Smart Cup Coaster Project

Group 25

Nicodemus Ong (22607943)

Siyuan Zhou (23861658)

Amir Yazdanian Kalashtari (24135694)

Jiewen Su (22687382)

Semester 2, 2023

CITS5506 - Internet of Things

The University of Western Australia

Table of Contents

1 ABSTRACT

Contemporary lifestyles underscore a preference for convenience and personalized experiences in beverage consumption, often compromised by non-ideal temperature conditions leading to beverage waste. In Australia, the high consumption of hot beverages highlights a need for maintaining optimal temperatures, enhancing the drinking experience. Addressing this, we introduce the Smart Cup Coaster, designed to transform how individuals consume their drinks.

The Smart Cup Coaster utilizes an Arduino Uno board for core processing functions, coupled with an IR temperature sensor for accurate real-time temperature readings. An ultrasonic sensor detects cup placement, triggering the device. Unique from traditional coasters, it features interactive real-time temperature monitoring and alerts through a TTGO T-beam connection module, keeping the user informed when the beverage reaches the ideal temperature, thereby minimizing neglect and waste.

The device stands out with its user-centric design, offering customizable settings that remember preferred temperatures for different drinks, catering to individual preferences and ensuring a consistent experience. This innovation marries convenience with personal preference, paving the way for future developments in smart kitchen appliances by offering a solution that not only promotes efficient consumption and reduces waste but also significantly enhances the user's overall beverage consumption experience.

2 Problem Statement

We are currently grappling with significant issues concerning the maintenance of beverages at their ideal temperatures, a situation that gives rise to both inconvenience and undue waste. Individuals often encounter the problem of hot beverages cooling down excessively before they can be fully enjoyed, while cold drinks become undesirably warm over the course of consumption.

1. **Temperature Monitoring:** The inability to keep beverages at the preferred temperature compromises the drinking experience, often leading to discomfort and dissatisfaction among consumers. People frequently find their hot drinks less enjoyable once they cool down, while a lukewarm status can mar the experience of drinking something that was initially chilled. This temperature dilemma disrupts the overall enjoyment anticipated from savoring beverages at their ideal warmth or coolness.
2. **Waste Generation:** Another critical facet of this issue is the contribution to food and beverage waste. When drinks are not maintained at the desired temperature, they often end up being discarded. This behavior not only signifies a personal loss of value but also culminates in a broader societal issue, contributing significantly to the global problem of consumable waste. Beverages that are thrown away due to temperature non-compliance result in wastage of resources used in their production, packaging, and distribution.

Confronting this challenge is essential in not only elevating the experience of beverage consumption but also in curtailing avoidable waste. By providing a solution, we can empower individuals with the choice to relish their beverages at personally tailored temperatures, enhancing satisfaction and promoting a more responsible, waste-conscious culture.

3 INTRODUCTION AND BACKGROUND

In today's fast-paced world, there is a distinct need for an ideal beverage experience, specifically a means to maintain or adjust a drink's temperature to the perfect level. Given the significant number of individuals indulging in hot beverages, as depicted in Figure 1,

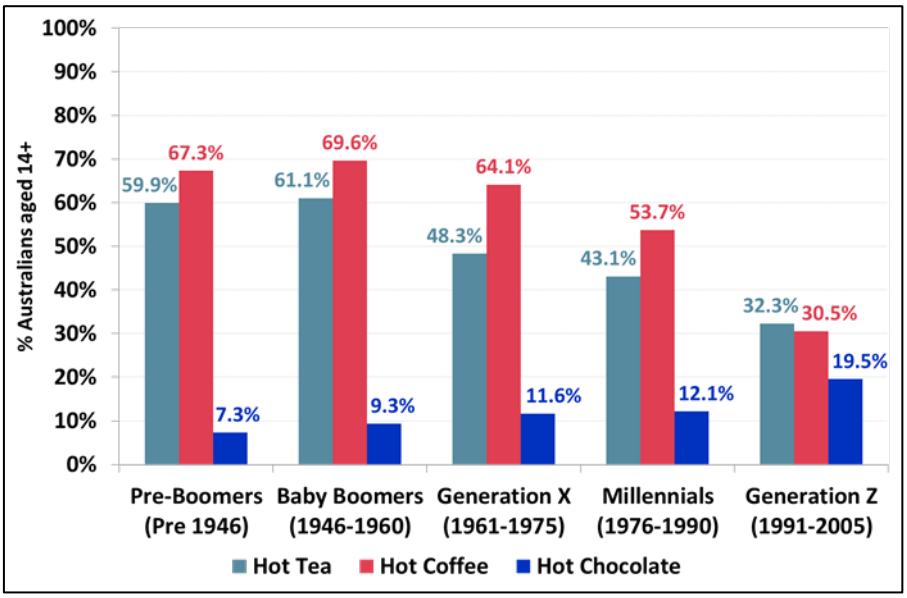

Figure 1: Hot drinks consumed by Australians in an average week by Generation. (Roy Morgan., 2017)

there is an escalating demand for smart appliances. These innovative products play a crucial role, efficiently signaling users when their food or beverage achieves the ideal temperature for consumption.

This is where the immense benefits of a smart coaster become evident. This device does more than merely measure temperature; it informs users when their drink has reached the optimal temperature for enjoyment without the risk of burning one's lips or tongue. Additionally, it alerts them if the temperature drops to a less desirable range, potentially compromising the beverage's taste. What's more, numerous smart coasters now offer features to either cool or warm beverages, preserving the desired temperature for extended durations.

Beyond the functionalities, smart coasters boast high portability and user-friendly operation. Their compact dimensions not only facilitate easy transport but also minimize clutter, offering a space-efficient alternative to bulkier items like thermos bottles. Importantly, the adoption of smart coasters could substantially diminish waste generated

from beverages discarded due to unsatisfactory temperatures, promoting a more sustainable consumption culture.

3.1 Primary Motivation for Smart Cup Coaster

The impetus for introducing the Smart Cup Coaster stems from a desire to significantly improve the beverage-consuming experience, particularly within the context of our contemporary, high-pressure lifestyle characterized by tight deadlines and relentless demands. Roy Morgan's recent research highlights that an overwhelming 74.1% of Australians aged 14 and up—amounting to over 15.2 million individuals—regularly consume hot beverages like coffee, tea, and hot chocolate weekly. This substantial figure underscores the necessity for a product capable of elevating the drinking experience by granting users the autonomy to adjust their beverages to the perfect temperature, thereby augmenting overall enjoyment, comfort, and reducing unnecessary beverage waste.

The Smart Cup Coaster is ingeniously designed with versatile, customizable settings, allowing users not only to establish their preferred beverage temperature but also to save these preferences for consistent future experiences. Such personalization can profoundly enhance user satisfaction while also saving precious time. It ensures that individuals can remain immersed in their tasks without the distraction caused by their drink reaching an undesirable temperature.

Furthermore, the Smart Cup Coaster is not just a convenience—it's an enhancement to our daily quality of life. It appreciates localized needs, understanding that, for instance, Perth's temperatures can soar to 40 degrees Celsius in the summer. While iced beverages are naturally enticing in such climates, studies led by Dr. Ollie Jay at the University of Ottawa indicate that consuming hot drinks can effectively cool the body, especially in areas with dry heat (McVean, A., 2023, July 7). This counterintuitive discovery implies a consistent demand for hot beverages regardless of the season. Beyond physical comfort, the psychological aspect also plays a part; the mere action of cradling a warm cup can evoke feelings of generosity and warmth towards others. Hence, the Smart Cup Coaster transcends physical utility, potentially nurturing a more empathetic and kind-hearted society, a need ever more pressing in the whirlwind of our modern existence.

3.2 Project Objectives

This project centres around the development of the Smart Cup Coaster, an IoT-enabled device poised to transform the beverage consumption landscape. Our approach is

methodical, breaking down the project into several nuanced phases. The overarching goal is to innovate a multifunctional smart coaster prototype, complete with an IR temperature sensor for precise beverage temperature readings. This venture is underscored by our dedication to ensuring users can savour their beverages at the perfect temperature, courtesy of a reactive feedback mechanism. This system employs various coloured LEDs, each representing a distinct temperature bracket, facilitating an immediate and instinctive understanding of the beverage's temperature status.

Moreover, our ambition extends beyond mere temperature regulation. We're integrating an ultrasonic sensor within the coaster to detect whether a cup or glass is present, ensuring a streamlined and automatic process. Another pivotal element is incorporating the TTGO module, establishing network connectivity that allows for remote beverage monitoring via a dedicated mobile application. This application is designed with user convenience in mind, offering an intuitive interface where users can specify their desired beverage temperatures, thereby tailoring a unique consumption experience.

Our vision for the Smart Cup Coaster is comprehensive, anticipating the need for real-time feedback. Additionally, we prioritize energy efficiency by utilizing low-power components and incorporating a sleep mode feature when the device is idle, significantly reducing the energy footprint.

In terms of quality assurance, we have instituted rigorous testing protocols to scrutinize every aspect of the device's performance, ensuring its optimal functioning and robust data management. These tests are designed to push the device to its operational boundaries, guaranteeing its reliability under various conditions. We're committed to exhaustive data recording during this phase, enabling a comprehensive review process for necessary adjustments, retrospective analyses, and efficiency assessments over the device's lifespan. This meticulous approach underscores our commitment to excellence and user satisfaction in the revolutionary Smart Cup Coaster project.

4 Project Genesis and Evolution

The inception of our project was marked by collaborative brainstorming sessions, wherein we aimed to converge on an idea that was not only practical but also imbued with an element of intrigue. Our initial proposition revolved around developing a 'Smart Locker,' acknowledging its utility in the IoT spectrum and the abundance of accessible online resources for reference. However, our project selection methodology was grounded in a 'first come, first serve' basis, and coincidentally, another group had already earmarked the Smart Locker concept.

This minor setback propelled us back to the drawing board, leading to further deliberation and research. Our collective intellectual curiosity was piqued by the concept of a 'Smart Cup Coaster'—a project that promised practicality and the prospect of augmenting everyday convenience.

Following the crystallization of our project theme, our focus shifted to meticulous research aimed at sourcing appropriate hardware. This phase was critical, necessitating that our selections met specific criteria: availability through university resources, cost-effectiveness, and prompt delivery timelines. Our university's provision of microcontrollers simplified this process, and our decision gravitated towards the Arduino due to its ubiquitous presence on campus. Its reputation for facilitating real-time control tasks efficiently and its user-friendly nature assured us that it was conducive to adhering to our project timeline.

However, our hardware requirements extended beyond what was available on campus, specifically necessitating the acquisition of two additional sensors: an IR temperature sensor and an ultrasonic sensor. The inclusion of the ultrasonic sensor was strategic, enabling the device to discern the presence of any object placed upon it. In parallel, the IR temperature sensor was integral to the functionality of reading temperature levels accurately.

This journey from conceptualization to actualization was a testament to our team's adaptability, commitment, and cohesive effort, steering us through decisions, re-evaluations, and final implementations in our pursuit to bring the Smart Cup Coaster to life.

4.1 Dynamic Task Allocation

Before starting work on the project, we discussed the task allocation about coding and hardware tasks. Ordinally, in the proposal document the task allocation is like:

TABLE I

ORDINAL TASK DISTIBUTION

Name of Student	**Work Assigned**
Jiewen Su	Documentation, and software programming
Siyuan Zhou	Software Programming, and hardware
Nicodemus Ong	Software Programming
Amir Yazdanian Kalashtari	Hardware and Documentation

As our team navigated the development process of the Smart Cup Coaster, we were acutely aware of the pressing time constraints we faced, with just weeks to bring our project to fruition. Given these limitations, we adopted a strategy of dynamic task allocation to expedite the realization of the coaster's essential functions, primarily its effective categorization and handling of beverages into three distinct temperature states: cold, warm, and hot. We collectively understood these core functionalities to be the crux of our project's primary objective.

This methodology wasn't merely about meeting deadlines; it was about paving the way for future enhancements and potential feature expansions. By concentrating on foundational elements first, we ensured the project's completion within the allocated timeframe while establishing a flexible and adaptable framework capable of evolving in response to the unpredictable challenges inherent in hardware and software development.

To bolster efficiency and maintain project momentum, we continuously reassessed and reallocated tasks, capitalizing on individual team members' strengths, availability, and the project's evolving demands. This fluid approach to task distribution was instrumental in optimizing resource management and overall productivity.

Below, we present the revised task distribution framework, reflecting our adaptive process and strategic response to the project's demands and constraints:

TABLE Ⅱ

NEW TASK DISTIBUTION

Name of Student	Work Assigned
Jiewen Su	Documentation
Siyuan Zhou	Software Programming, Software development, staging, troubleshooting and deployment, code documentation
Nicodemus Ong	Software development, staging, troubleshooting and deployment, code documentation Hardware design, implementation, troubleshooting, hardware documentation, report generation, report refinement, report editing/proof reading
Amir Yazdanian Kalashtari	Hardware and Documentation

This dynamic task allocation strategy underscored our commitment to agility and resilience in the face of project development hurdles, ensuring not just the completion of the Smart Cup Coaster within the deadline, but also laying a robust groundwork for its potential future evolution.

4.2 Project Timeline

The timeline for the Smart Cup Coaster project is tight. Considering the time for drafting the project idea, we had to complete the overall project in 5 weeks' time and with additional another week allocated for documentation. The following picture illustrate the timeline of the Smart Cup Coaster project.

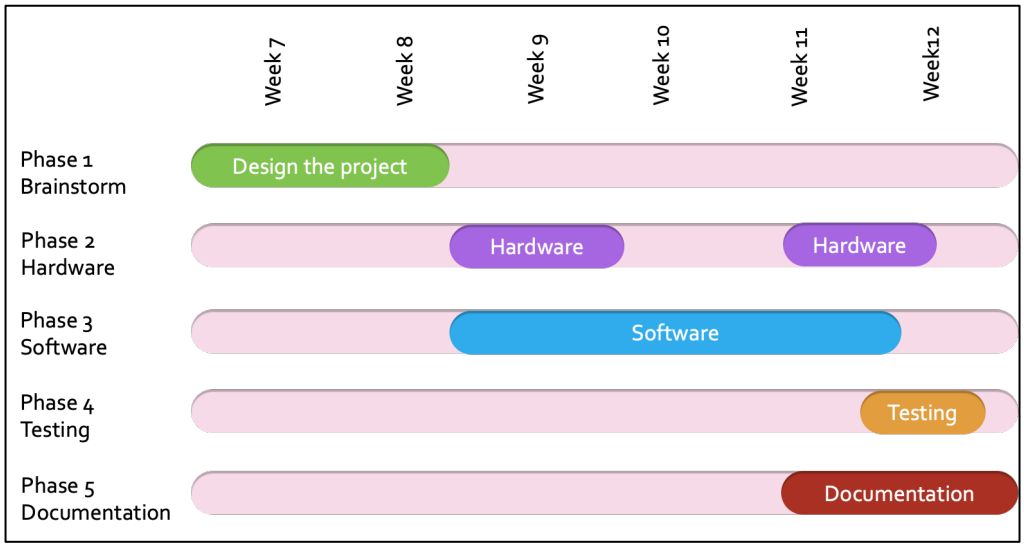

Figure 2: Project Timeline

We began the discussion for designing the project including the hardware and software we are going to use. This phase lasted from week 7 to week 8. Following the submission of the project proposal, we started on a period waiting for the hardware to arrive. We got most of the hardware component on the Monday of week 8 but because of sensor shortage problem of the original temperature sensor, we waited until week 11 for the new sensor to arrive. There is a gap between the implementation of hardware components. The software planning and implementation started as soon as the brainstorm phase ended. It began with the platform we would use in the project and then working on the code. The testing was done on the beginning of week 12. Documentation lasted for two and a half weeks starting from the end of week10.

4.3 Problems in Development

The journey through the development phase of our Smart Cup Coaster project was not without its challenges, particularly concerning hardware availability and connectivity issues. These hurdles tested our adaptability and problem-solving skills, pushing us to devise immediate and innovative solutions to keep the project on track.

4.3.1 Hardware Component Unavailability

In our initial phase, we faced an unexpected setback with the discontinuation of our chosen non-contact IR sensor module, leading to an unavoidable delay as we awaited the arrival of an alternative. To maintain our momentum, we adopted a hard-coding approach, allowing us to progress with the software components while we awaited the new hardware. This

workaround ensured continuity but also highlighted the need for flexible planning within hardware-dependent projects.

4.3.2 Connectivity Complications and Component Transition

The ESP8266 Mini initially promised a compact solution for our Wi-Fi connectivity needs. Still, its unreliable connection, prone to disruption due to a loose connector pin, prompted us to opt for the more stable TTGO T-Beam LilyGO module. This crucial pivot wasn't just about maintaining a connection; it was about ensuring consistent performance and reliability for the end-user. Our experience underlined the importance of selecting components not just for their capabilities but also for their durability and stability in real-world applications.

4.3.2.1 Notification System Dilemma:

Pushbullet, our chosen notification system, proved effective during Android testing but threw us a curveball with its sudden disappearance from the iOS App Store. This development restriction nudged us toward broader platform testing, emphasizing the need for foresight into cross-platform compatibility, especially for products intended for a diverse user base.

4.3.2.2 Revising Connection Methodology

Our initial Device-to-Device (D2D) communication strategy, while functional, restricted users' internet access, prompting a shift to a Device-to-Cloud (D2C) approach. Despite its advantages, D2C introduced latency issues, adversely affecting user interaction. We reverted to D2D, sacrificing concurrent internet functionality for a more responsive user experience. This decision, while difficult, prioritized immediate usability, crucial for our demonstrations.

4.3.2.3 Navigating Built-in Wi-Fi Limitations

The TTGO T-Beam board's built-in Wi-Fi, initially perceived as a boon, presented a significant hurdle: its inability to support simultaneous two-way communication with our cloud service and API. To navigate this, we engineered a dynamic connection-switching mechanism. By alternating between cloud and internet connections, the board could maintain uninterrupted service, albeit with temporary functional pauses during the transitions. This intricate dance between connections, a matter of mere seconds, involved halting all other functionalities, emphasizing uninterrupted communication as paramount. The

solution, while not without its compromises, underscored our commitment to delivering a seamless user experience.

Each challenge, a learning experience, imbued our team with invaluable insights into the intricacies of product development, particularly the balance between ideal functionality and practical adaptability. These lessons from the bedrock for future projects, highlighting the need for versatility, comprehensive testing, and user-centric decision-making.

5 SOLUTION AND IMPLEMENTATION

5.1 Hardware Design Overview

Throughout the Smart Cup Coaster project's evolution, specific hardware criteria were meticulously evaluated to guarantee the prototype's successful realization. Herein, we explore these pivotal criteria and their overarching significance:

5.1.1 Cost

5.1.1.1 Economic Efficiency:

Cost-efficiency was paramount in the prototype's developmental phase, necessitating a disciplined approach to budget adherence. Our ambition was to democratize access to the Smart Cup Coaster, making it an affordable option for diverse demographics, including students. We adopted a frugal budget strategy, capping expenses at $50. This budget excluded fundamental components such as the Arduino TTGO LilyGO, jumper wires, LEDs, and resistors, generously provided by the university. This fiscal prudence was vital in preserving the product's affordability, fostering its potential for mainstream adoption.

5.1.2 Availability

5.1.2.1 Proximity of Resources:

The project prioritized local sourcing within Perth to expedite the assembly process, ensuring prompt availability of necessary hardware components. This strategic localization minimized wait times associated with shipping, optimizing our team's efficiency and resource accessibility.

5.1.3 Size

5.1.3.1 Ergonomic Design:

The coaster's dimensions were critically assessed to guarantee its practicality. It was crucial for the device to have a sufficient surface area to accommodate various types of cups or glasses while maintaining accuracy in temperature measurement. The housing design was equally pivotal, conceived to encapsulate all integral hardware in a sleek, compact ensemble. This meticulous design philosophy ensured the Smart Cup Coaster's aesthetic minimalism and functional discretion.

5.1.4 Power Requirement

5.1.4.1 Universal Compatibility:

We standardized all hardware components to function optimally at five volts, a decision that underscored compatibility and consistent performance. The device's power was harnessed via a USB cable, a commonality among various powering interfaces—laptops, desktop computers, or standard charging adapters. This universal approach fortified the coaster's user-friendly persona, simplifying its operational setup.

5.1.5 Temperature Range

5.1.5.1 User-Centric Adaptability:

The coaster featured a sophisticated temperature sensor, capable of precise readings ranging from zero to one hundred degrees Celsius. A standout feature was the user's ability to define their unique temperature parameters for what constitutes cold, warm, or hot. Such personalized calibration meant the device resonated more intimately with individual user inclinations.

5.1.5.2 Colour-Based Communication:

For an immediate temperature reading, we integrated a color-coordinated LED system. Each color—blue, yellow, and red—was indicative of a specific temperature range (cold, warm, hot, respectively). This instant visual reference enhanced user interaction, making beverage temperature assessment instinctive and immediate.

In conclusion, every facet of the Smart Cup Coaster's hardware solution was diligently curated, balancing cost-efficiency, resource availability, design practicality, power compatibility, and temperature customizability. These collective efforts culminated in a

prototype that was not only user-centric and versatile but also economically accessible, meeting both the project's ambitions and prospective users' anticipations.

5.1.6 Hardware implementation

The hardware framework of the device is anchored by several key components, including an ESP32 Arduino network module found integrated on the TTGO board, a trio of LEDs, a breadboard, an ultrasonic sensor, and an IR temperature sensor. The ensuing diagram offers an in-depth exploration of the hardware orchestration.

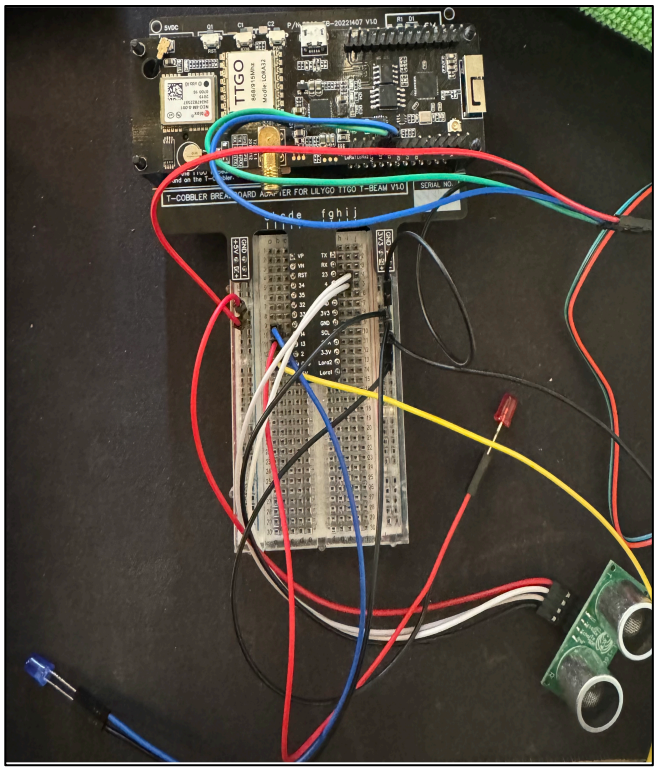

Figure 2a: implementation of hardware connection

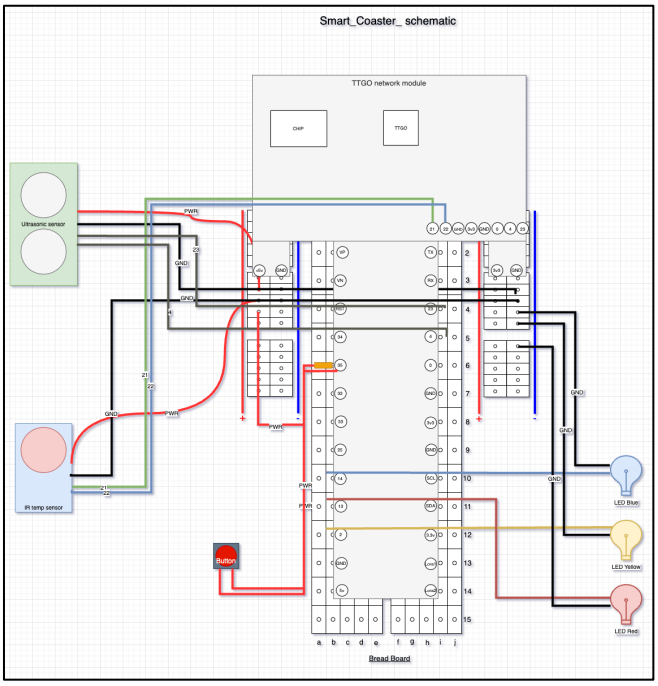

Figure 2b: Schematic of hardware connection

The device's electrical framework is streamlined through the ESP32 module, which supplies power directly to the breadboard and facilitates further distribution. Additional power is sourced from the positive pins, enhancing the operational capacities of the sensors.

Notably, the ultrasonic sensor distinguishes itself with dual-functionality pins: the 'trigger' and 'echo.' These are allocated to pins 23 and 4 on the TTGO board, respectively, handling detection and the initiation of a responsive action.

Complementing this, the IR temperature sensor, though powered similarly, operates under different dynamics. Unlike the ultrasonic counterpart, it engages in perpetual scanning, monitoring temperature variances. Such alterations are meticulously logged by the software and reflected within the mobile application interface. Connectivity for this sensor is established through pins 22 and 21 on the TTGO board, interfacing with the SDA and SCL pins of the IR apparatus.

Additionally, three distinct LEDs are interfaced with pins 14, 13, and 2 on the TTGO board, each designated for a specific temperature indication: blue for cold, yellow for warm, and red for hot conditions. These LEDs provide an intuitive visual guide for users to assess their beverage's temperature at a glance. All components within the system are electrically grounded through appropriate ground pins, ensuring stability and reliability in the device's operations.

In enhancing functionality, a reset button with pin 35, is integrated into the design, allowing for user-friendly interaction with the device. This button simplifies the transition between uses, swiftly resetting the notification state to a default state and making the system primed for the placement of a new cup or glass. This feature ensures a seamless, continuous experience for users, catering to the need for convenience in repeated use. To notify the user that the button is pressed, all three LEDs will flash as a signal. The figures below showcases the final product fitted into its custom 3d printed case.

Figure 2c: Fully fitted device in its 3d printed case (diagonal view)

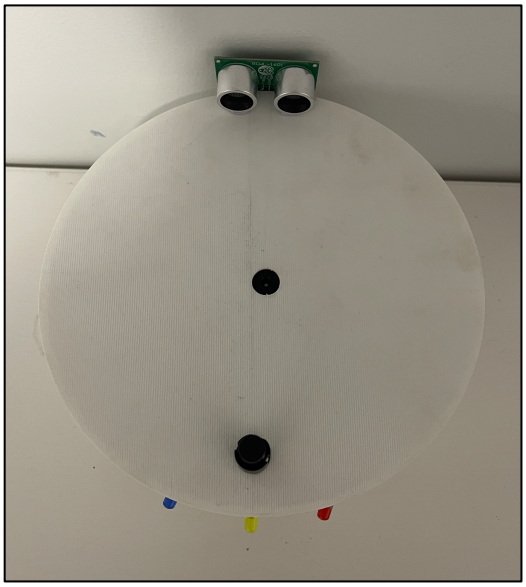

Figure 2d: Fully fitted device in its 3d printed case (top view)

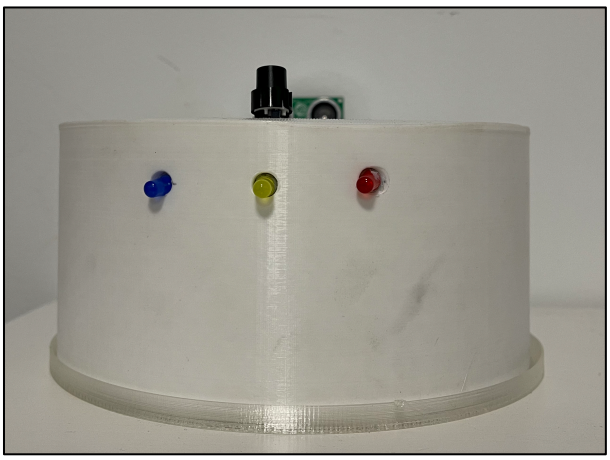

Figure 2e: Fully fitted device in its 3d printed case (front view)

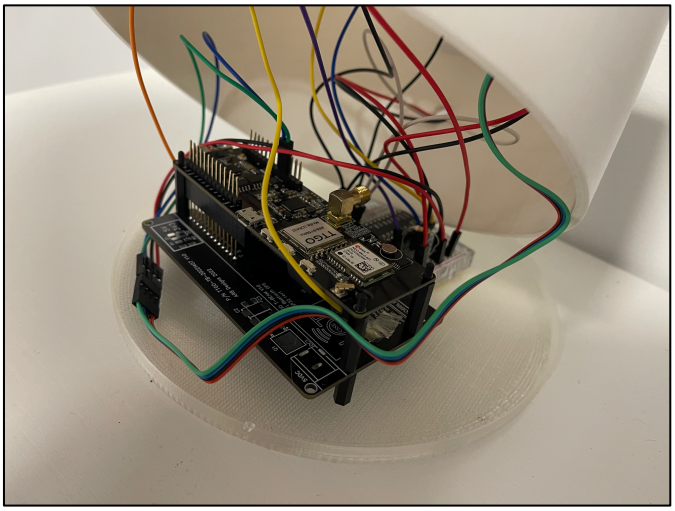

Figure 2f: Internal device look

Figure 2g: example of a cold beer placed on the device

Figure 2h: example of the button pressed

5.2 Software Architecture Overview

The software infrastructure of the Smart Cup Coaster ingeniously merges the capabilities of two distinct Application Programming Interfaces (APIs) - RemoteXY and Pushbullet - forming a robust framework that enhances both functionality and user experience. This dual-API architecture is fundamental in transforming the coaster into a smart, interactive gadget.

5.2.1 Integration of RemoteXY API:

Central to the coaster's interactive features is the implementation of the RemoteXY API. This powerful tool allows for the crafting of custom, intuitive graphical user interfaces (GUIs), significantly simplifying user interaction with the device. It is instrumental in the Smart Cup Coaster's ability to provide a visually rich and user-friendly experience.

- Interactive Control Panel:
    - The prowess of RemoteXY manifests in a comprehensive control dashboard, functioning as the primary interaction point for users. It integrates controls for various coaster functionalities, including temperature adjustment and beverage type selection, all accessible via a smartphone. This convenience enhances the overall user interaction, making beverage enjoyment a seamless experience.

5.2.2 Pushbullet API for Enhanced Connectivity:

- Complementing RemoteXY's user interface, the coaster incorporates the Pushbullet API, extending its utility through the provision of immediate, real-time notifications.
    - Direct User Alerts:
        - The system utilizes Pushbullet to keep users abreast of their beverage conditions, sending timely alerts directly to their smartphones. This feature ensures users are promptly informed when their drink reaches the optimal temperature, enhancing the beverage consumption experience.
    - Personalized Notification System:
        - Beyond basic alerts, Pushbullet offers users the flexibility to personalize their notification settings. Users can specify their preferred alert methods, whether through auditory signals, vibrations, or visual pop-ups, ensuring the notification process aligns with their individual needs and situations.

The strategic alliance of RemoteXY and Pushbullet within the Smart Cup Coaster's software ecosystem establishes a sophisticated, responsive, and user-oriented device. It transcends conventional coasters, redefining user engagement by ensuring optimal beverage enjoyment through a blend of aesthetic interfaces and customized, real-time communications.

5.2.3 Software Implementation

The implementation for a "Smart Coaster" system, leveraging the capabilities of ESP32, RemoteXY service, and Pushbullet's notification service. The system monitors the temperature of a drink placed on the coaster, controlling LEDs to indicate the drink's temperature range, and sending notifications to the user's device when the beverage reaches a desired temperature.

5.2.3.1 System overview

RemoteXY: A platform that facilitates the creation of GUIs for controlling Arduino and ESP32 devices. In this script, it's used for real-time monitoring and controlling various parameters like temperature settings and receiving notifications.

- ESP32: A powerful microcontroller with Wi-Fi capabilities, acting as the brain of the operation.

Pushbullet: A service that provides functionalities like sending notifications to devices. Here, it notifies when the drink has reached the specified temperature.

5.2.3.2 Functionality

- Setup and Configuration:
    - The system starts by initializing the RemoteXY service with the specified configuration, setting up a Wi-Fi access point.
    - Pins for LEDs and sensors are configured, and initial temperature settings are loaded into the corresponding RemoteXY variables.
    - The script initializes the serial communication for debugging and resets the IR temperature sensor.
- Temperature Reading and Distance Checking:
    - The program continuously reads the IR sensor to check the temperature of the object on the coaster.
    - An ultrasonic sensor determines whether an object is placed on the coaster.

- User Interaction and Control:
    - The user can interact via the RemoteXY interface, adjusting temperature settings for different beverage types (cold, warm, hot).
    - LED indicators provide a visual representation of the drink's current temperature state.
- Notification Mechanism:
    - If the temperature criteria meet, the system triggers a Pushbullet notification. The ESP32 connects to the Wi-Fi network, interacts with Pushbullet's API, sends the notification, and then disconnects to resume normal operation with RemoteXY.

5.2.3.3 Deep Dive into Notable Features

Handling Wi-Fi and RemoteXY Connection:

The script smartly manages its connections. While RemoteXY requires a continuous connection to interact with the GUI, the Pushbullet notification feature needs internet access. To achieve this, the script temporarily disconnects from the RemoteXY server to establish a Wi-Fi connection, send a Pushbullet notification, and then immediately disconnects from Wi-Fi to resume the RemoteXY session. This ensures seamless user experience while keeping the system responsive for real-time control. The code snippets below depicts the function that was described.

```
// Initialize WiFi for Pushbullet
void initializeWiFi() {
  // Connect to WiFi using the provided SSID and password
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    RemoteXY_delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");
}

void disconnectWiFi() {
  // Disconnect from WiFi
  WiFi.disconnect();
}
```

Figure 3a: Function for WiFi connection to the internet

```
if (!buttonFlag && distanceFlag) {
  if (RemoteXY.SMSswitch == 0) {
    if (cold - 2 <= temp && temp <= cold + 2) {
      initializeWiFi();
      sendNotification("cold");
      Serial.print("Sending Object celsius: ");
      Serial.print(temp);
      Serial.println(" °C");
      buttonFlag = true;
    }
  } else if (RemoteXY.SMSswitch == 1) {
    if (warm - 2 <= temp && temp <= warm + 2) {
      initializeWiFi();
      sendNotification("warm");
      Serial.print("Sending Object celsius: ");
      Serial.print(temp);
      Serial.println(" °C");
      buttonFlag = true;
    } else {
      if (hot - 2 <= temp && temp <= hot + 2) {
        initializeWiFi();
        sendNotification("hot");
        Serial.print("Sending Object celsius: ");
        Serial.print(temp);
        Serial.println(" °C");
        buttonFlag = true;
      }
    }
    disconnectWiFi();
    Serial.println("Disconnected from WiFi");
```

Figure 3b: Function that swaps between WiFi modes

Pushbullet Integration:

The integration with Pushbullet is achieved through HTTP requests. The system uses the ESP32's Wi-Fi capabilities to connect to the internet, formulates an HTTP POST request containing the notification's title and body, and uses the Pushbullet API endpoint to send this data. The user's API key authenticates the request. After the server's response, which is either confirmation or an error message, the system closes the HTTP connection and disconnects from Wi-Fi. The code snippet below shows the functionality that was described.

```
void sendNotification(String type) {
  // Send a Pushbullet notification with the specified type
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin("https://api.pushbullet.com/v2/pushes");
    http.addHeader("Authorization", "Bearer " + String(apiKey));
    http.addHeader("Content-Type", "application/json");

    String json = "{\"type\": \"note\", \"title\": \"SMART COASTER Notification\", \"body\": \"Your " + type + " drink is ready!!\"}";
    int httpResponseCode = http.POST(json);
    if (httpResponseCode > 0) {
      String response = http.getString();
      Serial.println(httpResponseCode);
      Serial.println(response);
    } else {
      Serial.println("Error on HTTP request");
    }
    http.end();
  }
}
```

Figure 3c: Push Bullet Function

In Conclusion, The Smart Coaster system is an innovative use of various technologies that creates a seamless and interactive experience for the user. By smartly toggling between different connection states, it maintains real-time control with RemoteXY while enabling external communications via Pushbullet. This project is a testament to the versatility of microcontrollers and the integration capabilities of different IoT services. Future improvements can include more stable connection handling, energy efficiency optimizations, and expanded notification features.

6 Result and Discussion

The development journey culminated in several key outcomes that underlined the Smart Cup Coaster's efficacy and user-centric design:

6.1 Intelligent Notifications

A core feature of the coaster is its ability to discern the beverage's temperature, categorizing it into one of three distinct levels: hot, warm, or cold. Upon reaching the user-preferred temperature, the device sends a one-time notification, ensuring users are informed without feeling overwhelmed by repetitive alerts.

6.2 Real Time Monitoring

Integration with a mobile application has enabled live tracking of the beverage's temperature, offering users immediate updates right on their screens. This interactive element ensures users are consistently aware of their drink's status.

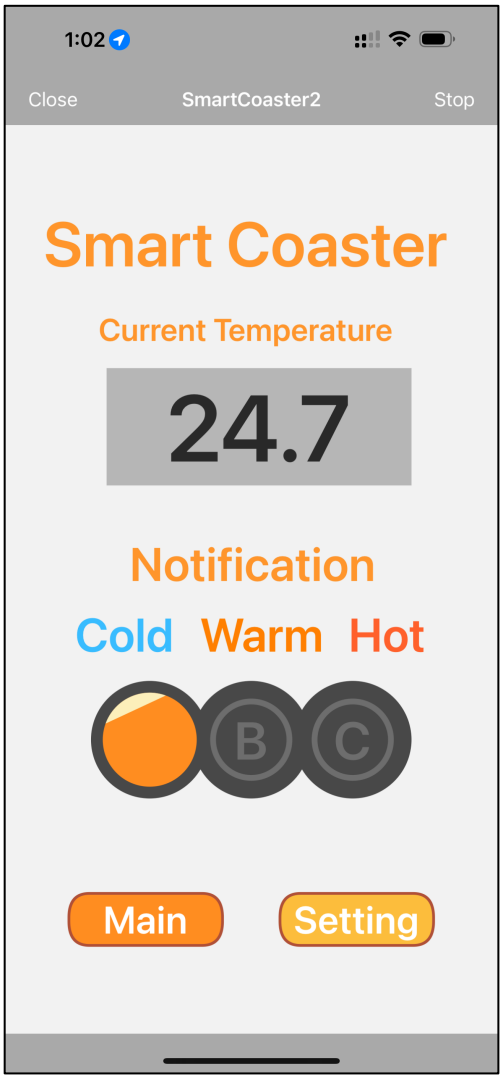

Figure 4a: Main UI page on phone app

6.3 Personalized Preferences

Embracing user individuality, the coaster allows for customized temperature notifications. Users dictate their ideal points for what constitutes cold, warm, or hot, enhancing the personal relevance of each notification received.

Figure 4b: Settings UI page on phone app

6.4 Physical Indicators

Beyond digital alerts, the coaster features an LED system as a visual aid reflecting the beverage's temperature:

- Red signals a hot beverage,
- Yellow denotes a warm drink,
- Blue indicates a cold one.

This immediate, at-a-glance system informs users, even if their mobile device is not immediately accessible.

6.5 Single-use Notification System

To avoid notification saturation, the design includes a one-alert-per-beverage protocol, striking a balance between informative and non-intrusive. This feature respects the user's space while keeping them informed.

6.6 Reset functionality

A practical addition is the physical reset button, signaling the placement of a new beverage on the coaster. This function supports the device's readiness to monitor a fresh round without confusion over residual temperature data from the previous cup.

7 Verification of Performance

To validate the coaster's functionality, rigorous testing phases were employed, examining the device's responsiveness and accuracy across various scenarios.

7.1 Accuracy Assessment:

Real-world testing involved exposing the coaster to beverages at different temperatures, observing the device's ability to accurately read and report these changes. The trials confirmed the device's sensitivity and precision in real-time temperature detection, as reflected on the mobile application interface. This meticulous process, demonstrated through the visual evidence provided below, certifies the product's readiness for everyday use.

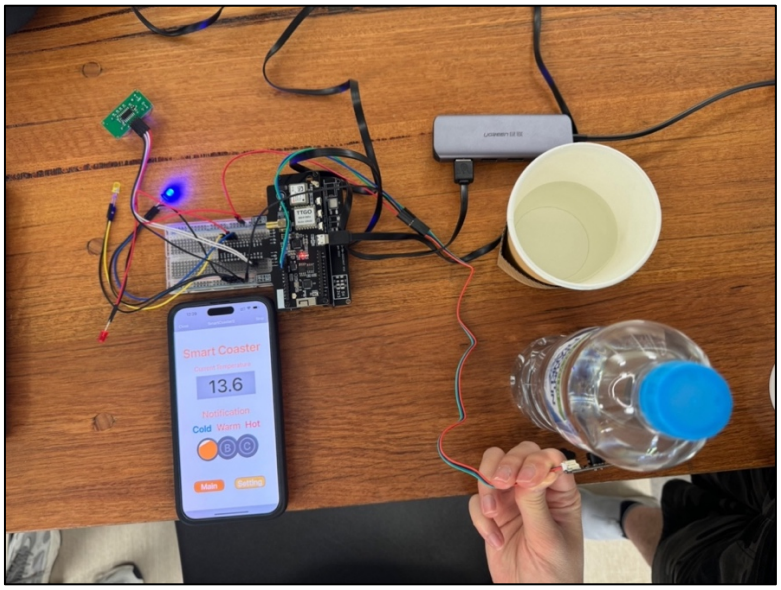

Figure 5a: result of a cold response

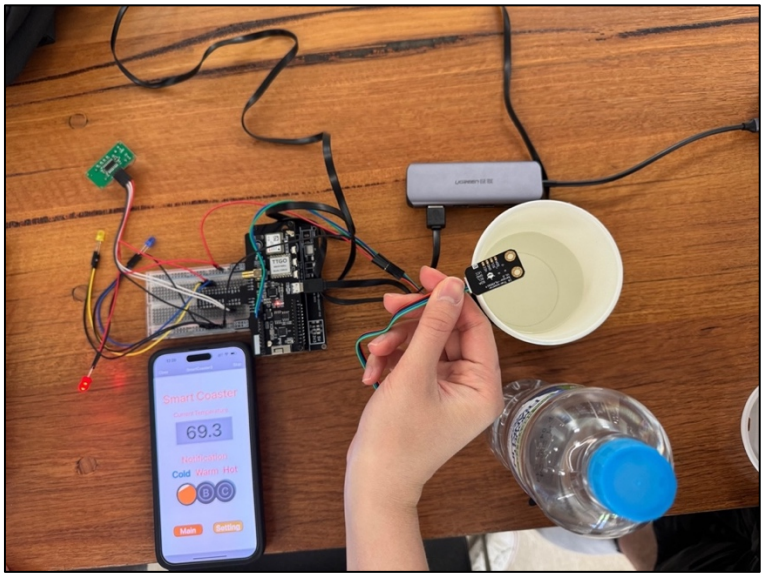

Figure 5b: result of a hot response

results					
cold		warm		hot	
Thermometer	Sensor	Thermometer	Sensor	Thermometer	Sensor
15.45	14.97	36.14	35.68	64.28	65.44
15.87	15.42	34.25	36.45	65.75	66.10
14.23	15.15	35.86	34.72	66.12	64.87
16.00	14.81	34.77	35.14	65.44	65.75
15.12	15.64	36.91	36.27	64.91	64.52
14.76	15.28	34.12	34.93	65.67	65.23
15.68	15.93	35.03	35.36	65.23	66.04
15.34	14.34	35.69	36.09	64.59	65.68
14.91	15.73	36.47	34.61	66.01	65.96
16.00	15.09	34.58	35.79	65.87	64.35
14.56	14.67	35.21	36.62	64.33	65.12
15.79	15.55	36.33	34.29	66.29	65.81
15.03	15.36	35.44	35.57	65.56	64.69
15.21	14.14	34.96	35.02	65.09	66.25
14.65	15.61	36.58	36.74	64.76	65.57

Through these tests and the resulting data, we have not only verified the coaster's operational effectiveness but also showcased its potential as a convenient and reliable addition to any setting, personal or professional.

The analysis of temperature data, juxtaposing readings from our sensor and a standard thermometer across diverse conditions, unveils critical insights into the sensor's performance and reliability.

7.1.1 Systematic Bias and Calibration Concerns

A persistent trend is observed in the sensor's propensity to record marginally elevated temperatures relative to the thermometer. This uniform bias points towards potential calibration discrepancies or inherent offsets within the sensor's measurement mechanics. Understanding the root of this systematic deviation is paramount, as it underscores the necessity for possible recalibration or a reassessment of the sensor’s baseline accuracy standards.

7.1.2 Conditional Accuracy Variances

Scrutinizing the sensor’s fidelity under various thermal conditions reveals a pronounced accuracy gradient. Interestingly, the sensor’s readings closely mirror the thermometer in cooler environments, while hotter contexts elicit a notable divergence. This inconsistency signals that the sensor’s technology and design might be less equipped to cope with higher temperature spectra, raising questions about its suitability and reliability under such extremities. Consequently, it’s imperative to confirm whether the sensor's operational parameters are aligned with the specific thermal ranges it encounters.

7.1.3 Precision Disparities Across Thermal Zones

The realm of mid-range temperatures, or the "warm" spectrum, unveils a broader scatter in the sensor’s readings, hinting at diminished precision within this thermal niche. This erratic behavior could stem from a multitude of sources, including the sensor's intrinsic calibration, its sensitivity, or external environmental influencers. Conducting a deep dive into these contributing factors is essential, aiming to fortify the sensor's consistency across the full temperature gamut.

7.1.4 Anomalies and Data Noise

Occasional stark outliers in the sensor’s data, divergent from expected trends, suggest the intrusion of 'noise' or disruptions in measurement integrity. Such anomalies could be the byproduct of electromagnetic disturbances, gradual sensor degradation (drift), procedural inconsistencies in data logging, or other extraneous variables. Mitigating these disruptions necessitates a holistic review of the data acquisition ecosystem, encompassing sensor maintenance protocols, calibration cycles, and the operational environment to curtail interference and crystallize data purity.

In essence, this comparative temperature analysis serves as a springboard for comprehensive sensor performance optimization. By untangling the web of factors influencing accuracy and precision, we can strategize targeted enhancements, from recalibration to redesign, ultimately steering the sensor functionality towards unerring reliability in diverse operational theatres.

8 Overcoming Project Constraints: Strategies for Enhancement

8.1 Navigating Software Restrictions

8.1.1 Pushbullet API's Limited Scope

Opting for Pushbullet's cost-effective solution influenced our initial direction, but its Android-centric development presents a significant hurdle, particularly concerning the iOS experience. To transcend this limitation, future development necessitates investing in a premium, cross-platform compatible API. Such an upgrade promises a harmonious experience across devices, enriching engagement by eliminating platform-induced disparity and positioning the product on a universal pedestal.

8.1.2 Hurdles with RemoteXY's Inflexibility

Our endeavors with RemoteXY confronted a bottleneck when its customization capabilities did not measure up to our innovative demands. This rigidity stymied development flow, embedding complications into the process. To circumvent this in subsequent iterations, we propose scouting for software solutions that champion customization or pioneering our proprietary interface. This strategy will grant us sovereign control over functionality and aesthetics, fostering an interface that resonates with our vision.

8.2 Hardware Bottlenecks and Resolutions

8.2.1 The Arduino Conundrum

While Arduino served preliminary needs, its deficiencies, especially in cloud connectivity linked to the TTGO module's restricted memory, surfaced starkly. Transitioning to Raspberry Pi, renowned for its superior computational prowess, can substantially alleviate these issues. By harnessing Raspberry Pi's expansive capabilities, we can fortify cloud interactions, thus optimizing performance where Arduino faltered.

8.3 Refining Physical Aesthetics

8.3.1 Embracing Compactness

The current iteration of the Smart Cup Coaster, albeit functional, is marred by its cumbersome form, an artifact of utilizing off-the-shelf components. Transitioning from a proof of concept to a consumer-ready model involves embracing miniaturization. Implementing bespoke or meticulously sourced components will enable us to shrink the footprint, boosting the device's portability and integration into everyday life, thereby enhancing consumer appeal through ergonomic elegance.

8.4 Horizon of Expansion

The Smart Cup Coaster stands on the precipice of evolution, with vast potential landscapes to explore.

8.4.1 IoT Ecosystem Symbiosis

The coaster's true potential is unlockable through its amalgamation into the sprawling IoT network. Imagine a symbiotic relationship with smart home infrastructures, enabling users to orchestrate an array of devices via a singular hub, their smartphone, or voice command. This integration will serve as a conduit for unparalleled convenience and home management efficiency.

8.4.2 Harnessing User Data

The coaster, a silent observer, can morph into a reservoir of user behavioral data, documenting intricate patterns from beverage preferences to consumption spaces. Pooling this intelligence with broader smart home analytics paves the way for hyper-personalized user experiences and catalyzes the innovation of IoT gadgets that resonate with consumer lifestyles on a nearly instinctual level.

8.4.3 Pioneering Hydration Wellness

Envision the coaster transcending its basic utility, evolving into a guardian of hydration. By synthesizing consumption data with personal health metrics, it can usher in proactive hydration reminders, aligning with daily intake goals personalized for each user. This transformation from a passive tool to an active health ally signifies not just an enhanced user journey, but a stride toward holistic well-being.

In essence, the pathway to refining the Smart Cup Coaster involves surmounting present limitations through strategic investments in software and hardware enhancements and foreseeing its metamorphosis from a standalone gadget to a vital cog in the IoT and personal wellness sphere. This journey mandates a vision that is both microscopic in addressing immediate technical constraints and telescopic, gazing at the boundless potential horizons.

In summary, the journey ahead for the Smart Cup Coaster is brimming with opportunities that fuse functionality with empathy. The roadmap is laid out toward a horizon where technology converges with human habits and needs, crafting experiences that resonate on a personal level, turning everyday living from a mundane routine into a finely tuned harmony of interactive wellness and convenience. The Smart Cup Coaster's potential lies in its evolution from a device of utility to a beacon of a holistic, intuitive lifestyle.

9 Conclusion

As we draw this discourse to its close, it is imperative to reflect on the journey of the Smart Cup Coaster project, an endeavor that began as a fusion of innovation and practicality aimed at enhancing the beverage-consuming experience. Through various phases of development, testing, and refinement, the project has not only achieved its preliminary objectives but also revealed a spectrum of potential waiting to be explored.

From its inception, the Smart Cup Coaster was envisioned as a bridge between convenience and technology. The real-time temperature monitoring and user-centric notifications stood testament to this, responding to a tangible need with elegance and sophistication. The challenges encountered, particularly highlighting the sensor's systematic bias and the precision discrepancies across different temperature conditions, were instrumental in understanding and emphasizing the importance of meticulous calibration, data integrity, and consistent performance metrics.

Addressing the limitations within the software and hardware realms unearthed opportunities rather than obstacles. The Pushbullet API's compatibility issues and the customization constraints with RemoteXY underscored the necessity for a versatile, robust platform that resonates with the needs and expectations of a diverse user base. Similarly, the transition from Arduino to Raspberry Pi is not merely a switch but a strategic enhancement to accommodate superior computational demands, particularly for seamless cloud connectivity.

The physical design of the Smart Cup Coaster, critiqued for its relatively large footprint, brought to light the quintessential balance between functionality and aesthetics. The recommendation for custom-designed components wasn't just about reducing size but about optimizing user interaction, portability, and overall product appeal.

The exploration didn't stop at remediation of the present but extended into the possibilities of the future. The Smart Cup Coaster's potential integration into the IoT ecosystem marks a paradigm shift from a standalone device to a cog in a much larger, interconnected mechanism. Its capability to contribute to data analytics offers a deeper dive into consumer behaviour, opening avenues for personalized user experiences. The envisioned hydration monitoring feature epitomizes the transformative impact of technology on health and wellness.

In essence, the Smart Cup Coaster project is a testament to the progressive relationship between human needs and technological advancement. Each limitation navigated, and innovation implemented, paved the way for a product that promises to be more than a convenient gadget but a lifestyle companion. As we look ahead, the scope for further development is vast, from integration with broader smart systems to becoming an active participant in promoting users' well-being. The journey forward is rife with opportunities for enhancement, innovation, and the redefinition of how we perceive and interact with the technology that permeates our daily lives.

10 REFERENCES

Hot drinks consumed by Australians in an average week by Generation. (2017). Roy Morgan. Retrieved October 9, 2023, from https://roymorgan-cms-dev.s3.ap-southeast-2.amazonaws.com/wp-content/uploads/2022/06/07050348/8054-c1.png

Williams, L. E., & Bargh, J. A. (2008). Experiencing physical warmth promotes interpersonal warmth. Science, 322(5901), 606-607. https://doi.org/10.1126/science.1162548

McVean, A. (2023, July 7). Drink hot drinks in hot weather to cool down faster. Office for Science and Society - McGill University. https://www.mcgill.ca/oss/article/did-you-know/drink-hot-drinks-hot-weather-cool-down-faster

Yasar, K. (2023, August). Smart home. IoT Agenda. https://www.techtarget.com/iotagenda/definition/smart-home-or-building

11 Code implementation

Notable libraries that need to be installed are:

- RemoteXY.h
- WiFi.h
- DFRobot_MLX90614.h
- HTTPClient.h

Note: no assembly hardware documentation was provided as it is assumed for the user that the product will arrived preassembled and will work like a plug and play.

```
/*
   -- smartCoasterNew --

   This source code of graphical user interface
   has been generated automatically by RemoteXY editor.
   To compile this code using RemoteXY library 3.1.11 or later version
   download by link http://remotexy.com/en/library/
   To connect using RemoteXY mobile app by link http://remotexy.com/en/download/
     - for ANDROID 4.11.4 or later version;
     - for iOS 1.9.1 or later version;

   This source code is free software; you can redistribute it and/or
   modify it under the terms of the GNU Lesser General Public
   License as published by the Free Software Foundation; either
   version 2.1 of the License, or (at your option) any later version.
*/

//////////////////////////////////////////////
//        RemoteXY include library          //
//////////////////////////////////////////////

// RemoteXY select connection mode and include library
#define REMOTEXY_MODE__ESP32CORE_WIFI_POINT
#include <WiFi.h>

#include <RemoteXY.h>

// RemoteXY connection settings
#define REMOTEXY_WIFI_SSID "SmartCoaster2"
#define REMOTEXY_WIFI_PASSWORD "88880000"
#define REMOTEXY_SERVER_PORT 6377


// RemoteXY configurate
#pragma pack(push, 1)
uint8_t RemoteXY_CONF[] =   // 268 bytes
  { 255,4,0,44,0,5,1,16,16,5,67,5,13,23,39,15,1,8,29,11,
  3,3,43,24,12,32,2,64,27,1,2,36,65,12,12,2,50,31,43,0,
  1,2,16,65,12,12,2,205,16,45,0,67,5,24,23,13,5,2,192,30,
  11,129,0,12,16,40,4,1,65,67,117,114,114,101,110,116,32,84,101,109,
  112,101,114,97,116,117,114,101,0,67,5,24,37,13,5,2,78,30,11,67,
  5,24,51,13,5,2,51,30,11,129,0,9,23,11,5,2,192,67,111,108,
```

```
  100,0,129,0,8,37,13,5,2,78,87,97,114,109,0,129,0,10,51,8,
  5,2,51,72,111,116,0,129,0,5,3,54,8,0,65,83,109,97,114,116,
  32,67,111,97,115,116,101,114,0,129,0,16,45,32,6,1,65,78,111,116,
  105,102,105,99,97,116,105,111,110,0,131,1,8,90,20,7,1,2,31,77,
  97,105,110,0,131,0,35,90,20,7,2,2,31,83,101,116,116,105,110,103,
  0,3,131,11,63,41,15,1,2,26,129,0,9,54,13,6,1,178,67,111,
  108,100,0,129,0,25,54,16,6,1,64,87,97,114,109,0,129,0,44,54,
  9,6,1,51,72,111,116,0 };

// this structure defines all the variables and events of your control interface
struct {

    // input variables
  uint8_t TempSwitch; // =0 if select position A, =1 if position B, =2 if position
C, ...
  uint8_t add; // =1 if button pressed, else =0
  uint8_t minus; // =1 if button pressed, else =0
  uint8_t SMSswitch; // =0 if select position A, =1 if position B, =2 if position
C, ...

    // output variables
  char TempDisplay[11];  // string UTF8 end zero
  char coldTemp[11];  // string UTF8 end zero
  char warmTemp[11];  // string UTF8 end zero
  char hotTemp[11];  // string UTF8 end zero

    // other variable
  uint8_t connect_flag;  // =1 if wire connected, else =0

} RemoteXY;
#pragma pack(pop)

/////////////////////////////////////////////
//           END RemoteXY include          //
/////////////////////////////////////////////
// Import necessary libraries
#include <HTTPClient.h>
#include <DFRobot_MLX90614.h>

// WiFi and Pushbullet Setup
const char* ssid = "IOTproject";        // WiFi SSID (network name)
const char* password = "hahaha123";     // WiFi password
const char* apiKey = <insert you own api key>;

// Define pins for LEDs, distance sensor and button
int yellowLED = 2;
int redLED = 13;
int blueLED = 14;
int DistTrigger = 23;                  // Ultrasonic distance sensor trigger pin
int DistEcho = 4;                      // Ultrasonic distance sensor echo pin
int button = 35;
```

```
int buttonState = 0;
bool buttonFlag = false;

// Default temperature settings (in Celsius)
float cold = 30.0;
float warm = 50.0;
float hot = 70.0;
float distance = 0.0;
bool distanceFlag = false;

// Initialize the IR temperature sensor
DFRobot_MLX90614_I2C sensor;

void setup() {
  RemoteXY_Init();                      // Initialize RemoteXY (an external library
for remote control)
  // Set the pin modes
  pinMode(yellowLED, OUTPUT);
  pinMode(redLED, OUTPUT);
  pinMode(blueLED, OUTPUT);
  pinMode(DistTrigger, OUTPUT);
  pinMode(DistEcho, INPUT);
  pinMode(button, INPUT);

  // Convert default temperature settings to strings and save to RemoteXY
  dtostrf(cold, 0, 1, RemoteXY.coldTemp);
  dtostrf(warm, 0, 1, RemoteXY.warmTemp);
  dtostrf(hot, 0, 1, RemoteXY.hotTemp);

  Serial.begin(9600);                   // Begin Serial communication

  // Reset the thermal sensor
  sensor.enterSleepMode();
  RemoteXY_delay(50);
  sensor.enterSleepMode(false);
  RemoteXY_delay(200);
}

float readDistanceSensor() {
  // Send a pulse from the trigger pin and measure the time for the echo pin to
receive the reflected signal
  digitalWrite(DistTrigger, LOW);
  RemoteXY_delay(2);
  digitalWrite(DistTrigger, HIGH);
  RemoteXY_delay(10);
  digitalWrite(DistTrigger, LOW);

  float duration = pulseIn(DistEcho, HIGH);
  float distance = duration * 0.0343 / 2.0;  // Convert duration to distance in
centimeters
  return distance;
```

```
}

void initializeWiFi() {
  // Establish WiFi connection
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    RemoteXY_delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");
}

void disconnectWiFi() {
  WiFi.disconnect();                      // Disconnect from the WiFi network
}

void sendNotification(String type) {
  // Send a notification to Pushbullet service
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin("https://api.pushbullet.com/v2/pushes");
    http.addHeader("Authorization", "Bearer " + String(apiKey));
    http.addHeader("Content-Type", "application/json");

    String json = "{\"type\": \"note\", \"title\": \"SMART COASTER Notification\",
\"body\": \"Your " + type + " drink is ready!!\"}";
    int httpResponseCode = http.POST(json);
    if (httpResponseCode > 0) {
      String response = http.getString();
      Serial.println(httpResponseCode);
      Serial.println(response);
    } else {
      Serial.println("Error on HTTP request");
    }
    http.end();
  }
}

void loop() {
  RemoteXY_Handler();                     // Update RemoteXY states

  // Read the distance sensor
  distance = readDistanceSensor();
  if (distance < 5 || distance > 750) {
    distanceFlag = true;
  } else {
    distanceFlag = false;
  }

  // Read temperature from the IR sensor
  float temp = sensor.getObjectTempCelsius();
```

```
  RemoteXY_delay(100);

  // Check button press and light up all LEDs when pressed
  buttonState = digitalRead(button);
  if (buttonState == HIGH) {
    buttonFlag = false;
    Serial.println("Button Pressed!");
    digitalWrite(blueLED, HIGH);
    digitalWrite(redLED, HIGH);
    digitalWrite(yellowLED, HIGH);
    RemoteXY_delay(100);
    digitalWrite(blueLED, LOW);
    digitalWrite(redLED, LOW);
    digitalWrite(yellowLED, LOW);
  }

  // Check distance and temperature to send notifications
  if (!buttonFlag && distanceFlag) {
    // ... (the code checks for various temperature ranges and sends appropriate
notifications)

    // Adjust the temperature settings remotely
    // ... (the code checks for RemoteXY inputs and adjusts temperature settings)

    // LED indications based on temperature readings
    if (distanceFlag) {
      if (temp > warm) {
        digitalWrite(redLED, HIGH);      // Hot temperature indication
        digitalWrite(blueLED, LOW);
        digitalWrite(yellowLED, LOW);
      } else if (temp <= warm && temp > cold) {
        digitalWrite(redLED, LOW);
        digitalWrite(yellowLED, HIGH);   // Warm temperature indication
        digitalWrite(blueLED, LOW);
      } else if (temp <= cold) {
        digitalWrite(redLED, LOW);
        digitalWrite(yellowLED, LOW);
        digitalWrite(blueLED, HIGH);    // Cold temperature indication
      }
    } else {
      // Turn off all LEDs if no cup is detected
      digitalWrite(blueLED, LOW);
      digitalWrite(redLED, LOW);
      digitalWrite(yellowLED, LOW);
    }
  }
}
```