# Home Kitchen Robot - UR10e

*Akansha Bhatia, Salma Riaz, Reia Menezes — MSc in Robotics, Middlesex University Dubai.*

***Abstract*—The Home Kitchen Robot is a collaborative robotic system that automates routine cooking chores; this paper explains its design and implementation. The UR10e robotic arm, computer vision, IoT devices, and a user-friendly Python-Tkinter GUI guide the user through the cooking process, from switching on an electric gas stove to dish completion. A RoboDK simulation, running with a working GUI, is presented, showing the control system architecture, computer application logic, sensor integration, and task sequencing. The system leverages real-time data processing, cloud storage, and embedded control logic, creating a scalable, intelligent solution that improves user convenience, safety, and cooking precision. Not only does this study confirm the feasibility of robotic kitchen automation, it also sets out a clear roadmap towards real-world deployment and future expansion.**

***Index Terms*—Home automation, collaborative robot, UR10e, RoboDK, IoT kitchen, robotic cooking, Python GUI, OpenCV, smart stove, control system architecture, user interface, embedded systems, kitchen robot, simulation, food tech.**

## I. Introduction

The increasing demand for automation in everyday environments has led to the growing use of collaborative robots in tasks that were once considered entirely human. These robots are currently being developed to help with repetitive, precise, or time-consuming tasks in both households and manufacturing environments. Particularly, kitchens can offer a perfect environment for the use of robotic systems that can help with day to day cooking. The incorporation of robotics into everyday cooking tasks has the potential to completely alter how families prepare meals while maintaining the standard and appeal of home-cooked meals. This project explores the possibility of creating a Home Kitchen Robot that integrates modern robotics with complex control systems, computer vision capabilities, and user-friendly interfaces in order to develop a complete solution that handles from ingredient selection, preparation, to a complete ready to eat dish all on its own. The paper addresses the mentioned robot systems architectural design, the choice and integration of important hardware and software elements, and the creation and explanation of a working simulation that shows how the idea is used in real-world scenarios. By reclaiming time, enhancing access for a wide range of users, and providing reliable, high-quality meals that meet dietary requirements and personal preferences, we hope to show not only the technical feasibility of kitchen automation but also its potential for transforming everyday cooking activities.

### A. Need for the Home Kitchen Robot

The need for automated kitchen solutions has increased significantly in today's fast-paced culture as a result of multiple converging causes. As the elderly population seeks independence in everyday tasks, upcoming generation lacking in basic cooking skills, time constraints seen among working professionals, and the growing interest in smart home technologies have all combined to provide an ideal environment for innovation in kitchen automation. Meals cooked by the robot can also be tailored to meet specific health needs by programming the intelligent system to take into account of the dietary restrictions, allergies, and nutritional requirements. The system allows users from constant kitchen monitoring by automating the entire cooking process, enabling them to engage in other responsibilities while meals are being cooked.

This paper was produced for the module PDE4435 - Robot Integration System.

Date of Submission: April 11, 2025

## II. Control System Design

The Home Kitchen Robot utilizes various hardware and software related components that communicate with a centralized core control system. The main processing unit of the control system would be the embedded core built with the control system as a circuit board [1]. By collecting inputs from all hardware and sensors, this embedded core conducts the processing and decision making for the system, along with communicating the next action to that specific component to produce an output [1]. The embedded core can be divided into 3 parts as follows:

- Device Driver Layer: This layer communicates with sensors to receive information along with sending signals to actuators to perform an action [1]. FreeRTOS is used for real time task management and execution along with handling safety measures, such as emergency stops, that need immediately interrupt the process [1]. The coding language used would be C/C++ for communicating with hardware components [2]. Typically, C/C++ is preferred for robotics control system due to having reliable and faster real time execution compared to other languages, such as Python [2].

- Middleware layer: This layer ensure communication between hardware and software to handle the sequence of task execution and messages between different components [1]. ROS (Robot Operating System) is used for managing communication between core system and robotics arm along with controlling the movements required [3]. FreeRTOS manages the timing of when the robotics arm should be moved along with sequence of task related to monitoring reading from sensors and other actuator components [1].

- Functional Module Layer: This layer focuses on processing and decision making of the control system

along with holding logic related to step-by-step task execution for each recipe database [1]. Furthermore, detecting errors and apply responsive action is handled in this layer to manage situations, such as low ingredient or abnormal weight in container [1]. For this layer, ROS would be used for complex decision making, such as detecting item is picked through computer vision and camera, along with being based on Python scripts that set the logic for decisions.

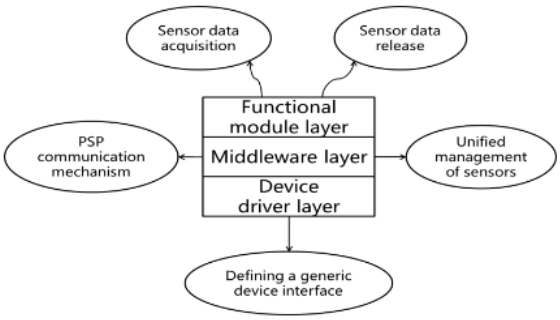


Fig. 1: Control System Design Structure [1]

Apart from the embedded core, the control system design can be broken down into 3 key sections with which the core control system communicates and are as follows:

*A. User Interface (Android Tablet & Cloud Server – Core System Control*

- The tablet provides a user interface in which the end user can provide input in terms of recipe section to start automating cooking or other input related to their user preferences, such as favorite dishes or places to eat. For recipe selection, end users can also provide information on portion size and heat level to control the dish being created. In addition, the emergency stop button is available to stop cooking immediately.

- This tablet would have an android-based operating system with Android version 12 or higher to be able to communicate with hardware components in the control system. The front end of the android application would show the GUI (Graphical User Interface) and be coded in Java or Kotlin depending on the compatibility of the SDK of the hardware component (Software Development Kit) [4]. The back end of application would be the embedded core that communicates with various components as mentioned in the above section.

- User data would be stored within a cloud server to hold vital information about recipes and user preferences along with system logs. In addition, this server would be managing the data in terms of sending the relevant data to the related user's robot ensuring only their data is being displayed. This management of data is essential for maintaining user's history and favorites along with creating a personalized experience for them. Furthermore, support team members can view the control system status through viewing error logs and other system logs shared with cloud server.

- The storage and management of this data would be conducted by AWS (Amazon Web Services) due to security practices and worldwide distribution of servers [5].For security, AWS uses AES-256 encryption when sending and receiving data through an API bridge [6]. When data is being sent from robot to server, it is first encrypted into an indecipherable format utilizing the 256-bit secret key [6]. Upon the data reaching the authorized user, the secret key is used to decrypt the data to change the data back to the original readable format [6]. Therefore, if an unauthorized user tried to get access to the data while it was passed through API, they would receive the unreadable encrypted format along with being enabled to decrypt it due to not having the secret key [6].

- The communication protocol used would be HTTPS and MQTT for data being transferred over the network. HTTPS would be used for sending data upon a specific request received for it and be able to send larger sum of data compared to MQTT [7]. For example, it would be used when tablet request for all fields related to recipe number 12 include descriptions and images resulting in HTTPS being used to handle the large data transfer. In addition, it is encrypted with SSL certificate to ensure data is not being viewed nor being interfered as it passes through the network [7]. On the other hand, MQTT is used for sending data continuously to provide essential updates about the control system [7]. For example, it would be used for sending notifications of actions complete by robotics arms, such as picked up ingredient 1 or start stove on high heat. MQTT transfers data that is light weight resulting data transfer being faster than HTTPS and ideal for IoT communication between different components in control system [7].

*B. Robotics Arm Control (UR10e Robotics Arm, Camera & Tool Box – Core System Control)*

- UR10e was the model of robotics arm used to automate the cooking process. The robot has a payload of 12.5 kgs which is adequate for lifting not only the ingredients but also the tools [8]. The reach of robotics arm is 1300 mm in which 2 robotics arm would have a total reach of 2600 mm [8] suitable for a kitchen size of 240 cm to 250 cm.

- A camera is attached to the robotics arms to verify whether the correct ingredient is picked and to ensure the ingredient is placed within the pan [9]. This verification can be done using a computer vision AI model in which the real time video surveillance is captures and sent to

core control system [9]. Within the core control system, the functional module layer in embedded core would hold the OpenCV python script and conduct the image processing along with an output predication [1]. Upon verification received, the core control system would communicate the next control action to the robotics arm [9].

- To automate the cooking process, the robotic arm would be the main component to execute the cooking from picking up ingredients and utensils to performing the cooking motion. In addition, a camera is attached to the robotic arm with computer vision model to verify whether the correct ingredient has been picked along with being placed in the accurate position in pan. Furthermore, tool box holds the all the tools required, such as gripper or spatula, allowing the robotics arm to perform variety of cooking task by changing the tool. Overall, the core control system manages the timing and task execution based on data received from camera along with sending signals to control the robotics arm and tool box.

- For the tool box, the core control system would need to communicate with the box to allow the robotics arm to change the tool the required [10]. First, the robotics arm would have a robot coupling which is attached to robotics arm's end effector [10]. This robot coupling has a locking and unlocking mechanism which allowing the tool to become attached or detached from the arm [10]. It would also include electrical contacts to power tools that have motion, such as rotating motion for whisking cooking motion [10]. Second, tool box would be shaped and positioned on the kitchen counter to ensure adequate space is present for robotic arm to position on top and on point for changing the tool [10]. To ensure the tool has been aligned accurately, force sensors are utilized to capture data on pressure applied when robot is pushing down on tool [10]. If the force data collected matches the ideal force data stored in core control system, this situation would indicate that the force is accurately coming from center and well balance resulting in tool being accurately aligned [10]. If there is a difference in force data, the robotics arm would need to conduct adjustment slight rotating or moving the arm until the correct force data is reached [10]. Once the alignment is completed, the robotics arm can lock to attach it to the arm. For detaching the tool, the alignment is more simplified as the robotics arm would need to ensure the right point is reached and unlock the tool to place the tool back in the correct slot [10].

*C. Ingredients Management (Customized Cupboards, IoT stove, Thermostat & Weight Scale – Core Control System)*

- IoT stove would be fully automatic in which the core control system would be turning it on and off. According to the sequence of task required, the core control system would manage the timing in which the stove needs to turn on by sending a signal to switch resulting the electrical circuit to be close [11]. As result, power would flow through the circuit powering the electric stove to be turned on [11]. Once the cooking task is completed, the core control system would send a signal to switch to open the circuit resulting in power flow to be interrupted and stove to be turned off [11]. Furthermore, the heat level is control by the duty cycle using Pulse Width Modulation (PWM) which controls the duration of current is each cycle [11]. For example, on low heat, the duty cycle would be set to have 30% of time to be powered on and 70% to be powered off for cooling [11]. This duty cycle is managed using short interval of time resulting in low heat would have 3 seconds on and 7 seconds off within a 10 seconds duration [11]. In addition, medium heat would have 60% being powered on and 40% being powered off while high heat would have 100% powered on [11].

- Customized cupboards are used for this control system to manage the opening and closing of containers according to robotics arms action along with inventory management, such as monitoring weights and expiry dates [12]. Opening and closing of containers is fully automate and managed by core control system to ensure that system doesn't require assistant from end user during the cooking process [12]. In addition, the robotics arm would be limited to only pick from the container that is open ensuring that the right ingredient is taken for cooking [12].

- In addition, when the end users filled the ingredient container, the core control system would request for expiry date of that item through the tablet [12]. This feature is to ensure rotten ingredients are not used in the cooking process and dish being made is safe for consumption [12]. In addition, the core control system would give reminders for ingredients that are close to expiry date along with restricting the container to be opened for expired ingredients [12].

- Furthermore, the cupboards are equipped with weight sensors to detect the addition or removal of ingredient along with any abnormalities [13]. The sensors are located at bottom of each individual container in the smart cupboard to measure the weight of ingredient placed and match it with the desired weight in kitchen to detect any abnormalities [13]. For example, the container for chicken would have a desired weight of 1 to 2 kgs resulting in a weight less than desired weight indicates an issue in end user placing the ingredient [13].

- Thermostat is located within the cupboard for refrigerated ingredients that needs to be stored within a colder temperature [14]. Regular checks on temperature is conducted to ensure that it is within the desired range

set in the core control system [14]. If the temperature goes out of the desired range, the thermostat adjusts the temperature to ensure the cupboard reaches the desired range [14].

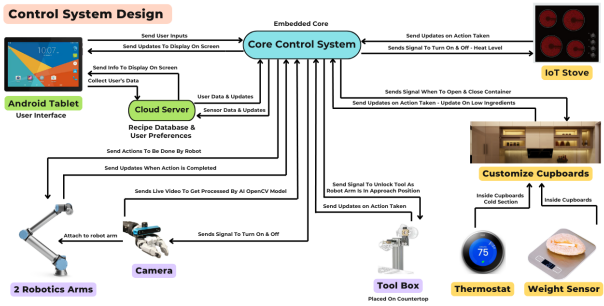


Fig. 2: Control System Design For Home Kitchen Robot

## III. Selection of Components

Currently, the KitchenBot Home system is software-driven and runs on the RoboDK simulator integrated with a Tkinter-based GUI written in Python. Although the simulation demonstrates the logic flow and robotic control thoroughly, executing it in practice requires a careful choice of physical parts and a precise integration architecture. This part describes the configuration of the simulation in detail, as well as the expected hardware parts and interconnections for practical implementation.

### A. Software-Based Implementation

The system is built around two main software components:

- The Graphical User Interface (GUI) was created using Python's Tkinter library.
- The RoboDK robotic simulation environment is accessed via the RoboLink API.

This GUI features multiple interactive screens, such as:

- Welcome Screen
- Control Panel
- Recipe Database
- Timer Interface
- Places to Eat Explorer
- Favorites Menu

These components communicate through Python bindings over a local TCP/IP interface using the RoboDK API. Each screen is triggered using Tkinter button widgets and dynamically clears and repopulates the window with relevant content. User actions such as pressing "Start Cooking" or "Apply Filters" invoke specific Python callback functions.

*1) RoboDK Integration:* When, for instance, a user selects the Omelette recipe and confirms the action, the GUI calls the runrecipe function. Internally, the following steps occur:

Connection Establishment:

```
from robodk import robolink
RDK = robolink.Robolink()
```

Fig. 3: Shows the code snippet

Figure 3 shows the connection between the GUI and the RoboDK API. This establishes a session with the RoboDK station running locally (by default on localhost:20500).

Robot Identification:

The robot item UR10e is selected using:
"robot = RDK.item('UR10e',robolink.ITEM-TYPE-ROBOT)"
The system verifies that this robot is active and responsive.

GUI-to-Robot Timing Synchronization:

Each command execution is synchronously handled via WaitFinished(), ensuring that the robot completes one action before the next begins. The progress is visualized in a separate Tkinter popup window with a progress bar and status label, allowing real-time tracking of the recipe.

- Emergency stops are also handled through GUI buttons, which invoke RDK.StopAll() to halt all motion programs and reset the robot.

*2) Simulated User-Guided Input:* The GUI includes a stove prompt, where users must select heat levels ("Low", "Medium", "High"). In the simulation, this input is symbolic and triggers the cooking sequence immediately. In real-world deployment, this step would be replaced by sensor feedback and GPIO verification. However, in simulation:

- The user's heat selection is simply stored as a variable and passed along to run-recipe-in-robodk() for visual confirmation.

*3) Data Flow and Event Handling:* The entire system is event-driven:

- Each button click is bound to a Python function (command=...) that carries out logical procedures.
- RoboDK is a passive execution engine—it only acts when the Python GUI issues a command.
- Since the simulation lacks physical feedback, all logical conditions (like "stove is hot") are simulated as successful upon user confirmation.

```
def emergency_kill():
    try:
        RDK = robolink.Robolink()
        RDK.StopAll()  # Stops all running programs in RoboDK
        messagebox.showwarning( title: "Emergency Stop",  message: "All robot activities have been halted!")
    except Exception as e:
        messagebox.showerror( title: "Error",  message: f"Failed to execute emergency stop:\n{e}")
```

Fig. 4: code snippet showing the emergency stop function as an event.

*4) Directory and File Usage:*

- Recipe and favourite data are stored as local JSON files (recipes.json, favourites.json) and dynamically updated by the GUI using Python's json module.
- These allow persistence across GUI sessions and simulate a user database or local storage system.

*5) Runtime Environment:*

- The full application is executed within a PyCharm virtual environment on Windows.
- Python 3.12 is used, along with modules such as tkinter, json, robolink, and webbrowser.

*6) Security and Fail-Safe in Simulation:*

- The robot interface includes a simulated Emergency Stop button that interrupts any ongoing action.

If RoboDK fails to locate a named program, an error dialog is displayed via:

messagebox.showerror("RoboDK Error", f"Program 'prog-name' not found")

- This helps test robustness before moving to hardware.

### *B. Real-World Component Selection and Sensors*

To implement the KitchenBot within a practical kitchen setting, the following hardware components will be utilized:

- Robotic Arm: The Universal Robots UR10e is a collaborative industrial robotic arm that offers 6 degrees of freedom, a payload capacity of 12.5 kg, and a reach of 1300 mm. It is equipped with force-torque sensing capabilities to ensure safe interactions.
- Controller: The integrated controller of the UR10e manages motion planning and real-time operations, connecting to a local processing unit through Ethernet or USB.
- Processing Unit: A Raspberry Pi 5 with 8GB of RAM serves as the host for the Python GUI application. It operates on a Linux operating system and interfaces with the robot controller via Ethernet, utilizing TCP/IP and the RoboDK API.
- Electric Stove: The Bosch Serie 6 Ceramic Hob (model: PKE611CA1E) measures 60 cm and offers three heating levels (low, medium, and high) with digital control interfaces.
- Thermal Sensors: An MLX90640 thermal imaging array is positioned above the stove to monitor surface temperatures. It transmits temperature data in real time to the Raspberry Pi via I²C, enabling the graphical user interface (GUI) to display this information.
- Limit Switches: These sensors are mounted on the stove knob to detect any movement associated with adjustments to the heat level. They are connected to the Raspberry Pi through GPIO.
- Force Sensors: To enhance the realism and safety of interactions, a Tekscan FlexiForce sensor has been placed beneath the pan to track pressure variations during stirring and flipping activities.

*1) Integration and Communication Logic:*

- Powering On the Robot: The UR10e robot has a direct 230V AC power source connection through a smart relay. So, as soon as the GUI gets activated, the smart relay comes on and powers up the robot. A GPIO pin from Raspberry Pi is used to control this relay, which in turn enables software-based activation for the robot.
- Communication Between GUI and Robot: The communication link between the robot and the GUI is created using Python API provided by RoboDK over an Ethernet connection (TCP/IP). Every button on the GUI has a corresponding programmable command stored within the RoboDK station, like "Attach Tool" or "Stirring Motion." These commands are executed using the associated computer command RunProgram(), and its completion is monitored with WaitFinished() so that proper automation can take place.
- Recipe Execution: After the user selects a recipe that he wants, the omelette, in this case, the GUI asks the user to tell what heat setting they prefer. Once the user selects either Low, Medium, or High, the system should wait for confirmation from the stove about its readiness.

*2) Heat Level:* The robot switches on the heat level based on the recipe chosen. For instance, if the robot needs to cook an omelette, it will choose the heat level as 'low'; however, if the robot needs to boil water, it will automatically choose the heat level as 'high.'

*3) Temperature Verification:*

- The MLX90640 thermal sensor provides continuous monitoring of the temperature on the stove surface.

- When the desired temperature corresponding to the chosen heat setting is attained (for instance, approximately 120°C for Medium), the graphical user interface (GUI) automatically initiates the cooking sequence of the robot.

*4) Cooking Actuation:* The robotic arm executes tasks in a predetermined order—such as adding oil, stirring, and folding the omelette—while maintaining verified heat levels and temperatures. A FlexiForce sensor positioned beneath the pan monitors force application, ensuring it remains within safe limits to prevent any movement or damage to the pan.

*5) Feedback Loop:* Throughout the cooking process, sensor data regarding temperature and force is continuously tracked. Any irregularities, such as pan displacement or excessive heat, activate an emergency stop via the graphical user interface (GUI), ceasing all robotic functions.

*6) Compatibility Mapping:*

- The chosen stove model, Bosch PKE611CA1E, is selected for its flat surface and reliable heat zones, which provide consistent temperature feedback.
- The UR10e robot is programmed with waypoints that correspond to the stove's configuration, facilitating safe operation within the pan's vicinity.
- The Raspberry Pi serves as the central hub for real-time decision-making, ensuring compatibility through I²C for sensor data and GPIO for stove status.

This well-integrated system exemplifies a smooth connection between simulated software environments and practical robotic kitchen automation. Each component has been carefully selected and connected to guarantee safety, precision, and ease of use.

## IV. Building Simulation

### A. Setup

The required software programs were thoughtfully selected, set up, and configured in order to execute a working simulation of the robotic cooking process. For a realistic depiction of the home kitchen cooking robot process, the preparation required setting up the virtual environment, installing the necessary tools, and defining key object placements.

- Installed OpenCV, Python, and RoboDK to enable vision-based and simulation-based processes.
- Based on the reach and payload required for efficient task execution, the robot type UR10e was chosen from the RoboDK library.
- Created a realistic kitchen atmosphere by setting up the floor and table along with 2 prism shapes to act as shelves where 1 holds all the ingredients and the other for utensils like bowls.
- The robot arm is placed on one corner of the table while the representation for a stove (square and circle) is placed on the opposite corner, along with three small cubes to act as buttons for flame intensity (low, medium, high).
- Next to the stove a tool box is placed on the table, along with different grippers which acts as utensils like the spatula or a spoon.
- Ingredients are depicted by arranging various geometric shapes in a range of colors on the shelves.
- Activated and mounted a simulated camera onto the robot's gripper for ingredient detection.
- Taught robot movement points, such as pick position, approach position, button pressing, placing position, etc.
- Programmed task sequences such as tool change, stirring motion, folding motion, pick-and-place actions.
- Integrated Python and OpenCV for color detection using the camera; logic implemented here is different colors represent different ingredients.

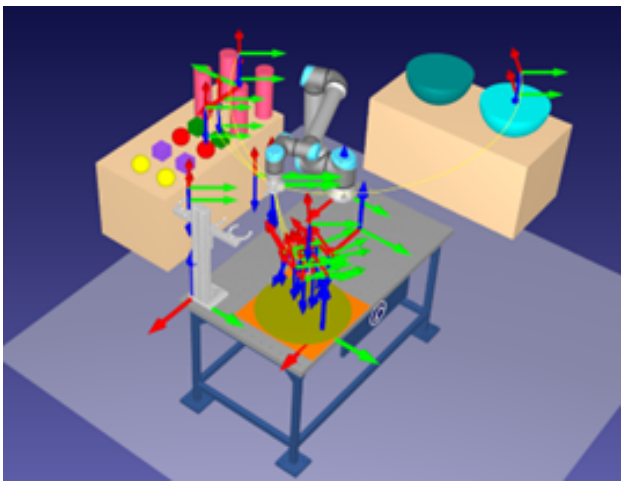

Fig. 5: Displays the RoboDK Simulation layout

- Installed Tkinter and necessary dependencies for the creation of GUI.
- Designed a user-friendly GUI layout, optimizing colors, symbols, images, and buttons for better visualization.
- Established a connection between the Python GUI and RoboDK, enabling simulation control through the GUI.
- Combined all programmed tasks to visualize the kitchen robot performing the cooking activities in a simulated environment.

### B. Working

From choosing a recipe to observing the robotic system carry out cooking activities, the user is guided by the user-friendly interface of the robotic cooking system. Every stage of the procedure has been systematically carried out to guarantee smooth interaction between the RoboDK, the GUI, and the robot's preprogrammed tasks.

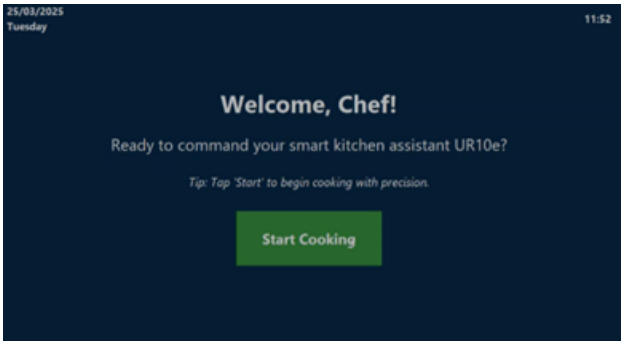


Fig. 6: Shows the home screen GUI layout

- The GUI presents the user with a "Start" option when the program is launched.

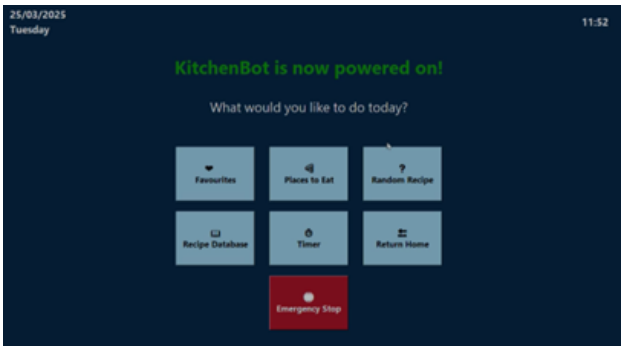


Fig. 7: Shows the user control panel

- The user is presented with the question "What would you like to do today?" along with mutliple options such as "Favourites", "Recipe Database" and many more.
- When selecting the "Recipe Database" option, recipes that are preprogrammed and stored are displayed.
- Once a recipe is chosen, the connected RoboDK simulation initiates the robotic cooking process.
- The robot switches on the stove and approaches the buttons on the stove to adjust the flame intensity based on the recipe selected(e.g., high flame for cooking rice, medium for boiling water).
- The robot selects the required tool from the toolbox.
- The bowl is picked from the shelf and placed onto the stove.
- The robot moves to the ingredient shelf, where each colored object represents a different ingredient.

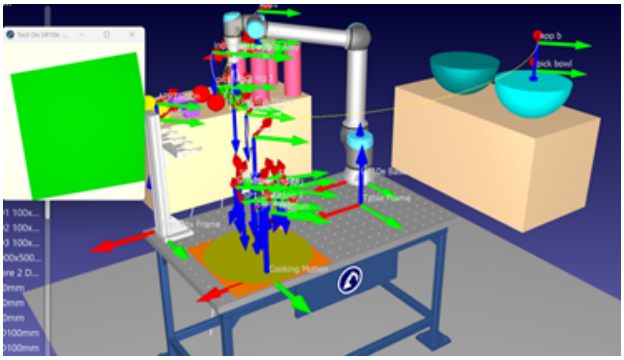

Fig. 8: The integration of OpenCV in RoboDK

- The robot then moves to the top of the required ingredient and pauses, now the camera detects the color using Python and OpenCV. This is done to represent the logic that when the robot moves to pick the ingredient it checks if it is the correct ingredient required for the dish, if not then the robot stops the process and returns to home position, indicating the user to place the ingredients correctly in the designated area on the shelf.
- Once the ingredient is recognized (OpenCV has detected the color accurately), the robot picks the ingredient and places it into the bowl.
- For bottle-shaped objects like the cylinder, the robot does not place them in the bowl but hovers over it, simulating liquid dispensing.
- If required, the robot switches tools and performs actions such as stirring and folding.
- Once the entire process is completed the stove is switched off and the dish is now ready to be served.

### C. Testing

Extensive testing was done to guarantee the simulation's accuracy and dependability. To reduce errors and guarantee a seamless workflow, each operation was verified to ensure an accurate representation of the simulation of the home kitchen robot.

- Checked for any object collisions to ensure the smooth movement of the robot arm and all associated objects.
- Verified that all the points taught for each position are accurate for the overall successful execution of the program.
- Ensured that the simulated camera starts and operates well within RoboDK.

- Confirmed that the intergration of OpenCV is accurate in image processing and color detection.
- Ensured a stable connection between the Python GUI and RoboDK.
- Confirmed that all preprogrammed recipe sequences follows the correct procedural flow without errors.
- Verified the emergency stop button terminates the entire process immediately when triggered.
- Conducted step-by-step testing to confirm individual program performance.
- Executed multiple test runs, simulating different conditions to identify and resolve potential errors.

## V. Challenges

Only very few challenges arose during the development of the robotic cooking simulation, careful problem-solving and choosing alternatives guaranteed the project's successful completion.

The restricted features of the RoboDK free version license was one of the main obstacles. The inability to add two robots to the simulation was one of the advanced features that were limited since a full license was not available. This restriction had an effect on the original concept of placing 2 robots which would have increased productivity and efficiency . As an alternative to the problem, the simulation was designed using a single robotic arm, ensuring all necessary operations could still be executed within the constraints of the free version.

## VI. Conclusion

Advancements within IoT and smart home technologies allows end users to automate various task within their home. Home Kitchen Robot aims to be part of advancement through automating the cooking process to deliver a home cooked meal for end users. This report explores the control system designed for this automation along with the various components required.

To start, the control system design section discussed the hardware and software component along with path of communication with each other. This section discusses how key component, such as embedded core and IoT devices, operate along with key technical details, such as communication protocols and AI model for decision making. Next, the selection of components was conducted based on the RobotDk simulation created along with describing the real world application of these components. This section provided more information of the compatibility mapping and feedback loop created to ensure the system is well integration with the various component. The final section would be building the simulation which explore the setup and process of creating the Home Kitchen Robot RobotDk simulation. The section explores the GUI created and programs for operation of robotics arm along with the testing conducted in simulation.

## VII. Future Implications

To further develop this project, research can be conducted on adding a feature for cleaning the dishes after the cooking has been completed. This feature would involve the robotics arm placing the cooking food on a plate to be served to the end user along with sending the pan to be clean. Dishwasher component can be added so that the robotics arm can simply place the pan in that component for cleaning. In addition, core control system can communicate with the robotics arm when the dish is cleaned to ensure the robotics arm picks up the pan from dishwasher area and places it in the right location.

Furthermore, the dishwasher component can also include a special compartment to place the tools used for cooking. This compartment would ensure that the tools also get cleaned as they are in contact with food when conducting the cooking process. Similar to the pan cleaning process, the core control system can also indicate the robotics arm when tool cleaning is completed to ensure it is placed back in tool box.

## References


[1] L. Shao, C. Wang, C. Chu, Y. Song, H. Hu, Y. Yang, W. Fei, and X. He, "Design and implementation of real-time robot operating system based on freertos," in *Journal of Physics: Conference Series*, vol. 1449, no. 1. IOP Publishing, 2020, p. 012115.

[2] S. Y. Cho, R. Delgado, and B. W. Choi, "Feasibility study for a python-based embedded real-time control system," *Electronics*, vol. 12, no. 6, p. 1426, 2023.

[3] M. Singh, J. Kapukotuwa, E. L. S. Gouveia, E. Fuenmayor, Y. Qiao, N. Murry, and D. Devine, "Unity and ros as a digital and communication layer for digital twin application: Case study of robotic arm in a smart manufacturing cell," *Sensors*, vol. 24, no. 17, p. 5680, 2024.

[4] S. Robertus, J. O. Chandra, W. Andriyani *et al.*, "A comparative study of java and kotlin for android mobile application development," 2020.

[5] N. Kewate, A. Raut, M. Dubekar, Y. Raut, and A. Patil, "A review on aws-cloud computing technology," *International Journal for Research in Applied Science and Engineering Technology*, vol. 10, no. 1, pp. 258–263, 2022.

[6] C. Manthiramoorthy, K. M. S. Khan *et al.*, "Comparing several encrypted cloud storage platforms," *International Journal of Mathematics, Statistics, and Computer Science*, vol. 2, pp. 44–62, 2024.

[7] S. Hong, J. Kang, and S. Kwon, "Performance comparison of http, https, and mqtt for iot applications," *International journal of advanced smart convergence*, vol. 12, no. 1, pp. 9–17, 2023.

[8] U. Robots, "UR10e Medium-sized, versatile cobot," https://www.universal-robots.com/products/ur10e/, 2025, [Accessed 06-04-2025].

[9] C. Bellinger and L. Lamarche-Cliche, "Learning visual tracking and reaching with deep reinforcement learning on a ur10e robotic arm," *arXiv preprint arXiv:2308.14652*, 2023.

[10] P. J. Leitão *et al.*, "Development and simulation of an automatic tool changer for an abb robot," 2019.

[11] M. N. K. Hamdani, I. Sulistiyowati, and S. D. Ayuni, "Automatic stove control system based on the nodemcu esp8266 microcontroller," *Journal of Electrical Technology UMY*, vol. 6, no. 2, pp. 103–111, 2022.

[12] D. Surie, H. Lindgren, and A. Qureshi, "Kitchen as-a-pal: Exploring smart objects as containers, surfaces and actuators," in *Ambient Intelligence-Software and Applications: 4th International Symposium on Ambient Intelligence (ISAmI 2013*. Springer, 2013, pp. 171–178.

[13] K. P. Amutha, C. Sethukkarasi, and R. Pitchiah, "Smart kitchen cabinet for aware home," in *SMART 2012, The First International Conference on Smart Systems, Devices and Technologies*, 2012, pp. 9–14.

[14] M. Mohammed, K. Riad, and N. Alqahtani, "Design of a smart iot-based control system for remotely managing cold storage facilities," *Sensors*, vol. 22, no. 13, p. 4680, 2022.