SENIOR DESIGN PROJECT

COCKTAIL BARTENDER

Mentor David Chkhaidze and Students

Andro Mazmishvili

Ioane Mania

Irakli Chichua

Anri Aleksandria

Erekle Shatirishvili

San Diego State University

17/05/2023

Contents

Abstract

This report presents the design and development of an automated cocktail bartender device, aimed at revolutionizing the cocktail making process. The project's objective was to create a sophisticated and user-friendly system that could accurately measure and dispense precise amounts of various ingredients to craft delicious cocktails.

The automated cocktail bartender device consists of a combination of hardware and software components. The hardware includes pumps, Peltier cooler, relays, tubes, and ESP, while the software encompasses a control system and a database of cocktail recipes and mobile application. The device allows users to select their desired cocktail from a mobile application, and it automatically dispenses the necessary ingredients with precise measurements. On the other hand, a user can pour a custom cocktail based on his/her desire.

The development process involved several stages, including conceptualization, prototyping, testing, and refinement. The hardware components were carefully selected and integrated to ensure efficient and reliable operation. The control system was programmed to synchronize the pumps and water cooler for accurate ingredient dispensing. Extensive testing was conducted to validate the device's performance and to fine-tune the user interface for intuitive operation.

Results demonstrated that the automated cocktail bartender device successfully produced a wide range of cocktails with consistent quality. The system exhibited precise measurement capabilities, ensuring that the correct proportions of ingredients were dispensed for each cocktail. The user interface provided a seamless and enjoyable experience, enabling users to easily navigate through the cocktail selection process.

Overall, this project demonstrates the successful creation of an automated cocktail bartender device, showcasing its potential to transform the cocktail industry by providing an efficient, consistent, and enjoyable cocktail-making experience.

Introduction

In today's fast-paced world, where efficiency and convenience are highly valued, we present your very own automated cocktail bartender. Bartender is an innovative and intelligent device designed to bring the art of mixology to your fingertips, revolutionizing the way you enjoy cocktails in the comfort of your home, at events, or even in bars and restaurants.

Imagine having a professional bartender available at any time, ready to create the perfect cocktail tailored to your taste preferences. With the Cocktail Bartender, the possibilities are endless. Whether you're hosting a party, enjoying a quiet evening at home, or even exploring the realm of mixology as an aspiring bartender, this device is your trusted partner in crafting exquisite cocktails with precision and flair.

Using a device is a breeze. Simply select your desired cocktail from the extensive menu or customize your own by specifying ingredients, measurements, and techniques. Its precise dispensing system ensures accurate proportions, delivering consistent and flavorful results every time.

To address the problem of boring cocktail crafting, our team at SDSI_Georgia dedicated our time and funding to researching the best components on the market and implementing them in our project. We bought water pumps that give a precise amount of liquid over a second. We ordered a Peliter water cooler to make our drinks fresh and enjoyable to drink. To control liquid flow we purchased ESP32 with an 8x relay module attached to it so that we have a perfect control system.

Overall, after dozens of testing and precise measures we have built a cocktail bartender which can make the cocktail crafting process easy and enjoyable.

Body

ESP and the Relay Module

The ESP32 microcontroller played a crucial role in our project by serving as the main control unit for various functionalities. The ESP32 is a powerful and versatile microcontroller that integrates Wi-Fi and Bluetooth capabilities, making it an ideal choice for our project's requirements. The ESP32 microcontroller had two important tasks in our project:

Control of Liquid Flow: One of the primary functions of the ESP32 microcontroller in our project was to regulate the flow of liquids through the pumps. By interfacing with the pumps through appropriate driver circuits, the microcontroller could control the activation and deactivation of the pumps, allowing us to pour different types of cocktails accurately. This control was essential to ensure precise measurements and consistent drink quality.

Direct control of pumps using an ESP32 microcontroller is not feasible due to the differences in voltage and current requirements between the microcontroller and the pumps.

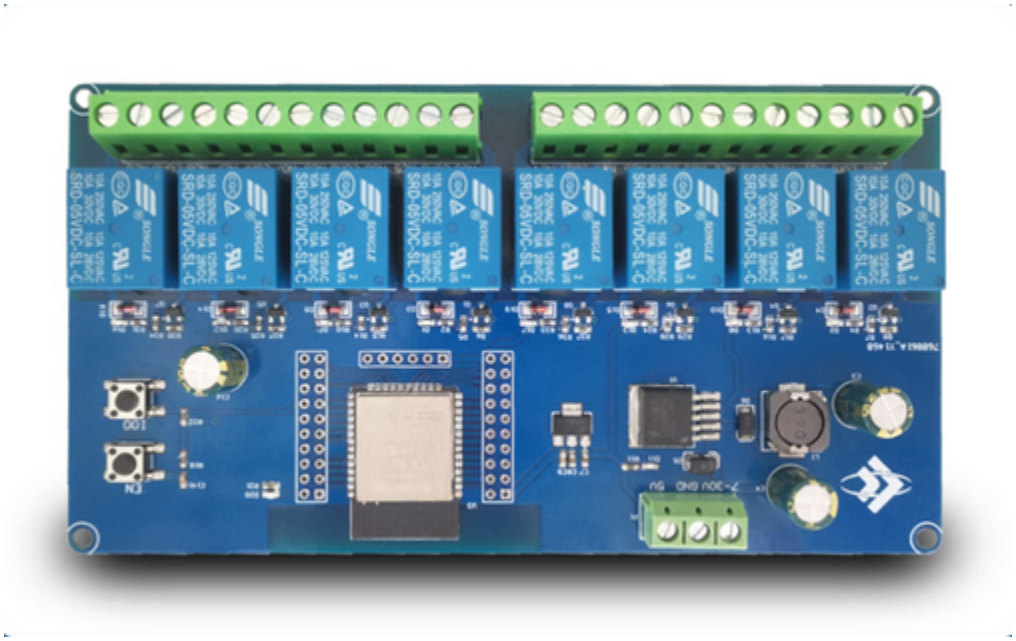
Pumps used in our cocktail bartender or other similar applications often operate at higher voltages and currents than what the ESP32 microcontroller can handle directly. The pumps used in our project operate at 12V and The ESP32 typically operates at 3.3V or 5V logic levels, which is not sufficient to power or control the pumps effectively.

Furthermore, pumps often draw significant amounts of current and can introduce electrical noise or voltage fluctuations that may adversely affect the stability and integrity of the microcontroller.

For the above reasons, direct connection of pumps with the ESP32 microcontroller is not an effective or a reliable solution, so we introduced relays. Relays are electromechanical devices that can handle higher voltage and current levels. They act as switches, allowing the microcontroller to control the flow of current to the pumps indirectly. By using a relay module, the microcontroller can energize or de-energize the relay coil with a low-power signal, while the relay itself handles the high-power current required to operate the pumps.

This means that our microcontroller circuit is separated from the pump circuit by the use of relays, providing us with electrical isolation and protection of the microcontroller from potential voltage spikes or back EMF (electromotive force) generated by the pumps.

We have decided to use an ESP32 microcontroller with a built in 8x relay module, which is a perfect fit for our use case.



This module contains two important components (full specifications can be found in the references):

- Mature and stable on-board ESP32-WROOM-32E module with a large capacity of 4MB Flash
- Onboard 8-channel 5V relay, output switch signal, suitable for controlling the load whose control voltage is within AC 250V/DC 30V;

Six of the onboard 5V relays are connected to a dedicated pump responsible for dispensing different cocktail liquids. Additionally, an extra relay is connected to a pump responsible for dispensing liquid from the PELTIER water cooler into a cup. To achieve optimal cooling, all the cocktail liquid pumps are combined into the PELTIER pump, ensuring the liquids are sufficiently cooled before being poured directly into a cup as a perfectly prepared cocktail.

The connection between the liquid and PELTIER pumps is made using an internal funnel. All of the liquid pumps are attached and poured into the funnel and the PELTIER water cooler is connected to the output of the funnel, where it collects all the dispensed liquids.

Remote Communication with a Mobile Interface: The ESP32 microcontroller facilitated the establishment of a Bluetooth interface, which enabled remote control of the pumps and the entire cocktail making process. With this interface, users could control the machine wirelessly from a mobile device using our mobile application. The microcontroller handled the Bluetooth communication protocols, allowing seamless interaction between the machine and the user's device.

The ESP32 microcontroller offers two Bluetooth modes: Classic Bluetooth and Bluetooth Low Energy (BLE). These modes provide different functionalities and are suitable for various use cases:

Classic Bluetooth: Classic Bluetooth, also known as Bluetooth Basic Rate/Enhanced Data Rate (BR/EDR), is the traditional Bluetooth technology that has been widely used for wireless audio streaming, file transfer between devices, and connecting peripherals like keyboards and mice. It is designed for relatively higher data transfer rates and is ideal for applications that require continuous streaming or bidirectional communication.

In the context of the ESP32, Classic Bluetooth mode allows the microcontroller to act as a Bluetooth Classic device, enabling it to establish connections and communicate with other Classic Bluetooth devices. This mode is suitable for scenarios where compatibility with existing Bluetooth devices or protocols is required, such as connecting the cocktail pouring machine to a smartphone for manual control or pairing with a Bluetooth speaker for audio feedback.

Bluetooth Low Energy (BLE): BLE is a power-efficient version of Bluetooth designed for applications that require low power consumption and intermittent communication. It is commonly used in applications like fitness trackers, smartwatches, and IoT devices, where battery life is a critical factor. BLE devices

are characterized by their ability to operate in a low-power sleep state and establish quick connections for short bursts of data transmission.

Both Bluetooth modes, Classic Bluetooth and Bluetooth Low Energy, provide distinct advantages and serve different purposes. The choice of which mode to use depends on the specific requirements of your project, such as power consumption.

Given that our cocktail bartender is a stationary device connected to a reliable power source, energy and power constraints are not significant concerns for us. This enables us to have more flexibility in power usage without worry. Consequently, we have opted to utilize the Classic Bluetooth mode to establish a connection between the bartender and our web application. This choice aligns with our needs and facilitates seamless communication between the two devices.

Our ESP32 microcontroller has been programmed with the necessary functionality to efficiently control the liquid flow and enable seamless Bluetooth communication.

In the setup phase, we start up the bluetooth module and configure the relay pin output modes:

```
#define RELAY1 32
#define RELAY2 33
#define RELAY3 25
#define RELAY4 26
#define RELAY5 27
#define RELAY6 14
#define RELAY7 12

BluetoothSerial SerialBT;
char bt_data[1024];

void setup() {
  Serial.begin(9600);
  SerialBT.begin("Cocktail Machine");

  pinMode(RELAY1, OUTPUT);
  pinMode(RELAY2, OUTPUT);
  pinMode(RELAY3, OUTPUT);
  pinMode(RELAY4, OUTPUT);
  pinMode(RELAY5, OUTPUT);
  pinMode(RELAY6, OUTPUT);
  pinMode(RELAY7, OUTPUT);
}
```

After bluetooth is set up, we expect a connection from a mobile device and consequent requests to pour certain cocktails. To accept the requests we constantly await for any information relayed on the bluetooth communication line:

```
void loop() {
  char *ptr_bt_data = bt_data;
  if(SerialBT.available() > 0) {
    while(SerialBT.available() > 0) {
      *ptr_bt_data = SerialBT.read();
      ptr_bt_data++;
    }
    *ptr_bt_data = '\0';
```

After we receive a request from the mobile device we write that information into a temporary buffer and try to parse it into a JSON object. The format of the request/message that we expect from the mobile application is the following:

A JSON array containing a list of objects that hold information about which relay pin to activate and the duration of the activation, for example a request message to activate three different relays would look like this:

```
[
  {"pin": 32,"timeMs":3000},
  {"pin": 33,"timeMs":2500},
  {"pin": 25,"timeMs":2000}
]
```

In order to read the JSON information inside the ESP32 we use ArduinoJson. ArduinoJson is a powerful and versatile JSON library specifically designed for embedded systems and microcontrollers, such as the ESP32. It provides a comprehensive set of tools and functions to parse, generate, and manipulate JSON data efficiently.

One of the key features of ArduinoJson is its lightweight and efficient design, making it highly suitable for resource-constrained environments. It is optimized to consume minimal memory and processing power, allowing it to operate seamlessly on microcontrollers with limited resources.

The library supports both parsing and generating JSON data. When parsing, ArduinoJson can extract values from JSON strings and convert them into appropriate data types, such as integers, floats, strings, or boolean values. This enables easy access and extraction of specific information from complex JSON structures.

After receiving the JSON request from the mobile application, we parse this information using ArduinoJson:

```
StaticJsonDocument<600> doc;
DeserializationError error = deserializeJson(doc, bt_data);

// Test if parsing succeeds.
if (error) {
  Serial.print(F("deserializeJson() failed: "));
  Serial.println(error.f_str());
  return;
}
```

In the event of parsing errors, it indicates a communication issue where the mobile application failed to send a correct request. In such cases, the request is promptly aborted to avoid further processing. Conversely, if parsing is successful, the process proceeds to the next step, involving the dispensing of liquids, cooling them, and finally pouring out the requested cocktail mixture.

```
    for(int i = 0; i<doc.size(); i++){
      JsonObject obj = doc[i];
      const int pin = obj["pin"];
      const int time = obj["timeMs"];
      digitalWrite(RELAY7, HIGH);
        if(pin == RELAY1 || pin == RELAY2
        || pin == RELAY3 || pin == RELAY4
        || pin == RELAY5 || pin == RELAY6
        || pin == RELAY7) {
          digitalWrite(pin, HIGH);
          delay(time);
          digitalWrite(pin, LOW);
        }

        delay(1000);

    }

    delay(6000);

    // everything done
    digitalWrite(RELAY7, LOW);

}
```

To initiate the dispensing process, we iterate through each of the received pins and activate the corresponding pumps sequentially. As the pumps engage, the liquids flow through the funnel, converging into the PELTIER water cooler. Inside the cooler, the PELTIER pump effectively transfers all the liquid from the cooler to the final cup. It is worth noting that, as evident in the source code, the PELTIER pump operates continuously throughout the entire process, ensuring a consistent and uninterrupted flow of liquids across the system.

As a result, we obtain a well-crafted cocktail mixture that is appropriately cooled and ready to be savored by the user.

PELTIER

The Peltier water cooler plays a crucial role in our cocktail bartender automated device project by ensuring that the liquids used in the cocktails are kept cool and refreshing. Its primary function is to maintain the optimal temperature of the ingredients, enhancing the taste and quality of the beverages produced by the Bartender.

A Peltier water cooler, also known as a thermoelectric cooler or TEC, uses the Peltier effect to transfer heat between two surfaces. The Peltier effect is named after Jean Charles Athanase Peltier, who discovered it in the 19th century. It is based on the principle that an electric current flowing through two dissimilar materials can cause a temperature difference at their junction.

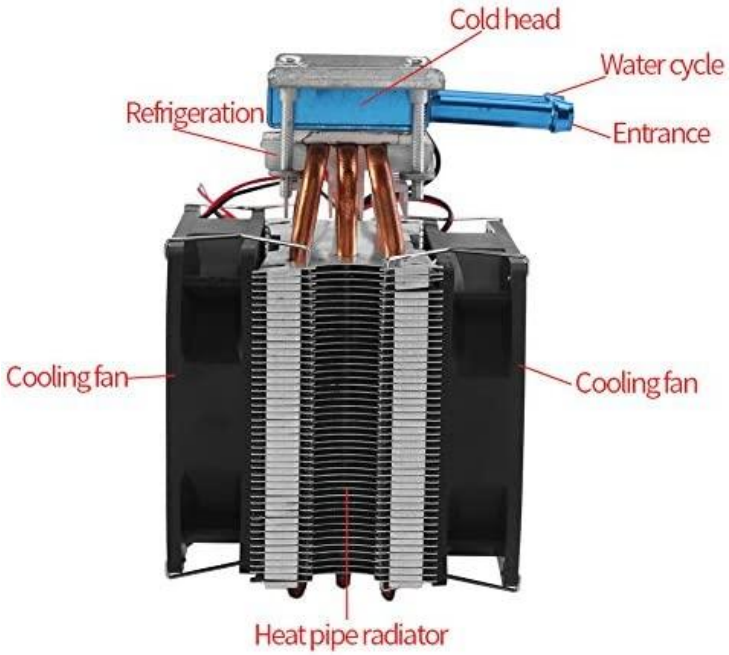
The Peltier device consists of two ceramic plates, usually made of bismuth telluride, with a series of semiconductor junctions between them. These junctions are formed by alternating p-type and n-type semiconductor materials. P-type semiconductors have an excess of positive charge carriers (holes), while n-type semiconductors have an excess of negative charge carriers (electrons).

When a DC electric current is applied to the Peltier device, it causes electrons to flow from the n-type semiconductor to the p-type semiconductor. This flow of electrons absorbs heat at the cold junction (where the heat needs to be removed) and releases heat at the hot junction (where the heat needs to be dissipated). This phenomenon is known as the Peltier effect.

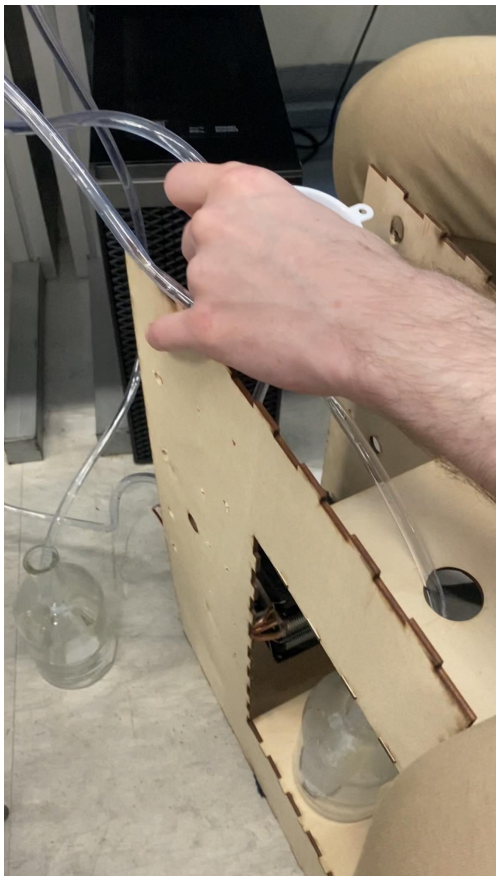
In a Peltier water cooler, one side of the Peltier device is in contact with the water or the object that needs to be cooled, and the other side is connected to a heat sink or a fan for heat dissipation. As the electric current flows through the Peltier device, heat is absorbed from the water, lowering its temperature. The absorbed heat is then released into the surrounding environment through the heatsink or fan.

In our implementation, we have carefully selected a high-quality Peltier water cooler with specific specifications to ensure efficient and effective cooling. The cooling system operates with a voltage of 12V and a current of 9A, while the fans require 12V and 0.2A.

As already mentioned our Peltier has two fans that play a crucial role in the cooling process. The fans help to circulate the air around the thermoelectric modules, which enhances the cooling effect by dissipating heat more efficiently. Without these fans, the cooling system would not be able to operate at optimal efficiency, and the temperature of the drinks would not be cooled down as effectively. The cooler features a robust construction with optimized heat dissipation capabilities to prevent overheating and maintain a consistent cooling performance. It incorporates advanced thermoelectric modules and a reliable power supply system to ensure reliable operation throughout the cocktail-making process.



Within our cocktail bartender automated device, the Peltier water cooler is strategically integrated into the system. It is always turned on to make sure that one side is always cold enough and cooling fans are always working as well. Whenever the user selects the desired cocktail, corresponding ingredients are poured in the funnel which is directly connected to the Peltier through a pipe. Eventually liquid passes through the cooler and pours into the cup.



In summary, the Peltier water cooler in our cocktail bartender automated device project plays a crucial role in maintaining the ideal temperature of the liquids used in cocktail preparation. With its implementation, we ensure that the drinks produced by Bartender are consistently cool, fresh, and enjoyable.

Water Pumps

Water pumps play a crucial role as the cornerstone physical component in our project, which revolves around the efficient transfer of liquid from multiple bottles into a single cup. Our meticulous consideration led us to select a 12v DC pump as the optimal choice, primarily due to its exceptional stability and formidable power output. This particular pump not only surpassed our expectations but also exceeded the necessary requirements for effectively transferring liquids through tubes.

The selection of a 12v DC pump offers several advantageous features that contribute to the success of our project. Firstly, its direct current (DC) operation ensures a consistent and reliable flow of power, eliminating the concerns associated with fluctuations that might compromise the liquid transfer process. This stability is paramount in maintaining the integrity of the overall system, guaranteeing a seamless and uninterrupted flow of liquid from the bottles to the cup.

Moreover, the 12v DC pump's commendable power capabilities surpass the minimum threshold needed for fluid transfer. By providing an ample amount of power, it enables the efficient movement of liquids through the interconnected tubing network. This ensures a swift and reliable transfer, minimizing any delays or inefficiencies that might arise from insufficient power supply.

The versatility of a 12v DC pump also allows for easy integration into our project setup. Its compatibility with a wide range of power sources and control mechanisms facilitates seamless integration into our existing framework. Furthermore, the compact size and lightweight nature of the pump enhance its practicality, enabling convenient placement within the system without consuming excessive space or hindering the overall functionality.

Liquid Transferring: The pumps we have carefully selected operate at a voltage of 12V and draw a current of 0.7A, resulting in a power consumption of 6W. Among the primary concerns we had regarding the pump's performance was its speed in transferring liquid through the tubes. To address this, we conducted a series of meticulous tests to accurately measure its liquid transfer rate. Through rigorous experimentation, we obtained a precise answer, revealing an approximate rate of 30ml/s.

This measured rate of 30ml/s proved to be quite satisfactory for our project, considering that many cocktails typically require a volume of around 100-120ml. With this transfer rate, it would take a mere 3-4 seconds to efficiently pour the desired amount of liquid into the cup. This impressive speed ensures a swift and seamless dispensing process, minimizing any potential delays and ensuring a prompt delivery of the desired quantity of liquid.

In our project, as you already know, we utilize the ESP-32 microcontroller, which operates at 5V and provides a current of 0.1A. However, this current is insufficient for the proper functioning of the pumps, as they require at least 0.5A. To address this issue, we made the decision to acquire a power supply that could cater to both the ESP and the pumps, with a maximum voltage of 12V and a current of 5A.

Since we have a total of seven pumps in our project, each pump requiring a current of 0.5A, the total current required amounts to 3.5A. Fortunately, this falls within the maximum amperage provided by the power supply, ensuring that all the pumps can operate optimally without any power-related concerns.

Another challenge we encountered was the unpredictable nature of the pumps, as they tend to generate a significant amount of electric noise. This noise has the potential to interfere with the ESP and potentially cause long-term damage. To mitigate this risk, we successfully separated the voltage used by the ESP and the water pumps using the power supply. This separation ensured that the ESP remained safe and secure from any electrical noise generated by the pumps.

To enable the ESP to control and activate the pumps when necessary, we incorporated an ESP-32 module with built-in relays. These relays provided a safe and reliable means for the ESP to control the pumps, while effectively isolating it from the electrical noise generated by the pumps.

Design

Now, let's discuss the design and placement of the pumps within our cocktail machine "box." The box features six holes designed for the external pipes, which are inserted into the desired bottles by the user. Inside the box, the pumps are positioned in parallel with the holes, allowing the pipes to pass through them and connect at one end. With this setup, we have a total of six pumps and six external pipes, facilitating the smooth transfer of liquids into the Peltier cooler.

Additionally, we have an extra pump dedicated to extracting the liquid from the Peltier cooler and dispensing it outside into the provided cup, completing the fluid circulation within our cocktail machine.

By addressing the power requirements, electrical noise concerns, and implementing an effective design for pump placement, we have created a reliable and efficient system that enables the seamless transfer and circulation of liquids in our cocktail machine.

Power Supply

A power supply is an electronic device that converts electrical power from a source, such as a wall outlet or battery, into a form of power suitable for use by electronic devices. The power supply provides a constant and regulated output voltage and current that meets the requirements of the device it is supplying power to.

Our power supply is providing reliable and regulated electrical power to the cocktail Machine. With a DC voltage of 12.00 V, DC amperage of 16.50 A, and a power output of 200.00 W, this power supply is specifically chosen to meet our project's requirements. We chose this high amperage power supply for the Peltier water cooler especially. Among all of the components, Peltier uses highest power (12*9 = 108 watts).

This power supply that we bought has some very good specifications:

AC input range selectable by switch:

The power supply incorporates an AC input range that can be conveniently selected using a switch. This flexibility allows the device to accommodate different voltage standards, making it suitable for a wide range of power sources, ensuring compatibility in various regions or locations.

Protections: **Short circuit/Overload/Over temperature**:

To ensure the safety and longevity of the power supply and the entire system, it is equipped with comprehensive protection mechanisms. These include safeguards against short circuits, overloads, and over-temperature conditions. In the event of any abnormality or excessive load, the power supply automatically detects and mitigates potential risks, preventing damage to the components and ensuring the device's smooth operation.

Cooling by free air convection:

Efficient cooling is essential to maintain the power supply's optimal performance and prevent overheating. The power supply utilizes a free air convection cooling method, which leverages the natural flow of air to dissipate heat. This approach eliminates the need for additional cooling mechanisms such as fans, reducing noise levels and improving the overall reliability of the system.

100% full load burn-in test:

To guarantee the power supply's reliability and performance under real-world operating conditions, it undergoes a rigorous 100% full load burn-in test. This test subjects the power supply to maximum load and ensures that it can withstand prolonged operation, identifying any potential issues or weaknesses before it is integrated into the cocktail bartender automated device.

Compact size, lightweight:

Considering the space constraints and portability requirements of our project, the power supply is designed with a compact form factor and lightweight construction. This compact size allows for easy integration within the limited space available in the cocktail bartender device, while its lightweight nature ensures the overall weight of the device remains manageable and portable.

Fixed switching frequency at 25 kHz:

The power supply operates at a fixed switching frequency of 25 kHz. This stable and consistent frequency allows for efficient power conversion while minimizing

electromagnetic interference and ensuring reliable operation. The fixed frequency also simplifies the design and optimization of other components in the system, contributing to overall efficiency and performance.

Overally, The power supply drives various subsystems in the cocktail machine, such as pumps, relays, and PELTIER cooling systems. Its stable and regulated power output ensures precise liquid dispensing, consistent mixing, and accurate control of functions, enhancing the overall quality of the cocktails. Such results would be much harder to achieve if we were working with an AC power source.

Mobile App

Our mobile application serves the purpose of providing users with a convenient and intuitive interface to interact with a cocktail machine. Through the application, users can remotely send commands to the cocktail machine, specifying the drinks and quantities for their cocktails. To establish seamless communication between the mobile application and the cocktail machine, we have integrated the ESP32 Bluetooth module into our solution.

The ESP32 Bluetooth module plays a crucial role in bridging the connection between the mobile application and the cocktail machine. It enables wireless communication, allowing real-time transmission of user commands from the mobile application to the cocktail machine. This integration ensures accurate and prompt execution of the desired drink combinations, enhancing the overall user experience.

In addition to the ESP32 Bluetooth module, we have leveraged the power of React Native to develop our mobile application. React Native offers a practical framework for building cross-platform applications using JavaScript. By utilizing

React Native, we can write code once and deploy it on multiple platforms, such as iOS and Android, reducing development time and effort.

The inherent benefits of React Native, including its ability to deliver a native-like user experience and its simplified development process, have contributed to the seamless and intuitive interface of our mobile application so users can enjoy a smooth and visually appealing experience. By choosing React Native for our mobile application, we ensure its compatibility with future updates and changes in the mobile ecosystem.

Application overview

Development Environment and Tools: To develop the application, we used the following tools and technologies:

 • **React Native**: A JavaScript framework for building user interfaces.

 • **Node.js**: A JavaScript runtime environment for executing JavaScript code outside of a web browser.

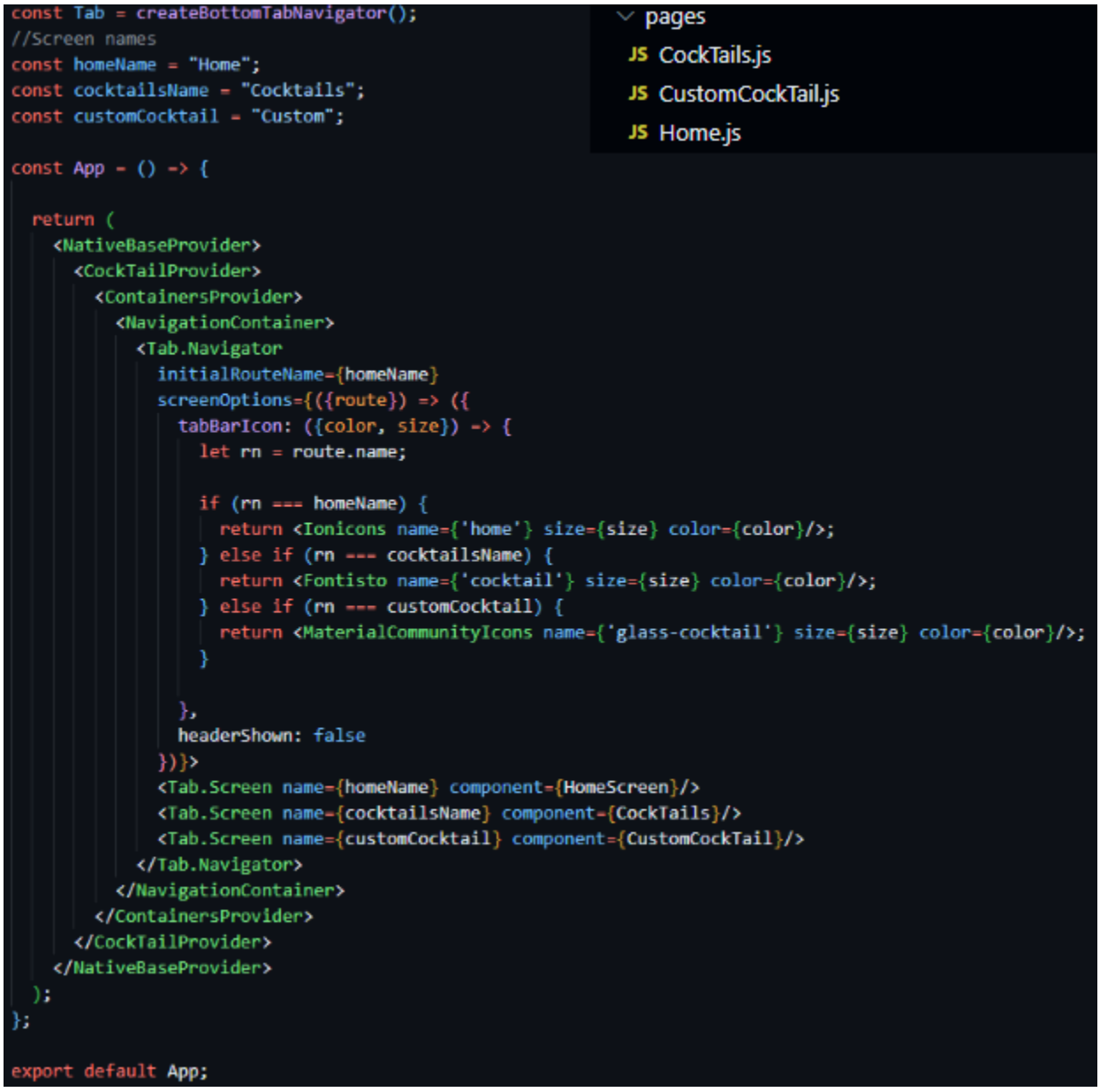
 • **WebStorm**: A powerful and intelligent integrated development environment (IDE) specifically designed for JavaScript development.

 • **Expo**: A set of tools and services for building and deploying React Native applications.

Architecture and Components: Our application follows a modular architecture, utilizing various components and libraries to achieve the desired functionality. The main components of our application include:

• **Screens and Navigation**: React Native screens that represent different views of the application, such as home, cocktails, and custom cocktail screens. We used React Navigation, a popular routing and navigation library, to handle navigation between screens and create a seamless user experience.

Screens and Navigation code snippet:



```
const Tab = createBottomTabNavigator();
//Screen names
const homeName = "Home";
const cocktailsName = "Cocktails";
const customCocktail = "Custom";

const App = () => {

  return (
    <NativeBaseProvider>
      <CockTailProvider>
        <ContainersProvider>
          <NavigationContainer>
            <Tab.Navigator
              initialRouteName={homeName}
              screenOptions={({route}) => ({
                tabBarIcon: ({color, size}) => {
                  let rn = route.name;

                  if (rn === homeName) {
                    return <Ionicons name={'home'} size={size} color={color}/>;
                  } else if (rn === cocktailsName) {
                    return <Fontisto name={'cocktail'} size={size} color={color}/>;
                  } else if (rn === customCocktail) {
                    return <MaterialCommunityIcons name={'glass-cocktail'} size={size} color={color}/>;
                  }

                },
                headerShown: false
              })}>
              <Tab.Screen name={homeName} component={HomeScreen}/>
              <Tab.Screen name={cocktailsName} component={CockTails}/>
              <Tab.Screen name={customCocktail} component={CustomCockTail}/>
            </Tab.Navigator>
          </NavigationContainer>
        </ContainersProvider>
      </CockTailProvider>
    </NativeBaseProvider>
  );
};

export default App;
```

pages
JS CockTails.js
JS CustomCockTail.js
JS Home.js

• **useContext and AsyncStorage**: We utilize the useContext hook to create a context called CockTailContext. This context provides access to the cocktails state variable, as well as functions to add, edit, and delete cocktails. By using this context, different components can access and update the cocktails' data without prop drilling.

To persist the cocktail data, we utilize AsyncStorage from the **@react-native-async-storage/async-storage** library. When a cocktail is added, edited, or deleted, the corresponding function updates the cocktails state and then stores the updated data in AsyncStorage. This ensures that the cocktail data is retained even if the application is closed or restarted.

useContext and AsyncStorage code snippet:

components > contexts > JS CockTailContext.js > ...

```
1   import React, {createContext, useEffect, useState} from 'react';
2   import AsyncStorage from '@react-native-async-storage/async-storage';
3   import {defaultCocktails} from "../utils/utility";
4
5   export const CockTailContext = createContext({ ···
13  });
14
15  export const CockTailProvider = ({children}) => {
16    const [cocktails, setCocktails] = useState(defaultCocktails);
17
18    const addCocktail = (newCocktail) => { ···
26    };
27
28    const editCocktail = (index, updatedCocktail) => { ···
36    };
37
38    const deleteCocktail = (index) => { ···
46    };
47
48    useEffect(() => {
49      // Get data from AsyncStorage and update the state
50      AsyncStorage.getItem('cocktails').then(data => {
51        if (data) {
52          setCocktails(JSON.parse(data));
53        }
54      }).catch(error => {
55        console.error('Error retrieving data from AsyncStorage:', error);
56      });
57    }, []);
58
59
60    return (
61      <CockTailContext.Provider value={{
62        cocktails: cocktails,
63        editCocktail: editCocktail,
64        addCocktail: addCocktail,
65        deleteCocktail: deleteCocktail
66      }}>
67        {children}
68      </CockTailContext.Provider>
69    );
70  };
```

components
contexts
JS CockTailContext.js
JS ContainersContext.js

Getting cocktails in component:

```
const {cocktails, deleteCocktail} = useContext(CockTailContext)
```

• **Bluetooth integration and Request**: In our React Native application, we integrate the Bluetooth module from the "react-native-bluetooth-classic" library. This integration allows us to seamlessly communicate with external devices, such as our cocktail pouring machine. By leveraging the Bluetooth module's functions, we establish connections, exchange data, and perform operations with the hardware components.

```
async function requestBluetoothPermissions(): Promise<boolean> {
  const result = await PermissionsAndroid.requestMultiple([
    PermissionsAndroid.PERMISSIONS.BLUETOOTH_CONNECT
  ]);

  return Object.values(result).every((val) => val == "granted")
}

function App(): JSX.Element {
  const isDarkMode = useColorScheme() === 'dark';

  const backgroundStyle = {
    backgroundColor: isDarkMode ? Colors.darker : Colors.lighter,
  };

  useEffect(() => {
    (async () => {
      if(! await Bluetooth.isBluetoothEnabled()) {
        await Bluetooth.requestBluetoothEnabled();
      }

      if(! await requestBluetoothPermissions()) {
        console.debug("FAILED TO AQCUIRE REQUIRED PERMISSIONS");
        return;
      }

      const paired = await Bluetooth.getBondedDevices();
      console.log(paired);

      const device = paired.find(device => {
        console.log(device.name);
        return device.name === "Cocktail Machine"
      })

      if(device === undefined) {
        console.debug("FAILED TO IDENTIFY THE COCKTAIL MACHINE IN PAIRED DEVICES");
        return;
      }
```

Through this Bluetooth integration, we enable real-time interaction and control over the cocktail pouring process directly from the mobile application. We utilize a request mechanism via Bluetooth, where we send requests to the cocktail pouring machine using the Bluetooth module's functions. These requests initiate communication, allowing us to retrieve relevant information, send instructions, and receive responses from the machine.

Request example:

```
await device.write(JSON.stringify([{"pin": 32,"timeMs":2000}, {"pin":
33,"timeMs":2000}, {"pin": 25,"timeMs":2000}, {"pin": 26,"timeMs":2000},
{"pin": 27,"timeMs":2000}, {"pin": 14,"timeMs":2000}, {"pin":
12,"timeMs":2000}]));
```

• **Packages and UI Components**: We utilized various UI component libraries, such as React Native Elements or Material-UI, to create visually appealing and consistent user interfaces.

```
"dependencies": {
  "@expo/webpack-config": "^18.0.1",
  "@react-native-async-storage/async-storage": "1.17.11",
  "@react-navigation/bottom-tabs": "^6.5.7",
  "@react-navigation/native": "^6.1.6",
  "@react-navigation/native-stack": "^6.9.12",
  "@react-navigation/stack": "^6.3.16",
  "@rneui/base": "^4.0.0-rc.7",
  "@rneui/themed": "^4.0.0-rc.7",
  "expo": "^48.0.9",
  "expo-status-bar": "~1.4.4",
  "native-base": "^3.4.28",
  "react": "18.2.0",
  "react-dom": "18.2.0",
  "react-native": "0.71.7",
  "react-native-safe-area-context": "4.5.0",
  "react-native-screens": "~3.20.0",
  "react-native-svg": "13.4.0",
  "react-native-svg-transformer": "^1.0.0",
  "react-native-vector-icons": "^9.2.0"
},
"devDependencies": {
  "@babel/core": "^7.21.3"
},
```

Challenges and Considerations: While developing with React Native, we encountered a few challenges specific to this framework. Some of them include:

• **Performance**: React Native relies on a bridge to communicate between JavaScript and native code, which can introduce some performance overhead. To mitigate this, we optimized our code by minimizing unnecessary renders and using native modules for computationally intensive tasks.

• **Platform-Specific Differences**: Although React Native provides a unified development experience, there are still some platform-specific differences that need to be considered. We carefully handled these differences by utilizing conditional rendering and platform-specific code blocks.

• **Third-Party Library Compatibility**: React Native has a vast ecosystem of third-party libraries, but not all of them are compatible with both iOS and Android. We made sure to choose libraries that have good cross-platform support or implemented platform-specific alternatives when necessary.

User interface design

When designing the user interface (UI) for our mobile application, we followed specific design principles and considerations to create an appealing and user-friendly experience. These principles include simplicity, consistency, accessibility, responsive layout, and visual hierarchy.

We prioritized a clean and intuitive UI design, ensuring that the application's interface is straightforward and easy to navigate for users. Maintaining visual consistency throughout the application, including typography, color schemes, and visual elements, creates a cohesive and familiar experience.

To ensure accessibility, we closely followed guidelines for factors such as color contrast and font sizes. We also made accommodation for users with different abilities, aiming to provide an inclusive experience.

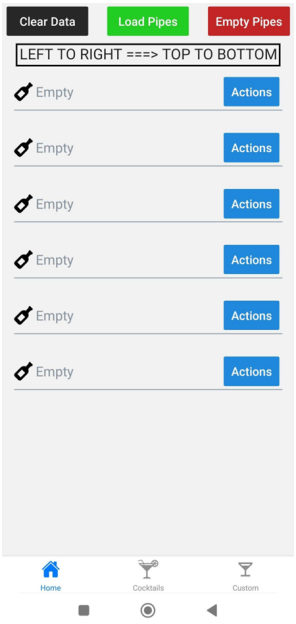
Our UI design is responsive, seamlessly adapting to various screen sizes and orientations. This guarantees optimal usability on different devices, enhancing the overall user experience.

Employing visual hierarchy techniques, we prioritized important information, guided users' attention, and made the UI more scannable and digestible. By utilizing these design principles, we created an interface that is visually appealing, user-friendly, and optimizes user engagement.

Functionality and Features:

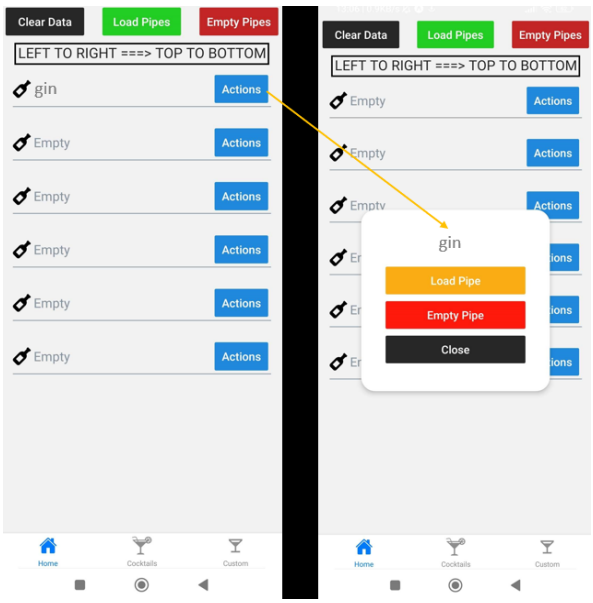
Our mobile application, designed for a cocktail machine, provides users with convenient control over the pouring process. The application offers the following core functionality and features:

Home Page:



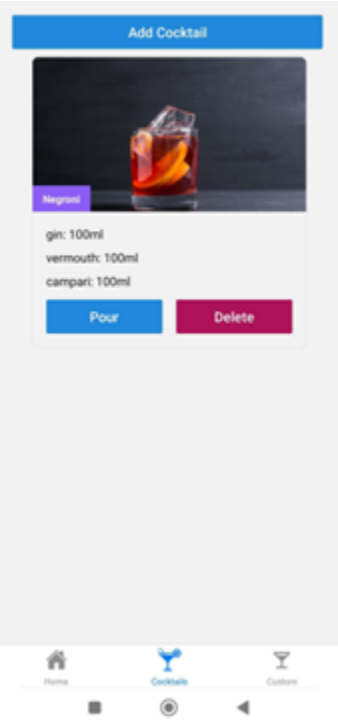
• **Drink Configuration Display**: The Home page displays information about the assignment of each drink to a specific pipe. Users can easily identify which drink is assigned to each pipe and make changes accordingly.

• **Drink Assignment Modification**: Users have the flexibility to modify the drink assignments based on their preferences or the available drink options. They can easily update the configuration by selecting a different drink for a specific pipe.



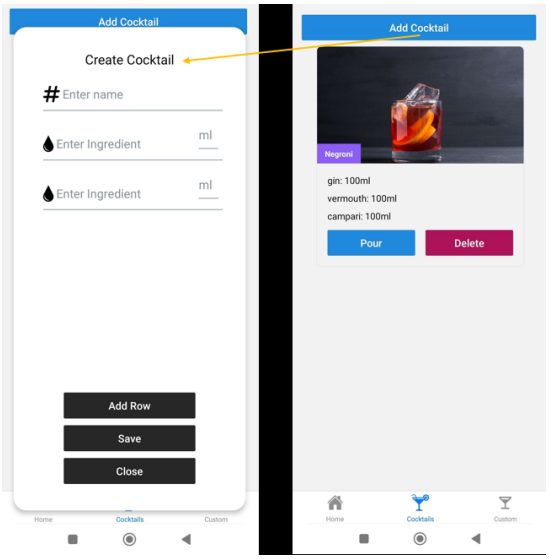
• **Load Pipe and Empty Pipe Actions**: To facilitate the management of the cocktail machine, the Home page includes action buttons for each pipe. These buttons provide users with the option to load a pipe with a specific drink or empty the contents of a pipe. This functionality allows users to easily control the drink inventory and make adjustments as needed.

Cocktails Page:



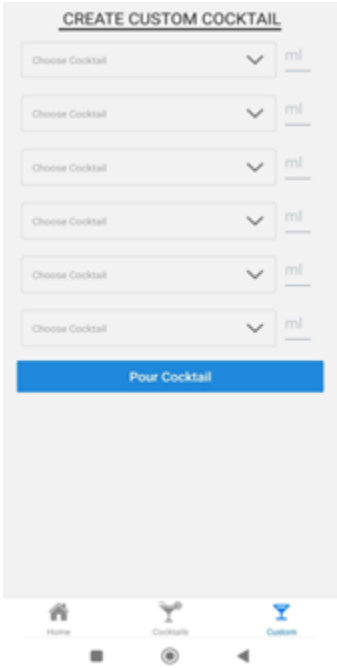
• **Predefined Cocktails Display**: The Cocktails page displays a list of predefined cocktails. Each cocktail is presented with its name and other relevant information.

• **Automated Drink Pouring**: When a user selects a predefined cocktail from the list, the cocktail machine automatically pours the corresponding drinks assigned to the pipes based on the drink configuration set on the Home page. This seamless integration ensures accurate and convenient cocktail preparation.



• **Add Cocktail Option**: The Cocktails page features an "Add Cocktail" button, which opens a modal or form for users to create their own custom cocktails. Within the modal, users can specify the desired combination of drinks and save the custom cocktail for future use.

Custom Cocktail Page:



• **Drink Selection**: Users can select the drinks for their custom cocktail from the available options. The available drink options are based on the drink configuration set on the Home page, where users assign drinks to each pipe of the cocktail machine.

• **Pouring Quantities**: Once the drinks are selected, users can specify the pouring quantities for each drink. This feature

allows users to customize the proportions of different drinks in their custom cocktail, providing flexibility and control over the final result.

These features provide users with control over the drink configuration of the cocktail machine, allowing them to customize their drink combinations and conveniently pour the desired cocktails.

Overall, the design and development of our mobile application have provided users with a convenient and customizable way to control their cocktail machine. We have overcome challenges, gained valuable insights, and achieved a functional and user-friendly application. With potential future enhancements, the application can continue to evolve and cater to the needs and preferences of cocktail enthusiasts.

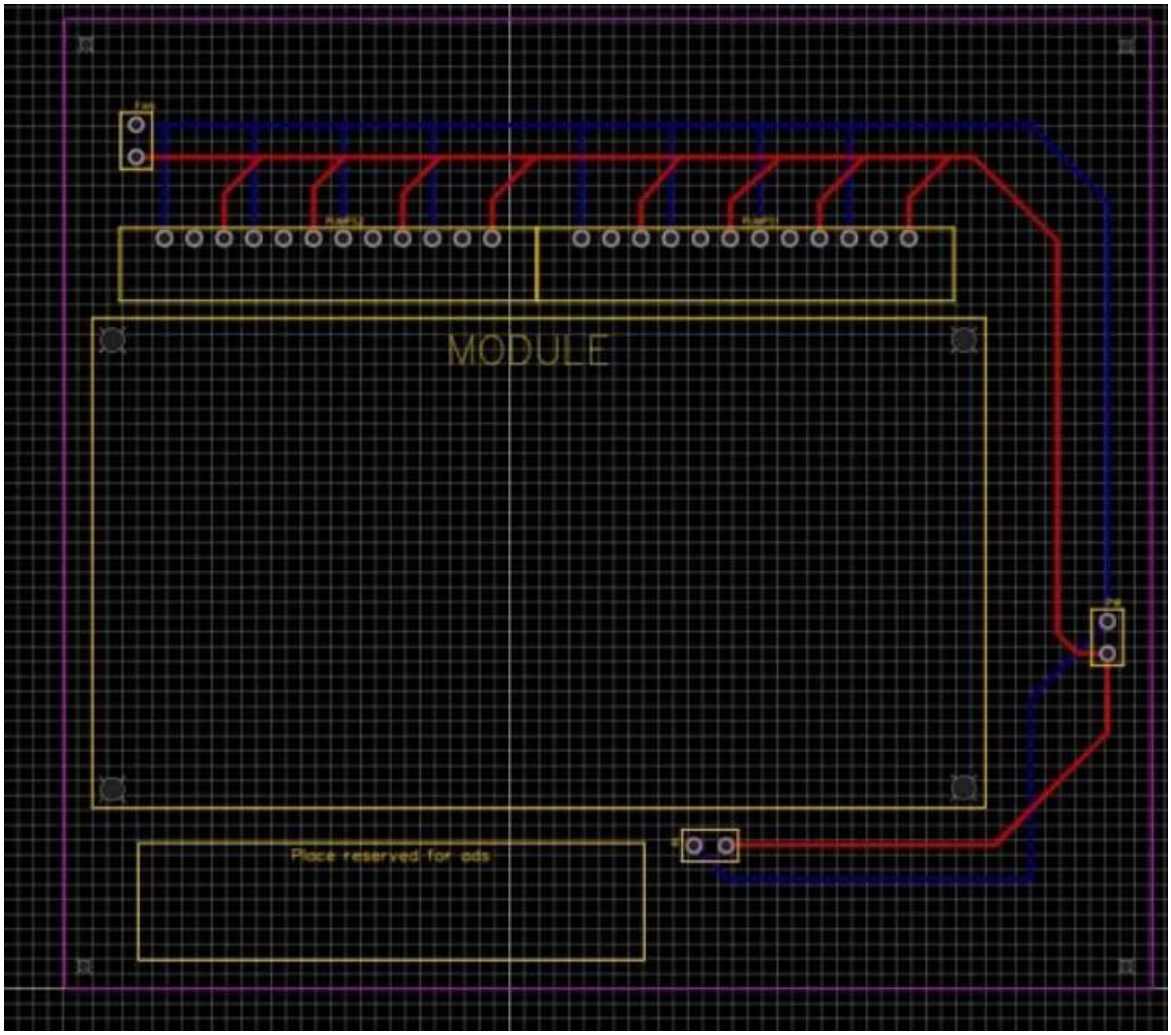
PCB

The PCB board is crucial for our cocktail machine to work. A PCB, or Printed Circuit Board, is a vital component that provides a platform for mounting and connecting electronic parts. It consists of a non-conductive material with a layer of copper foil. The copper is etched to create pathways called traces, which enable the flow of electrical signals between components.

Almost every component is connected to each other with the help of PCB. In our case, the board has only a bottom layer as there was no need for more than one layer. Traces are wider (0.762mm) than standard (0.254mm), however, it didn't have much of an impact. In terms of size, our PCB is 185.3 X 140.2 mm^2, which is larger than you would expect for PCB. The reason for that is the relay module, which should be placed right on top of the PCB board and which itself is 148 X 79.8 mm^2 in size. There are 4 holes in the center of the board for the relay module to be placed and 4 additional holes for the PCB board itself. We made holes in the center slightly wider to have more freedom with relay module placement in case calculations were wrong. Gladly, the relay module was placed on PCB just as it was supposed to be.

To understand how the cocktail machine works, it is important to know the purpose of each element, pin, or terminal and the relationship between them. The placement of different terminals was done in a way that would be useful for the overall design of the cocktail machine. PW terminals (on the bottom right side of the PCB) are connected to the power source. They are entry points for 12V DC voltage to the entire PCB. IN terminals (in the bottom half of the PCB) are connected to the relay module, they take 12 Volts from PW terminals and power the entire relay module. Usually 12 Volts would be too much for ESP32 that's inside the module, however, the relay module itself has a terminal specifically for 7-30V current. That is because there is a device called a "buck converter" inside the module, which brings the applied voltage down to 5 Volts. A buck converter is a type of DC-DC converter that steps down a higher DC voltage to a lower DC voltage. It works based on the principle of pulse width modulation (PWM). There is another buck converter to bring 5 Volts down to 3.3V. 5V electricity is needed for ESP32 to operate, which sends 3.3V signals to the relays accordingly.

Now, the way these relays work is this: each relay is connected to 3 terminals (NO, COM, and NC). Depending on the state of the relay, either NC or NO terminal is connected to COM. We decided to use NO terminals for powering pumps. There are 24 PUMP terminals (in the upper half of the PCB) on our board, 8 of them are used to provide 12V DC voltage to NO terminals. Once ESP32 sends a 3.3V signal to the respective relay, the NO terminal becomes short connected to the COM terminal, and the respective pump is activated. Another 8 PUMP terminals are used as ground pumps. The last 8 PUMP terminals are not used at all. PCB could have just as easily had 16 PUMP terminals, but it was much easier and practical to use 24 terminals instead.



The last Terminals on the PCB are FAN terminals (in the top left part of the PCB). They are used to provide power to PELTIER fans with 12V DC voltage. The only connections that do not happen through PCB are the connection between ESP32 and relays and the connection between the power supply and PELTIER. The first connection is already hardwired in the relay module, so there was no need for PCB to connect ESP32 outputs to the relay. The second connection was too risky for PCB. PELTIER requires 9A and 12V DC voltage to operate at its best. That is equal to 108 watts of power, and it was too risky to connect a device that consumes such an amount of power to the source via PCB trace. Not that it was impossible, it was certainly possible, but we decided to play it safe and just connect the power supply to the PELTIER directly.

Conclusion

In conclusion, our senior project, the cocktail machine, successfully integrated various components to create a fully functional and innovative system. The key elements of our project included the utilization of the ESP-32 microcontroller, the incorporation of seven pumps, and the implementation of Peltier cooling technology. We developed a mobile application that served as a user-friendly interface, providing control over the entire system, including the ability to create custom cocktails and dispense pre-existing recipes.

By leveraging the capabilities of the ESP-32 microcontroller, we achieved efficient control and coordination of the cocktail machine's components. The ESP-32 not only facilitated seamless communication between the pumps and the mobile application using bluetooth but also ensured the reliable and precise activation of the pumps as required. With the ESP-32's ability to handle multiple tasks simultaneously, we were able to create a responsive and dynamic user experience.

The integration of seven pumps into our system allowed for the simultaneous handling of various ingredients and liquids. This capability, combined with the power and stability provided by the chosen 12V DC pumps, ensured a swift and accurate transfer of liquids from multiple bottles into a single cup. The measured transfer rate of approximately 30ml/s provided an efficient and timely dispensing process, meeting the requirements for a wide range of cocktails.

To maintain the cool temperature for the cocktails, we incorporated Peltier cooling technology into our project. This cooling system effectively regulated the temperature within the cocktail machine, ensuring that the beverages were served at the desired chilled state. By efficiently transferring heat away from the liquids, the Peltier cooling system enhanced the overall quality of the cocktails.

The development of a mobile application served as a user-friendly interface, allowing individuals to control the cocktail machine with ease. The app provided a range of functionalities, including the creation of custom cocktails by selecting specific ingredients and their corresponding quantities. Additionally, users had the

convenience of accessing and pouring pre-existing cocktail recipes from a comprehensive database. The mobile app's intuitive design and seamless connectivity with the ESP-32 microcontroller made the entire cocktail-making process straightforward and enjoyable.

Overall, our senior project successfully achieved its objectives by integrating the ESP-32 microcontroller, implementing seven pumps, incorporating Peltier cooling, and developing a user-friendly mobile application. The combination of these components resulted in a sophisticated cocktail machine capable of creating custom cocktails and dispensing pre-existing recipes. Through this project, we demonstrated our ability to leverage technology and creativity to enhance the beverage-making experience, providing convenience and enjoyment to users.

Team Member Contributions

Andro Mazmishvili

As the team leader, my primary responsibility was to oversee and coordinate the entire project. I played a pivotal role in ensuring effective communication among team members, organizing regular team meetings, and facilitating collaboration. Additionally, I took charge of delegating tasks, tracking progress, and ensuring that the project remained on schedule.

In addition to my managerial role, I actively contributed to the technical aspects of the project. Specifically, my contributions included:

ESP32 Code Development:

Together with Ioane Mania, I collaborated on writing the code for the ESP32 microcontroller, which serves as the brain of our automated cocktail bartender device. We designed and implemented the code to control various functionalities such as ingredient dispensing and communication with external components.

Device Design:

I took the lead in conceptualizing and designing the overall structure and aesthetics of the cocktail bartender device. Drawing upon my knowledge of industrial design principles, I worked closely with the team to create an ergonomic and visually appealing device that seamlessly integrates the necessary components, ensuring user-friendliness and efficient operation.

Job Responsibility Division:

To maximize efficiency and productivity, I played a significant role in dividing job responsibilities among team members based on their skills and interests. I conducted thorough assessments of each member's capabilities and assigned tasks accordingly, considering factors such as programming, electrical systems, mechanical components, and user experience design.

By actively participating in both managerial and technical aspects of the project, I ensured smooth coordination and progress, while also contributing directly to the core development and design processes. Through effective leadership and collaboration, I aimed to foster a cohesive team environment that motivated and empowered each member to excel in their respective roles.

Irakli Chichua

As a team member, my main focus throughout the senior design project was the development of the mobile application. However, I actively participated in every aspect of the project creation, collaborating with my teammates and contributing to various tasks and responsibilities.

In terms of the technical aspects, I played a significant role in designing and implementing the user interface (UI) and user experience (UX) components of the mobile application. Leveraging my knowledge of React Native, I worked closely with the team leader and other members to create an intuitive and visually appealing interface that seamlessly integrated with the cocktail pouring machine. I contributed to the development of key features, such as the home page for drink configuration, the cocktails page with predefined recipes, and the custom cocktail page for user-generated drinks.

Beyond my primary responsibilities in application development, I actively collaborated with other team members in different areas. For instance, I provided input and suggestions during the development of the ESP32 functionality. I worked closely with my teammate responsible for the ESP32 code to ensure seamless communication between the mobile application and the microcontroller. This involved understanding the requirements and constraints of the ESP32, exploring communication protocols, and facilitating data exchange between the bartender and the mobile app.

Furthermore, I actively participated in discussions and decision-making processes related to the procurement of parts and components. By providing insights and

researching suitable suppliers, I contributed to the smooth progress of the project by ensuring the timely acquisition of necessary items.

Overall, my contribution extended beyond the application development aspect of the project. By actively participating in various tasks, collaborating with team members, and offering insights and suggestions, I aimed to foster a cohesive and productive team environment. Together, we worked towards the successful realization of our cocktail pouring machine, integrating the mobile application with the hardware components and ensuring an exceptional user experience.

Ioane Mania

As a valued team member, I held several key responsibilities throughout the project. Primarily, I took charge of developing the ESP32 functionality, which involved programming and configuring the microcontroller to control various aspects of the cocktail pouring machine. This included managing the liquid flow, integrating relay modules, and ensuring smooth communication with other components.

Furthermore, I conducted extensive research on Bluetooth communication, delving into its underlying principles and exploring how it could be effectively implemented within both the ESP32 and the React Native mobile application. This involved understanding Bluetooth protocols, establishing connections, and facilitating seamless data exchange between the bartender and the mobile app.

In addition to these core responsibilities, I also played a crucial role in the procurement process by purchasing the required parts and components for the project. This involved identifying suitable suppliers, comparing prices, and ensuring timely delivery of the necessary items, thereby contributing to the smooth progress of the development phase.

Beyond my primary tasks, I actively contributed to the overall development of the project. This involved collaborating with team members, sharing ideas and insights, and participating in discussions and decision-making processes. By

actively engaging in the project's development, I provided valuable input and helped shape the final outcome.

In summary, my responsibilities as a team member encompassed developing the ESP32 functionality, researching and implementing Bluetooth communication, managing procurement tasks, and contributing to the overall development of the project. Through these efforts, I aimed to ensure the successful realization of our cocktail pouring machine project.

As the team leader, my primary responsibility was to oversee and coordinate the entire project. I played a pivotal role in ensuring effective communication among team members, organizing regular team meetings, and facilitating collaboration. Additionally, I took charge of delegating tasks, tracking progress, and ensuring that the project remained on schedule.

In addition to my managerial role, I actively contributed to the technical aspects of the project. Specifically, my contributions included:

Anri Aleksandria

As a valued team member, my primary role was to assist Andro in various tasks related to project organization and planning. This involved purchasing the necessary components, coordinating with team members to assign tasks, and actively participating in the development of the overall design. Furthermore, Andro and I dedicated substantial time to conducting extensive testing of various components, such as pumps, Peltier coolers, and the ESP-32. These experiments took place in the laboratory, allowing us to assess the performance and suitability of each component for our project. Through our collaborative efforts, we ensured that all aspects of the project were meticulously examined and prepared for successful implementation.

Testing and buying components

Every week, Andro and I would purchase and test various components for our project. Our initial focus was on finding the perfect pump that would suit our system. Through extensive testing of multiple pumps, we eventually identified the

ideal one that met our requirements. In addition, I conducted testing at home using different pumps and tubes to determine the optimal way to connect all the components together. This experimentation allowed us to visualize and plan the intricate connections needed for our project's success.

Working on Design

After purchasing the components and conducting tests, the next step was to determine the design. As a team, we collectively brainstormed and established an overall design concept. However, the challenge remained in assembling the six pumps and creating an efficient system for transferring liquid into and out of the cooler. During this phase, I contributed several ideas on how to tackle this issue. One particular challenge we encountered was connecting the pipes that extended from these six pumps into a single point. To address this, I proposed the implementation of a funnel where all six pipes could converge and flow seamlessly. This solution effectively resolved the issue and ensured a cohesive and functional design for the project.

Working on Presentations and website for showcase

In this project, my responsibilities centered around the development of presentations and a website for showcase purposes. This involved a range of activities aimed at creating compelling and informative materials to captivate the audience and effectively communicate the desired message. For the presentations, I conducted thorough research, organized content in a logical and coherent manner, designed visually appealing slides, and crafted persuasive and concise text. Similarly, for the website, I analyzed the target audience's preferences, developed a user-friendly and visually appealing layout, created and organized content, implemented appropriate multimedia elements, and utilized programming languages to develop a functional and responsive website. Throughout the project, I conducted rigorous testing and debugging, incorporated search engine optimization techniques, and revised and refined the materials multiple times to ensure accuracy, coherence, and a professional finish.

Erekle Shatirishvili

As the sole electrical engineer in a team predominantly composed of computer engineers, my main contribution to the project revolved around the design and implementation of the Printed Circuit Board (PCB). My expertise in electrical engineering played a crucial role in ensuring the successful integration of various electronic components and circuits within the cocktail bartender device.

Specifically, my contributions included:

PCB Design:

I took charge of designing the PCB layout, carefully considering the placement and routing of components to optimize functionality, minimize signal interference, and ensure efficient use of space. Drawing upon my knowledge of electrical principles, I designed the PCB to accommodate the ESP32 microcontroller, power supply and other necessary circuitry.

Component Selection and Integration:

Working closely with the team leader and other members, I provided input and recommendations on the selection of electronic components for the PCB. I assessed factors such as performance, compatibility, and availability to choose components that best fit the project requirements. I also collaborated with other team members to seamlessly integrate these components into the PCB design.

Circuit Implementation:

Once the PCB layout was finalized, I proceeded with implementing the circuit design. Using specialized software tools, I created the necessary electrical connections, vias, and traces to establish the desired electrical pathways on the PCB. I ensured proper grounding, and power distribution throughout the board, following industry best practices and design guidelines.

Testing and Troubleshooting:

Throughout the project, I conducted rigorous testing and troubleshooting of the PCB. I verified its functionality, checked for potential issues like short circuits or voltage fluctuations, and rectified any problems that arose. I collaborated with other team members to ensure seamless integration and proper functioning of the entire cocktail bartender device.

By designing and implementing the PCB, I contributed to the overall functionality, reliability, and performance of the cocktail bartender device. Additionally, my collaboration with the team leader and other members fostered effective communication and coordination, ensuring that the electrical aspects of the project were aligned with the broader objectives and timeline.

References

DC5-30V power ESP32 WIFI bluetooth ble eight-way Relay Module. Retrieved from http://www.chinalctech.com/m/view.php?aid=540

Power supply 7282 PW SUP S-201-24. Retrieved from https://dac.ge/en/?product=4289&category=10&subcategory=24

Hilitand 12V 108W Thermoelectric Cooler Peltier Semiconductor Refrigeration DIY Water Cooling System Cooler Device with Fan. Retrieved from https://www.amazon.com/dp/B07DS8WBT8?psc=1&ref=ppx_yo2ov_dt_b_product_details

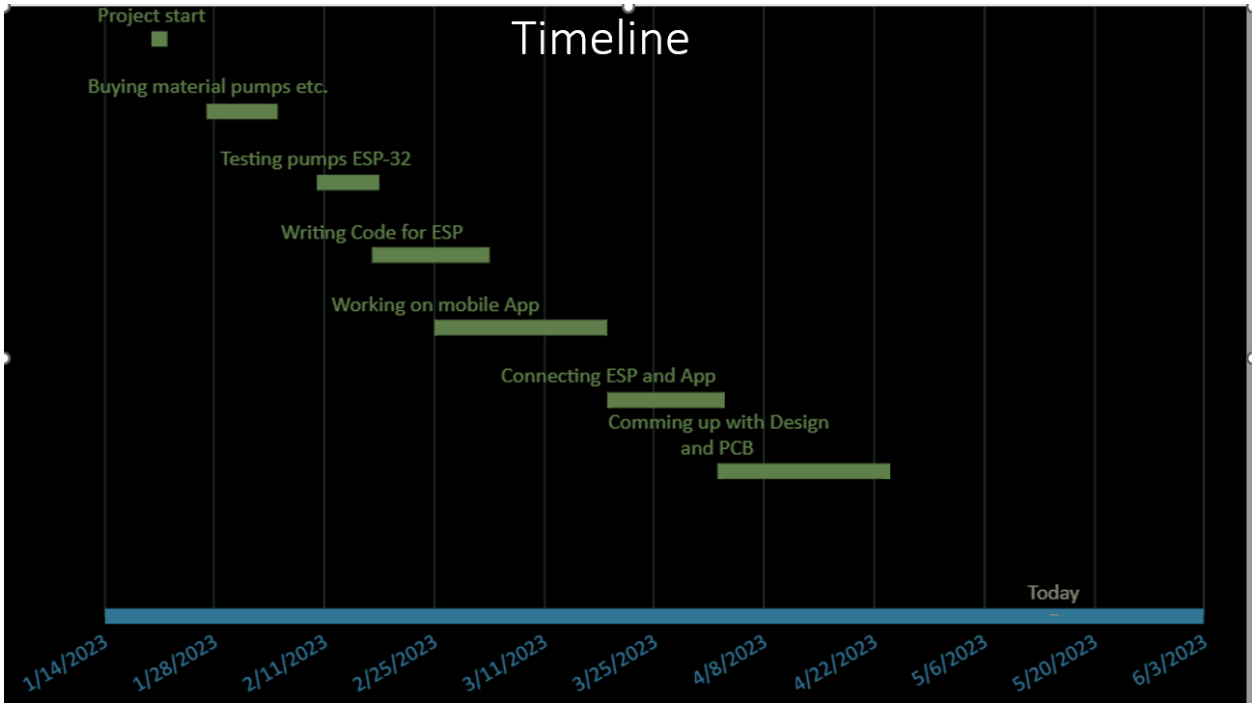
Efficient JSON serialization for embedded C++. Retrieved from https://arduinojson.org/

React native · learn once, write anywhere (no date) React Native RSS. Available at: https://reactnative.dev/.

Getting started with Bluetooth Classic. Retrieved from: https://kenjdavidson.com/react-native-bluetooth-classic/

Appendices

Timeline



Bill Of Materials

Components	Price in $	Price in GEL
ESP 32 x3		75
ESP 32 Relay Module x3	60$ + 20$	
Peltier Water Cooler	50$ + 20$	
Water Pumps x16		256
Water Tubes		110
Digital Multimeter		120
Arduino		20
Power Supply		90
Sparkling Waters		300
Wires		40
Relays		10
Funnels		15
Glass Bottle containers		35
Other Expenses		200