IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

Wi-Fi Controlled Self Balancing Bot Using ESP32

[1]C G Gokul , [2]Dr.Sudha M S, [3]D R Balaji, [4]Chetas J, [5]Chinmay, [6]G N Harish Kumar

[13456]Student, [2]Asso.Professor,

[1]Department of Electronics and Communication Engineering,

[1]Cambridge Institute of Technology, K R Puram, Bengaluru-36, India

Abstract: This paper presents the design and implementation of a self-balancing robot controlled via Wi-Fi using the ESP32 microcontroller. The proposed system leverages the capabilities of the ESP32, a powerful and versatile microcontroller, to achieve stable and responsive control of the robot's motion. The self-balancing mechanism is achieved through the integration of an inertial measurement unit (IMU) and a PID controller, allowing the robot to maintain its balance while moving. The Wi-Fi connectivity provided by the ESP32 enables remote control of the robot using a smartphone or computer, offering flexibility and convenience to the user. The system's hardware and software components are detailed, including the design of the chassis, motor control, sensor integration, and communication protocols. Experimental results demonstrate the effectiveness and reliability of the proposed system in achieving stable and precise control of the self-balancing robot over Wi-Fi. This project serves as a practical example of utilizing modern microcontroller technology for developing interactive and autonomous robotic systems.

Index Terms - MPU6050, gyroscope, Wi-Fi controlled, ESP32

I. INTRODUCTION

In recent years, the field of robotics has witnessed remarkable advancements, with self-balancing robots emerging as a fascinating area of research and development. These robots, equipped with sensors and control systems, possess the ability to maintain stability and navigate autonomously, making them versatile platforms for various applications. This research paper delves into the design, development, and analysis of a self-balancing robot utilizing the ESP32 microcontroller and the MPU6050 inertial measurement unit (IMU).

The integration of the ESP32, a powerful and versatile microcontroller, and the MPU6050, a highly sensitive and accurate IMU, offers a robust foundation for constructing a sophisticated self-balancing robot. The ESP32 provides the computational power and connectivity required for real-time control and communication, while the MPU6050 contributes precise motion sensing capabilities critical for maintaining equilibrium.

The significance of this research lies in its potential applications across diverse industries, ranging from education and research to industrial automation and surveillance. Self-balancing robots can navigate through challenging terrains, making them suitable for scenarios where traditional wheeled robots may face limitations. Furthermore, the incorporation of the ESP32 and MPU6050 allows for a cost-effective and efficient solution, making this technology accessible to a broader audience.

This paper aims to address key aspects of the self-balancing robot's design and implementation. Starting with an overview of the ESP32 and MPU6050, the paper will explore the integration process and the development of control algorithms essential for stabilizing the robot. Additionally, the research will investigate the impact of various parameters on the robot's performance, providing insights into optimization strategies.

The research methodology involves hardware and software implementation, testing, and analysis. Through systematic experiments, the study aims to evaluate the robot's stability, responsiveness, and overall performance in different scenarios. The findings will contribute valuable insights into the design considerations and potential enhancements for self-balancing robots using the ESP32 and MPU6050.This research endeavors to provide a comprehensive understanding of the development and optimization of self-balancing robots using the ESP32 microcontroller and MPU6050 IMU. By shedding light on the intricacies of this technology, the paper aims to inspire further research and innovation in the realm of autonomous robotics.

Unveiling the ESP32: At the heart of this marvel lies the ESP32, a versatile microcontroller renowned for its robust performance and extensive connectivity capabilities. Equipped with WiFi, Bluetooth, and a myriad of GPIO pins, the ESP32 serves as the backbone for our futuristic creation.

Mastering Self-Balancing Technology: Building upon the principles of dynamic stabilization, our bot boasts an intricate system that enables it to maintain equilibrium on two wheels. Utilizing gyroscopic sensors and intelligent algorithms, it navigates obstacles and terrain with remarkable precision, offering a glimpse into the realm of autonomous mobility.

Empowering Control via WiFi: Gone are the days of tethered control; our bot harnesses the power of WiFi to offer seamless remote operation. Through a user-friendly interface, enthusiasts can effortlessly maneuver the bot, commanding it to traverse various environments with unparalleled ease.

Delving into the Build: From hardware assembly to software integration, our project encompasses a comprehensive guide for enthusiasts eager to embark on their own journey of creation. With detailed instructions and insightful tips, individuals of all skill levels can partake in the excitement of building their very own WiFi-controlled self-balancing bot.
Embracing Innovation: Beyond its practical applications, our creation serves as a testament to the boundless possibilities of innovation. Whether utilized for educational purposes, recreational pursuits, or even as a platform for further experimentation, the WiFi-Controlled Self-Balancing Bot stands as a beacon of technological ingenuity.

II. LITERATURE SURVEY

Self-balancing robots, with their ability to stay upright on two wheels, have captivated researchers and hobbyists alike. The MPU6050 sensor, combining an accelerometer and gyroscope, plays a key role in their operation. Here's an overview of the existing literature on self-balancing robots:

Yan Zhang et al.[1] presented a Desing of Self Balancing Robot Based on ESP32.
This paper designs a two wheeled self balancing robot based on esp32. The robot collects inertia data through mpu9250 and motor data through Hall encoder. PID algorithm is used to control motor to realize robot balance. At the same time, the upper computer software is designed, which can change the PID parameters of the robot in real time, obtain the pitch angle and velocity of the robot in real time, and control the motion state of the robot.

Ghanta Sai Krishna et al.[2] proffered a Two-Wheeled Self Balancing Robot Using PID Controller And Deep Reinforcement Learning.
The fundamental concept of the proposed framework "Epersist" is to overcome the challenge of counterbalancing an initially unstable system by delivering robust control mechanisms, Proportional Integral Derivative (PID), and Reinforcement Learning (RL). Moreover, the micro-controller NodeMCU ESP32 and inertial sensor in the Epersist employ fewer computational procedures to give accurate instruction regarding the spin of wheels to the motor driver, which helps control the wheels and balance the robot. This framework also consists of the mathematical model of the PID controller and a novel self-trained advantage actor- critic algorithm as the RL agent.

A.V. Putov et al.[3] demonstrated Self-balancing Robot Autonomous Control System.
Stabilization and balancing are both very important and widespread problems in robotics and control systems spheres of interest nowadays. This paper describes the design of the body, control algorithm and performance results of such two-wheeled vertically self-balancing robot. The robot was designed and assembled in Saint Petersburg Electrotechnical University "LETI" at the automatic control systems department.

R.G. Vidhya et al.[4] depicted Smart Design and Implementation of Self Adjusting Robot using Arduino.
This idea is based on the basic pendulum theory. A framework goes through a cycle of self-adjustment to reinforce itself from inside. This study intends to overcome the difficulties of stabilizing an unstable structure by supplying the robot with the essential components for self-adjustment and balance. Data from

the accelerometer and gyrator are used to compute the robot's accurate three-dimensional location, which is then relayed to a controller.

Anmol Singh Shekhawat et al.[5] came out with Design and Control of Two-Wheeled Self-Balancing Robot using Arduino.
This paper discusses the design, construction, and control of a two-wheel study level, small self-balancing robot. The system architecture comprises of Arduino Nano microcontroller board, DC stepper motor, gyroscope, and accelerometer sensor. PID controller controls it through the program written in Arduino IDE. A detailed description of constituting materials in making the robot has been provided. Flow chart and closed loop control diagrams have been given for understanding the controlling of the robot.

III. METHADOLOGY

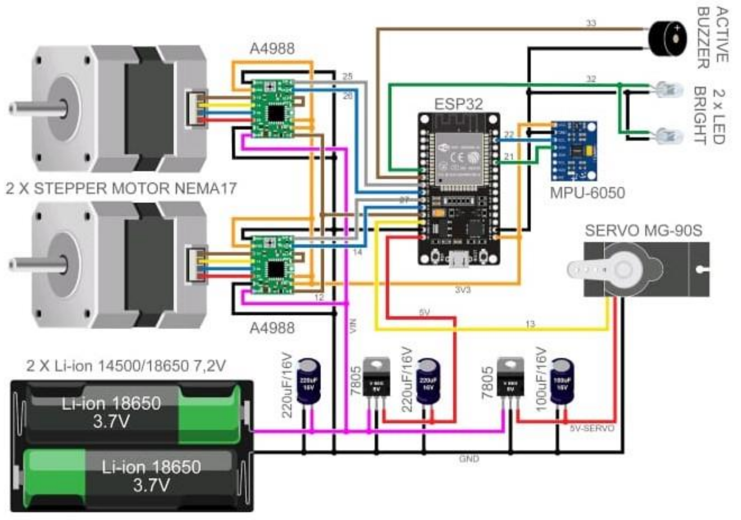

Fig.1 Connections of the Bot

The above figure Fig.1 shows the connections that are made for the Bot to work. The components includes Stepper Motor NEMA17, Motor drivers, MPU6050, ESP32 and capacitors. It also contains batteries to power the circuit.

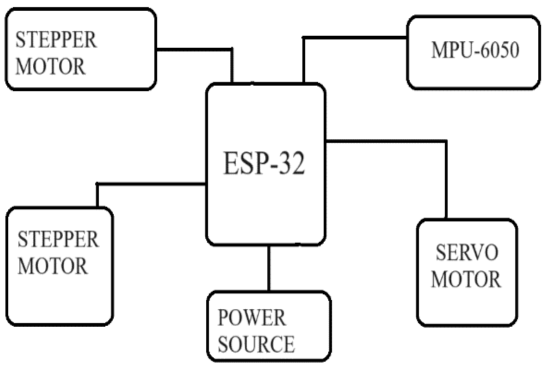

Fig.2 Block Diagram of the Bot

The above figure Fig.2 shows the block diagram of the Self balancing Bot in which the main component is ESP32, to which the other components like Stepper motor, MPU6050, Servo motor are connected and powered by the power source.
Creating a Wi Fi controlled self-balancing robot using an ESP32 involves integrating various components, including sensors, motors, and the ESP32 microcontroller, and programming them to work together. Below

is a general overview of the key components and steps involved: Two geared DC motors with sufficient torque for balancing are connected to the motor driver A4988 which controls the DC motors. The motor driver inputs are connected to the appropriate General Purpose Input/Output (GPIO) pins on the ESP32. The Inertial Measurement Unit (IMU) that is MPU6050 is connected to the ESP32, ensuring the correct wiring for power, ground, and data lines. Power the motors and the ESP32 with a suitable power source as shown in Fig1 and Fig 2, ensuring it provides enough current for the motors. Securely mount the motors, wheels, motor driver, and ESP32 on the robot chassis. Write python code to read data from the gyroscope and accelerometer of MPU6050.

Implement a PID (Proportional-Integral-Derivative) control algorithm that uses the sensor data to adjust the motor speeds and keep the robot balanced. Set up the ESP32 as a Wi-Fi station or access point. Use Message Queuing Telemetry Transport (MQTT) to handle communication between the robot and the control device. Write code to receive control commands from a remote device through Wi-Fi.

Test the self-balancing algorithm by placing the robot on a flat surface. The motors should adjust to maintain balance. Send commands from a remote device to the ESP32 over Wi-Fi. Commands can include instructions to move forward, backward, turn, etc.

The C programming code that is put into ESP32:

```
#include <Wire.h>
#include <WiFi.h>
#include <ArduinoOTA.h>
#include <Arduino.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include "Control.h"
#include "MPU6050.h"
#include "Motors.h"
#include "defines.h"
#include "globals.h"
#include <stdio.h>
#include "esp_types.h"
#include "soc/timer_group_struct.h"
#include "driver/periph_ctrl.h"
#include "driver/timer.h"
#include "driver/ledc.h"
#include "esp32-hal-ledc.h"

const char* PARAM_FADER1 = "fader1";
const char* PARAM_FADER2 = "fader2";
const char* PARAM_PUSH1 = "push1";
const char* PARAM_PUSH2 = "push2";
const char* PARAM_PUSH3 = "push3";
const char* PARAM_PUSH4 = "push4";
const char* PARAM_TOGGLE1 = "toggle1";
const char* PARAM_FADER3 = "fader3";
const char* PARAM_FADER4 = "fader4";
const char* PARAM_FADER5 = "fader5";
const char* PARAM_FADER6 = "fader6";

/* Wifi Crdentials */
String sta_ssid = "";     // set Wifi network you want to connect to
String sta_password = "";       // set password for Wifi network

unsigned long previousMillis = 0;

AsyncWebServer server(80);

void initMPU6050() {
  MPU6050_setup();
  delay(500);
  MPU6050_calibrate();
}

void initTimers();

void notFound(AsyncWebServerRequest *request) {
  request->send(404, "text/plain", "Not found");
}

void setup() {
  Serial.begin(115200);        // set up seriamonitor at 115200 bps
  Serial.setDebugOutput(true);
  Serial.println();
  And goes on.......
```

This is a c programming code that can be compiled in Visual Studio, CLion, NetBeans, Eclipse, CodeLite, QtCreator etc.

IV. RESULT

Building a self-balancing robot using an MPU6050 sensor offers a wealth of valuable outcomes. Beyond the immediate thrill of seeing, it defies gravity, the project serves as a powerful demonstration of key concepts in feedback control, sensor fusion, and microcontroller programming. This hands-on experience fosters essential skills in problem-solving, electronics, mechanics, and coding, laying the groundwork for further exploration in advanced control algorithms, additional functionalities, and integration with other technologies. It's not just about building a robot; it's about igniting a passion for learning, creativity, and innovation, ultimately paving the way for exciting advancements in the field of robotics and its potential applications

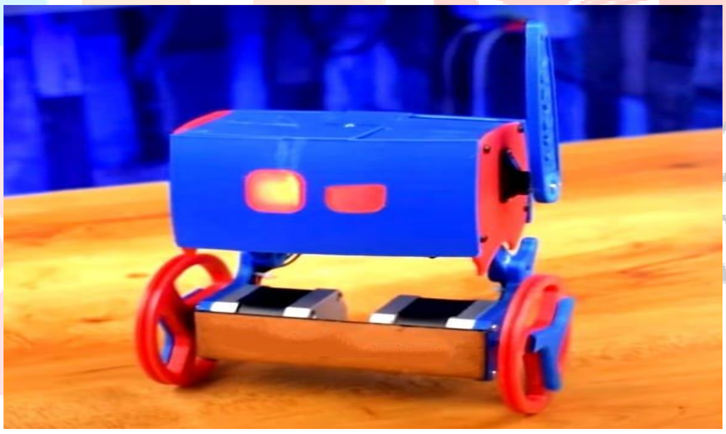

Fig.3 Picture of the Bot

The above figure Fig.3 shows the outlook of the finished project. It is made of two wheels and a small extended from right that is moveable which is controlled by ESP32 and MPU6050.

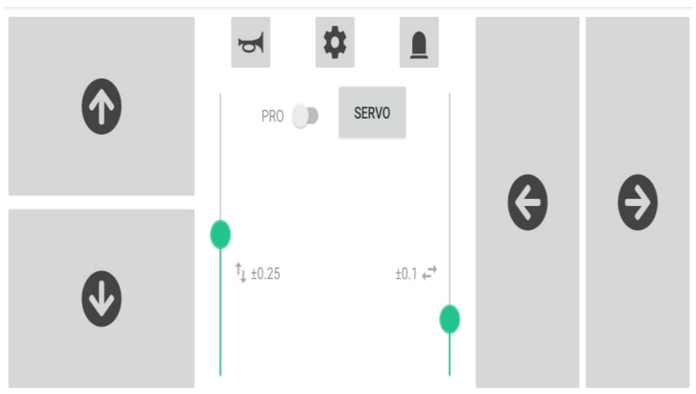

Fig.4 Picture of the Controller

The above figure Fig.4 shows the control system indicating the directions in which we can move the bot. This bot is controlled using Wi-Fi. The speciality in this bot is that it can balance on its own in only two wheels with the help of MPU6050. The arrow marks in the Fig.4 indicates the movement of bot and the servo button indicates the control of servo motor in the bot.

V. FUTURE SCOPE

The future scope for a Wi-Fi controlled self-balancing bot using ESP32 is quite promising. Here are some potential avenues for development:
Enhanced Stability and Control Algorithms: Further refining the control algorithms to improve the stability and maneuverability of the bot, allowing it to navigate various terrains and obstacles more effectively.
Integration with IoT Ecosystems: Integrate the bot into larger IoT ecosystems, enabling it to interact with other smart devices and systems, such as home automation systems or industrial IoT networks.
Autonomous Navigation: Develop autonomous navigation capabilities using techniques such as SLAM (Simultaneous Localization and Mapping) to enable the bot to navigate and explore environments without direct human control.

VI. CONCLUSION

In conclusion, the development of a Wi-Ficontrolled self-balancing bot utilizing the ESP32 microcontroller presents a significant advancement in the field of robotics and IoT. Through the integration of advanced technologies, such as accelerometers, gyroscopes, and WiFi connectivity, this project offers a versatile and efficient solution for various applications.
The ESP32 microcontroller's capabilities, including its low power consumption, robust processing power, and built-in WiFi module, make it an ideal choice for controlling the self-balancing bot. By leveraging the ESP32's Wi-Fi connectivity, users can remotely control the bot from a smartphone or computer, enhancing its accessibility and usability in different environments.
In summary, the Wi-Ficontrolled self-balancing bot using ESP32 offers a combination of advanced features, ease of use, and flexibility, making it a valuable asset for hobbyists, educators, and professionals alike. As technology continues to evolve, this project serves as a testament to the limitless potential of IoT and robotics in shaping the future of automation and connectivity.

VII. REFERENCES

[1]Yan Zhang and Wenjie Cai,"Design of Self-Balancing Robot Based on ESP32,"2021 IEEE 3rd International Conference on Civil Aviation Safety and Information Technology (ICCASIT).
https://ieeexplore.ieee.org/abstract/document/9633673

[2]Ghanta Sai Krishna, Dyavat Sumith, and Garika Akshay,"Epersist: A Two-Wheeled Self Balancing Robot Using PID Controller And Deep Reinforcement Learning,"2022 22nd International Conference on Control, Automation and Systems (ICCAS).
https://ieeexplore.ieee.org/abstract/document/10003940

[3]A.V. Putov, E.V. Ilatovskaya, and M.M. Kopichev,"Self-balancing Robot Autonomous Control System,"2021 10th Mediterranean Conference on Embedded Computing (MECO).
https://ieeexplore.ieee.org/abstract/document/9459720

[4]R.G. Vidhya, Kamlesh Singh, P John Paul, T. Aditya Sai Srinivas, Jyoti Prasad Patra, and K.V.Daya Sagar,"Smart Design and Implementation of Self Adjusting Robot using Arduino,"2022 International Conference on Augmented Intelligence and Sustainable Systems (ICAISS).
https://ieeexplore.ieee.org/abstract/document/10011083

[5]Anmol Singh Shekhawat and Yogesh Rohilla, "Design and Control of Two-wheeled Self-Balancing Robot using Arduino,"2020 International Conference on Smart Electronics and Communication (ICOSEC).
https://ieeexplore.ieee.org/abstract/document/9215421