"WAITRON: THE SMART WAITER"

Submitted in partial fulfilment of the requirements
for the award of the degree of

BACHELOR OF ENGINEERING
IN
ELECTRONICS & TELECOMMUNICATION

Submitted by

Mr. Niranjan Vajramushti	**D19A**	**52**
Mr. Rohan Bhamre	**D19A**	**58**
Mr. Vignesh Poojary	**D19A**	**61**
Mr. Paarth Rane	**D19A**	**64**

Under the Guidance of

Dr. Rasika Naik
Assistant Professor, Department of EXTC, VESIT

UNIVERSITY OF MUMBAI
Mumbai - 400032

DEPARTMENT OF ELECTRONICS & TELECOMMUNICATION ENGINEERING

VIVEKANADA EDUCATION SOCIETY'S
INSTITUTE OF TECHNOLOGY

MUMBAI, CHEMBUR - 400074

2022-23

Project Report Approval for B.E.

Project entitled **WAITRON: THE SMART WAITER** by **Mr. Niranjan Vajramushti**, **Mr. Rohan Bhamre**, **Mr. Vignesh Poojary**, **Mr. Paarth Rane** , is approved for the Electronics and Telecommunication degree of Bachelor of Engineering.

Examiners

1. _____

2. _____

Supervisors

1. _____

2. _____

Date: _____

Place: **V. E. S. Institute of Technology, Chembur, Mumbai**

CERTIFICATE

This is to certify that **Group-A6** have completed the project report on the topic **WAITRON: THE SMART WAITER** satisfactorily in partial fulfilment of the requirements for the Bachelor's Degree of Engineering in Electronics and Telecommunication under the guidance of **Dr. Rasika Naik** during the year **2022-2023** as prescribed by University of Mumbai.

Dr. Rasika Naik
Guide

Dr. Chandansingh Rawat
Head of the Department

Dr. J. M. Nair
Principal

Examiner 1

Examiner 2

DECLARATION

We declare that this written submission represents our ideas in our words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misinterpreted or fabricated or falsified any idea/data/fact/source in my submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Mr. Niranjan Vajramushti

Mr. Rohan Bhamre

Mr. Vignesh Poojary

Mr. Paarth Rane

Date: _____

ACKNOWLEDGEMENT

We are highly indebted to **Dr. Rasika Naik** for her guidance and constant supervision as well as for providing necessary information regarding the project & also for her support in completing the project.

We would like to express our gratitude towards **Dr. Chandan Singh Rawat** (Head Of Department, EXTC) for his kind cooperation and encouragement which helped us in the completion of this project.

We would also like to express our special gratitude and thanks to our principal **Mrs. J. M. Nair** for giving us ample facilities and support in the course of our project work. We would also like to thank our friends who have been there for us all the time and supported us.

We are thankful towards **Mr. Abhishek Chaudhari** (Asst. Prof., ECS Dept.) for providing us with lab utilities and hardware required for the project & also for his timely assistance in problem-solving in the project.

Last but most importantly we would like to thank our lab assistants **Mr. Nitin Jadhav** and **Mrs. Shilpa Kedare** who have helped us totally in every aspect of our project work.

ABSTRACT

The adoption of technology in the food service industry has led to the emergence of "Intelligent Restaurants," where technology is integrated at every touch-point to enhance the overall dining experience. One example of this trend is the use of smart waiters, also known as waiter bots or automated waiters, equipped with advanced hardware and software tools, including sensors, wheel encoders, wireless systems, and various path-planning algorithms. These mobile robots can serve food and drinks in various settings, such as restaurants, hotels, businesses, or even homes, and interact with customers to take orders, simplifying the dining experience for both staff and patrons. Ongoing research is continuously evaluating and improving the current advancements in technology used to build smart waiters, making them an exciting and promising development in the food service industry.

Keywords— Intelligent autonomous vehicles, LIDAR, mobile robot systems, Automated Service, Obstacle avoidance, Simultaneous Localization and Mapping, Chatter-Bot.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

AMCL - Adaptive Monte Carlo Localization
AVR - Alf and Vegard's RISC processor
BLE - Bluetooth Low Energy
IDE - Integrated Development Environment
LDR - Light Dependent Resistor
LIDAR - Light Detection and Ranging
MVC - Model-View-Controller
NFC - Near Field Communication
NiMH - Nickel Metal Hydride
RFID - Radio Frequency Identification
RPM - Revolutions Per Minute
RTS - Robot Training System
SLAM - Simultaneous Localization and Mapping
TEB - Time Elastic Band
WCET - Worst-Case Execution Time

Chapter 1
INTRODUCTION

1.1 INTRODUCTION

Restaurants have traditionally relied on human employees to handle various tasks, such as taking orders and handling customer inquiries. However, as technology continues to advance, there is a growing trend toward the concept of an "Intelligent Restaurant", which seeks to integrate all customer touch-points, streamline processes, and enhance the overall dining experience.

One way that restaurants are adopting technology is through the use of digital menus and robotic systems to replace traditional services provided by waiters. These systems offer a range of options for marketing meals and snacks, and some of the most advanced restaurants now use robots equipped with technologies like LIDAR [6] [9] for smooth navigation, customer interaction, and a user-friendly online ordering interface. This not only replaces the traditional method of taking orders with pen and paper but also enhances the overall dining experience for consumers.

These "smart waiters" are the result of advanced technology and ongoing research efforts. For instance, one example of a smart waiter is an RFID-based [2] [15] [12] line follower robot with a high level of efficiency, around 96.66%. Another type of smart waiter utilizes the SLAM (Simultaneous Localization and Mapping) technique for robot navigation and LIDAR for generating maps using data obtained from the robot's surroundings. However, ongoing research and development are being done to further enhance the capabilities of these smart waiters.

Chapter 2

LITERATURE SURVEY

2.1 REVIEW OF LITERATURE

T. Akhund et al. [2] has developed a prototype of an autonomous waiter robot that could potentially work in restaurants and replace human servers. The robot uses a Line follower algorithm and RFID-based table recognition to collect the requested meal from the chef and serve it to the customer. The customer places their order by scanning a QR code on the table and on the body of the robot. The robot successfully travels along the black line to reach its destination without any issues. The hardware implementation rate was 96.66% effective, and the hardware prototype could be built at a significantly lower cost. This type of technology also creates job opportunities and maintenance profiles. However, further research and improvement of performance, as well as a more user-friendly interface, is desired. In the future, the robot may incorporate machine learning and computer vision.

S. Vongbunyong et al. [3] used Unity software to develop a simulation program to study the operations of a fleet of AMRs in a specific room. The program was built with the NavMesh module, which allows for the control of the robots through the use of NavMesh agents to mimic real-life AMRs. The simulation also includes the automatic generation of movement pathways and obstacle avoidance. A Unity script manages the battery level and charging procedure. The simulation was conducted for two scenarios based on meal serving time for patients, with different levels of operational requirements. The key factor that can lead to failure in the simulation is the depletion of the battery. The robots will not be able to perform their tasks if the charging rate is insufficient to compensate for the battery depletion. Using a more precise model for the rates of battery recharge and consumption could enhance the simulation outcomes.

In their research, Zahid Abbas et al. [4] have proposed a solution for reducing the human cost of small cafes in Pakistan through the development of an autonomous cafe management system. The system was tested in a real-world setting and showed promising results. The low cost of the system, estimated to be around 100S, makes it accessible to small businesses and has the potential to have a positive impact on the local economy. The authors believe that their solution can effectively balance the cost of labor and profit for small cafes and are confident in its potential for success.

In their research, F. Voinea et al. [5] have developed a robotic system that uses LIDAR technology and the SLAM (Simultaneous Localization and Mapping) approach for tracking the location of the robot in space. The system was designed to have a graphical interface that displays a 2D map created from LIDAR data, which allows the user to monitor the robot's movements in real-time. The system operates in three modes: direct control, text command control, and automated mode. In the automated mode, the software uses a pathfinding algorithm to search for a path on the map and guide the robot to the user-specified location. The use of graph theory in the software allows for real-time adjustments to the robot's path, avoiding potential obstacles and ensuring smooth navigation.

J. López et al. [6] in their paper described an automatic hotel assistant system named 'Bell-Bot' shown in **Figure 2.1.1** that utilizes mobile platforms to assist guests and service personnel with various tasks. These tasks include delivering small items, guiding guests to different points of interest in the hotel, escorting them to their rooms, and providing general information. The robots are capable of autonomously performing some daily scheduled tasks in addition to responding to user-initiated and pre-planned tasks. The system also enables robots to perform tasks based on events triggered by the building's automation system (BAS). The robots and the BAS are connected to a central server via a local area network.

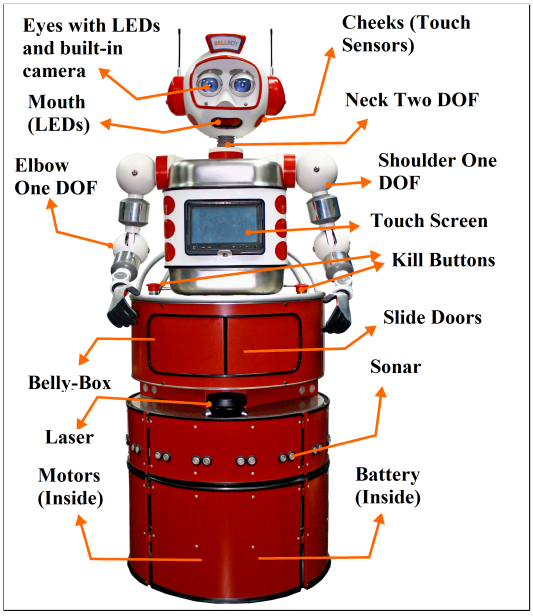


Figure 2.1.1: Hotel Assistant- BellBot

Figure 2.1.1 shows that the navigation devices are located in the robot's base, while the social interaction devices are mostly housed in its body. The base includes a Sick LMS100/10000 laser, bumpers that are integrated into the skin, two brushless Longway motors rated at 80 W/24 V with a torque of 6.2 N/m, and a LiFePO4 battery with a capacity of 26 V/40 Ah that provides power autonomy for 3 to 5 hours. For the BellBot hotel assistant robot [6], it is important to have several key features such as the ability to operate independently, easy customization, robust perception systems, and a focus on close interaction with humans.

M. Pinto et al. [8] aimed to investigate the use of robotic systems with temporal limitations and to govern their operations using the Robot Training System (RTS). They emphasized the importance of having a deterministic system in order to ensure time predictability and showed how to build a low-cost differential wheeled robot while maintaining time limitations. They used an embedded system with a microprocessor running an RTOS and used Rate-Monotonic scheduling to manage real-time processes. By estimating the Worst-Case Execution Time (WCET), they were able to create a robot that met all its time requirements in practice.

P. Denysyuk et al. [9] developed a mobile robot system that can automatically plan and travel to a designated location on a map. An Arduino DUE microcontroller controls the system, which utilizes SLAM techniques and LIDAR technology, chosen for their speed and compatibility. The software development process was conducted using Java programming language and JavaSwing library for interface development, while the development environments were Arduino IDE and IntelliJ IDEA. The software was designed using the Model-View-Controller (MVC) approach, separating the user interface, control logic, and data. The use of graph theory in the software model allowed for real-time analysis of potential obstacles and adjustments to the robot's path. The outcome of the project was a successfully functioning system with the robot's movements properly executed. The use of graph theory in the software model sets this development apart scientifically, enabling real-time adjustments and obstacle identification for the mobile robot's path.

A. Antony et al. [12] presented a hybrid solution for automating food service in restaurants. The system involves a central coordination system that manages a team of automated robots.

The selection algorithm determines the appropriate robot by ranking them based on various parameters processed through MATLAB and providing a merit value. The robot serves customers through a central server using a modified A* path planning algorithm for efficient navigation in transporting food or receiving orders. The algorithm is stored in the ATMega 128 microcontroller. The paper also explores the challenges and limitations of implementing collaborative robotics in a restaurant setting, with multiple robots and a central station. Further investigation is suggested to examine the coordination between robots, including communication between a robot that follows a set path and a robot that moves freely without restrictions.

Md. Kamruzzaman et al. [13] proposed a system in which customers can place an order for tea or coffee using a wireless communication system. The robot would reach the customer's table by counting the number of steps taken. Customers would use an Android app to place their orders when they are seated at the restaurant. The waiter would receive the order from the kitchen crew after the order was placed and the robot would travel to the designated table with the help of wheel encoders that count the steps. After the order was delivered, the waiter would wait for it to be unloaded and then inform the customers. Finally, the waiter would return to the kitchen following the same path.

Karan Kaushal et al. [14] implemented a system called "E-Restaurant" that utilizes "E-trolleys" powered by Light Detecting Resistors (LDR) and a line-following method for navigation. They give customers the option to place their order by selecting one of three buttons that correspond to the desired number of servings. Once the order is placed through a website accessible to both customers and restaurant staff, the e-trolley begins its journey to the customer's table using the line-following technique. The trolley is equipped with two Infrared (IR) sensors at the bottom which detect the black line on the floor. Once the order is delivered, the customer can press the button on the trolley to initiate its return journey to the kitchen.

A Cheong et al. [15] made a special robot waiter system that can help serve food and clean up at restaurants. It uses different technology like sensors, transmitters, and coasters that tell it where to go. Customers can order food using a tablet and then get a pager that vibrates when their food is ready. This page also tells the robot which table to bring the food to. We use a special wireless network to help the robot find its way around and locate tables. We want to try out the robot in a real restaurant and see if it works well. It could also be used in hospitals or other places where people need help with serving and cleaning up. The robot waiter system can be customized to fit different needs and use different technology.

Eksiri and T. Kimura's [19] paper describes the development of service robots that were tested in a natural restaurant environment. Two types of robots were created: one to take customer orders and another to deliver orders to tables. The Order robot would move to the designated table after the operator entered the table number and take the customer's order through its user interface before returning to the robot station. On the other hand, the Server robot would move after the operator placed the food in its container and entered the table number, the Order and the Server robots are shown in **Figure 2.1.2**. The operator and robot would then proceed to the table together, with the operator placing the food on the table before the robot returned to the station. The authors used a Distributed Control System Concept to manage various tasks simultaneously, such as robot localization, and obstacle detection, and for entertainment and warning systems they have incorporated robotic arm movement. The slave controllers controlled hardware directly, while the master controlled the slaves. To simplify the circuit design, the master and slaves were connected via a data communication bus.

Figure 2.1.2: The figure shows various order and serve Robots proposed in [19] since 2009 until 2012. At the leftmost and rightmost corners, we have two Serve-Robots namely ServeTwo and ServeOne developed in the years 2010 and 2012 respectively. Besides the Serve-Robots, we have the Order-robots which are OrderOne and OrderTwo both developed in the year 2009. At the center, we have the most recently developed Slim Robot in the year 2012.

G. A. Haider et al. [20] have developed "Jemma", an intelligent server that utilizes AI to cater to various beverage needs, especially in work environments. It's a groundbreaking solution that enables individuals and workers to obtain drinks easily while working for prolonged periods without hydrating themselves. The system mainly consists of a camera-equipped head for facial recognition and voice projection, an AI system that identifies employees or targeted groups, a body that houses the beverage dispensing machine and a touch screen for ordering, and a base with a smart obstacle avoidance mechanism. Raspberry Pi powers the system for self-directed mobility between offices. "Jemma" can operate in two ways: remotely, via a mobile app connected to Wi-Fi, or autonomously, using a smart internal positioning system.

Yu Hing et al. [21] has created a highly advanced senior care robot in their research paper. The cost-effective 2D SLAM system based on laser radar and the G mapping SLAM techniques are among the standout features of the robot. The robot's ability to navigate its surroundings is made possible through the use of the ROS "move base" module and the Adaptive Monte Carlo Localization (AMCL), which integrates data from laser radar and encoders to determine its position in the environment. Additionally, the robot is equipped with facial recognition, face tracking, ChatterBot technology, and an Android application, making it highly capable and versatile. This robot has the potential to greatly improve the lives of the elderly by providing personalized care and support in a cost-effective and efficient manner.

A. Abdulkareem et al. [22] have implemented a mobile robot that replaces a waiter in a cafeteria, delivering food from the main counter to customers' tables to improve service efficiency. An Arduino microcontroller powers the robot, and features RFID technology, a line-following module, and obstacle detection. An RFID reader fixed onto the robot can read passive tags placed under serving trays to identify the correct table and navigate accordingly. The unique aspect of this robot design is the integration of obstacle detection and collision avoidance with RFID technology. The study's results demonstrate the robot's ability to follow its intended path.

P. Guo et al. [24] have discussed the development of an autonomous mobile robot designed to complete delivery tasks in a catering facility, with the goal of reducing the spread of COVID-19 and other dangerous bacteria and viruses. The robot's path is planned and executed efficiently using a Time Elastic Band (TEB) motion control method and an upgraded Dijkstra algorithm. In addition, the robot is equipped with voice and image recognition capabilities, which are trained using open-source AI platforms, allowing it to recognize voice signals of varying tones and loudness, and identify images using You Only Look Once version 4 (YOLOv4) and Scale-Invariant Feature Transform (SIFT). Experimental tests, including China's 16th National University Student Intelligent Car Race, demonstrated the feasibility of the proposed robot architecture, suggesting that it could provide an effective solution to reducing the transmission of harmful viruses and bacteria in catering facilities.

Y. Ashra et al. [25] described a system that utilizes an Android application and delivery robots to streamline cafeteria services. The system includes key components such as an Arduino microcontroller and an ESP8266-based Node MCU for internet connectivity. The Node MCU is essential as it connects the robot to Firebase and other internet services through Wi-Fi. The Arduino and Node MCU are linked via a serial COMM port, while all other sensing and control components are directly connected to the Arduino. They utilize a line follower robot to transport food from the pantry to the customer. The robot is equipped with an OLED display that displays information such as the customer's destination table number and battery percentage remaining, among other data required by the server. During the delivery process, a servo motor is used to secure the shelf and prevent any tampering with the food, which remains locked until the user unlocks it using the keypad and OLED display. Additionally, the Android SDK is employed to develop applications for the Android Platform.

T. Y. Lin et al. [26] have identified some shortcomings caused during the COVID-19 pandemic in the catering industries thus, presented a solution in the form of an automated system called Mots for contactless meal ordering and takeout. This system is powered by AI-assisted smart robots and aims to resolve the issues. Our Mots system is designed to group robots using the Welsh-Powell coloring algorithm to create a bump-free schedule for them to move without colliding. Our simulation results indicate that Mots can significantly enhance takeout efficiency and service accuracy, resulting in a substantial increase in business profits of up to 95.4% under various cafeteria scales and shop popularity differences, compared to the traditional takeout method. Our experiments demonstrate that Mots can also handle a sudden influx of customers during peak hours. Additionally, we have developed a proof-of-concept prototype to showcase the automated operations of our Mots system.

A. H. S. Hamdany et al. [28] engineered a waiter robot utilizing Arduino technology that features a user-friendly keypad and LCD display for customers to place orders. The robot traverses a circular path from the kitchen to the tables, which are arranged both inside and outside the path, enabling efficient order collection. The robot also features integrated payment hardware located on its body for secure and hassle-free payments. Communication with the restaurant is established via Bluetooth technology, and Near Field Communication (NFC) capability is also available. Once orders are collected, they are distributed to specific customers, and the credit card device located within the robot's body allows for easy payment from the customer's seat. This innovative waiter robot combines technological advancement with customer convenience, potentially revolutionizing the restaurant industry.

2.2 COMPARISON OF LITERATURE REVIEW

Table 2.1 below shows a comparative study that evaluates the range of currently available smart waiter robots. The efficiency and versatility of such robots increase with the addition of sensors and actuators, which allow them to perform a variety of activities. For instance, a robot with a greater number of sensors can accurately detect and avoid obstacles while navigating through the restaurant also, it can be more interactive customers for enhancing their experience.

Table 2.1: Comparison between different existing smart waiters reported in Literature

Reference Paper	Number of Trays	Wheel Rotation (rpm)	Capacity (kilograms)	Scanning Angles (degrees)	No. of Sensors
Filofetia Valentina Voinea et al. [5]	1	160	-	-	6
Md. Kamruzzaman et al. [13]	1	50-250	0.6	-	6
Karan Kaushal et al. [14]	3	-	-	180	8
A. Cheong et al. [15]	3	267-300	-	-	5
G. Abou Haidar et al. [20]	1	-	20	-	8
Yu Hing et al. [21]	-	-	0.5	180	12
A. Abdulkareem et al. [22]	1	200	-	30	4
P. Guo et al. [24]	1	-	-	360	8

The number of trays that a robot can carry is determined by its weighing capacity. Robots with a higher maximum weighing capacity can carry more trays, allowing them to deliver more food and drinks to customers in a single trip. This feature is helpful in busy restaurants, as it reduces the number of trips the robot needs to make to serve a large number of customers.

The speed of the robot is a crucial determinant of its performance and is primarily dependent on its wheel rotation measured in RPM (revolutions per minute). A robot can operate at varying speeds, ranging from as low as 50 RPM to as high as 300 RPM. Robots with higher RPM have a faster wheel rotation and can cover more distance in less time. This attribute is especially valuable in a busy restaurant setting.

Additionally, the robots' scanning angles play a vital role in the accuracy of their operations. A higher scanning angle(i.e. 360 degrees) allows the robot to scan the surrounding environment more accurately, ensuring that it can detect obstacles and navigate accordingly. This feature also ensures that the robot can locate and deliver food to the correct table without making any errors.

Chapter 3

PROJECT DESCRIPTION

3.1 PROBLEM STATEMENT

In traditional restaurant ordering systems, customers have to rely on waiters to take their orders. This process can be time-consuming and frustrating, especially during peak hours when there is a high demand for service. Additionally, there is always a possibility of miscommunication, leading to incorrect orders, delays, and unhappy customers. This can result in lost revenue for businesses, as dissatisfied customers are less likely to return and may leave negative reviews.

Smart Waiters aim to address these issues by automating and personalizing the ordering process. With Smart Waiters, customers can use a user-friendly interface to place their orders, reducing the reliance on waiters and minimizing wait times. This technology can also offer personalized recommendations based on the customer's preferences and previous orders, increasing customer satisfaction and loyalty. Moreover, businesses can use Smart Waiters to collect customer data and analyze customer behavior, allowing them to make data-driven decisions to improve their services and increase revenue.

Furthermore, Smart Waiters can provide additional benefits, such as contactless ordering and payment options, which have become increasingly important during the COVID-19 pandemic. With contactless ordering, customers can place their orders and make payments without physical contact with menus, credit card terminals, or waiters. This can help reduce the risk of infection and provide customers with a safer dining experience. Overall, Smart Waiters offer a solution to the inefficiencies and limitations of traditional restaurant ordering systems, providing a more efficient, personalized, and safer dining experience for customers, while also increasing revenue and customer loyalty for businesses.

3.2 PROPOSED SOLUTION'S BLOCK DIAGRAM

The proposed system shown in **Figure 3.2.1**, consists of two major Micro-controller blocks, one being the ESP32 for wireless communication while the other being Atmega2560 which looks after the majority of the hardware part. The combination of these microcontrollers is used as the brains and brawl of the machine. Both are communicating and sharing data with the Database. The database stores the data of the host website as well as the orders placed by the customer.

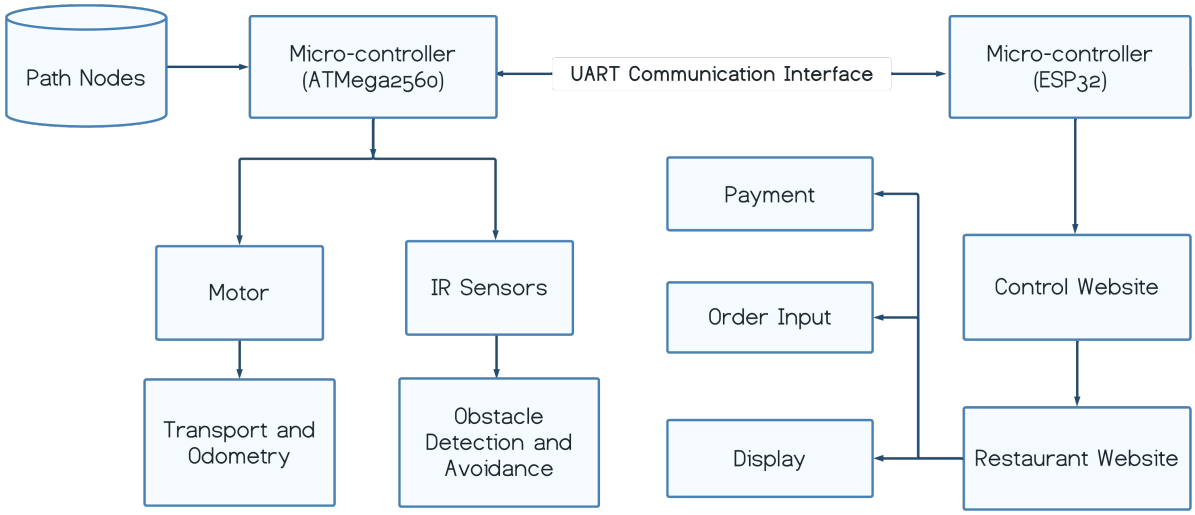


Figure 3.2.1: Block Diagram of Proposed Solution

The atmega2560 is connected to the hardware components of the system. The Transport mechanism consists of wheels connected to the motors which are internally controlled by encoders that contribute to the movement of the robot. Spatial recognition gives information about the environment around the robot and sends the data to the ESP32. Spatial data can be used to give an estimation of where the robot is standing and to calculate the distance from a nearby stop point. Obstacle detection helps with the detection of any or every obstacle in the path of the robot and when detected is prepared to avoid them as well.

The ESP32 is the communicator of this project; it connects the hardware part of the robot with the IoT world. It is connected to the restaurant website which takes order input from the user and displays it with other information on a Display. the website also includes a payment gateway designed for payment-reducing human interaction. The ESP32 gets all the data and sends the coordinates to the ATmega2560 to follow and reach the particular table.

3.3 PROPOSED RESTAURANT LAYOUT

This proposed restaurant layout **Figure 3.3.1** consists of two separate lanes: an even lane and an odd lane. At one end of the restaurant, you'll find the bustling kitchen, where skilled chefs prepare dishes.

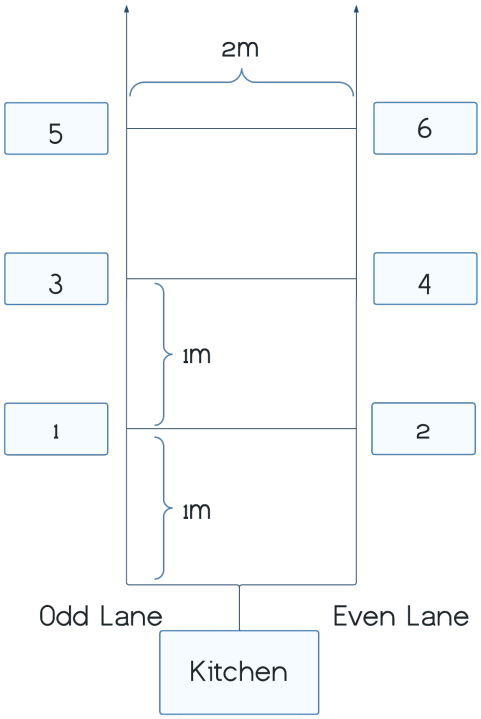


Figure 3.3.1: Proposed Layout of the Restaurant.

The even lane is equipped with three tables, numbered 2, 4, and 6, while the odd lane boasts tables numbered 1, 3, and 5. This distinct arrangement allows for a cozy and intimate dining experience, perfect for couples or small groups.

These lanes are separated from each other by a distance of 2 meters and the tables in each lane are separated from each other by a distance of 1 meter. These specifications are crucial for navigation using odometry. The robot can be adapted to any restaurant layout plans with proper distance specifications for the table and the kitchen.

3.4 OVERVIEW OF THE WORKING

The Smart Waiter Robot Uses the principle of the Path Planning Algorithm. By Defining the system and Allowing tasks to different nodes the whole system works efficiently. The combination of two micro-controllers effectively divides the tasks and reduces load over a single system. The ATmega micro-controller handles the hardware part while the ESP32 takes care of communication and the Website. The ESP32 takes information from the website, when the robot is called the ESP32 commands the robot to get to the table. The robot then follows a set of specific coordinates to the table. The ATmega microcontroller handles the hardware part of the system. This includes locomotion and movement, obstacle detection, spatial recognition, and Path recognition.

The first step is a "Call" from the customer in terms of his order. The customer has to order food based on their choices and availability from the website by scanning the QR code placed on the table and entering the table number. The second step is the "Call" from the Kitchen, Based on this counter the employee in the kitchen orders the robot to deliver the order to a certain table. The robot takes the ordered food and maps the closest route to the customer's table. Using this route alongside spatial and obstacle detection the robot then delivers the order to the customer. It may come directly to the customer or will get to other orders before reaching the customer. Once at the table, the Robot shares the order with the customer. Once the order is received by the customers the robot returns to the kitchen.

Once the customer wishes to leave the restaurant he needs to do the payment. The payment will be cashless and will be done through the website. The payment gateway will be available on the website itself and the customer can pay the bill using UPI or card transaction. An in-detail demonstration is discussed in **Section 5.2** of **Chapter 5**.

3.4.1 Working

When the kitchen orders the robot to go to a certain table, the algorithm is executed. For example when the kitchen orders the waiter to move to table number 4.The `MoveToTable()` function is called with `tableNumber` parameter value of 4.The `initialDirection` variable is set to left so the robot needs to move left to reach the target table. The lane selection function selects the correct lane for the robot to move toward the target table. The robot is then moved towards the target table using the Move function with a distance of `lanewidth * 2` (where `lanewidth = 1000mm`) since the target table is 4. The robot is then rotated by 90 degrees towards the target table using the `rotate()` function with `rotateDegree` parameter value of 90 and `rotateDirection = left` since the target table is an odd-numbered table.

The `CurrTable` variable is used to update the current table i.e. the table number where the robot is currently standing and in this case is updated to 4 since the robot has reached the target table. So, in summary, the robot will move to the lane corresponding to the target table's position, move towards the target table in the lane, and rotate towards the target table before updating the `CurrTable` variable to the target table number.

Once the robot delivers the order to the corresponding table number, it will then head towards the kitchen where it will wait while the next order is being prepared. The robot traces the same path which it has traveled to reach the target table to deliver the order while coming back to the kitchen side. All the `rotate()` functions are reversed, while the distance required by the robot to travel in order to reach the kitchen remains the same.

3.5 COMPONENTS DESCRIPTION

3.5.1 Fire Bird V ATMEGA2560

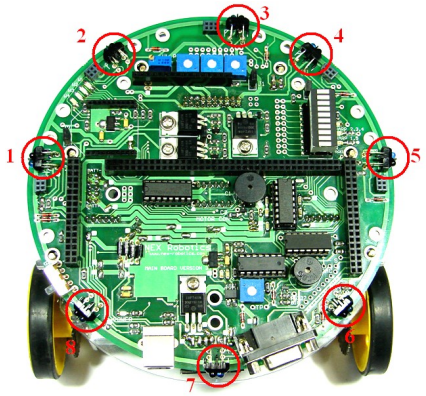
1. **Hardware Components:** Both the below-mentioned micro-controllers are AVR based

   (a) **Micro-controllers:**

      i. **Master micro-controller:** Atmel ATMEGA2560 - the main controller that processes and executes instructions received from the user or operator.

      ii. **Slave micro-controller:** Atmel ATMEGA8 - a secondary controller that manages low-level operations, such as sensor readings and motor control.

   (b) **Infrared proximity and directional light intensity sensors:** Infrared proximity sensors are used to detect the proximity of any obstacles in the short range. IR proximity sensors have about a 10cm sensing range. These sensors sense the presence of obstacles in the blind spot region of the Sharp range sensors. Fire Bird V robot has 8 IR proximity sensors. **Figure 3.5.1** shows the location of the 8 IR proximity sensors.



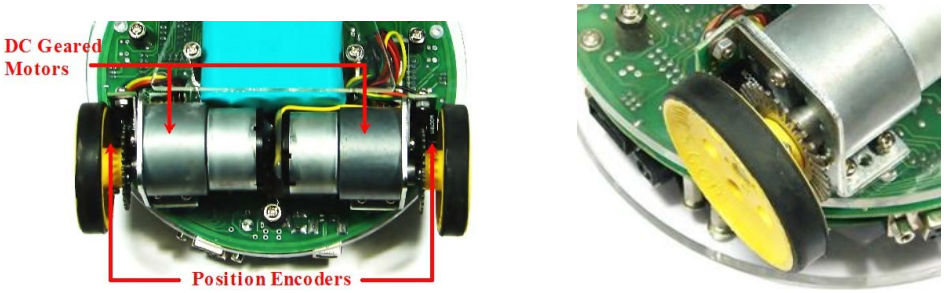
Source: http://elsi.e-yantra.org/resources#datasheets_and_Manuals
Figure 3.5.1: Eight IR proximity sensors on Fire Bird V.

   (c) **Buzzer:** Robot has a 3 KHz piezo buzzer. It can be used for debugging purposes or as an attention seeker for a particular event. The buzzer shown in **Figure 3.5.2** is connected to the PC3 pin of the microcontroller. Buzzer is driven by a BC548 transistor. Resistor R40 of 100K is used to keep the transistor off if the input pin is floating. Buzzer will get turned on if the input voltage is greater than 0.65V.

Source: http://elsi.e-yantra.org/resources#datasheets_and_Manuals
Figure 3.5.2: Buzzer

   (d) **Position Encoders:** Position encoders shown in **Figure 3.5.3** give position / velocity feedback to the robot. It is used in a closed loop to control the robot's

position and velocity. The position encoder consists of a slotted disc that rotates between the optical encoder (optical transmitter and receiver). When the slotted disc moves in between the optical encoder we get a square wave signal whose pulse count indicates position.



Source: http://elsi.e-yantra.org/resources#datasheets_and_Manuals
Figure 3.5.3: DC geared motors and position encoders.

Optical encoder MOC7811: It is used for position encoder on the robot. It consists of an IR LED and a phototransistor mounted in front of each other separated by a slot in the black opaque casing with a small slot-shaped window facing each other. When IR light falls on the phototransistor it gets into saturation and gives logic 0 as the output. In the absence of the IR light, it gives logic 1 as output. A slotted encoder disc is mounted on the wheel and placed between the slot. When the encoder disc rotates it cuts IR illumination alternately because of which the phototransistor gives a square pulse train as output.

(e) **Compact size and weight:** With a diameter of 16cm, a height of 8.5cm, and a weight of 1100 grams, the Fire Bird V is portable and can be used in different environments.

(f) **Reliable power source:** The 9.6V Nickel Metal Hydride (NiMH) battery pack provides long-lasting power and can be recharged using the external auxiliary power from the battery charger.

(g) **Battery monitoring and intelligent charging:** The platform has features that monitor the battery's health and prevent overcharging or undercharging.

2. **Communication Options:**

   (a) **USB:** Used to establish a wired connection with a computer or other devices.

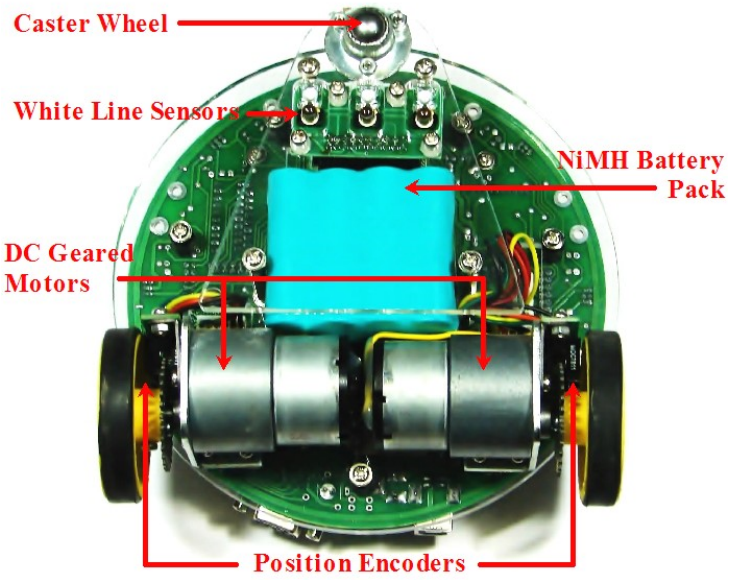
   (b) **Wired RS232:** Another wired communication option that uses the RS232 standard for serial communication.

   (c) **Wireless serial:** Allows wireless communication between the robot and other devices that support serial communication.

   (d) **WiFi:** With a WiFi module installed, enables the robot to connect to WiFi networks and communicate over the internet.

   (e) **Bluetooth:** With a Bluetooth wireless module installed, the robot can communicate with Bluetooth-enabled devices, such as smartphones and tablets.

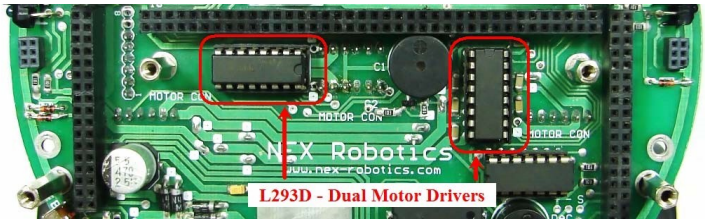
3. **Motion Control:** Fire Bird V robot has two 75 RPM DC geared motors in differential drive configuration along with the third caster wheel as shown in **Figure 3.5.4** for the support. Robot has a top speed of about 24cm per second. Using this configuration, the robot can turn with a zero turning radius by rotating one wheel in a clockwise direction



Source: http://elsi.e-yantra.org/resources#datasheets_and_Manuals
Figure 3.5.4: Fire Bird V ATmega2560 robot bottom view

and the other in a counterclockwise direction. Position encoder discs are mounted on both the motor's axle to give position feedback to the microcontroller.

Motion control involves direction control and velocity control. Motors are controlled by L293D dual shown in **Figure 3.5.5** motor driver which can provide up to 600mA of current to each motor. To change the direction of the motor, appropriate logic levels (High/Low) are applied to L293D's direction control pins. Velocity control is done using Pulse Width Modulation (PWM).



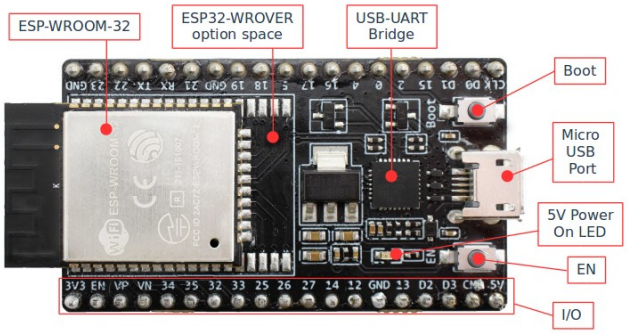
Source: http://elsi.e-yantra.org/resources#datasheets_and_Manuals
Figure 3.5.5: Motor Driver

3.5.2 Espressif Systems' ESP32

1. **Processor and Performance:**
   (a) **Dual-core Processor:** The ESP32 microcontroller shown in **Figure 3.5.6** features a dual-core processor that provides enhanced performance.
   (b) **Clock Frequency:** The microcontroller has a clock frequency of up to 240 MHz, which ensures faster execution of instructions.
2. **Communication Protocols:**
   (a) **Wifi:** The ESP32 micro-controller has built-in WiFi capabilities that support the 802.11b/g/n standard.

(b) **Bluetooth:** The microcontroller supports Bluetooth v4.2 BR/EDR and Bluetooth Low Energy (BLE) protocols.

(c) **Range of Hardware Interfaces:** The micro-controller includes several hardware interfaces, such as Universal Asynchronous Receiver Transmitter(UART), Serial Peripheral Interface(SPI), Inter-Integrated Circuit (I2C), and I2S.



Source: http://esp32.net/images/Espressif/ESP32-DevKitC/Espressif_ESP32-DevKitC_V4_DiagramInfo.jpg
Figure 3.5.6: ESP WROOM-32

3. **Power Options:**

   (a) **Power Sources:** The ESP32 can be powered by various sources, including USB, battery, and external power supply.

   (b) **Low Power Consumption:** The microcontroller's low power consumption makes it suitable for applications that require wireless connectivity without using too much power.

4. **Security:** The ESP32 microcontroller supports a range of security protocols, including WPA2, WPA3, and TLS.

5. **Small Size:** The microcontroller's small size makes it suitable for applications with limited space.

6. **Applications:** The ESP32 is a versatile and reliable Wi-Fi module that can be used in a variety of applications, from home automation to industrial IoT.

Chapter 4

IMPLEMENTATION

4.1 SOFTWARE IMPLEMENTATION

4.1.1 Website Implementation

The website plays an important role in accepting orders and providing a vivid range of available food items. A QR code will be prominently displayed on each tabletop with the help of which the customers can place their desired order. They are instructed to scan the code with their smartphones in order to gain access to the website. After scanning the code, customers are immediately directed to the restaurant's online ordering platform containing the "Food Item List", "Price" and "My Cart".

The 'WAITRON' website is implemented using ReactJS and Stripe and offers a smooth and easy-to-use interface for customers to order food and pay for it online. The website is divided into different sections such as home, menu, ordering, payment, and contact us, allowing customers to easily select the dishes they wish to order. Once they have made their selections, customers are asked to input their respective table numbers before submitting the order.

Upon receiving the order, the kitchen staff is promptly notified and begins preparing the food with the utmost care and attention to detail. As the dishes are completed, the kitchen staff calls out to the robot. The robot, which has been programmed to respond to the call, moves over to the kitchen staff member, who then carefully places the dishes on the robot's tray. Upon receiving an order, the staff member proceeds to use the controller to input the corresponding table number to initiate food delivery services.

We have incorporated the Landing, Menu, and Cart sections in our 'WAITRON' website which are discussed further in **Section 5.1** of **Chapter 5**. This cutting-edge system streamlines the restaurant's operations and improves the overall dining experience. Customers no longer have to wait in long queues to place their orders, and the use of the robot adds a unique and exciting element to the dining experience.

4.1.2 Interfaces to Control Robot

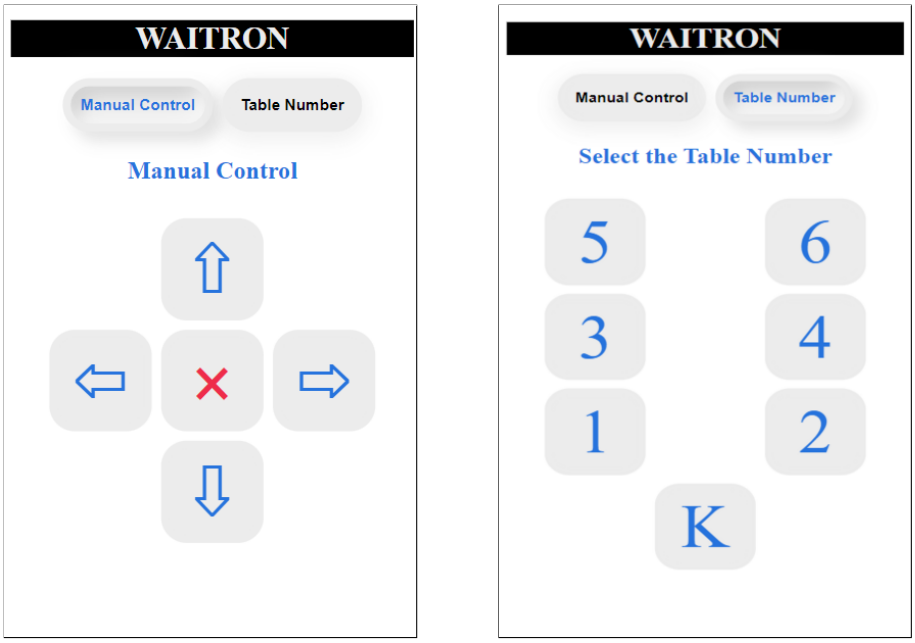
The Interface developed with the help of HTML and CSS hosted on the ESP Wroom-32 board, which can be displayed on a smartphone or tablet, plays a vital role in controlling and determining the target table for the robot. This Interface will only be visible to the coordinator/admin on the kitchen side. Once the order corresponding to a table gets ready to dispatch, the kitchen coordinator will place that order onto the trays of the robot and will enter the desired target table number.

As shown in **Figure 4.1.1** below there are two interfaces available to control the robot's movement which can be switched as and when required, one is the manual control and the other one is used to select the desired table number according to the order received.

The manual control interface shown below in **Figure 4.1.1a** consists of five buttons for manually controlling the robot's movements. The buttons are represented by upward, left, backward, and right arrow keys while the central **'X'** button is used to stop the robot i.e., it executes the functionality of the `brake()` function.

The other interface shown by **Figure 4.1.1b** below is used to select the desired table number for delivering the order or collecting used plates to and from the table respectively. They are

arranged in accordance with the proposed layout of the restaurant. The button to return to the kitchen from an of the table is represented by **'K'**, while the other six buttons namely **'1', '2', '3', '4', '5'** and **'6'** represent the table numbers. Initially when the robot is in the kitchen and when an order is received and gets ready, the kitchen coordinator with the help of this interface selects the desired table from where the order has been received.



(a) Interface for Manual Control **(b)** Interface to Select Table Number

Figure 4.1.1: Interfaces for Controlling/ Determining the WAITRON's target table.

For Example, If the order has been received from table six then, the coordinator will select or will press '6' on the interface. The same information is then forwarded to the robot through serial communication which has been established between ESP Wroom-32 and the Fire Bird V robot by means of UART serial communication. There are some predefined functions that will be executed by the robot on pressing each of the buttons specified in both interfaces.

4.1.3 Installation and Set-up of Required Software

1. **Arduino IDE:**

   (a) Download and install the Arduino IDE version 1.6.12.

   (b) Open the Arduino IDE and navigate to the "Tools" menu and select "Boards Manager".

   (c) Check the version of the Arduino AVR boards in the Boards Manager and make sure it matches the IDE version, which is 1.6.12.

   (d) Locate the "boards.txt" file located in the install location of the Arduino IDE, which is typically located at "C:\Program Files (x86)\Arduino\hardware\arduino\avr\boards.txt".
   Note that sometimes the boards.txt file is stored in 2 locations as a backup. The second location is "C:\Users\"username"\AppData\Local\Arduino15\packages\arduino\hardware\avr\boards.txt".

(e) Edit the "boards.txt" file to modify the part related to the Firebird microcontroller with custom crystal frequency(14745600) as shown below in **Figure 4.1.2**. This typically involves removing everything above and below the Arduino Mega and Mega ADK parts, except the first two lines of code.

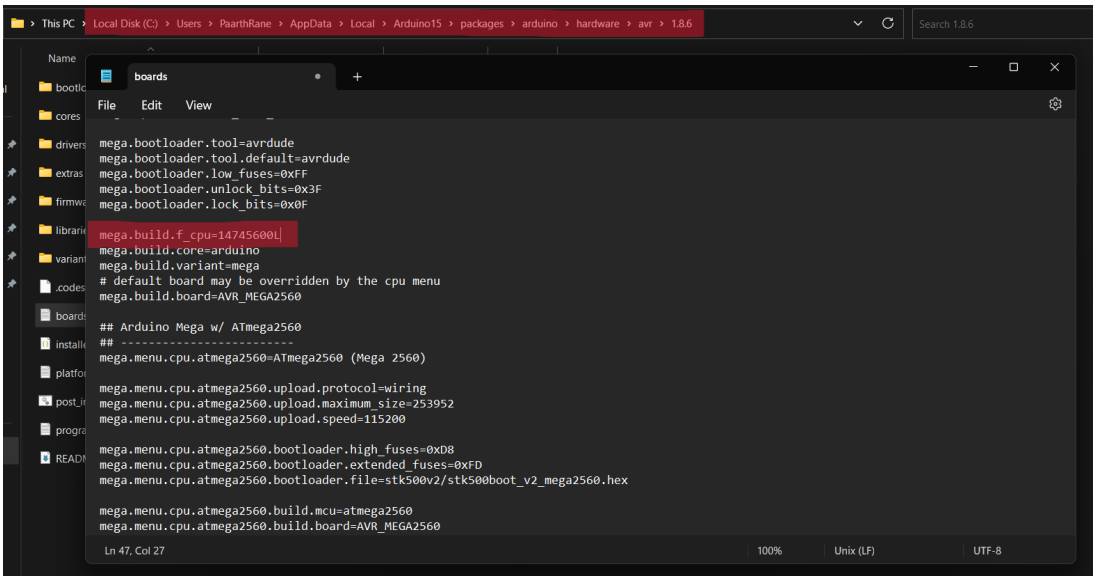


Figure 4.1.2: Modification of boards.txt

(f) Start writing your code in the Arduino IDE.

(g) Once your code is complete, click on "Verify" to check for any errors. This will generate a temporary file in the temp folder.

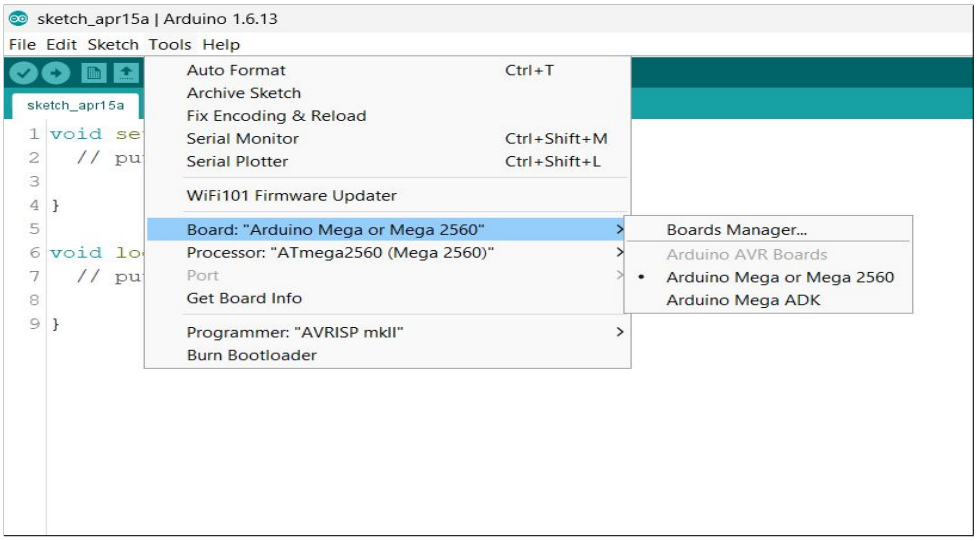


Figure 4.1.3: Arduino IDE after changing the boards.txt file.

(h) To locate the temporary folder, press the Windows key + R and type "%temp%" in the dialog box, and hit enter. This will take you to the temp folder where an Arduino-labeled folder can be found. Your hex file with your project name can be found in this folder.

Note: After changing the "boards.txt" file, there will be only two boards shown in your Arduino IDE. Select the first board as shown below in **Figure 4.1.3** and then "Verify"

the code in order to generate the ".hex" file which will then be uploaded to the Fire Bird V using the AVR Bootloader Software.

2. **AVR Bootloader:** All AVR microcontrollers can be programmed using In-System Programming (ISP), an external programmer, or using boot loader. The advantage of the bootloader is that you don't need any external hardware to load the .hex file on the microcontroller. It also protects the robot's hardware from possible damage due to static electricity and prevents any accidental changes in the fuse settings of the micro-controller.

   Boot loader operating principle:

   If Bootloader firmware is loaded on the micro-controller, it allows in-system programming directly via serial port without any need for the external ISP programmer. Code responsible for In System Programming via serial port (boot loader) resides in the configurable boot memory section of the micro-controller. When signaled using an external switch while resetting the microcontroller it gets active and waits for communication from the PC for copying the .hex file on the microcontroller's flash memory.

   PC sends the .hex file to the microcontroller. Code residing in the boot section loads the .hex file on the microcontroller's flash memory. After the boot loading process is complete, the newly loaded code can be executed by pressing reset. Once the code is loaded on the microcontroller UART is free and can be used for other applications. Bootloader gets invoked only if the boot switch is kept pressed while the microcontroller is reset using the reset switch.

   Note:
   Boot loader code is loaded on the ATMEGA2560 of the Fire Bird V robot using an ISP programmer. It is recommended that you only use a Boot loader for the robot. If you use an ISP programmer with the robot then the boot section code might get erased and the robot will no longer support boot loading.

   (a) **Step 1:**

      i. Ensure that Microsoft .NET Framework version 2 or higher is installed on your Windows system.
      ii. Download the AVR Boot loader and ATmega2560 firmware from NEX Robotics' official website.
      iii. Install the AVR Boot loader and firmware on your Windows system at the default path specified by the installer as shown in **Figure 4.1.4** below.
      iv. To upload code to the Firebird 5 robot, a .hex file is required. This .hex file can be generated using different compilers, but in this project, we use the Arduino IDE.

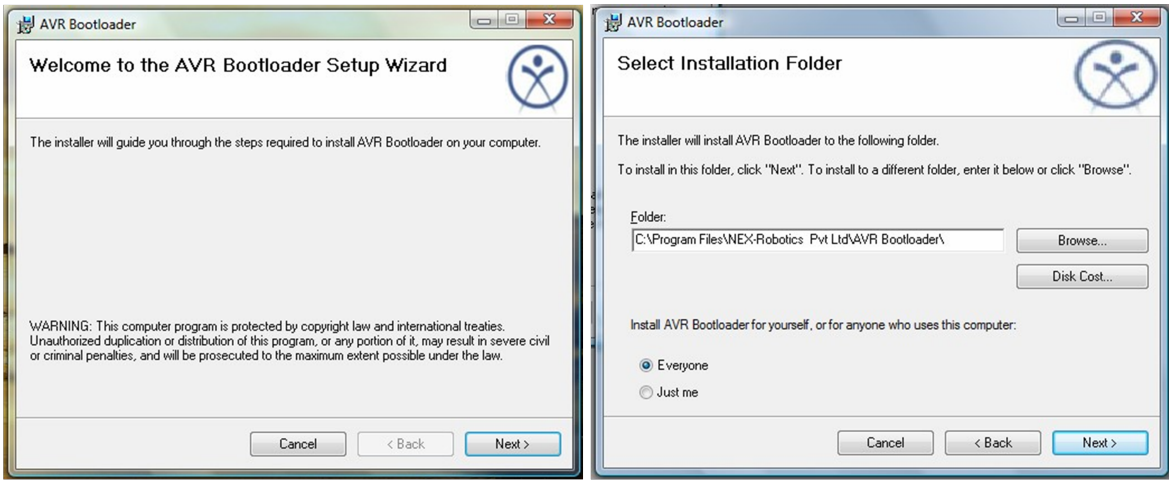


Figure 4.1.4: Installation of AVR Boot loader

(b) **Step 2:**

i. Open the Arduino IDE and write your code for the Fire Bird V robot.

ii. Compile the code by clicking on the "Verify" button. This will generate a .hex file in a temporary folder.

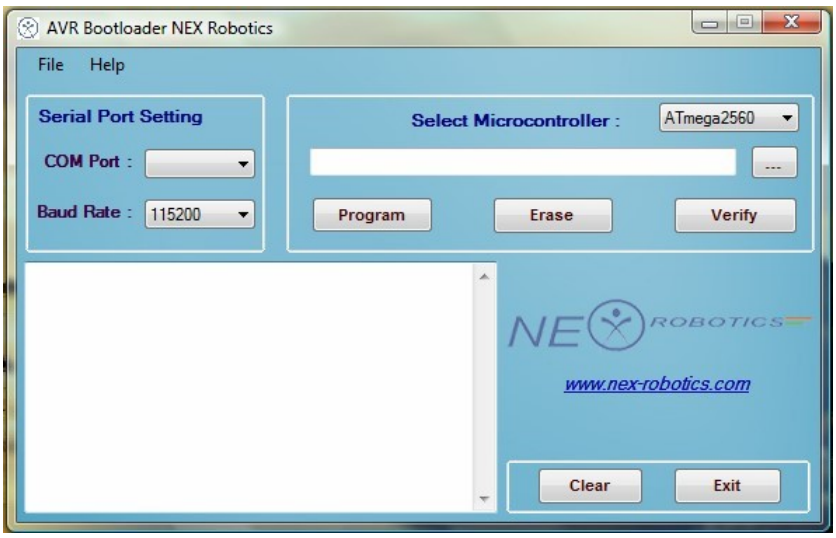


Figure 4.1.5: AVR Boot Loader software

iii. Locate the .hex file in the temporary folder by pressing the Windows key + R and typing "%temp%" in the dialog box. Hit enter and navigate to the Arduino-labeled folder to find the .hex file with your project name. (Don't use the "_withbootloader.hex")

iv. Turn on the Fire Bird V robot.

v. Connect the USB cable between the robot and the PC and wait for 2 seconds

vi. Open the AVR Bootloader software **Figure4.1.5**. It will auto-detect the COM port number associated with the robot.

vii. Click on the detected COM port number. If multiple COM port numbers are detected, refer to section 6.6 in the Hardware Manual to identify the COM port number associated with the robot.

viii. If the robot's COM port number is more than COM8, change it to any COM port number between COM1 to COM8.

ix. Set the Baud rate at 115200 bps.
Note: The execution of steps form vi to ix is represented in **Figure 4.1.6**

x. Select the ATmega2560 micro-controller.

xi. Browse for the target .hex file generated in Step 6 of the previous instructions.

xii. First press the boot switch (switch on the right). Press and release the reset switch with a time interval of 1 second while holding the boot switch. The boot loader is written in such a way that when the reset switch is pressed while holding PE7 low, ATMEGA2560 will go into boot loader mode.

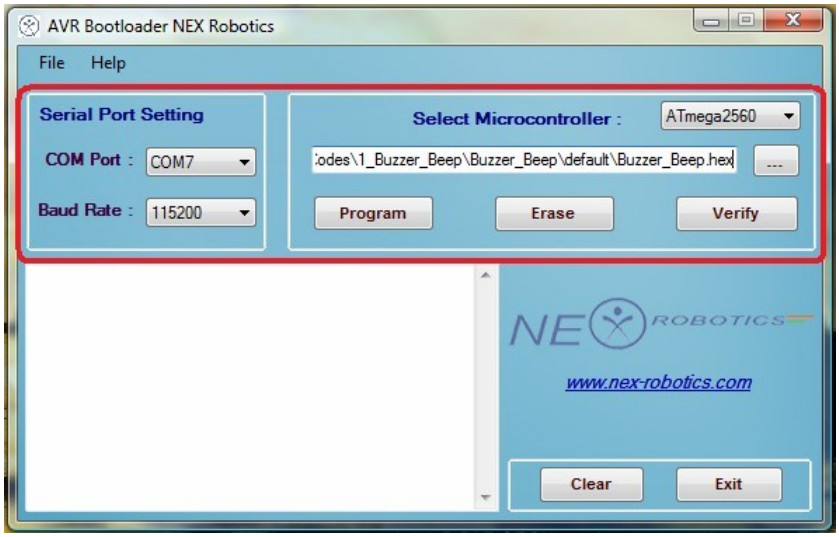


Figure 4.1.6: Uploading the .hex file

xiii. Press the program button on the GUI. Within 2 seconds, the .hex file loading will start. You can see the activity on the TX and RX LEDs on the ATMEGA2560 microcontroller adaptor board. These LEDs are located just below the FT232 USB to Serial Converter chip.

xiv. **After successfully programming following text will appear in the dialogue box as shown in Figure 4.1.7 below:**
" Serial port timeout set to 5 sec.
Found AVRBOOT on COM7!
Entering programming mode...
Parsing XML file for device parameters...
Parsing '.\ATmega2560.xml'...
#######
Saving cached XML parameters...
Signature matches device!
Erasing chip contents...
Reading HEX input file for flash operations...
##############################
##############################
##############################
Programming Flash contents...
Using block mode...
#######

Reading Flash contents...
Using block mode...
#######
Comparing Flash data...
Equal!
Leaving programming mode...”

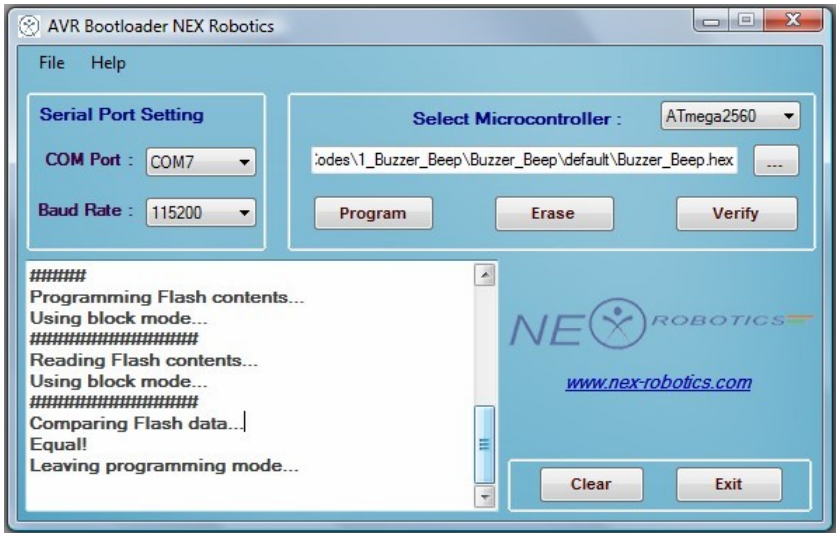


Figure 4.1.7: Successfully uploaded the code through AVR Boot Loader

4.1.4 Path Planning Algorithm

The floor plan of the restaurant is divided into two separate lanes: an even lane and an odd lane. At one end of the restaurant, you'll find the bustling kitchen, where skilled chefs prepare dishes. The even lane is equipped with three tables, numbered 2, 4, and 6, while the odd lane boasts tables numbered 1, 3, and 5.

The algorithm shown in **Figure 4.1.8** used here stands apart in its usage for planning the path to the destination/ target table number. This algorithm considers only one entry, i.e. the table number to which the robot has to move. The robot gets input from the kitchen to which table it has to move to. The algorithm checks a run to see if the robot has reached the required check-point, if yes then the task is done and completed.

We have defined two functions `Move(int Distance_in_mm, int direction)` and `rotate(int Degree, int direction)`. The `Move()` functions takes two argument for its working:

1. `Distance_in_mm`**:** Accepts distances in mm. Here we have considered it as a multiple of 1000,i.e. `lane_width`.

2. `direction`**:** Here we give the direction in which the robot has to move, it will be either '0' or '1'. If `direction = 1` then the robot moves `forward()` on the other hand if `direction = 0` then the robot moves `reverse()`.

On the other hand the `rotate()` function takes two argument:

1. `Degree`**:** It gives the robot the information by how much degree it has to rotate. It can take any value ranging from 0 to 360.

2. `direction`: Here we give the direction in which the robot has to rotate, it will be either '0' or '1'. If `direction = 1` then the robot turns in `right()` direction else, if `direction = 0` then the robot rotates in `left()`direction.

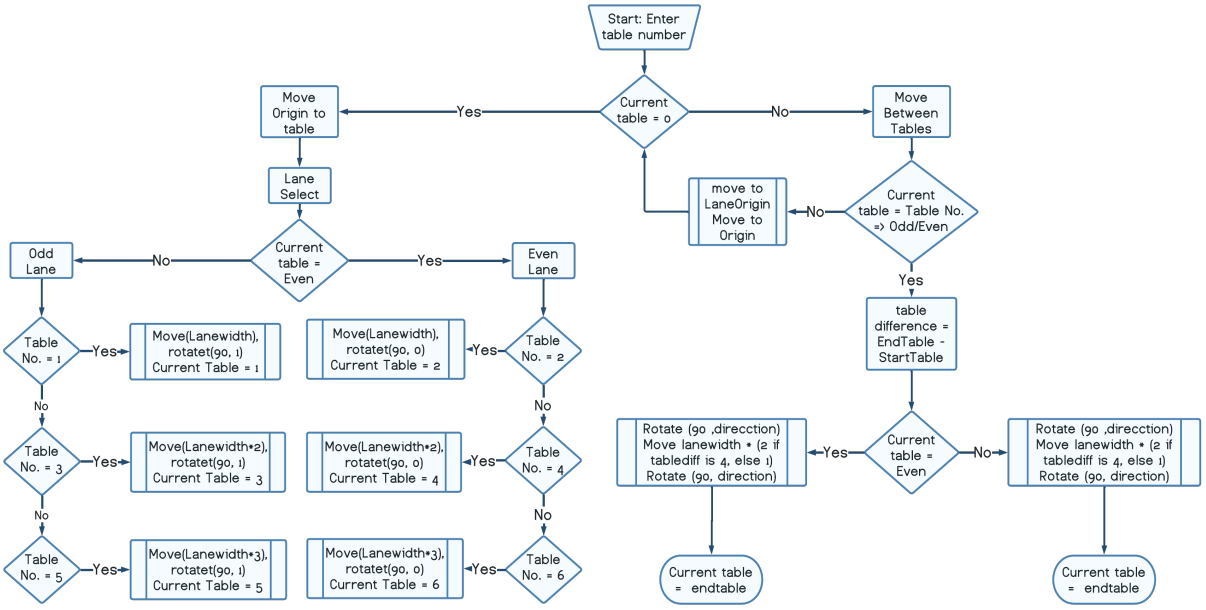


Figure 4.1.8: Path Planning Algorithm

We have globally defined two parameters namely `CurrTable` and `EndTable` which notify the robot about its current table number or the table number it has last visited and its destination table number,i.e. the table number to which it has to travel respectively.

Initially when the robot is present in the kitchen, the variable `CurrTable` is initialized to '0'. When the order is received from a table via the website, the information gets updated in the database for that particular table number. Once the order is ready to get delivered, the kitchen co-ordinator will select the desired table number from the ESP32 Interface as shown in **Figure 4.1.1b** and the `MoveOrigintoTable(int tableNumber)` function will be executed, where the `tableNumber` corresponds to the table where the order needs to be delivered.

The `MoveOrigintoTable()` function makes use of the `Move()` and `rotate()` functions for executing the task of order delivery to the specified table number. It is executed only when `CurrTable = 0` i.e. the robot is present at the kitchen side. Once the robot reaches its target table i.e. the `EndTable`, the `CurrTable` value is updated such that the `CurrTable = EndTable`.

The `MoveBetweenTables()` function is used when the robot is currently at a table and needs to move to a new table for collecting the used dishes. The function checks for the `EndTable` if it is on a different lane than the robot's `CurrTable` value. If yes, then the robot first moves to the kitchen by means of `MovetoKitchen()` function and then to the destination table by means of `MoveOrigintoTable()` function.

Whereas, if the destination table is on the same lane, the robot calculates the table difference and rotates 90 degrees to the left or right depending on the direction of the destination table. The robot moves a distance of either one or two lane widths, depending on the table difference, and updates the current table number to the destination table. All of the above mention processes are executed by means of `MoveLaneTables(int tableNumber)` function.

4.2 HARDWARE IMPLEMENTATION

The Fire Bird V robot serves as the backbone of the system, with various hardware components connected to it. As shown in the **Figure 4.2.1**, the robot is equipped with 8 IR sensors, 2 wheel encoders, and 2 DC motors. The sensors and encoders allow the robot to detect and respond to its environment, while the motors enable it to move in different directions.

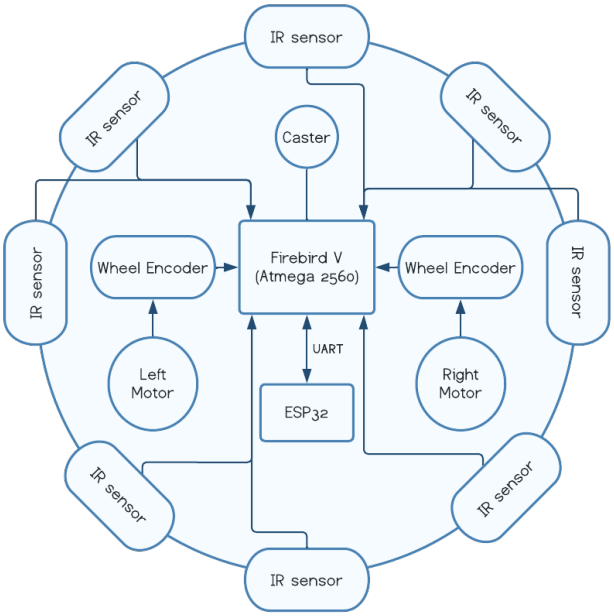


Figure 4.2.1: Hardware Implementation Block Diagram

The DC motors and wheel encoders share a 6-pin connector, with 2 pins reserved for motor AB and 3 for the wheel encoder. This connector is connected to the main board of the Fire Bird V robot, allowing it to control the motors and receive data from the encoders. Two such motors are connected to the Fire Bird V, which allows the robot to move forward, backward, and turn.

The IR sensors are PCB soldered onto the main board at a 45-degree angle to each other, providing a total of 8 sensors on the board. These sensors enable the robot to detect obstacles and avoid collisions. For communication, the Fire Bird V robot is connected to an ESP32 module via the TXRX pins. The ESP32 is connected externally using the Expansion Header shown in **Figure 4.2.2** above the Fire Bird V robot. The pin mapping at the Fire Bird V is shown in **Table 4.1**.

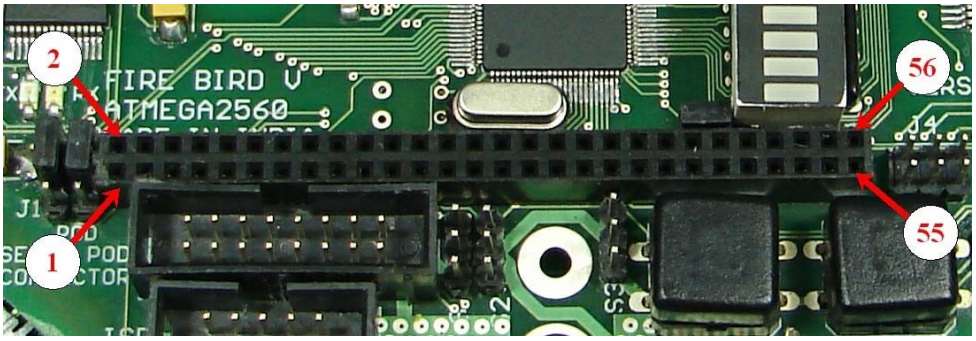


Figure 4.2.2: Expansion Header on Microcontroller Board

The ESP32 is connected to the Fire Bird V according to **Table 4.2**. The whole system is powered by a battery pack, which provides the necessary power to the Fire Bird V robot and the ESP32 module. Overall, the hardware components of the system work together to enable the Firebird5 robot to move and sense its environment, while the ESP32 module provides a means of wireless communication. This allows the robot to be remotely controlled and monitored, making it a valuable tool as a robotic waiter.

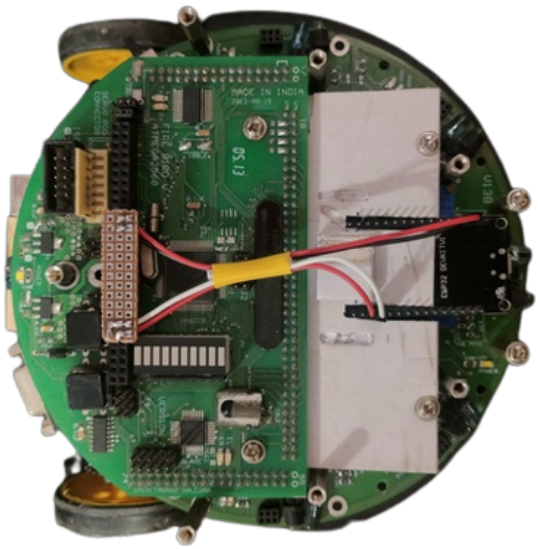


Figure 4.2.3: Integrating ESP32 with FireBird V

Table 4.1: ATMEGA2560 microcontroller board expansion header table.

Pin No.	***Function***
21	5V System Voltage. Can be used for powering up any digital device. Current Limit: 400mA
22	5V System Voltage. Can be used for powering up any digital device. Current Limit: 400mA
23	Ground
24	Ground
45	RXD3
46	TXD3

Table 4.1 shows the expansion header table for the ATMEGA2560 microcontroller board. Pins 21 and 22 both provide a system voltage of 5V, which can be used to power up any digital device. The current limit for both pins is 400mA. Pins 23 and 24 are ground pins. Pin 45 is used for RXD3 or reception, while pin 46 is used for TXD3 or transmission. These pins are essential for communication with other devices and can be used to send and receive data between the ATMEGA2560 microcontroller board and other digital devices.

Table 4.2: Connection of ESP32 with Fire Bird V's Expansion Header

ESP32 Pin	Function of ESP32 Pin	Expansion Header Pins of Fire Bird V	Function of Expansion Header Pins of Fire Bird V
Vin	Power(5V)	21 or 22	Power(5V—400mA)
Gnd	Ground	23 or 24	Ground
GPIO1	TXD0	46	RXD3
GPIO3	RXD0	45	TXD3

The Firebird5 pin mapping mentioned in **Table 4.2** includes four pins that are essential for communication with other devices. The power pins are either 21 or 22, and the ground pins are either 23 or 24. TxD3, which stands for transmission, uses pin 46, while RxD3, which stands for reception, uses pin 45. Meanwhile, the ESP32, another device connected to Firebird5, uses Vin for power, Gnd for ground, GPIO1 for TxD0 or transmission, and GPIO3 for RxD0 or reception. GPIO pins, in general, are called general-purpose input/output pins, which can be programmed to either receive or transmit signals, depending on the user's needs.

Chapter 5
RESULTS

5.1 WAITRON WEBSITE

Our 'WAITRON' website is designed to provide customers with a seamless and convenient way to order food online and pay for it securely using Stripe's payment processing system. It's built using ReactJS, a popular JavaScript library for building user interfaces, which makes the website fast, responsive, and easy to navigate.

Here are some of the key features of the restaurant website:

1. **Landing Page:** The landing/ home page shown in **Figure 5.1.1** showcases our project title and the logo for the same, and provides an overview of the project's unique features. It also includes a slideshow of some popular dishes and promotions to entice customers to order. Here are some elements that are included on our landing page.

   (a) Header: The top of the page features a header with the Waitorn logo, name, and navigation menu. The navigation menu could include links to different sections of the website, such as the menu, ordering, payment, and category.

   (b) Call to Action: A call to action button is included which says "Order Now" on the home page to encourage customers to start browsing the menu or placing an order.

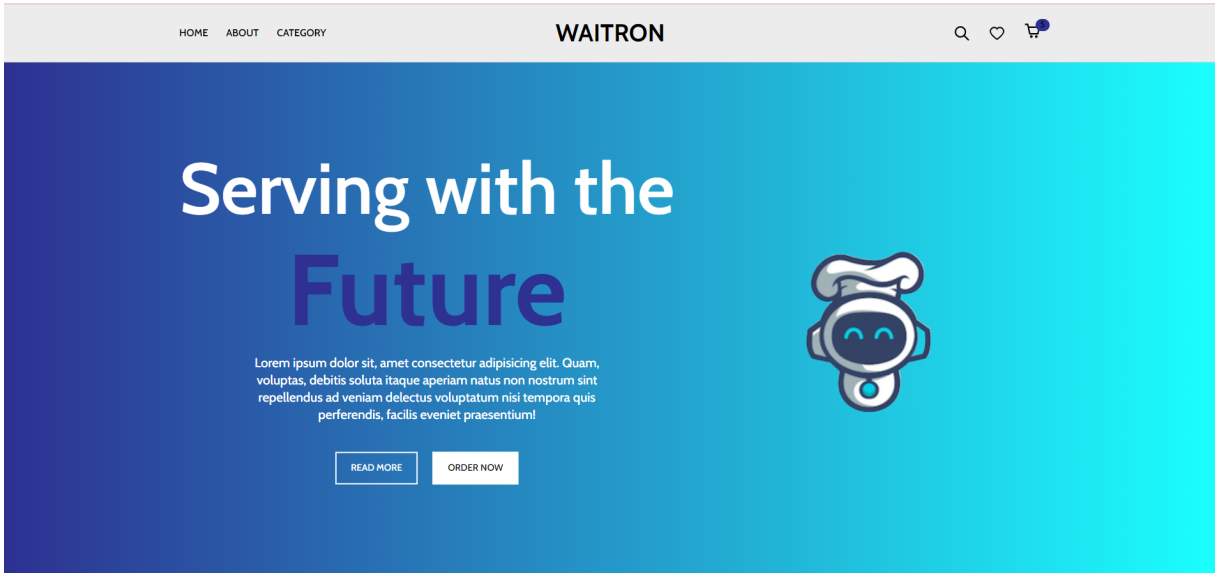


Figure 5.1.1: WAITRON Landing Page.

The home page of our website implemented using ReactJS and Stripe is designed to create a strong first impression and encourage customers to start browsing the menu and placing orders. It is visually appealing, informative, and easy to navigate, with clear calls to action and social media integration.

2. **Menu Page:** The menu page of the website shown by **Figure 5.1.2** is where customers can browse through the different food items that are available to order. The menu page is designed to be easy-to-use, visually appealing, and informative. Here are some elements included on our menu page:

   (a) Items: Each food item on the menu is displayed with a picture, name, description, and price. Customers could hover over the item to see more details or add it to their cart.

(b) Filters: The menu page includes filters to help customers narrow down their search based on dietary restrictions or preferences, such as gluten-free, vegetarian, or spicy.

(c) Nutrition Information: For health-conscious customers, the menu page displays nutritional information such as calories, fat, and protein content for each food item.

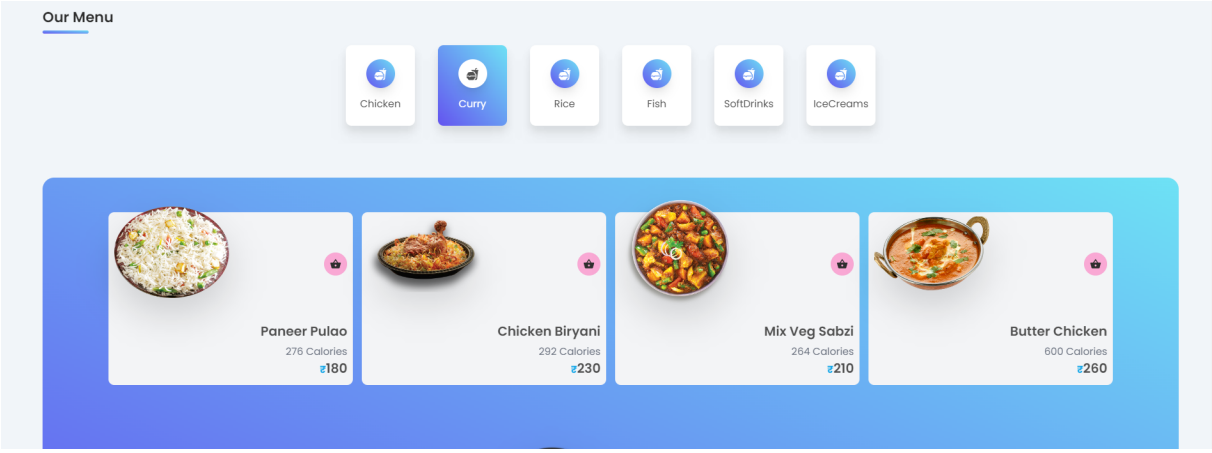


Figure 5.1.2: WAITRON Menu Page.

The Menu page provide customers with a comprehensive overview of the restaurant's offerings, while also allowing them to easily navigate and customize their orders. With clear calls to action and filters to help customers find what they're looking for, the menu page is for sure an integral part of this online ordering experience.

3. **Cart Page:** The cart page is where customers can review and adjust their orders before proceeding to the payment page. Some elements that are included on our cart page are:

(a) Cart Items: The cart page would display all the items that the customer has added to their cart. Each item would be listed with its name, quantity, price, and any customizations or special instructions.

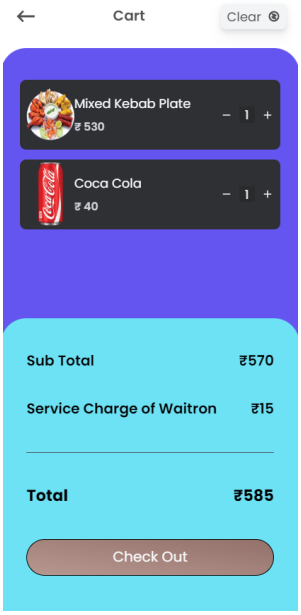


Figure 5.1.3: WAITRON Cart Page.

(b) Subtotal: The cart page displays the subtotal of the order, which would be calculated based on the prices of the items in the cart.

(c) Discounts: If the restaurant offers any discounts or promotions, the cart page would display the discount amount and apply it to the subtotal.

(d) Taxes and Fees: The cart page displays any taxes or fees that are applicable to the order, such as sales tax or delivery fees.

(e) Total: The cart page displays the total amount that the customer will be charged for their order, including taxes, fees, and any discounts.

(f) Adjustments: The cart page allows customers to adjust their orders by adding or removing items, changing quantities, or adjusting customizations or special instructions.

(g) Checkout: The cart page features a prominent checkout button that allows customers to proceed to the payment page and complete their order.

The Cart page of our website as shown in provides customers with an overview of their order, including the items, prices, and any applicable taxes or fees. With a clear checkout button and easy-to-use interface it allow customers to make adjustments to their orders before proceeding to the payment page.

5.2 DEMONSTRATION OF PROJECT

Considering you have read the above chapters which were discussed so far, here is an example demonstrating our proposed solution.

Supposedly a family comes to our restaurant and occupies table number 5 scans the QR placed on the table and orders food according to their preferences. But before placing the order, they need to enter their respective table number. After doing the needful, the order isreceived at the kitchen and our skilled chefs start preparing for the respective order. As and when the order gets prepared, the kitchen co-ordinator stacks the order onto the trays of our waiter robot and selects the required table number, the order of which is ready by means of the **Select Table Number** Interface built and hosted upon the ESP Wroom-32 board as shown by the **Figure 4.1.1b**.

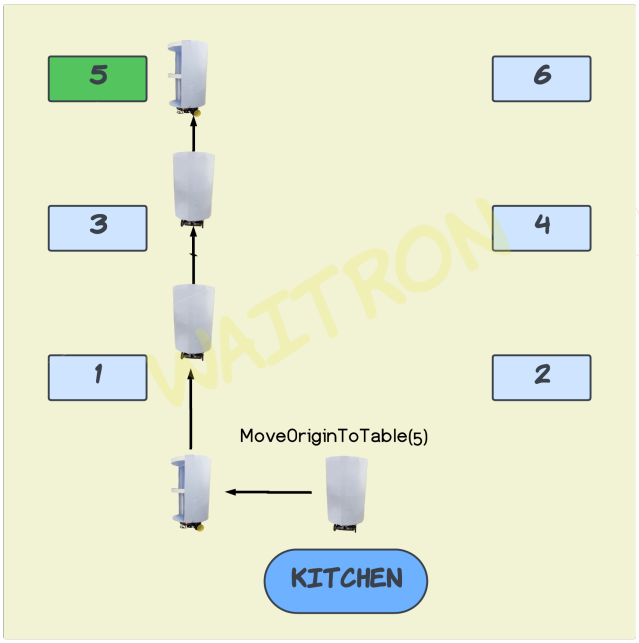


Figure 5.2.1: WAITRON delivers the order at the desired table.

Since the robot was initially present at the kichen side therefore the `CurrTable` variable was **initialised to '0'**. As soon as the co-ordinator gives command to the robot in terms of the target table number, the `MoveOriginToTable(int tableNumber)` function was executed with the fucntion parameter `tableNumber = 5`. The robot then performs the predefined activities defined by the `MoveOriginToTable(5)` function and the destination table number 5. This process of delivering order from kitchen to the target table is depicted with the help of **Figure5.2.1**

Now the order has been successfully delivered to the desired table number, the `CurrTable` variable gets updated to the table number where the robot is currently present i.e. `CurrTable = 5`. If suppose the customers of table number 4 have finished with their dining and the used dishes needs to be taken to the kitchen for washing for doing this task, the kitchen co-ordinator will select that particular table number i.e. table number 4 from the ESP Wroom-32 interface. As soon as the co-ordinator clicks on table number four, the `EndTable` variable gets updated with the selected table number i.e. `EndTable = 4`.

For performing the above mentioned task, the robot executes the `MoveBetweenTable(int tableNumber)` function with the `tableNumber` parameter been updated with the `EndTable` value,

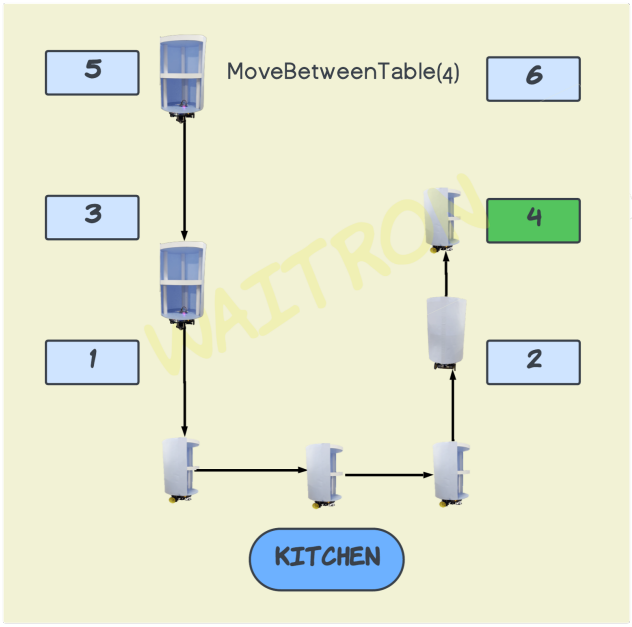


Figure 5.2.2: WAITRON move towards another table.

i.e. `tableNumber = 4` and now the finally the function `MoveBetweenTable(4)` gets executed. The process described above is depicted with the help of **Figure 5.2.2** shown above.

Note: Inside the `MoveBetweenTable(int tableNumber)` function we have included the `MoveToKitchen()` and `MoveOrigintoTable(int tableNumber)` functions respectively.i.e. When the `MoveBetweenTable(4)` function gets called, the robot will first reach the kitchen by means of `MoveToKitchen()` function and after reaching the kitchen the `MoveOriginToTable(4)` function gets executed respectively.

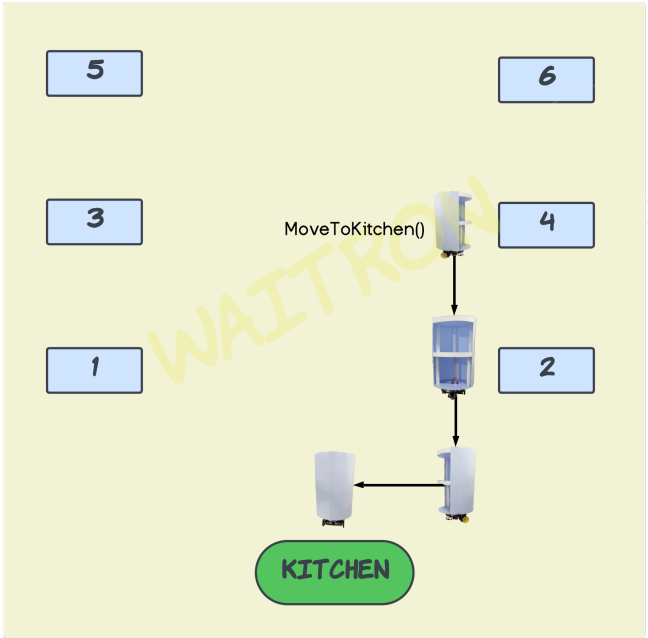


Figure 5.2.3: Robot returning towards the kitchen after collecting used dishes

At the time when the robot reaches table number '4', the `Currable` variable gets updated with the `EndTable` value, since the robot reaches the `EndTable` therefore, `Currtable = 4`. After collecting the dishes from table '4', it is time for the robot with used dishes to return back at the kitchen where the dishes will be cleaned, washed and dried before using them to serve any of the further incoming order. The task of carrying the used dishes back to the kitchen from the table is executed by the `MoveToKitchen()` function. This is shown with the help of **Figure 5.2.3** below.

With this the robot is ready again to deliver the order and collect dishes from the required tables with utmost efficiency and enthusiasm.

5.3 COMPARISON WITH EXISTING SOLUTIONS

Table 5.1 below shows a comparative study betwen our proposed solution i.e. 'WAITRON: The Smaer Waiter" and some of the currently available smart waiter robots. The comparison parameters taken into consideration are the 'Number of Trays' which determines the food carraying 'Capacity' of the robot, we also have considered 'Wheel Rotation' as one of the comparable parameters. The 'Number of Sensors' and 'Scanning Angles' of the smart waiters play avital role in determining their operational functional accuracy.

Table 5.1: Comparison of proposed solution with some existing smart waiters reported in Literature

Reference Paper	Number of Trays	Wheel Rotation (rpm)	Capacity (kilograms)	Scanning Angles (degrees)	No. of Sensors
WAITRON: The Smart Waiter	2	75	0.5	180	8
Filofetia Valentina Voinea et al. [5]	1	160	-	-	6
Md. Kamruzzaman et al. [13]	1	50-250	0.6	-	6
Karan Kaushal et al. [14]	3	-	-	180	8

It is clear from the above table that our proposed solution 'Waitron: The Smart Waiter' robot is having a neck-to-neck advantage over some of the existing smart waiters.

Chapter 6
DISCUSSION

6.1 CONCLUSION

Robots are transforming our world, offering unprecedented opportunities to improve our daily lives. From manufacturing and transportation to healthcare and education, robots are revolutionizing industries and increasing efficiency while reducing costs. The integration of robots in the service industry, such as the use of robot waiters, has the potential to revolutionize the way we interact with businesses and restaurants. With their ability to work tirelessly without the need for breaks or rest, robot waiters can offer consistent and reliable service, reducing human error and wait times for customers. Moreover, with the ability to carry multiple dishes at once, they can help restaurants save time and money while increasing their profits.

In addition to their practical applications, robots also have the potential to inspire and intrigue people, generating interest and attracting customers to businesses that use them. As more and more people become familiar with robots, they may become increasingly comfortable with their use in a wide range of settings, further enhancing their impact on society. As the technology behind robots continues to evolve, it is likely that their impact on our lives will only continue to grow, offering new and exciting possibilities for the future.

6.2 FUTURE SCOPE

6.2.1 Future Scope with Odometry

This Odometry-based smart waiter has the potential to evolve and expand in various ways in the future. Here are some possible future scopes for our smart waiter robot:

1. **Integration with other technologies:** The smart waiter can be integrated with other technologies, such as sensors, GPS, and artificial intelligence, to enhance their performance and functionality. This can improve its navigation and obstacle detection capabilities and thus enable it to operate more efficiently and safely.

2. **Customization for different industries:** It can be customized to meet the specific needs of different industries, such as hospitality, healthcare, and retail. For example, in healthcare, smart waiters could be used to transport medications or lab samples within a hospital or clinic, while in retail, they could be used to deliver products to customers.

3. **Improved communication capabilities:** The smart waiter can be equipped with better communication capabilities to interact with customers and staff more effectively. This could include voice recognition technology, touchscreens, and other interactive features.

4. **Expansion to outdoor environments:** This smart waiter will mostly be used in indoor environments such as restaurants and hotels. In the future, it can be adapted for outdoor environments such as theme parks, stadiums, and outdoor events, where it will be used to deliver food, drinks, and other products to customers.

Overall, this smart waiter robot has the potential to become more advanced and versatile, with expanded capabilities and applications across various industries. It could play an important role in improving efficiency, safety, and customer service in businesses of all kinds.

6.2.2 Integrating with other Technologies

Apart from the advancements alone with only the odometer, the smart waiter integrated with other technologies can greatly enhance its functionality and expand the scope of applications. Here are some possible future scopes for the smart waiter by integrating it with other technologies:

1. **Computer Vision:** Integrating with computer vision technology can enhance the ability to detect and avoid obstacles and navigate in complex environments. Computer vision can also enable the smart waiter to recognize and interact with customers, improving customer service.

2. **Artificial Intelligence:** By integrating artificial intelligence, it can learn and adapt to the customer's communication, improving its answering capabilities and operational efficiency. The smart waiter can also be programmed to understand customer preferences and make personalized recommendations for food and drinks.

3. **Augmented Reality:** Integrating the smart waiter with augmented reality technology can enhance the customer experience by providing interactive menus and digital information about food and drink options. Augmented reality can also be used to provide directions and other useful information to customers.

4. **Colony of robots:** Multiple robots can simultaneously work together without being an obstacle to each other. Thus improving work efficiency and effectiveness. The restaurant can then deploy multiple robots to work at different tables

References

[1] C. Sowmya, A. Menon, G. Manvitha, M. Jayakumar and S. Vijayraghavan, "Fibonacci Path Planning Algorithm for Robotic Waiters", in IEEE Second International Conference on Control, Measurement and Instrumentation (CMI), Kolkata, India, 2021.

[2] T. Akhund, M. Siddik, M. Hossain, M. Rahman, N. Newaz and M. Saifuzzaman, "IoT Waiter Bot: A Low-Cost IoT based Multi Functioned Robot for Restaurants", in 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, 2020, pp. 1174-1178.

[3] S. Vongbunyong, S. Tripathi, K. Thamrongaphichartkul, N. Worrasittichai, A. Takutruea, and T. Prayongrak, "Simulation of Autonomous Mobile Robot System for Food Delivery in In-patient Ward with Unity", in 15th International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP), Bangkok, Thailand, 2020.

[4] Zahid Abbas, Saad Ahmed khan, Shafqat Abbas, Aleena Mazhar, Moshin Gulzar, "Automatic Café Management System Using Waiter Robot", in 2nd International Conference on Communication, Computing and Digital Systems(C-CODE), 2019.

[5] F. Voinea, I. Deaconu, A. Chirilă, A. Lupaşcu, and V. Năvrăpescu, "4WD Automatic Robot for Feeding Animals from a Shelter", in Electric Vehicle International Conference, Romania, 2019.

[6] J. López, D. Pérez, E. Zalama, and J. Gómez-García-Bermejo, "Bellbot-a hotel assistant system using mobile robots," International Journal of Advanced Robotic Systems, vol. 10, no. 1, p. 40, 2013.

[7] R. Yang and L. Cheng, "Path Planning of Restaurant Service Robot Based on A-star Algorithms with Updated Weights", in 12th International Symposium on Computational Intelligence and Design (ISCID) Date of Conference, Hangzhou, China, 2019.

[8] M. Pinto and A. de Oliveira, "Robotics Development Using Real-Time Embedded System", in Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium on Robotics (SBR) and 2019 Workshop on Robotics in Education (WRE), Rio Grande, Brazil, 2019.

[9] P. Denysyuk and V. Teslyuk, "Development of mobile robot using LIDAR technology based on Arduino controller", in XIV-th International Conference on Perspective Technologies and Methods in MEMS Design (MEMSTECH), Lviv, Ukraine, 2018.

[10] A. Yadav, D. Yadav, S. Gupta, D. Kumar, and P. Kumar, "Online Food Court Payment System using Blockchain Technology", in 5th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON), Gorakhpur, India, 2018.

[11] C. Domokos, B. Séra, K. Simon, L. Kovács and T. Szakács, "Netfood: A Software System for Food Ordering and Delivery", in IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY), Subotica, Serbia, 2018.

[12] A. Antony and Sivraj P, "Food Delivery Automation in Restaurants Using Collaborative Robotics", in International Conference on Inventive Research in Computing Applications (ICIRCA 2018), Coimbatore, India, 2018.

[13] M. Kamruzzaman and M. Tareq, "Design And Implementation of A Robotic Technique Based Waiter", in 3rd International Conference on Electrical Information and Communication Technology (EICT), Khulna, Bangladesh, 2017.

[14] K. Kaushal, K. Yadav, V. Vaibhav, C. Sharma, L. Gupta, and T. Tripathy, "The E-Restaurant", in Ninth International Conference on Contemporary Computing (IC3), Noida, India, 2016.

[15] A. Cheong, M. Lau, E. Foo, J. Hedley, and J. W. Bo, "Development of a Robotic Waiter System," in International Federation of Automatic Control, 2016.

[16] G. Sawadwuthikul et al., "Visual Goal Human-Robot Communication Framework With Few-Shot Learning: A Case Study in Robot Waiter System," IEEE Transactions on Industrial Informatics, Mar. 2022.

[17] M. Asif, M. ur Rahman, and Z. H. Khan, "Waiter Robot - Solution to Restaurant Automation," in Multidisciplinary And Data Science Research Center, Nov. 2015.

[18] D. Salgotra, A. H. Khan, H. Mithaiwala, M. Mithaiwala, and R. B. Kakkeri, "Restaurant Waiter Robot," SAMRIDDHI, 2020.

[19] A. Eksiri and T. Kimura, "Restaurant Service Robots Development in Thailand and Their Real Environment Evaluation," J. Robot. Mechatron., Vol.27, No.1, pp. 91-102, 2015.

[20] G. Abou Haidar, N. Hachem, and M. Awad, "Smart device waiter using AI image and Voice Recognition with an IPS navigation system," in 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS). IEEE, 2021, pp. 16–21.

[21] S. Yu Hing, C. Chi Hung and L. Kam Fat, "Smart Elderly Care Robot", in IEEE 2nd International Workshop on System Biology and Biomedical Systems (SBBS) Date of Conference, Taichung, Taiwan, 2020.

[22] A. Abdulkareem, V. Ogunlesi, A. Afolalu, and A. Onyeakagbu, "Development of a Smart Autonomous Mobile Robot for Cafeteria Management," vol. 10, pp. 1672–1685, 01 2019.

[23] P. R. Kshitiz Rathore, Monica Chhabi and Prof. J.S.Morbale, "Smart restaurant food menu system," Journal of Emerging Technologies and Innovative Research, vol. 5, no. 5, pp. 963–964, 2018. P. Guo, H. Shi, S. Wang, L. Tang, and Z. Wang, "An ROS Architecture for Autonomous Mobile Robots with UCAR Platforms in Smart Restaurants," Machines, vol. 10, no. 10, p. 844, 2022.

[24] P. Guo, H. Shi, S. Wang, L. Tang, and Z. Wang, "An ROS Architecture for Autonomous Mobile Robots with UCAR Platforms in Smart Restaurants," Machines, vol. 10, no. 10, p. 844, 2022.

[25] Y. Ashra, A. Kholapure, P. Sawant, and D. Dalvi, "Automation of cafeteria services in covid-19," International Journal of New Innovations in Engineering and Technology, vol. 18, no. 2, 2021.

[26] T.-Y. Lin, K.-R. Wu, Y.-S. Chen, W.-H. Huang, and Y.-T. Chen, "Takeout Service Automation With Trained Robots in the Pandemic-Transformed Catering Business," IEEE Robotics and Automation Letters, vol. 6, no. 2, pp. 903–910, 2021.

[27] A. Y. S. Wan, Y. D. Soong, E. Foo, W. L. E. Wong, and W. S. M. Lau, "Waiter robots conveying drinks," Technologies, vol. 8, no. 3, p. 44, 2020.

[28] A. H. S. Hamdany, L. H. Albak, and R. R. O. Al-Nima, "Wireless waiter robot," TEST Engineering and Management, The Mattingley Publishing Co., Inc, vol. 81, pp. 2486–2494, 2019.

APPENDIX

Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 135 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-Chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 64K/128K/256K Bytes of In-System Self-Programmable Flash
  - 4K Bytes EEPROM
  - 8K Bytes Internal SRAM
  - Write/Erase Cycles:10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/ 100 years at 25°C
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
    Endurance: Up to 64K Bytes Optional External Memory Space
- JTAG (IEEE std. 1149.1 compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - Four 16-bit Timer/Counter with Separate Prescaler, Compare- and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four 8-bit PWM Channels
  - Six/Twelve PWM Channels with Programmable Resolution from 2 to 16 Bits (ATmega1281/2561, ATmega640/1280/2560)
  - Output Compare Modulator
  - 8/16-channel, 10-bit ADC (ATmega1281/2561, ATmega640/1280/2560)
  - Two/Four Programmable Serial USART (ATmega1281/2561,ATmega640/1280/2560)
  - Master/Slave SPI Serial Interface
  - Byte Oriented 2-wire Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 54/86 Programmable I/O Lines (ATmega1281/2561, ATmega640/1280/2560)
  - 64-pad QFN/MLF, 64-lead TQFP (ATmega1281/2561)
  - 100-lead TQFP, 100-ball CBGA (ATmega640/1280/2560)
  - RoHS/Fully Green
- Temperature Range:
  - -40°C to 85°C Industrial
- Ultra-Low Power Consumption
  - Active Mode: 1 MHz, 1.8V: 500 µA
  - Power-down Mode: 0.1 µA at 1.8V
- Speed Grade:
  - ATmega640V/ATmega1280V/ATmega1281V:
    0 - 4 MHz @ 1.8 - 5.5V, 0 - 8 MHz @ 2.7 - 5.5V
  - ATmega2560V/ATmega2561V:
    0 - 2 MHz @ 1.8 - 5.5V, 0 - 8 MHz @ 2.7 - 5.5V
  - ATmega640/ATmega1280/ATmega1281:
    0 - 8 MHz @ 2.7 - 5.5V, 0 - 16 MHz @ 4.5 - 5.5V
  - ATmega2560/ATmega2561:
    0 - 16 MHz @ 4.5 - 5.5V

8-bit AVR® Microcontroller with 64K/128K/256K Bytes In-System Programmable Flash

ATmega640/V
ATmega1280/V
ATmega1281/V
ATmega2560/V
ATmega2561/V

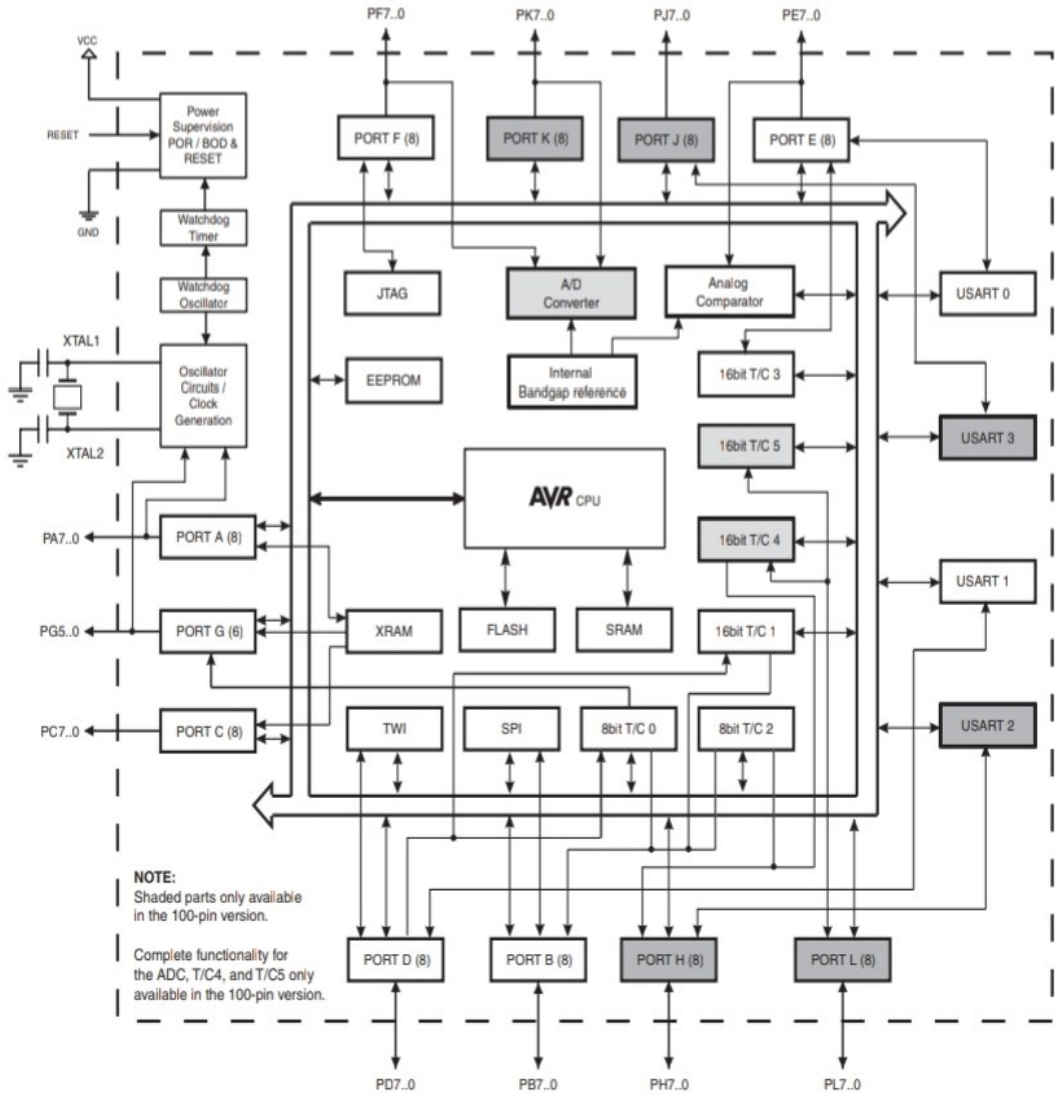
Preliminary

3. Overview

The ATmega640/1280/1281/2560/2561 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega640/1280/1281/2560/2561 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

3.1 Block Diagram

Figure 3-1. Block Diagram



SHARP GP2D120

GP2D120

General Purpose Type Distance Measuring Sensors

■ Features

1. Less influence on the color of reflective objects, reflectivity
2. Line-up of distance output/distance judgement type
   Distance output type (analog voltage) : **GP2D120**
   Detecting distance : 4 to 30cm
3. External control circuit is unnecessary

■ Applications

1. TVs
2. Personal computers
3. Amusement equipment
4. Copiers

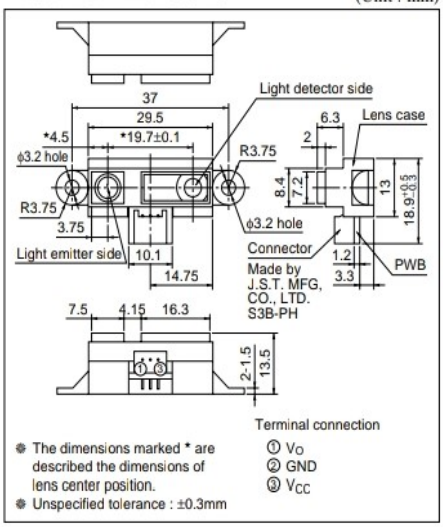
■ Absolute Maximum Ratings

(Ta=25°C, Vcc=5V)

Parameter	Symbol	Rating	Unit
Supply voltage	V_{CC}	−0.3 to +7	V
Output terminal voltage	V_O	−0.3 to Vcc +0.3	V
Operating temperature	T_{opr}	−10 to +60	°C
Storage temperature	T_{stg}	−40 to +70	°C

■ Outline Dimensions

(Unit : mm)



SHARP GP2D120

■ Recommended Operating Conditions

Parameter	Symbol	Rating	Unit
Operating supply voltage	V_{CC}	4.5 to +5.5	V

■ Electro-optical Characteristics

(Ta=25°C, V_{CC}=5V)

Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Distance measuring range	ΔL	*1 *2	4	–	30	cm
Output terminal voltage	V_O	L=30cm *1	0.25	0.4	0.55	V
Difference of output voltage	ΔV_O	Output change at L=30cm to 4cm *1	1.95	2.25	2.55	V
Average Dissipation current	I_{CC}	L=30cm *1	–	33	50	mA

Note) L : Distance to reflective object.

*1 Using reflective object : White paper (Made by Kodak Co. Ltd. gray cards R-27 · white face, reflective ratio ; 90%).

*2 Distance measuring range of the optical sensor system.

Fig.1 Internal Block Diagram

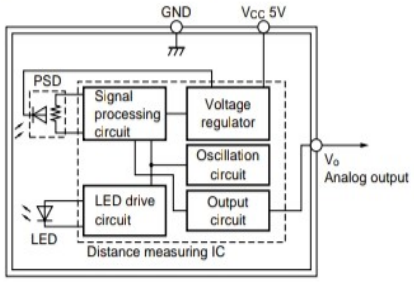
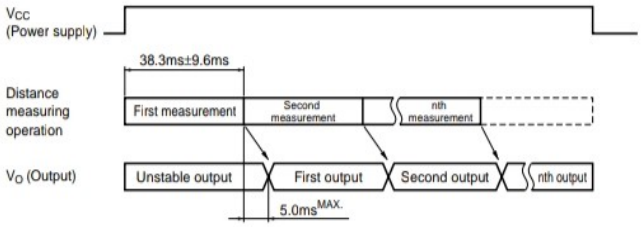


Fig.2 Timing Chart



1. Features

1.1 Hardware Features

- Single chip USB to asynchronous serial data transfer interface.
- Entire USB protocol handled on the chip - No USB-specific firmware programming required.
- UART interface support for 7 or 8 data bits, 1 or 2 stop bits and odd / even / mark / space / no parity.
- Fully assisted hardware or X-On / X-Off software handshaking.
- Data transfer rates from 300 baud to 3 Megabaud (RS422 / RS485 and at TTL levels) and 300 baud to 1 Megabaud (RS232).
- 256 byte receive buffer and 128 byte transmit buffer utilising buffer smoothing technology to allow for high data throughput.
- FTDI's royalty-free VCP and D2XX drivers eliminate the requirement for USB driver development in most cases.
- In-built support for event characters and line break condition.
- New USB FTDIChip-ID™ feature.
- New configurable CBUS I/O pins.
- Auto transmit buffer control for RS485 applications.
- Transmit and receive LED drive signals.
- New 48MHz, 24MHz,12MHz, and 6MHz clock output signal Options for driving external MCU or FPGA.
- FIFO receive and transmit buffers for high data throughput.
- Adjustable receive buffer timeout.
- Synchronous and asynchronous bit bang mode interface options with RD# and WR# strobes.
- New CBUS bit bang mode option.
- Integrated 1024 Bit internal EEPROM for storing USB VID, PID, serial number and product description strings, and CBUS I/O configuration.
- Device supplied preprogrammed with unique USB serial number.
- Support for USB suspend and resume.
- Support for bus powered, self powered, and high-power bus powered USB configurations.
- Integrated 3.3V level converter for USB I/O .
- Integrated level converter on UART and CBUS for interfacing to 5V - 1.8V Logic.
- True 5V / 3.3V / 2.8V / 1.8V CMOS drive output and TTL input.
- High I/O pin output drive option.
- Integrated USB resistors.
- Integrated power-on-reset circuit.
- Fully integrated clock - no external crystal, oscillator, or resonator required.
- Fully integrated AVCC supply filtering - No separate AVCC pin and no external R-C filter required.
- UART signal inversion option.
- USB bulk transfer mode.
- 3.3V to 5.25V Single Supply Operation.
- Low operating and USB suspend current.
- Low USB bandwidth consumption.
- UHCI / OHCI / EHCI host controller compatible
- USB 2.0 Full Speed compatible.
- -40°C to 85°C extended operating temperature range.
- Available in compact Pb-free 28 Pin SSOP and QFN-32 packages (both RoHS compliant).

1.2 Driver Support

Royalty-Free VIRTUAL COM PORT (VCP) DRIVERS for...

- Windows 98, 98SE, ME, 2000, Server 2003, XP.
- Windows Vista / Longhorn*
- Windows XP 64-bit.*
- Windows XP Embedded.
- Windows CE.NET 4.2 & 5.0
- MAC OS 8 / 9, OS-X
- Linux 2.4 and greater

Royalty-Free D2XX *Direct* Drivers (USB Drivers + DLL S/W Interface)

- Windows 98, 98SE, ME, 2000, Server 2003, XP.
- Windows Vista / Longhorn*
- Windows XP 64-bit.*
- Windows XP Embedded.
- Windows CE.NET 4.2 & 5.0
- Linux 2.4 and greater

The drivers listed above are all available to download for free from the FTDI website. Various 3rd Party Drivers are also available for various other operating systems - see the FTDI website for details.

* Currently Under Development. Contact FTDI for availability.

1.3 Typical Applications

- USB to RS232 / RS422 / RS485 Converters
- Upgrading Legacy Peripherals to USB
- Cellular and Cordless Phone USB data transfer cables and interfaces
- Interfacing MCU / PLD / FPGA based designs to USB
- USB Audio and Low Bandwidth Video data transfer
- PDA to USB data transfer
- USB Smart Card Readers
- USB Instrumentation
- USB Industrial Control
- USB MP3 Player Interface
- USB FLASH Card Reader / Writers
- Set Top Box PC - USB interface
- USB Digital Camera Interface
- USB Hardware Modems
- USB Wireless Modems
- USB Bar Code Readers
- USB Software / Hardware Encryption Dongles

3. Block Diagram

3.1 Block Diagram (Simplified)

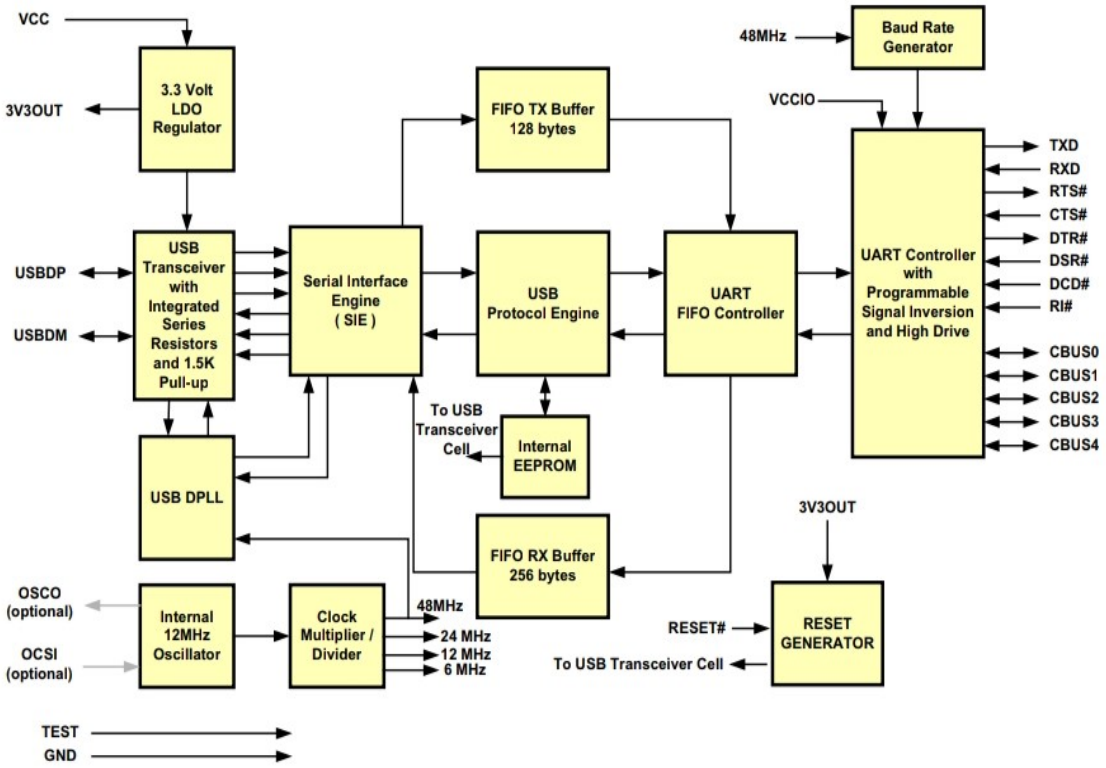


Figure 1 - FT232R Block Diagram

3.2 Functional Block Descriptions

3.3V LDO Regulator - The 3.3V LDO Regulator generates the 3.3V reference voltage for driving the USB transceiver cell output buffers. It requires an external decoupling capacitor to be attached to the 3V3OUT regulator output pin. It also provides 3.3V power to the 1.5kΩ internal pull up resistor on USBDP. The main function of this block is to power the USB Transceiver and the Reset Generator Cells rather than to power external logic. However, external circuitry requiring a 3.3V nominal supply at a current of around than 50mA could also draw its power from the 3V3OUT pin, if required.

USB Transceiver - The USB Transceiver Cell provides the USB 1.1 / USB 2.0 full-speed physical interface to the USB cable. The output drivers provide 3.3V level slew rate control signalling, whilst a differential receiver and two single ended receivers provide USB data in, SEO and USB Reset condition detection. This Cell also incorporates internal USB series resistors on the USB data lines, and a 1.5kΩ pull up resistor on USBDP.

USB DPLL - The USB DPLL cell locks on to the incoming NRZI USB data and provides separate recovered clock and data signals to the SIE block.

Internal 12MHz Oscillator - The Internal 12MHz Oscillator cell generates a 12MHz reference clock input to the x4 Clock multiplier. The 12MHz Oscillator is also used as the reference clock for the SIE, USB Protocol Engine and UART FIFO controller blocks

Clock Multiplier / Divider - The Clock Multiplier / Divider takes the 12MHz input from the Oscillator Cell and generates the 48MHz, 24MHz, 12MHz, and 6MHz reference clock signals. The 48Mz clock reference is used for the USB DPLL and the Baud Rate Generator blocks.

Serial Interface Engine (SIE) - The Serial Interface Engine (SIE) block performs the Parallel to Serial and Serial to Parallel conversion of the USB data. In accordance to the USB 2.0 specification, it performs bit stuffing / un-stuffing and CRC5 / CRC16 generation / checking on the USB data stream.

USB Protocol Engine - The USB Protocol Engine manages the data stream from the device USB control endpoint. It handles the low level USB protocol (Chapter 9) requests generated by the USB host controller and the commands for controlling the functional parameters of the UART.

FIFO TX Buffer (128 bytes) - Data from the USB data out endpoint is stored in the FIFO TX buffer and removed from the buffer to the UART transmit register under control of the UART FIFO controller.

FIFO RX Buffer (256 bytes) - Data from the UART receive register is stored in the FIFO RX buffer prior to being removed by the SIE on a USB request for data from the device data in endpoint.

UART FIFO Controller - The UART FIFO controller handles the transfer of data between the FIFO RX and TX buffers and the UART transmit and receive registers.

UART Controller with Programmable Signal Inversion and High Drive - Together with the UART FIFO Controller the UART Controller handles the transfer of data between the FIFO RX and FIFO TX buffers and the UART transmit and receive registers. It performs asynchronous 7 / 8 bit Parallel to Serial and Serial to Parallel conversion of the data on the RS232 (RS422 and RS485) interface. Control signals supported by UART mode include RTS, CTS, DSR , DTR, DCD and RI. The UART Controller also provides a transmitter enable control signal pin option (TXDEN) to assist with interfacing to RS485 transceivers. RTS / CTS, DSR / DTR and X-On / X-Off handshaking options are also supported. Handshaking, where required, is handled in hardware to ensure fast response times. The UART also supports the RS232 BREAK setting and detection conditions. A new feature, programmable in the internal EEPROM allows the UART signals to each be individually inverted. Another new EEPROM programmable feature allows a high signal drive strength to be enabled on the UART interface and CBUS pins.

Baud Rate Generator - The Baud Rate Generator provides a x16 clock input to the UART Controller from the 48MHz reference clock and consists of a 14 bit prescaler and 3 register bits which provide fine tuning of the baud rate (used to divide by a number plus a fraction or "sub-integer"). This determines the Baud Rate of the UART, which is programmable from 183 baud to 3 million baud.

The FT232R supports all standard baud rates and non-standard baud rates from 300 Baud up to 3 Megabaud. Achievable non-standard baud rates are calculated as follows -

Baud Rate = 3000000 / (n + *x*)

where n can be any integer between 2 and 16,384 (= 2^{14}) and *x* can be a sub-integer of the value 0, 0.125, 0.25, 0.375, 0.5, 0.625, 0.75, or 0.875. When n = 1, *x* = 0, i.e. baud rate divisors with values between 1 and 2 are not possible.

This gives achievable baud rates in the range 183.1 baud to 3,000,000 baud. When a non-standard baud rate is required simply pass the required baud rate value to the driver as normal, and the FTDI driver will calculate the required divisor, and set the baud rate. See FTDI application note AN232B-05 for more details.

RESET Generator - The integrated Reset Generator Cell provides a reliable power-on reset to the device internal circuitry on power up. A RESET# input pin is provided to allow other devices to reset the FT232R. RESET# can be tied to VCCIO or left unconnected, unless it is a requirement to reset the device from external logic or an external reset generator I.C.

Internal EEPROM - The internal EEPROM in the FT232R can be used to store USB Vendor ID (VID), Product ID (PID), device serial number, product description string, and various other USB configuration descriptors. The internal EEPROM is also used to configure the CBUS pin functions. The device is supplied with the internal EEPROM settings preprogrammed as described in Section 10.

4. Device Pin Out and Signal Descriptions

4.1 28-LD SSOP Package

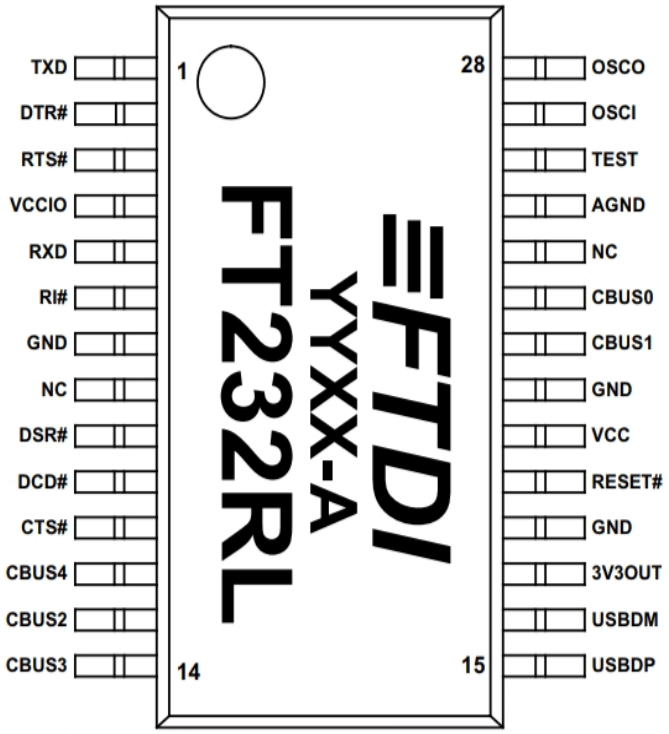


Figure 2 - 28 Pin SSOP Package Pin Out

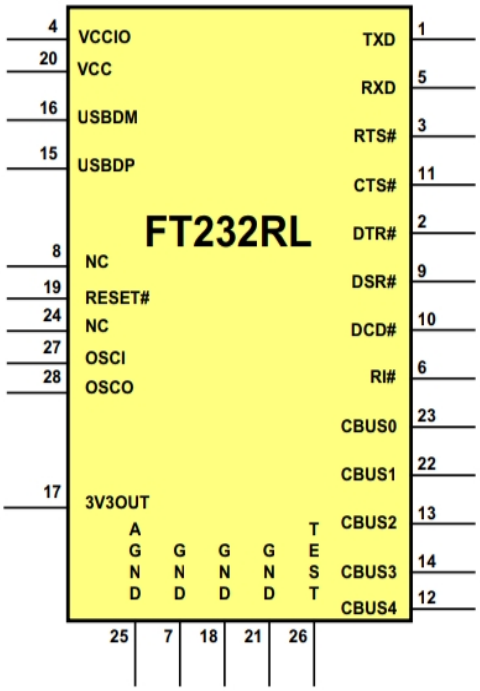


Figure 3 - 28 Pin SSOP Package Pin Out (Schematic Symbol)

4.4 QFN-32 Package Signal Descriptions

Table 2 - QFN Package Pin Out Description

Pin No.	Name	Type	Description
USB Interface Group			
14	USBDP	I/O	USB Data Signal Plus, incorporating internal series resistor and 1.5kΩ pull up resistor to 3.3V
15	USBDM	I/O	USB Data Signal Minus, incorporating internal series resistor.
Power and Ground Group			
1	VCCIO	PWR	+1.8V to +5.25V supply to UART Interface and CBUS group pins (2,3, 6, ...,11, 21, 22, 30,..32). In USB bus powered designs connect to 3V3OUT to drive out at 3.3V levels, or connect to VCC to drive out at 5V CMOS level. This pin can also be supplied with an external 1.8V - 2.8V supply in order to drive out at lower levels. It should be noted that in this case this supply should originate from the same source as the supply to Vcc. This means that in bus powered designs a regulator which is supplied by the 5V on the USB bus should be used.
4, 17, 20	GND	PWR	Device ground supply pins
16	3V3OUT	Output	3.3V output from integrated L.D.O. regulator. This pin should be decoupled to ground using a 100nF capacitor. The prime purpose of this pin is to provide the internal 3.3V supply to the USB transceiver cell and the internal 1.5kΩ pull up resistor on USBDP. Up to 50mA can be drawn from this pin to power external logic if required. This pin can also be used to supply the FT232R's VCCIO pin.
19	VCC	PWR	3.3V to 5.25V supply to the device core.
24	AGND	PWR	Device analog ground supply for internal clock multiplier
Miscellaneous Signal Group			
5, 12, 13, 23, 25, 29	NC	NC	No internal connection.
18	RESET#	Input	Can be used by an external device to reset the FT232R. If not required can be left unconnected or pulled up to VCCIO.
26	TEST	Input	Puts the device into I.C. test mode. Must be tied to GND for normal operation.
27	OSCI	Input	Input to 12MHz Oscillator Cell. Optional - Can be left unconnected for normal operation. *
28	OSCO	Output	Output from 12MHz Oscillator Cell. Optional - Can be left unconnected for normal operation if internal oscillator is used. *
**UART Interface and CBUS Group ** **			
30	TXD	Output	Transmit Asynchronous Data Output.
31	DTR#	Output	Data Terminal Ready Control Output / Handshake signal.
32	RTS#	Output	Request To Send Control Output / Handshake signal.
2	RXD	Input	Receive Asynchronous Data Input.
3	RI#	Input	Ring Indicator Control Input. When remote wake up is enabled in the internal EEPROM taking RI# low can be used to resume the PC USB host controller from suspend.
6	DSR#	Input	Data Set Ready Control Input / Handshake signal.
7	DCD#	Input	Data Carrier Detect Control input.
8	CTS#	Input	Clear to Send Control input / Handshake signal.
9	CBUS4	I/O	Configurable CBUS I/O Pin. Function of this pin is configured in the device internal EEPROM. Factory Default function is SLEEP#. See CBUS Signal Options, Table 3.
10	CBUS2	I/O	Configurable CBUS I/O Pin. Function of this pin is configured in the device internal EEPROM. Factory Default function is TXDEN. See CBUS Signal Options, Table 3.
11	CBUS3	I/O	Configurable CBUS I/O Pin. Function of this pin is configured in the device internal EEPROM. Factory Default function is PWREN#. See CBUS Signal Options, Table 3.
21	CBUS1	I/O	Configurable CBUS I/O Pin. Function of this pin is configured in the device internal EEPROM. Factory Default function is RXLED#. See CBUS Signal Options, Table 3.
22	CBUS0	I/O	Configurable CBUS I/O Pin. Function of this pin is configured in the device internal EEPROM. Factory Default function is TXLED#. See CBUS Signal Options, Table 3.

* Contact FTDI technical support for details on how to use an external crystal, ceramic resonator, or oscillator with the FT232R.

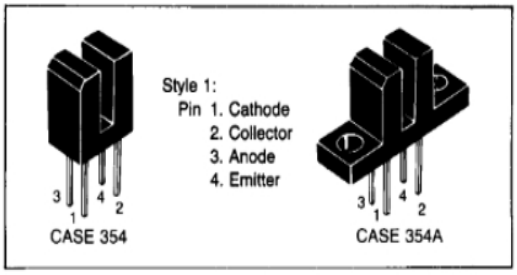
** When used in Input Mode, these pins are pulled to VCCIO via internal 200kΩ resistors. These pins can be programmed to gently pull low during USB suspend (PWREN# = "1") by setting an option in the internal EEPROM.

OPTOELECTRONICS — (continued)

Couplers/Interrupters

Slotted Couplers/Interrupter Modules

Slotted couplers consist of an infrared emitting diode facing a photodetector in a molded plastic housing. A slot in the housing between the emitter and the detector provides a means of interrupting the signal. A wide selection of standard and custom housings and detector functions is available. All IREDs and photodetectors in the miniature Case 349 (see Silicon Photodetectors) can be used in these housings.



Transistor Output ($V_{(BR)CEO}$ = 30 V)

Device	Current Transfer Ratio (CTR) % Min	@ I_F mA	V_{CE} Volts	$V_{CE(sat)}$ Volts Max	@ I_F mA	I_C mA	t_{on}, t_{off}* Typ µs	V_{CC} Volts	R_L Ω	I_F mA	V_F Volts Max	@ I_F mA	Case
MOC7811	5.0	20	5.0	0.4	30	1.8	12/60	5.0	2.5K	30	1.8	50	354A
MOC7812	10	20	5.0	0.4	20	1.8	12/60	5.0	2.5K	30	1.8	50	Style 1
MOC7813	20	20	5.0	0.4	20	1.8	12/60	5.0	2.5K	30	1.8	50	
MOC7821	5.0	20	5.0	0.4	30	1.8	12/60	5.0	2.5K	30	1.8	50	354
MOC7822	10	20	5.0	0.4	20	1.8	12/60	5.0	2.5K	30	1.8	50	Style 1
MOC7823	20	20	5.0	0.4	20	1.8	12/60	5.0	2.5K	30	1.8	50	

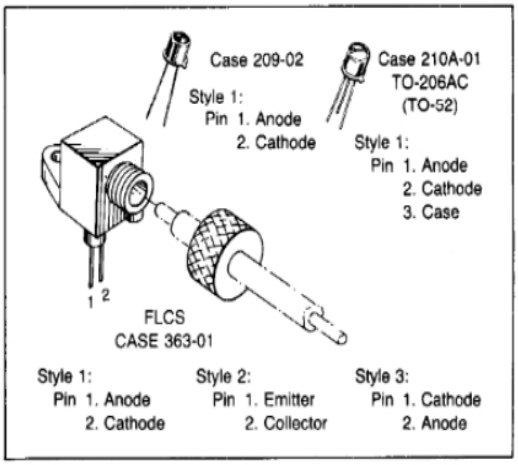
3

Fiber Optic Devices

Motorola offers high performance Infrared Emitters and Detectors for fiber optic systems. Devices are available for systems requiring greater than 100 MHz analog bandwidth over several kilometers or requiring very low cost with up to 10 MHz bandwidth over short distances.

The packages fit directly into standard fiber optic connector systems. All devices are spectrally matched to minimum attenuation regions of most fiber optic cables.

The Fiber Optic Low Cost System (FLCS) package houses infrared emitters and detectors and has a molded lens which efficiently couples the light to and from the cable. The package is complete with the fiber alignment and locking mechanism and the means for attaching to a board.



Infrared Emitters

Designed as infrared sources for fiber optic systems. MFOE200 is compatible with AMP #227015; MFOE1200, MFOE1201 and MFOE1202 are compatible with AMP #228756-1 and Amphenol #905-138-5001 receptacles.

Device	Total Power Output mW Typ	@ I_F mA	t_{on}/t_{off} ns Typ	λ nm Typ	Case
MFOE71	3.5	100	25	820	363-01 Style 1 (FLCS)
MFOE200	3.0	100	250	940	209-02 Style 1
MFOE1200	0.9	100	(>70 MHz bw)	820	210A-01
MFOE1201	1.5	100	(>100 MHz bw)	820	Style 1
MFOE1202	2.4	100	(>100 MHz bw)	820	

INTEGRATED CIRCUITS
UNITRODE

L293
L293D

Push-Pull Four Channel Driver

FEATURES

- Output Current 1A Per Channel (600mA for L293D)
- Peak Output Current 2A Per Channel (1.2A for L293D)
- Inhibit Facility
- High Noise Immunity
- Separate Logic Supply
- Over-Temperature Protection

DESCRIPTION

The L293 and L293D are quad push-pull drivers capable of delivering output currents to 1A or 600mA per channel respectively. Each channel is controlled by a TTL-compatible logic input and each pair of drivers (a full bridge) is equipped with an inhibit input which turns off all four transistors. A separate supply input is provided for the logic so that it may be run off a lower voltage to reduce dissipation.

Additionally the L293D includes the output clamping diodes within the IC for complete interfacing with inductive loads.

Both devices are available in 16-pin Batwing DIP packages. They are also available in Power SOIC and Hermetic DIL packages.

TRUTH TABLE

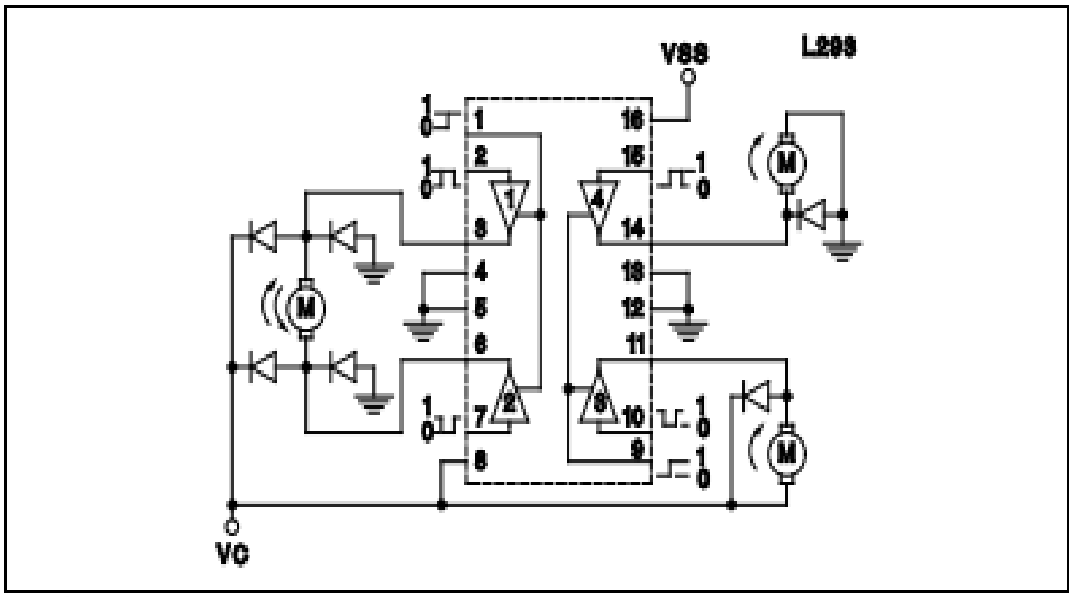
Vi (each channel)	V_{INH}	Vo
H	H	H
L	H	L
H	L	X**
L	L	X**

'Relative to the considered channel
''High output impedence

ABSOLUTE MAXIMUM RATINGS

Collector Supply Voltage, Vc 36V
Logic Supply Voltage, Vss 36V
Input Voltage, Vi 7V
Inhibit Voltage, V_{INH} 7V
Peak Output Current (Non-Repetitive), I_{OUT} (L293) 2A
I_{OUT} (L293D) 1.2A
Total Power Dissipation
at Tground-pins = 80°C, N Batwing pkg, (Note) 5W
Storage and Junction Temperature, Tstg, Tj . . . . . -40 to +150°C
Note: Consult packaging section of Databook for thermal limitations and considerations of packages.

BLOCK DIAGRAM



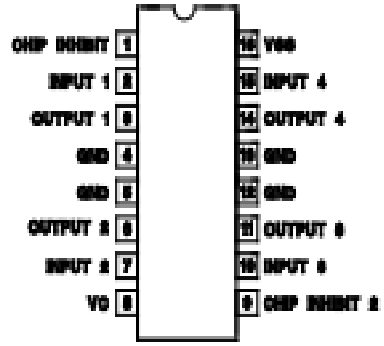
Note: Output diodes are internal in L293D.

8/94

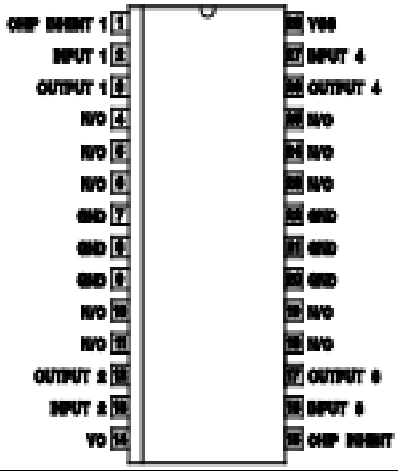
L293
L293D

CONNECTION DIAGRAMS

DIL-16 (TOP VIEW)
N Package, SP Package



SOIC-28 (TOP VIEW)
DWP Package



ELECTRICAL CHARACTISTICS: (For each channel, V_C = 24V, V_{SS} = 5V, T_{AMB} = 25°C, unless otherwise specified; $T_A = T_J$)

PARAMETER	SYMBOL	TEST CONDITION	MIN.	TYP.	MAX.	UNITS
Collector Supply Voltage	V_C				36	V
Logic Supply Voltage	V_{SS}		4.5		36	V
Collector Supply Current	I_C	V_I = L, I_O = 0, V_{INH} = H		2	6	mA
		V_I = H, I_O = 0, V_{INH} = H		16	24	mA
		V_{INH} = L			4	mA
Total Quiescent Logic Supply Current	I_{SS}	V_I = L, I_O = 0, V_{INH} = H		44	60	mA
		V_I = H, I_O = 0, V_{INH} = H		16	22	mA
		V_{INH} = L		16	24	mA
Input Low Voltage	V_{IL}		-0.3		1.5	V
Input High Voltage	V_{IH}	$V_{SS} \leq 7V$	2.3		V_{SS}	V
		$V_{SS} \geq 7V$	2.3		7	V
Low Voltage Input Current	I_{IL}	V_I = 0V			-10	µA
High Voltage Input Current	I_{IH}	V_I = 4.5V		30	100	µA
Inhibit Low Voltage	$V_{INH,L}$		-0.3		1.5	V
Inhibit High Voltage	$V_{INH,H}$	$V_{SS} \leq 7V$	2.3		V_{SS}	V
		V_{SS} >7V	2.3		7	V
Low Voltage Inhibit Current	$V_{INH,L}$			-30	-100	µA
High Voltage Inhibit Current	$V_{INH,H}$				10	µA
Source Output Saturation Voltage	$V_{CE(sat)H}$	I_O = -1A (-0.6A for L293D)		1.4	1.8	V
Sink Output Saturation Voltage	$V_{CE(sat)L}$	I_O = 1A (0.6A for L293D)		1.2	1.8	V
Clamp Diode Forward Voltage (L293D only)	V_F	I_F = 0.6A		1.3		V
Rise Time	T_R	0.1 to 0.9 V_O (See Figure 1)		100		ns
Fall Time	T_F	0.9 to 0.1 V_O (See Figure 1)		350		ns
Turn-on Delay	T_{ON}	0. 5 V_I to 0.5 V_O (See Figure 1)		750		ns
Turn-off Delay	T_{OFF}	0. 5 V_I to 0.5 V_O (See Figure 1)		200		ns