
          ),
          const SizedBox(height: 30),
          MaterialButton(
            onPressed: () {
              if(success){
                writeData("N");
              }else{
                null;
              }
            },
            child: Container(
              decoration: BoxDecoration(
                color: success? primaryColor: Colors.grey,
                borderRadius: BorderRadius.circular(10.0),
                border: Border.all(color: secondryColor, width: 2.0),
              ),
              padding: const EdgeInsets.symmetric(horizontal: 15,
vertical: 10),
              child: Row(
                mainAxisSize: MainAxisSize.min,
                children: [
```

```
                    Icon(
                      Icons.numbers_sharp,
                      color: Colors.white,
                      size: 25,
                    ),
                    const SizedBox(width: 8),
                    const Text(
                      "Move 2",
                      style: TextStyle(
                        fontSize: 25,
                        fontWeight: FontWeight.bold,
                        color: Colors.white,
                      ),
                    ),
                  ],
                ),
              ),
            ),
            const SizedBox(height: 30),
            Text(
              success
                  ? _receivedData
                  : "No connection",
              style: TextStyle(
                fontSize: 20,
                fontWeight: FontWeight.w700,
                color: success ? Colors.green : Colors.red,
              ),
            )

          ],
        ),
      );
  }
}
```

- ControlPage.dart:

```
import 'dart:async';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'Bluetooth.dart';
import 'colors.dart';

Future<bool?> getSwitch() async {
  final SharedPreferences prefs = await
SharedPreferences.getInstance();
  bool? switchCondition = prefs.getBool('_switchCondition');
```

```
  return switchCondition;
}

class ControlPage extends StatefulWidget {
  @override
  State<ControlPage> createState() => _ControlPageState();
}


class _ControlPageState extends State<ControlPage> {

  bool _switchCondition = true;

  void _loadSwitchCondition() async {
    bool? condition = await getSwitch();
    setState(() {
      _switchCondition = condition ?? true;
    });
  }

  static const MethodChannel _channel =
MethodChannel('bluetooth_channel');
  static const EventChannel _eventChannel =
EventChannel('bluetooth_event_channel');

  StreamSubscription? _streamSubscription;
  TextEditingController _messageController = TextEditingController();
  String _receivedData = "No Message!";

  String msg = "No Message!";

  Future<void> connectBluetooth() async {
    try {
      await _channel.invokeMethod('connectBluetooth');
    } on PlatformException catch (e) {
      print("Error: ${e.message}");
    }
  }

  Future<bool> writeData(String data) async {
    try {
      final bool success = await _channel.invokeMethod('writeData',
{'message': data});
      print("Send from Flutter");
      return success;
    } on PlatformException catch (e) {
      print("Error: ${e.message}");
      return false;
    }
  }

  // Future<String> readData() async {
```

```
  //   try {
  //     final String result = await
_channel.invokeMethod('readData');
  //     return result;
  //   } on PlatformException catch (e) {
  //     print("Error: ${e.message}");
  //     return '';
  //   }
  // }

  @override
  void initState() {
    super.initState();
    _streamSubscription =
_eventChannel.receiveBroadcastStream().listen((event) {
      setState(() {
        _receivedData = event;
        print(_receivedData);
      });
    }, onError: (error) {
      print("Error receiving data: $error");
    });
    _loadSwitchCondition();
  }

  @override
  void dispose() {
    _streamSubscription?.cancel();
    _messageController.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Container(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Text(
            'Control the arm',
            style: TextStyle(fontSize: 27, fontWeight:
FontWeight.bold),
          ),
          const SizedBox(height: 35),
          Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              MaterialButton(
                onPressed: _switchCondition && success? () =>
writeData("P") : null,
                child: Container(
                  decoration: BoxDecoration(
```

```
                    color: _switchCondition && success? primaryColor :
Colors.grey,
                    borderRadius: BorderRadius.circular(10.0),
                    border: Border.all(color: secondryColor, width:
2.0),
                  ),
                  padding: const EdgeInsets.symmetric(horizontal: 20,
vertical: 10),
                  child: Row(
                    mainAxisSize: MainAxisSize.min,
                    children: [
                      Icon(
                        Icons.play_arrow,
                        color: Colors.white,
                        size: 25,
                      ),
                      const SizedBox(width: 8),
                      const Text(
                        "Play",
                        style: TextStyle(
                          fontSize: 25,
                          fontWeight: FontWeight.bold,
                          color: Colors.white,
                        ),
                      ),
                    ],
                  ),
                ),
              ),
              MaterialButton(
                onPressed: _switchCondition && success? () =>
writeData("S") : null,
                child: Container(
                  decoration: BoxDecoration(
                    color: _switchCondition && success? primaryColor :
Colors.grey,
                    borderRadius: BorderRadius.circular(10.0),
                    border: Border.all(color: secondryColor, width:
2.0),
                  ),
                  padding: const EdgeInsets.symmetric(horizontal: 15,
vertical: 10),
                  child: Row(
                    mainAxisSize: MainAxisSize.min,
                    children: [
                      Icon(
                        Icons.stop,
                        color: Colors.white,
                        size: 25,
                      ),
                      const SizedBox(width: 8),
                      const Text(
```

```
                        "Stop",
                        style: TextStyle(
                          fontSize: 25,
                          fontWeight: FontWeight.bold,
                          color: Colors.white,
                        ),
                      ),
                    ],
                  ),
                ),
              ),
            ],
          ),
          const SizedBox(height: 27),
          Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              MaterialButton(
                onPressed: _switchCondition && success? () =>
writeData("V") : null,
                child: Container(
                  decoration: BoxDecoration(
                    color: _switchCondition && success? primaryColor :
Colors.grey,
                    borderRadius: BorderRadius.circular(10.0),
                    border: Border.all(color: secondryColor, width:
2.0),
                  ),
                  padding: const EdgeInsets.symmetric(horizontal: 15,
vertical: 10),
                  child: Row(
                    mainAxisSize: MainAxisSize.min,
                    children: [
                      Icon(
                        Icons.save,
                        color: Colors.white,
                        size: 25,
                      ),
                      const SizedBox(width: 8),
                      const Text(
                        "Save",
                        style: TextStyle(
                          fontSize: 32,
                          fontWeight: FontWeight.bold,
                          color: Colors.white,
                        ),
                      ),
                    ],
                  ),
                ),
              ),
              MaterialButton(
```

```
                onPressed: _switchCondition && success? () =>
writeData("R") : null,
                child: Container(
                  decoration: BoxDecoration(
                    color: _switchCondition && success? primaryColor :
Colors.grey,
                    borderRadius: BorderRadius.circular(10.0),
                    border: Border.all(color: secondryColor, width:
2.0),
                  ),
                  padding: const EdgeInsets.symmetric(horizontal: 15,
vertical: 10),
                  child: Row(
                    mainAxisSize: MainAxisSize.min,
                    children: [
                      Icon(
                        Icons.restart_alt,
                        color: Colors.white,
                        size: 25,
                      ),
                      const SizedBox(width: 8),
                      const Text(
                        "Reset",
                        style: TextStyle(
                          fontSize: 25,
                          fontWeight: FontWeight.bold,
                          color: Colors.white,
                        ),
                      ),
                    ],
                  ),
                ),
              ),
            ],
          ),
          const SizedBox(height: 30),
          Text(
            _switchCondition && success
                ? _receivedData
                : "No connection",
            style: TextStyle(
              fontSize: 20,
              fontWeight: FontWeight.w700,
              color: _switchCondition && success? Colors.green :
Colors.red,
            ),
          )
        ],
      ),
    );
  }
}
```

- ManualPage.dart:

```
import 'dart:async';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:roboarmv2/colors.dart';
import 'package:shared_preferences/shared_preferences.dart';

import 'Bluetooth.dart';
Future<bool> getSwitch() async {
  final SharedPreferences prefs = await
SharedPreferences.getInstance();
  bool? switchCondition = prefs.getBool('_switchCondition');
  return switchCondition ?? true;
}

class ManualPage extends StatefulWidget {
  @override
  State<ManualPage> createState() => _ManualPageState();
}

class _ManualPageState extends State<ManualPage> {
  bool _switchCondition = true; // Initial default value

  void _loadSwitchCondition() async {
    bool condition = await getSwitch();
    setState(() {
      _switchCondition = condition;
    });
  }
  double slider1 = 20;
  double slider2 = 90;
  double slider3 = 0;
  double slider4 = 90;
  double slider5 = 0;
  double slider6 = 120;

  String slider1S = "";
  String slider2S = "";
  String slider3S = "";
  String slider4S = "";
  String slider5S = "";
  String slider6S = "";

  static const MethodChannel _channel =
MethodChannel('bluetooth_channel');
  static const EventChannel _eventChannel =
EventChannel('bluetooth_event_channel');

  StreamSubscription? _streamSubscription;
```

```
  TextEditingController _messageController = TextEditingController();
  String _receivedData = "No Message!";

  String msg = "No Message!";


  Future<void> connectBluetooth() async {
    try {
      await _channel.invokeMethod('connectBluetooth');
    } on PlatformException catch (e) {
      print("Error: ${e.message}");
    }
  }

  Future<bool> writeData(String data) async {
    try {
      final bool success = await _channel.invokeMethod('writeData',
{'message': data});
      print("Send from Flutter");
      return success;
    } on PlatformException catch (e) {
      print("Error: ${e.message}");
      return false;
    }
  }

  // Future<String> readData() async {
  //   try {
  //     final String result = await
_channel.invokeMethod('readData');
  //     return result;
  //   } on PlatformException catch (e) {
  //     print("Error: ${e.message}");
  //     return '';
  //   }
  // }

  @override
  void initState() {
    super.initState();
    _streamSubscription =
_eventChannel.receiveBroadcastStream().listen((event) {
      setState(() {
        _receivedData = event;
        print(_receivedData);
      });
    }, onError: (error) {
      print("Error receiving data: $error");
    });
    _loadSwitchCondition();
  }
```

```
  @override
  void dispose() {
    _streamSubscription?.cancel();
    _messageController.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return SingleChildScrollView(
      child: Padding(
        padding: const EdgeInsets.all(8),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            const SizedBox(height: 20),
            Text(
              'Manual',
              style: TextStyle(fontSize: 27, fontWeight:
FontWeight.bold),
            ),
            const SizedBox(height: 20),
            buildSliderRow('Waist', slider1, 20, 180, 32, (value) {
              setState(() {
                slider1 = value;
                slider1S = "s1" + slider1.toString();
                writeData(slider1S);
                print(slider1S);
              });
            }, success && _switchCondition == false),
            const SizedBox(height: 25),
            buildSliderRow('Shoulder', slider2, 90, 150, 12, (value) {
              setState(() {
                slider2 = value;
                slider2S = "s2" + slider2.toString();
                writeData(slider2S);
                print(slider2S);
              });
            }, success && _switchCondition == false),
            const SizedBox(height: 25),
            buildSliderRow('Elbow', slider3, 0, 90, 18, (value) {
              setState(() {
                slider3 = value;
                slider3S = "s3" + slider3.toString();
                writeData(slider3S);
                print(slider3S);
              });
            }, success && _switchCondition == false),
            const SizedBox(height: 25),
            buildSliderRow('Wrist Roll', slider4, 90, 180, 18, (value)
{
              setState(() {
```

```
                slider4 = value;
                slider4S = "s4" + slider4.toString();
                writeData(slider4S);
                print(slider4S);
              });
            }, success && _switchCondition == false),
            const SizedBox(height: 25),
            buildSliderRow('Wrist Pitch', slider5, 0, 180, 36, (value)
{
              setState(() {
                slider5 = value;
                slider5S = "s5" + slider5.toString();
                writeData(slider5S);
                print(slider5S);
              });
            }, success && _switchCondition == false),
            const SizedBox(height: 25),
            buildSliderRow('Grip', slider6, 120, 180, 12, (value) {
              setState(() {
                slider6 = value;
                slider6S = "s6" + slider6.toString();
                writeData(slider6S);
                print(slider6S);
              });
            }, success && _switchCondition == false),
            buildActionButtonRow(),
            const SizedBox(height: 30),
            Text(
              success && _switchCondition == false
                  ? _receivedData
                  : "No connection",
              style: TextStyle(
                fontSize: 20,
                fontWeight: FontWeight.w700,
                color: success ? Colors.green : Colors.red,
              ),
            ),
          ],
        ),
      ),
    );
  }

  Widget buildSliderRow(String label, double value, double min, double
max, int divisions, ValueChanged<double> onChanged, bool isEnabled) {
    return Card(
      elevation: 4.0,
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(10.0),
      ),
      child: Padding(
        padding: const EdgeInsets.all(16.0),
```

```
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(
              label,
              style: TextStyle(fontSize: 18, fontWeight:
FontWeight.w700),
            ),
            const SizedBox(height: 10),
            Container(
              height: 50,
              child: SliderTheme(
                data: SliderTheme.of(context).copyWith(
                  activeTrackColor: primaryColor,
                  inactiveTrackColor: secondryColor,
                  trackHeight: 10.0,
                  thumbColor: Colors.amberAccent,
                  thumbShape:
RoundSliderThumbShape(enabledThumbRadius: 12.0),
                  overlayShape: RoundSliderOverlayShape(overlayRadius:
24.0),
                  tickMarkShape: RoundSliderTickMarkShape(),
                  valueIndicatorShape:
PaddleSliderValueIndicatorShape(),
                  inactiveTickMarkColor: primaryColor,
                  valueIndicatorColor: primaryColor,
                  valueIndicatorTextStyle: TextStyle(
                    color: Colors.white,
                    fontWeight: FontWeight.bold,
                  ),
                ),
                child: Slider(
                  value: value,
                  min: min,
                  max: max,
                  divisions: divisions,
                  label: value.round().toString(),
                  onChanged: isEnabled ? onChanged : null, // Disable
the slider if isEnabled is false
                ),
              ),
            ),
          ],
        ),
      ),
    );
  }



  Widget buildActionButtonRow() {
    return Column(
```

```
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        const SizedBox(height: 35),
        Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            buildMaterialButton(Icons.play_arrow, "Play", "P"),
            buildMaterialButton(Icons.stop, "Stop", "S"),
          ],
        ),
        const SizedBox(height: 25),
        Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            buildMaterialButton(Icons.save, "Save", "V"),
            buildMaterialButton(Icons.restart_alt, "Reset", "R"),
          ],
        ),
      ],
    );
  }

  Widget buildMaterialButton(IconData icon, String label, String
command) {
    return MaterialButton(
      onPressed: success ? () => writeData(command) : null,
      child: Container(
        decoration: BoxDecoration(
          color: success ? primaryColor : Colors.grey,
          borderRadius: BorderRadius.circular(10.0),
          border: Border.all(color: secondryColor, width: 2.0),
        ),
        padding: const EdgeInsets.symmetric(horizontal: 20, vertical:
10),
        child: Row(
          mainAxisSize: MainAxisSize.min,
          children: [
            Icon(
              icon,
              color: Colors.white,
              size: 20,
            ),
            const SizedBox(width: 8),
            Text(
              label,
              style: TextStyle(
                fontSize: 20,
                fontWeight: FontWeight.bold,
                color: Colors.white,
              ),
            ),
          ],
```

```
          ),
        ),
      );
    }
}
```

---

e-mail templates

- index.html: validate the new e-mail:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="icon" type="image/x-icon" href="images/roboarmlogo.png">
    <title>Email Verification Success</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #EFEFEF;
            margin: 0;
            padding: 0;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
        }

        .container {
            text-align: center;
            background-color: #FFF;
            padding: 30px;
            border-radius: 10px;
            box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.1);
            animation: fadeIn 0.5s ease-out;
            position: relative;
        }

        .container img {
            width: 200px;
```

```
    border-radius: 50%;
    margin-bottom: 20px;
}

h1 {
    color: #4D4D4D;
    margin-bottom: 20px;
    animation: slideInDown 0.5s ease-out;
}

p {
    color: #666;
    margin-bottom: 10px;
    animation: slideInUp 0.5s ease-out;
}

#p {
    color: #3186A0;
    text-decoration: none;
    font-weight: bold;
    transition: color 0.3s ease;
}

#p:hover {
    color: #0056b3;
}

@keyframes fadeIn {
    from {
        opacity: 0;
    }

    to {
        opacity: 1;
    }
}

@keyframes slideInDown {
    from {
        transform: translateY(-50%);
    }

    to {
        transform: translateY(0);
    }
```

```
        }

        @keyframes slideInUp {
            from {
                transform: translateY(50%);
            }

            to {
                transform: translateY(0);
            }
        }
    </style>
</head>

<body>
    <div class="container">
        <img src="images/roboarmlogo.png" alt="roboarmlogo">
        <h1>Email Verified Successfully!</h1>
        <p>Welcome back! Your email has been successfully verified.</p>
        <p id="p">Please close this window and return to the app to login</p>
    </div>
</body>

</html>
```

- resetPassword.html: Page to reset the password

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="icon" type="image/x-icon" href="../images/roboarmlogo.png">
    <title>RoboArm Reset Password</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #EFEFEF;
            margin: 0;
            padding: 0;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
```

```
}

.container {
    text-align: center;
    background-color: #FFF;
    padding: 30px;
    border-radius: 10px;
    box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.1);
    width: 370px;
}

h1 {
    color: #4D4D4D;
    margin-bottom: 20px;
}

.form-group {
    margin-bottom: 20px;
}

label {
    display: block;
    margin-bottom: 5px;
}

input {
    width: 100%;
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 5px;
}

button {
    padding: 10px 20px;
    background-color: #3186A0;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}

button:hover {
    background-color: #0056b3;
}
```

```
        #error-msg {
            color: red;
            margin-top: 10px;
        }
    </style>
</head>

<body>
    <div class="container">
        <h1>Reset Password</h1>
        <form id="passwordForm">
            <div class="form-group">
                <label for="email">Email:</label>
                <input type="email" id="email" name="email">
            </div>
            <div class="form-group">
                <label for="newPassword">New Password:</label>
                <input type="password" id="newPassword" name="newPassword">
            </div>
            <div class="form-group">
                <label for="confirmPassword">Confirm Password:</label>
                <input type="password" id="confirmPassword"
name="confirmPassword">
            </div>
            <button type="submit">Submit</button>
            <p id="error-msg"></p>
        </form>
    </div>
</body>
<script type="module" src="../js/resetPassword.js"></script>
</html>
```

- resetPassword.js:

```
import { createClient } from 'https://esm.sh/@supabase/supabase-js@2';

const supabase = createClient('https://xditbsgzrqnyqrlrxset.supabase.co',
'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6InhkaXR
ic2d6cnFueXFybHJ4c2V0Iiwicm9sZSI6ImFub24iLCJpYXQiOjE3MDgxOTI3NjcsImV4cCI6MjAy
Mzc2ODc2N30.uwQA5-Bt4T7Jv5QtjSdkvnOiCIesePoumYEGU3bUv64');

const emailInput = document.getElementById("email");
const passwordForm = document.getElementById('passwordForm');
const newPasswordInput = document.getElementById('newPassword');
```

```
const confirmPasswordInput = document.getElementById('confirmPassword');
const errorMsg = document.getElementById('error-msg');

passwordForm.addEventListener('submit', async function (event) {
    event.preventDefault();
    if (validatePassword()) {
        errorMsg.textContent = '';
        try {
            const { data, error } = await supabase.auth.updateUser({
                email: emailInput.value,
                password: newPasswordInput.value
            });
            window.location.href = "updateChanged.html";
        } catch (error) {
            errorMsg.textContent = 'An error has occurred, please try
again!';
        }
    }
});

function validatePassword() {
    const newPassword = newPasswordInput.value;
    const confirmPassword = confirmPasswordInput.value;
    const email = emailInput.value;

    if (newPassword.length < 8) {
        errorMsg.textContent = 'Password must be at least 8 characters long
and contain at least one lowercase letter, one uppercase letter, one digit,
and one special character';
        return false;
    }

    if (newPassword !== confirmPassword) {
        errorMsg.textContent = 'Passwords do not match';
        return false;
    }
    if (email == "") {
        errorMsg.textContent = 'Email filed is empty';
        return false;
    }

    return true;
}
```

- updateChanged.html:

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="icon" type="image/x-icon" href="../images/roboarmlogo.png">
    <title>Update Success</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #EFEFEF;
            margin: 0;
            padding: 0;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
        }

        .container {
            text-align: center;
            background-color: #FFF;
            padding: 30px;
            border-radius: 10px;
            box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.1);
            animation: fadeIn 0.5s ease-out;
            position: relative;
        }

        .container img {
            width: 200px;
            border-radius: 50%;
            margin-bottom: 20px;
        }

        h1 {
            color: #4D4D4D;
            margin-bottom: 20px;
            animation: slideInDown 0.5s ease-out;
        }

        p {
            color: #666;
            margin-bottom: 10px;
```

```
            animation: slideInUp 0.5s ease-out;
        }

        #p {
            color: #3186A0;
            text-decoration: none;
            font-weight: bold;
            transition: color 0.3s ease;
        }

        #p:hover {
            color: #0056b3;
        }

        @keyframes fadeIn {
            from {
                opacity: 0;
            }

            to {
                opacity: 1;
            }
        }

        @keyframes slideInDown {
            from {
                transform: translateY(-50%);
            }

            to {
                transform: translateY(0);
            }
        }

        @keyframes slideInUp {
            from {
                transform: translateY(50%);
            }

            to {
                transform: translateY(0);
            }
        }
    </style>
</head>
```

```
<body>
    <div class="container">
        <img src="../images/roboarmlogo.png" alt="roboarmlogo">
        <h1>Updated Successfully!</h1>
        <p>Your update has done uccessfully</p>
        <p id="p">Please close this window and return to the app to login</p>
    </div>
</body>

</html>
```

- 404.html: Page to the error 404 not found.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Page Not Found</title>

    <style media="screen">
      body { background: #ECEFF1; color: rgba(0,0,0,0.87); font-family:
Roboto, Helvetica, Arial, sans-serif; margin: 0; padding: 0; }
      #message { background: white; max-width: 360px; margin: 100px auto
16px; padding: 32px 24px 16px; border-radius: 3px; }
      #message h3 { color: #888; font-weight: normal; font-size: 16px;
margin: 16px 0 12px; }
      #message h2 { color: #ffa100; font-weight: bold; font-size: 16px;
margin: 0 0 8px; }
      #message h1 { font-size: 22px; font-weight: 300; color:
rgba(0,0,0,0.6); margin: 0 0 16px;}
      #message p { line-height: 140%; margin: 16px 0 24px; font-size: 14px; }
      #message a { display: block; text-align: center; background: #039be5;
text-transform: uppercase; text-decoration: none; color: white; padding:
16px; border-radius: 4px; }
      #message, #message a { box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px
2px rgba(0,0,0,0.24); }
      #load { color: rgba(0,0,0,0.4); text-align: center; font-size: 13px; }
      @media (max-width: 600px) {
        body, #message { margin-top: 0; background: white; box-shadow: none;
}
        body { border-top: 16px solid #ffa100; }
      }
    </style>
```

```
  </head>
  <body>
    <div id="message">
      <h2>404</h2>
      <h1>Page Not Found</h1>
      <p>The specified file was not found on this website. Please check the
URL for mistakes and try again.</p>
      <h3>Why am I seeing this?</h3>
      <p>This page was generated by the Firebase Command-Line Interface. To
modify it, edit the <code>404.html</code> file in your project's configured
<code>public</code> directory.</p>
    </div>
  </body>
</html>
```

- Email Templates: Customize the emails that will be sent out to the users:
  - Confirm Signup:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Confirm Your Signup</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            margin: 0;
            padding: 0;
        }
        .container {
            max-width: 600px;
            margin: 20px auto;
            padding: 20px;
            background-color: #fff;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0,0,0,0.1);
        }
        h2 {
            color: #333;
        }
        p {
            color: #666;
```

```
            margin-bottom: 20px;
        }
        a {
            display: inline-block;
            background-color: #007bff;
            color: #fff;
            text-decoration: none;
            padding: 10px 20px;
            border-radius: 5px;
        }
        a:hover {
            background-color: #0056b3;
        }
    </style>
</head>
<body>
    <div class="container">
        <h2>Confirm your signup</h2>
        <p>Follow this link to confirm your user:</p>
        <p><a href="{{ .ConfirmationURL }}">Confirm your
mail</a></p>
    </div>
</body>
</html>

```

- RoboArm Invite:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>RoboArm Invite</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            margin: 0;
            padding: 0;
        }
```

```
        .container {
            max-width: 600px;
            margin: 20px auto;
            padding: 20px;
            background-color: #fff;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0,0,0,0.1);
        }
        h2 {
            color: #333;
        }
        p {
            color: #666;
            margin-bottom: 20px;
        }
        a {
            display: inline-block;
            background-color: #007bff;
            color: #fff;
            text-decoration: none;
            padding: 10px 20px;
            border-radius: 5px;
        }
        a:hover {
            background-color: #0056b3;
        }
    </style>
</head>
<body>
    <div class="container">
        <h2>You have been invited</h2>
        <p>You have been invited to create a user on {{ .SiteURL }}. Follow this link to accept the invite:</p>
        <p><a href="{{ .ConfirmationURL }}">Accept the invite</a></p>
    </div>
</body>
</html>
```

- Change email address:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title>Confirm Change of Email</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            margin: 0;
            padding: 0;
        }
        .container {
            max-width: 600px;
            margin: 20px auto;
            padding: 20px;
            background-color: #fff;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0,0,0,0.1);
        }
        h2 {
            color: #333;
        }
        p {
            color: #666;
            margin-bottom: 20px;
        }
        a {
            display: inline-block;
            background-color: #007bff;
            color: #fff;
            text-decoration: none;
            padding: 10px 20px;
            border-radius: 5px;
        }
        a:hover {
            background-color: #0056b3;
        }
    </style>
</head>
<body>
    <div class="container">
        <h2>Confirm Change of Email</h2>
        <p>Follow this link to confirm the update of your email from
{{ .Email }} to {{ .NewEmail }}:</p>
        <p><a href="{{ .ConfirmationURL }}">Change Email</a></p>
    </div>
</body>
</html>
```

- Reset Password:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Reset Password</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            margin: 0;
            padding: 0;}
        .container {
            max-width: 600px;
            margin: 20px auto;
            padding: 20px;
            background-color: #fff;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0,0,0,0.1);
        }
        h2 {
            color: #333;
        }
        p {
            color: #666;
            margin-bottom: 20px;
        }
        a {
            display: inline-block;
            background-color: #007bff;
            color: #fff;
            text-decoration: none;
            padding: 10px 20px;
            border-radius: 5px;
        }
        a:hover {
            background-color: #0056b3;
        }
    </style></head><body>
    <div class="container">
        <h2>Reset Password</h2>
        <p>Follow this link to reset the password for your user:</p>
        <p><a href="{{ .ConfirmationURL }}/p/resetPassword.html">Reset Password</a></p></div></body></html>
```

References

- *Flutter Documentation - https://docs.flutter.dev/*
- *Firebase Documentation - https://firebase.google.com/docs?hl=en*
- *Supabase Flutter Documentation - https://supabase.com/docs/reference/dart/installing*
- *Using OAuth 2.0 to Access Google APIs - https://developers.google.com/identity/protocols/oauth2*
- *Flutter packages - https://pub.dev/*
- *Modeling and Analysis of a 6 DOF Robotic Arm Manipulator https://www.researchgate.net/profile/Jamshed-Iqbal-2/publication/280643085_Modeling_and_analysis_of_a_6_DOF_robotic_arm_manipulator/links/55c0a56b08aed621de13cf59/Modeling-and-analysis-of-a-6-DOF-robotic-arm-manipulator.pdf*

Summary

Project Description: It is an innovative robotic arm designed to improve material handling in industrial environments. This state-of-the-art project is designed to move boxes seamlessly between moving conveyor belts with precision and adaptability.

The arm, equipped with an advanced sensor system, detects incoming bins on the belt, quickly stopping the conveyor's movement. Upon detection, the robotic arm maneuvers efficiently, smoothly grabs the box and moves it to the designated belt.

The user interface provides multi-faceted control mechanisms. With dedicated desktop software and an easy-to-use mobile app, operators can seamlessly manage arm procedures. It is worth noting that the lever holder, which is regulated by Joysticks, allows fine-tuning and fine adjustments.

One great feature is the ability to record and store various arm movements. Users can pre-select and save specific movements, allowing for easy selection via arm interface buttons, desktop software, or mobile app.

Designed primarily for the integration of industrial plants, the arm revolutionizes logistics operations, ensuring efficient automated box transfers between conveyor belts while providing comprehensive control and adaptability for various tasks.