Gymnázium Pezinok

Senecká 2, 902 01 Pezinok

**RC controller**

High school research paper

No. of field:

12-Electricalngineering, Hardware, Mechatronics

Authors

Adam Hikl

Pezinok

2025

Year of Study: VI. A sexta

Gymnázium Pezinok

Senecká 2, 902 01 Pezinok

**RC controller**

High school research paper

No. of field:

12-Electricalngineering, Hardware, Mechatronics

Establishing Body: Authors:

Bratislava Self-Governing Region Adam Hikl

Pezinok

2025 Year of Study: VI. A sexta

Consultant:

Prof. Ing. František Duchoň, PhD

**Statement of Honor**

I declare that I have independently prepared the High school research paper on the topic 'Wireless Controller' using the listed literary sources. I have not submitted or presented this work in any other competition under the auspices of the MŠVVaŠ SR. I am aware of the consequences if the stated information is not truthful.

**Acknowledgment**

I would like to express my gratitude to my consultant for providing professional resources, valuable information, and guidance in the preparation of this work. I also sincerely thank my parents for their support, trust, and assistance. Additionally, I extend my appreciation to PCBWay for their material support in the development of PCB boards.

# Table of Contents

[Table of Contents 4](#_Toc190635061)

[List of Abbreviations, Notations, and Symbols 6](#_Toc190635062)

[List of Tables, Graphs, and Illustrations 7](#_Toc190635063)

[0 Introduction 8](#_Toc190635064)

[1 Problem Statement and Literature Review 9](#_Toc190635065)

[1.1 Microcontroller 9](#_Toc190635066)

[1.1.1 Programming 9](#_Toc190635067)

[1.1.2 Communication Protocol 9](#_Toc190635068)

[1.1.3 Size and Efficiency 9](#_Toc190635069)

[1.1.4 Integration Simplicity and Cost 9](#_Toc190635070)

[2 Objectives of the Paper 10](#_Toc190635071)

[2.1 Goals 10](#_Toc190635072)

[2.2 Approach 10](#_Toc190635073)

[3 Materials and Methodology 12](#_Toc190635074)

[3.1 Components 12](#_Toc190635075)

[3.1.1 Joysticks 12](#_Toc190635076)

[3.1.2 Battery 12](#_Toc190635077)

[3.1.3 OLED Display 12](#_Toc190635078)

[3.1.4 Buttons 13](#_Toc190635079)

[3.1.5 Speakers 13](#_Toc190635080)

[3.1.6 Vibration motors 13](#_Toc190635081)

[3.1.7 Microcontroller 13](#_Toc190635082)

[3.2 Hardware design 14](#_Toc190635083)

[3.2.1 PCB sketch 14](#_Toc190635084)

[3.2.2 Controller Body Design 18](#_Toc190635085)

[3.3 Software 20](#_Toc190635086)

[3.3.1 Introduction 20](#_Toc190635087)

[3.3.2 Libraries 20](#_Toc190635088)

[3.3.3 Setup 21](#_Toc190635089)

[3.3.4 Loop 21](#_Toc190635090)

[3.3.5 Methods 21](#_Toc190635091)

[3.3.6 Process 23](#_Toc190635092)

[4 Results and discussion 25](#_Toc190635093)

[4.1 Communication 26](#_Toc190635094)

[4.2 Battery 26](#_Toc190635095)

[4.3 Customizability 26](#_Toc190635096)

[4.4 Components 26](#_Toc190635097)

[4.5 Simplicity 27](#_Toc190635098)

[4.6 User Feedback 27](#_Toc190635099)

[4.7 Competitor Analysis 28](#_Toc190635100)

[5 Conclusions of the Paper 29](#_Toc190635101)

[5.1 Goals and results 29](#_Toc190635102)

[5.2 Outlook for the Future 29](#_Toc190635103)

[5.2.1 Third version 29](#_Toc190635104)

[5.2.2 Market Introduction 29](#_Toc190635105)

[6 Summary 31](#_Toc190635106)

[List of Used Literature 32](#_Toc190635107)

[Attachments 33](#_Toc190635108)

# List of Abbreviations, Notations, and Symbols

**List of Abbreviations**

MŠVVaŠ SR – Ministry of Education, Science, Research, and Sport of the Slovak Republic

PCB – printed circuit board

BMS – battery management system

DC – direct current

V – volt

A – ampere

CC – constant current

CV – constant voltage

PD – power delivery

IMU – inertial measurement unit

LED – light emitting diode

OLED – organic light emitting diode

PLA - polylactic acid

MOSFET - metal-oxide-semiconductor field-effect transistor

UI – user interface

I²S - Inter-Integrated Circuit Sound

TTS – text to speech

**List of Notations and Symbols**

**Symbol Unit Meaning – Description**

U V Electric Voltage

I A Electric Current

Q Ah Electric Current per hour

E Wh Electric power per hour

f Hz Frequency

# List of Tables, Graphs, and Illustrations

**List of Illustrations**

**Illustration number Illustration name Page**

[Figure 1 The sketch of the first version PCB (author: Hikl, A., 2025) 14](#_Toc190635166)

[Figure 2 The sketch of the second version PCB (author: Hikl, A., 2025) 16](#_Toc190635167)

[Figure 3 The first design of the controller body (author: Hikl, A., 2025) 18](#_Toc190635168)

[Figure 4 The second design of the controller body (author: Hikl, A., 2025) 19](#_Toc190635169)

[Figure 5 The second version of the controller (author: Hikl, A., 2025) 25](#_Toc190635170)

[Figure 6 Insides of the second version of the controller (author: Hikl, A., 2025) 25](#_Toc190635171)

[Figure 7 Xbox game controller (author: Microsoft, 2025) 28](#_Toc190635172)

[Figure 8 Professional drone controller (author: Unmanned Tech Limited, 2025) 28](#_Toc190635173)

# Introduction

Robot control has undergone significant changes in recent decades, with developers and enthusiasts increasingly favoring solutions that combine simplicity, flexibility, and efficiency. Smartphone applications are often considered a convenient and accessible option for many users. However, despite their advantages, many still prefer physical controllers—a trend evident in the use of drones and RC cars, where dedicated controllers continue to dominate.

Nevertheless, there is a gap in the market. A lack of controllers designed specifically for robot developers or custom drone builders who require a high level of customization. This market gap has created the need for a device that combines physical control with modern technology and an open-source approach.

Instead of every developer having to create their own robot controller from scratch, this open-platform solution would allow them to simply use an existing, highly customizable system where they can program the functionalities, they need in just a couple of hours.

# Problem Statement and Literature Review

## Microcontroller

The most important part of the controller is the microcontroller, which we needed to select based on several criteria.

### Programming

It is essential that developers can easily connect to the controller and upload their own code. We also need to choose a platform with extensive documentation which can be used with RC controllers. This means we need a microcontroller that supports Linux or Arduino operating systems.

### Communication Protocol

We need the microcontroller to support both Wi-Fi and Bluetooth. This eliminates most Arduino models.

### Size and Efficiency

The microcontroller must be small and optimized for battery operation. This leaves us with smaller Raspberry Pi modules and the ESP32.

### Integration Simplicity and Cost

Since our testing robot operates on the ESP32 chip, and these chips are more cost-effective than other alternatives, we decided to use this microcontroller.[[1]](#footnote-2)

# Objectives of the Paper

The main idea is to develop a controller that can be fully customized both in terms of software and hardware. For instance, if a developer’s robot uses a custom communication interface, there is no need to design a custom controller, which would take a lot of time. Instead, they can use a solution like this—already having a functional platform with rich features and components, into which they can easily add their own code. This is a faster and more resource efficient solution. Thus, the primary objective of this paper is to create a functional wireless controller that aligns with this concept.

This controller must meet the following criteria, listed from most important to less essential, with the less critical being optional but desirable to have.

All important criteria must be fulfilled for the controller to be a viable solution to our problem and also in the case of future market release.

## Goals

1. Communication with the robot
2. Functional joysticks and sending data from the joysticks to the robot
3. A functional battery, so that the controller operates wirelessly without any external batteries or connections.
4. Open source, allowing developers to customize code, 3D-printed body, and various other components
5. Low cost
6. USB C charging
7. Information output to the user thorough a built-in display.
8. Additional inputs besides the joysticks
9. Localized vibration motors
10. Comfortable grip
11. Simplicity
12. USB C programming and debugging
13. Long battery life

## Approach

The following steps outline the approach to creating the controller:

1. Test communication between the ESP32 development board and the robot
2. Test individual new components considered for use
3. Design a PCB
4. Design and 3D print the controller body
5. Assemble the controller and connect all components
6. Develop or modify the software
7. Test the controller
8. Identify errors
9. After testing, determine what causes issues
10. Find the best and most effective solutions for these problems
11. Create a new version by repeating steps 2 to 11
12. If this version is sufficiently suitable for production, after finding potential user interest and thorough testing, production can gradually begin.

# Materials and Methodology

The controller has open-source code, 3D models, and PCB sketches.[[2]](#footnote-3)

## Components

All components except BMS are powered by a 3.3V voltage regulator in the second version and a 5V regulator in the first version, which is directly placed on the PCB..

### Joysticks

Joysticks are a key part of the physical control system, as they provide intuitive and accurate analog input for controlling robots. We used two joysticks: one for controlling the left track and the other for controlling the right track of the test robot. For this control, we utilize the joystick's 44-degree mobility on the Y-axis. The values are later adjusted in the microcontroller to a range from -255 to 255. The X-axis is left free for later use, or developers can customize the functionality of the X-axis as needed. Additionally, we get another input in the form of a digital signal from pressing the joysticks, corresponding to the Z-axis. This input can be assigned to any function as desired.

### Battery

Since the controller is wireless, this applies to both power and communication. We chose two lithium cells of type INR21700-48G[[3]](#footnote-4). They are connected in series, resulting in a total nominal voltage of 8.4 volts. The capacity of the cells is 4.800 Ah, and the maximum current is 9.6 A. The total energy of the battery is 34Wh, which is sufficient for over 20 hours of operation under heavy use. It is no problem to replace the cells with other ones, if they are the same size.

### OLED Display

The OLED display allows us to create a customizable user interface for advanced control, it also provides the ability to display data that the user can then modify through inputs on the controller, like settings on a smartphone.

Additionally, we can display necessary information such as the robot’s speed, battery percentage of both the robot and the controller, connectivity, and other details that developers can add later. The decision to use an OLED display was based on its advantages, such as low power consumption and high contrast between black and white.

### Buttons

In the first version of the controller, we used touch-sensitive metal contacts that only required a touch to register input on the ESP32. After careful consideration, we decided to use traditional physical buttons in the second version of the controller, similar to those found on Xbox or PS controllers. The input from these buttons is digital, meaning binary – they are either pressed or not. The signal is processed by the ESP32, where the developer can customize what function the button will trigger upon pressing. Multi-touch, where multiple buttons are pressed simultaneously to increase the number of inputs, is also supported.

### Speakers

The speakers provide yet another output for the user. We decided to use a stereo setup, with one speaker positioned on the lower left and the other mirrored on the lower right. These can provide meaningful information or play tones and songs. They add another communication channel between the controller and the user, enriching the display. The developer can use the speakers for any purpose they deem appropriate. In the first version, a chip malfunction caused issues with amplifying the signal from the ESP32 to the speakers. In the second version, we switched the chip and the mode of communication from digital I²S to classic analog. Speakers are now fully functional, and we can play various tones, melodies, notes, songs, or internet radio. We have implemented a TTS model, however, due to processor and ram constraints, its capabilities are highly limited.

### Vibration motors

The controller features another user output in the form of four vibration motors placed as far apart as possible in all four corners. We chose four vibration motors to give the controller multiple options for indicating different data. The user can differentiate which of the four motors is vibrating, thus distinguishing multiple cues. We can also differentiate cues based on the duration and strength of the vibrations. Combining these parameters allows for a broader range of information to be conveyed to the user using vibration motors. Further details can be found in video - <<https://youtu.be/7rRjT4-H6NM>>

### Microcontroller

We chose the ESP32-WROOM-32U microcontroller as it allows for easy integration with the robot and offers a wide range of functions at a relatively low cost. Additionally, it supports an external antenna connection to increase the operational range.

## Hardware design

### PCB sketch

#### First version

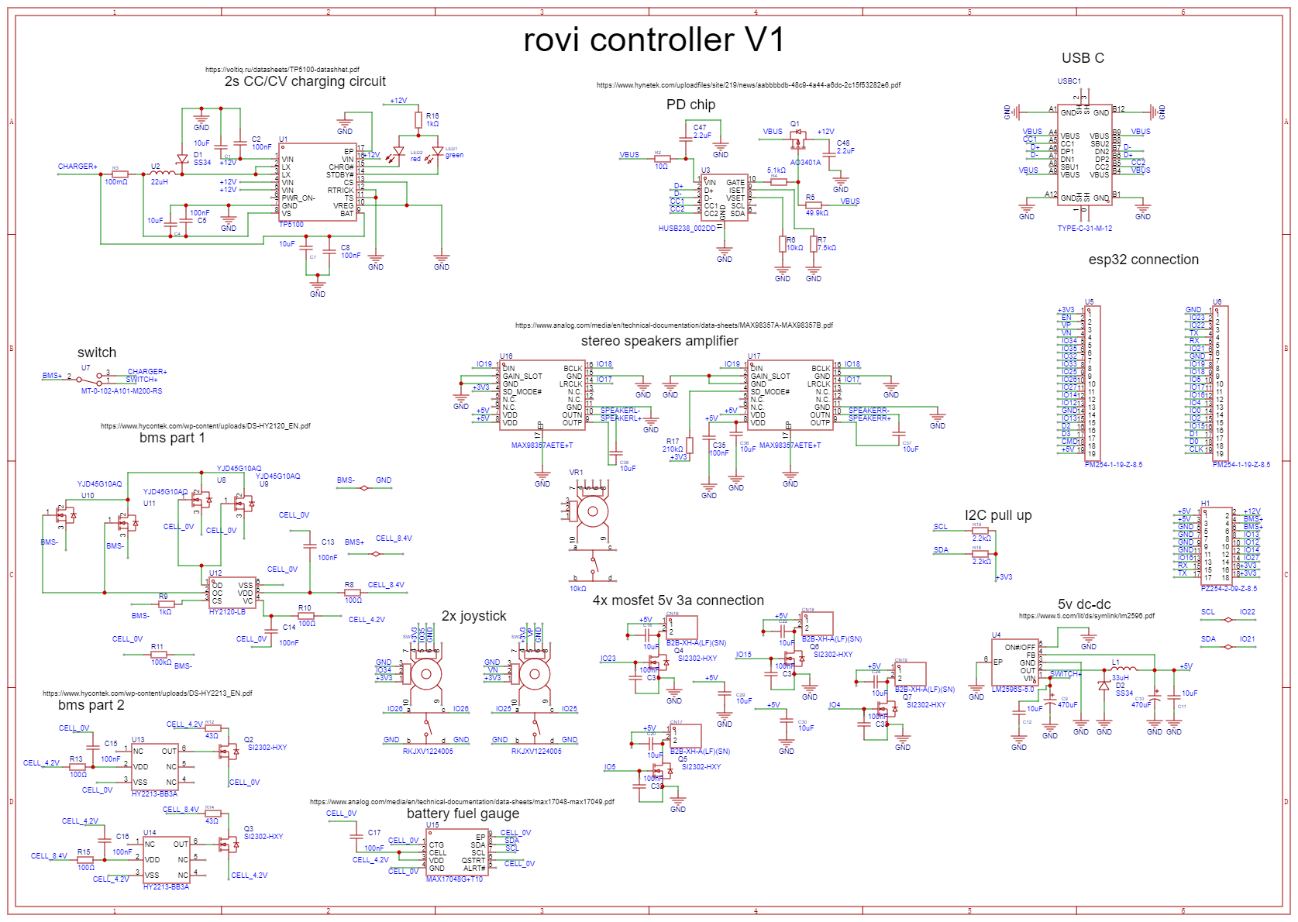


Figure 1 The sketch of the first version PCB (author: Hikl, A., 2025)

##### BMS

BMS, a Battery Management System, is essential for proper functioning of lithium cells. In the diagram, we have divided the BMS into two parts: BMS Part 1 and BMS Part 2.

BMS Part 1 is responsible for protecting the battery from excessive discharge or charge currents, as well as from low or high voltages of the battery. If such a situation occurs, the chip[[4]](#footnote-5) in BMS Part 1 interrupts the connection between the battery and the rest of the controller using four transistors. BMS Part 2 is responsible for balancing the voltage of the individual battery cells. This is crucial for maintaining a long lifespan of the lithium battery and ensuring its proper operation.[[5]](#footnote-6) Further details can be found in video - <<https://youtu.be/4ZWASpv6YaQ>>

##### 5V DC-DC Regulator

The 5V DC-DC regulator reduces the voltage from the battery to a suitable and constant 5 volts. All electrical energy consumed by the controller, except for the BMS, passes through this regulator. Specifically, we used the LM2596S-5.0[[6]](#footnote-7) chip, which has a maximum current of 3A per hour, more than sufficient for our controller. Further details can be found in the video - <<https://youtu.be/lNbmb66H4ww>>

##### CC/CV Charging Circuit

This part of the PCB is responsible for charging the battery using the CC and CV methods. However, the chip heats up and is not functional, we were unable to determine the cause of the malfunction. We tested this on multiple PCB units.

##### Stereo Speakers amplifier

These chips are designed to convert the digital signal from the ESP32 to analog and amplify it for use with speakers. However, the chip heats up and is not functional, we were unable to determine the cause of the malfunction. We tested this on multiple PCB units.

##### Switch

Mechanically disconnects the BMS from the rest of the PCB circuit

##### PD Chip

This PD chip is responsible for communication through the USB-C port with the adapter plugged into the electrical socket. It requests the correct voltage and current. The electrical power then continues to the battery charger.

##### Battery Fuel Gauge

This chip is responsible for measuring the charge status of the first battery cell. It communicates via I2C with the ESP32, where the total remaining energy in the battery is calculated as a percentage and then displayed to the user. We use the MAX17048 chip[[7]](#footnote-8).

##### I2C Pull Up

Two resistors between the SDA and SCL pins and 3.3V are required for proper communication via I2C.

##### MOSFET Transistors

Four MOSFET transistors control electrical current to four vibration motors. The input consists of four pins from ESP32, which use PWM to control how often the transistor gate opens, thus the intensity of the vibrations.

##### ESP32 Connector

We inserted ESP32 using this connector, ESP32 pins can then be used across the PCB.

##### USB C

We used USB C connector, which supports PD charging.

##### Additional Connectors

These connectors consist of 18 pins as an output from the PCB to buttons and display.

#### Second version

A diagram of a machine

Description automatically generated with medium confidence

Figure 2 The sketch of the second version PCB (author: Hikl, A., 2025)

##### BMS

BMS stays the same except lowering the number of transistors from 4 to 2 in BMS 1.

##### PD Chip

The power delivery chip stays the same without any changes.

##### I2C Pull Up

The I2C pull up circuitry stays the same without any changes.

##### 3.3V DC-DC Regulator

We replaced the initial LM2596S-5.0 chip with the LMR51430[[8]](#footnote-9), which is newer, smaller, and more efficient at both higher and lower currents. We also changed the output voltage from 5V to 3.3V. We added a green LED to indicate proper functioning of the regulator.

##### CC/CV Charging Circuit

We replaced the chip with a newer and more technologically advanced one. Despite our best efforts, we made a mistake and forgot to add a resistor between two pins. As a result, the chip mistakenly determined the battery’s temperature inappropriate for charging. After adding an external resistor, we found that the chip is functional and correctly charges the battery.

##### Stereo Speakers Amplifier

We replaced two chips with one and, instead of sending a digital signal, we now send an analog signal through pins 25 and 26. We have maintained the stereo configuration. We are using the MAX98306[[9]](#footnote-10) chip.

##### Switch

We replaced the mechanical switch with a smaller toggle button, which controls MOSFET transistor, that connects BMS to the rest of the PCB

##### USB C

USB C connector stays the same without any changes

##### MOSFET transistors

Circuit with transistors stays the same except for pull down resistors on their inputs.

##### Battery Fuel Gauge

Chip connector stays the same without any changes

##### ESP32 Connector

Connectors stay the same without any changes.

##### Additional Connectors

The output from the PCB has been changed to 7 plug-in connectors, each having 4 pins. The reason for the change is for better integration with components and cable management.

##### Buttons

We replaced the external metal contacts with physical buttons directly connected to the PCB. Each button is connected to an individual ESP32 pin.

##### IMU

The IMU unit provides information about the rotations of the controller. It also returns the controller's acceleration along the X, Y, and Z axes, as well as the acceleration of tilting, pitching, and rotating. It communicates directly with the ESP32 via the I2C communication protocol. We used the BNO086 chip[[10]](#footnote-11).

### Controller Body Design

We designed the 3D models in Autodesk Fusion 360[[11]](#footnote-12) .

#### First Version

A close-up of a device

Description automatically generated

Figure 3 The first design of the controller body (author: Hikl, A., 2025)

##### Material

We used PLA plastic as the material, and the controller body was 3D printed in two parts and then glued together. We chose PLA due to its ease of printing and good physical properties. The color we selected was white.

##### Component Placement

The entire controller was designed with the goal of compactness, to fit many components into a relatively small space. The later assembly proved to be challenging, mainly due to the bad cable management. The dimensions of the controller are as follow, width – 149mm, length – 100mm, height – 31mm, and it weighs 306 grams.

The battery cells were arranged symmetrically at the edges. The vibration motors are also symmetrically arranged, as far apart as possible in each corner, to create strong localized vibrations. The display is positioned directly in the center above the ESP32 connectors.

The controller was designed to be primarily practical, and in future versions, once the controller is fully functional, we can focus more on the aesthetics and ergonomics of the 3D model. The controller can either be laid flat or placed vertically on a surface, where it will stand on two flat surfaces at the lower ends of the battery cells.

#### Second Version

A computer generated image of a game controller

Description automatically generated

Figure 4 The second design of the controller body (author: Hikl, A., 2025)

##### Material

Apart from changing the color from white to black, no other factors were changed. Instead of gluing, we were able to add 4 screws this time, which we recessed into the body, so they won't interfere with operation. Thus, opening the controller has been greatly simplified.

##### Component Placement

In the second version, we improved the practical design. We added curvature to the wall around the joysticks to enhance rotation and prevent contact with the wall. We also succeeded in making the design more compact. The controller's dimensions are as follow, width – 141mm, length – 99mm, height – 35mm, and it weighs 279 grams, which is a 9% reduction in weight.

The battery cells were again arranged symmetrically at the opposing edges. The vibration motors are placed in each corner. The display is positioned to the right of the ESP32 to reduce the height of the controller. The ESP32 is right in the middle.

Rotating the ESP32 connectors by 90 degrees allowed for better access to the USB micro connector on the ESP32.

##### Connectors

From the back or bottoms side of the controller we added a connector. This connector allows the user to seamlessly add other attachments to their controller. For example, we experimented with a holder that serves as an arm to which end is, through yet another connector, attached a holder for mobile phone. In this simple way, the user can have both hands on the controller and simultaneously look at the phone and, for example, record something. This unfortunately is not a universal connector, as we needed to limit its dimensions to fit the design, however, it’s fairly simple and users can 3D print or buy attachments with this connector.

## Software

### Introduction

The code was written in Arduino IDE [[12]](#footnote-13) and adapted for operation with ESP32.

### Libraries

The following libraries are used and imported: WiFi.h, WiFiMulti.h, TTS.h, WiFiClientSecure.h, Arduino.h, WebSocketsClient.h, Adafruit\_GFX.h, Adafruit\_SSD1306.h, Adafruit\_MAX1704X.h, Adafruit\_BNO08x.h

We use these libraries across the whole code. Their authors and sources are listed in the list of used literature.

### Setup

In this method, which is called just once when the ESP32 is powered on, we initialize variables, set up pins, and establish connections to the display, IMU, and the MAX17048. We also start 3 processes: one that repeats every 40 milliseconds, another every second, and the third every 10 seconds.

### Loop

The loop is the main method that is being continuously executed in the code. We decided to handle most of the code outside of the loop and used separate processes. Inside the loop, we only call the webSocket.loop() method.

### Methods

#### Button\_1Change

This method is connected to the ESP32 pin and is called on the interrupt of the corresponding pin. We change the button state, either pressed or not pressed. We use separate methods for each of the four buttons.

#### compassCheck

This method is used to update the data from the IMU. First, we check if the IMU has been reset or powered on. If so, we call the setReports method.

We then request data from the IMU. If the IMU successfully updates and provides the data, we convert this data from the Quaternion system to Euler angles using the quaternionToEulerRV method, this gives us yaw, pitch, and roll.

After, we store the values in a variable and update the list of the last five values, we prepare the values for display on the screen.

#### setReports

We configure the IMU and specify which type of data it should send upon request.

#### quaternionToEuler

Here we convert values from the Quaternion coordinate system to Euler angles.

#### connectedToRoviProcedure

In this method, we perform the necessary steps for communication with the robot. Then we connect to the web server, where communication will take place using web sockets. We also notify the user of a successful connection by displaying a Wi-Fi icon on the screen and a quick 300-millisecond vibration from all vibration motors using the vibrationMotorsTaskMethod.

#### handleWebSocketMessage

We decode the message sent from the robot, identify the type of message based on predefined codes, and then perform the corresponding action.

#### webSocketEvent

This method is called when we receive any message or change from the web server. We are interested in three types of messages: DISCONNECTED; CONNECTED; and TEXT - meaning we received a text message from the robot.

#### vibrationMotorsTaskMethod

According to the parameters specified when calling this method, we activate certain vibration motors for a set duration and then turn them off.

#### updateJoysticks

In the following method, we read the analog values of the X and Y axes for both joysticks and the logical value of the Z axis, which indicates whether it is pressed or not. We then adjust the data range from 400–3600 to 255 to -255. Since the axes are oriented differently, we also adjust the direction, plus and minus of the resulting values accordingly. For the primary Y-axis, we use an exponential normalization, which helps with robot control by providing greater precision at low values—allowing fine control when the robot moves slowly—while still maintaining the full range up to 255. These values are then stored in variables.

#### sendJoystickData

First, we create a text string where we will add the current values from the joysticks. If a value has not changed, we append only "n" instead of the actual value. At the end, if at least one value has changed, we send the entire text string.

#### dinoGameLoop

A simple game where the user controls the letter "D" using the first button, giving the object the ability to jump. The goal is to avoid the incoming "#" from the right by jumping over it. The game gradually speeds up, and "#" moves more unpredictably and faster. If we detect that the "D" object intersects with the "#" object, the game is interrupted, and the method dinoGameOver is triggered. After three seconds, the game resets and restarts. We use these games for fun demonstration of the controller’s abilities and features.

#### dinoGameOver

We display the score that the user achieved in the game on the screen. We wait for 3 seconds and then reset the necessary variables to their original values.

#### gyroGameLoop

This game uses data from the IMU unit. It displays a small circle on the screen, and the user can move it by tilting and pitching the controller. If the controller is tilted forward, the circle on the screen moves forward as well.

#### drawDefaultUI

In this method, we display the updated Y-axis values from both joysticks and the current pitch, roll, and yaw values from the IMU unit, ranging from -180 to 180.

#### gameSlectionUI

We display a game selection menu on the screen, allowing the user to navigate and start a game or play a sample sound using buttons or return to the default UI.

#### playImperialMarch & playFuturisticSound

We use these methods to demonstrate the functioning of speakers. In playImperialMarch we play a short part of the Imperial March[[13]](#footnote-14) melody, it uses only 6 various sounds and from these it creates the melody. During every startup of the controller we execute playFuturisticSound method in which a frequency is played that exponentially decreases from 10.000Hz to 100Hz.

#### saySomeWords

This method is used to show how the TTS works, unfortunately due to the lack of storage space, memory, and low processing power it is limited in precision. It can play only 255 different sounds. From these, it then creates words and sentences.

### Process

#### Every 40 milliseconds

First, we update the joystick input using the method updateJoysticks and then check if the controller is connected to the robot. If it is, we send the data using the method sendJoystickData. We call the method compassCheck, in which we update data from the IMU unit. We verify if the display is connected and if we are not in a game, if not, we execute the drawDefaultUI method.

For each of the four buttons, we check whether it is pressed or not. If it is pressed, we activate the corresponding vibration motor. It will only be activated if we are not currently in a game. If we press the first (A), second (B), and third (X) buttons simultaneously, a new window will appear on the display using the gameSelectionUI method, allowing us to start one of the games. If we are already in a game during such an action, we will do the opposite and exit the game, returning to the original interface.

#### Every second

In this process, we update or retrieve data that is not as time sensitive as the data in every 40 milliseconds process. First, we send a connection confirmation to the robot and check the received confirmation from the robot. If we were connected to the robot and did not receive a connection confirmation within two seconds, we evaluate that the robot is no longer connected and inform the user. Next, we determine the battery charge percentage using the MAX17048 chip. If this percentage has changed, we update it on the display.

#### Every 10 seconds

The only thing we do in this process is attempt to establish communication, if none is established yet, that is, in our case, connect to the robot via Wi-Fi. If we succeed, we execute the connectedToRoviProcedure method

# Results and discussion

We will only focus on the current version, in this case, the second version.

A close-up of a remote control

AI-generated content may be incorrect.

Figure 5 The second version of the controller (author: Hikl, A., 2025)

A broken electronic device with wires

AI-generated content may be incorrect.

Figure 6 Insides of the second version of the controller (author: Hikl, A., 2025)

In the image, we can see the current version of the controller. After turning it on by pressing the button on the top, the system starts up within one to two seconds. The first thing the user notices is the display, which lights up and shows basic information. The controller is now ready for operation. You can watch the video demonstrating robot control at the following link <<https://youtu.be/JOBfqMeLTB0>>

## Communication

Connecting to the robot is very simple. Both devices just need to be powered on. Since we have full control over the software on both the robot and the controller, we can customize and implement our own handshake protocol. When the controller connects to the robot, they exchange identification and perform basic authentication. Once verified, the controller establishes a connection and can communicate with the robot. This connection is indicated by a Wi-Fi icon appearing on the display and a short vibration lasting 300 milliseconds.

Both the controller and the robot use ESP32 for communication, allowing us to leverage Espressif Systems' specialized "Long Range" (LR) protocol. This significantly extends the maximum communication range to approximately 1,000 meters.

## Battery

The low power consumption of the chip and other components, combined with the relatively large battery capacity for the device's size, provides excellent battery life and many hours of operation on a single charge. Users don’t have to worry about the controller running out of power and can rely on its long-lasting operation.

One drawback is the non-functional USB-C charging circuit. For now, the battery must be charged using an external charger, which can be connected via a dedicated connector.

## Customizability

One of the most important achievements is the ability to customize the controller for individual needs. Robot developers can customize it not only through software but also make hardware modifications. This is possible thanks to the source code, PCB design, and 3D models being open source. If a developer has a drone or robot using Wi-Fi, like our test robot, or Bluetooth, they can create their own communication protocol between the controller and the robot without any limitations imposed by the original system.

## Components

The diversity and quality of components is another key achievement. Components such as vibration motors, display, buttons, and the IMU further enhance the controller’s customizability. Both developers and end users can configure how each component behaves and for what purpose they want to use it. This flexibility allows users to create entirely new UI on the display, with controls tailored to their needs, whether for the interface itself or for robot control. Additionally, users can connect custom components to the PCB, replace vibration motors with LEDs, upgrade to a larger display, and customize the code accordingly.

## Simplicity

Despite the immense customization possibilities of nearly every aspect of the controller, its operation remains very simple. It can be compared to a "plug and play" experience, just turn it on, install custom code if needed, and the controller works seamlessly without any problems.

## User Feedback

For a product like this, where there is potential to enter the market, receiving feedback from as many people as possible, including experts and potential users, is crucial. We identified several important issues that need to be addressed in the future, such as charging and programming or rather debugging through a single USB-C port. Another improvement would be allowing users to choose between an integrated and an external antenna. The integrated antenna could be placed directly on the ESP32 module or the PCB, making the controller more compact and aesthetically pleasing. At the same time, users could still opt for an external antenna mounted on top of the controller when increased range is priority.

Additionally, we now have a clearer understanding of which aspects of the controller require more focus, time, and resources during development. One of the most frequently mentioned suggestions was improving ergonomics.

A significant challenge for future versions is cable management and organization. In this version, we solved this issue quickly with a brute-force approach. In future version, we aim to design the controller’s body to allow for a more practical and aesthetically pleasing way to organize the cables without interfering with controller’s functionality.

On the other hand, we received a great deal of positive feedback, particularly regarding the technology and components used. The open and highly customizable nature of the controller was also widely appreciated. An unexpected benefit, which we initially overlooked, is the level of control that developers have over the hardware. Since the software is fully customizable and serves as a direct interface with the hardware, supported by freely accessible documentation, the controller is entirely open and transparent.

This openness offers various advantages. For example, developers can optimize the controller for their specific needs, reducing power consumption or repurposing components for unique applications.

## Competitor Analysis

A black remote control with buttons and buttons

AI-generated content may be incorrect.

Figure 7 Xbox game controller (author: Microsoft, 2025)

Figure 8 Professional drone controller (author: Unmanned Tech Limited, 2025)

In some cases, a game controller like the one seen in image no. 7 can be used, but such controllers only work via Bluetooth, which limits their range and use. They also lack the ability to customize their software in any way or the use of custom components. Additionally, they do not have a display for showing various data related to the robot or drone. They are very limited in their use for robot development.

In image no. 8, we see a professional controller used for controlling drones. These controllers are highly advanced, offering a much better range and control over drones compared to our controller. However, where our controller excels is versatility. Our controller has no boundaries that would define which device it should control. The developer has full control and transparency over both the software and hardware of our controller. They can customize it and replace components to make the controller suitable for controlling not only robots and drones but also RC cars, RC boats, robotic arms, and more.

Both controllers also lack the ability to customize the body of the controller. For instance, if the developer needed a specific connector to attach or hang the controller, they would have to create their own controller.

Our controller fills the gap between gaming controllers (which are flexible but limited in hardware and software) and professional controllers (which are powerful but often locked into ecosystems). It offers a customizable, developer-friendly, and modular solution that can be used to control robots and drones without any restrictions. In terms of price, we aim to position it between cheaper, gaming controllers and more expensive, professional drone controllers. We combine the best features and components from various controllers into one solution.

# Conclusions of the Paper

## Goals and results

The achieved results show that most of the goals were met and the controller is functional. The controller has fully customizable hardware and software. This important result serves as the primary distinguishing factor of our controller from the competition. We believe that for developers and users alike, this will be one of the most important factors when choosing how to control their robot. Easy integration, long battery life, localized vibrations, and the OLED display also play an important role in differentiating our controller from others.

Despite a few shortcomings, such as USB C charging or programming and comfort in holding the controller, which will be improved later, we consider our work to be a great success. We clearly see the future trend, which is the exponential increase in the number of robots. Many of these robots will be autonomous, but many will require a human operator. Regardless, even autonomous robots need a controller for initial development. Given the large diversity of robots and their developers, it makes sense to move toward a simple and global solution that can be applied to various types of robots and drones. This is a step forward not only for the future of robot control but also for our personal development. Working on this extensive project, which fortunately does not end in the near future, has provided us with tremendous experience and knowledge in areas such as programming, PCB design, 3D model design, marketing, market research, and more.

## Outlook for the Future

### Third version

We plan to further develop this work and create a third version of the controller, with the following main objectives. Integrate the ESP32 module directly onto the PCB and utilize the newer ESP32-S3 chip. This will eliminate the use of the DEV model and reduce the space occupied by the ESP32 module. Implement USB-C charging and USB-C programming. Add the option for an external or internal antenna. After achieving these functional goals, we will need to focus more on the comfort of holding the controller and the aesthetics of its body.

### Market Introduction

This phase may seem distant, but still feasible in 2025. Before we introduce our product to the market, we need to meet all the goals. The controller must also be properly tested by several independent entities.

Market research shows that there is currently no direct alternative to our controller. Apart from game controllers and professional drone controllers, there are also mobile apps. We also developed such a mobile app in the past and used it in the early stages of robot development, as it was easier to create than a controller and quicker to adapt. What these apps lack is mostly on the hardware side. The software is fully customizable, but it is limited by the hardware, mainly the range of about 20m, and the constant need to look at the screen and the lack of physical feedback, such as feeling the joystick, button or slider move, made the control feel unnatural and difficult to manage.

We will also need to improve the manual assembly of the controller, which is currently time-consuming and inefficient.

Due to the large number of high-quality components and the overall low number of controllers, their price is relatively higher than that of game controllers, which is understandable, and we expect that developers won’t find this a big problem as our controller has extensive advantages. Creating a custom controller would still be more expensive and time-consuming for developers. The approximate cost of the components is currently around 135 USD per controller.

# Summary

The goal of this project was to create a fully customizable and functional controller for robots. We have largely achieved this goal. It is intended to serve as a cost- and time-effective solution for robot and drone developers, as well as for users who require a high degree of customization and control over the controller’s software and hardware.

The development of the controller itself was very time-consuming, as we had to go through two versions. A large number of components, the hardware design of both the PCB and controller body 3D model, and the software made this project extensive, and it will continue with a third version and gradual market introduction.

Feedback from multiple people has confirmed that we are on the right track. The variety of components, rich features, simplicity, open approach, and adaptability were highly appreciated. To these valued features, we still need to add improvements in ergonomics, design, and assembly efficiency.

# List of Used Literature

Adafruit Industries, 2023 [cit. 12. 02. 2025], Adafruit\_BNO08x.h <<https://github.com/adafruit/Adafruit_BNO08x>>

Adafruit Industries, 2024 [cit. 12. 02. 2025], Adafruit\_GFX.h - <<https://github.com/adafruit/Adafruit-GFX-Library>>

Adafruit Industries, 2024 [cit. 12. 02. 2025], Adafruit\_MAX1704X.h - <<https://github.com/adafruit/Adafruit_MAX1704X>>

Adafruit Industries, 2024 [cit. 12. 02. 2025], Adafruit\_SSD1306.h - <<https://github.com/adafruit/Adafruit_SSD1306>>

Arduino, 2019 [cit. 12. 02. 2025], Arduino.h - <<https://github.com/arduino/ArduinoCore-avr/blob/master/cores/arduino/Arduino.h>>

Espressif Systems, 2023 [cit. 12. 02. 2025], ESP32WROOM32D & ESP32WROOM32U Datasheet- <<https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf>>

Crane, S., 2023 [cit. 16. 02. 2025], TTS.h - <<https://github.com/jscrane/TTS/tree/master>>

Espressif Systems, 2024 [cit. 12. 02. 2025], WiFi.h – <<https://github.com/espressif/arduino-esp32/blob/master/libraries/WiFi/src/WiFi.h>>

Espressif Systems, 2024 [cit. 12. 02. 2025], WiFiMulti.h – <<https://github.com/espressif/arduino-esp32/blob/master/libraries/WiFi/src/WiFiMulti.h>>

HIKL, A., 2025 [cit. 12. 02. 2025], custom\_controller - <<https://github.com/Adam113311777/HSC-controller>>

Links2004, 2024 [cit. 12. 02. 2025], WebSocketsClient.h - <<https://github.com/Links2004/arduinoWebSockets/blob/master/src/WebSocketsClient.h>>

Pycom, 2024 [cit. 12. 02. 2025], rtc\_cntl\_reg.h - <<https://github.com/pycom/esp-idf-2.0/blob/master/components/esp32/include/soc/rtc_cntl_reg.h>>

Samsung SDI, 2015 [cit. 12. 02. 2025], SPECIFICATION OF PRODUCT for Lithium-ion rechargeable cell Model : INR21700-48G – <<https://www.dnkpower.com/wp-content/uploads/2019/02/SAMSUNG-INR21700-48G-Datasheet.pdf>>

# Attachments

1. Espressif Systems.: In: *ESP32WROOM32D & ESP32WROOM32U Datasheet* [online]. 2023 [cit. 12. 02. 2025]. Available: <<https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf>> [↑](#footnote-ref-2)
2. HIKL, A..: In: *custom\_controller* [online]. 2025 [cit. 12. 02. 2025] Available: <<https://github.com/Adam113311777/HSC-controller>> [↑](#footnote-ref-3)
3. Samsung SDI.: In: *SPECIFICATION OF PRODUCT for Lithium-ion rechargeable cell Model : INR21700-48G* [online]. 2015 [cit. 12. 02. 2025]. Available: <<https://www.dnkpower.com/wp-content/uploads/2019/02/SAMSUNG-INR21700-48G-Datasheet.pdf>> [↑](#footnote-ref-4)
4. HYCON Technology.: In: *HY2120 Data Sheet* [online]. 2015 [cit. 12. 02. 2025]. Available: <<https://www.hycontek.com/wp-content/uploads/DS-HY2120_EN.pdf>> [↑](#footnote-ref-5)
5. HYCON Technology.: In: *HY2213 Datasheet* [online]. 2015 [cit. 12. 02. 2025]. Available: <<https://www.hycontek.com/wp-content/uploads/DS-HY2213_EN.pdf>> [↑](#footnote-ref-6)
6. Texas Instruments.: In: *LM2596 SIMPLE SWITCHER® Power Converter 150-kHz 3-A Step-Down Voltage Regulator* [online]. 2023 [cit. 12. 02. 2025]. Available: <<https://www.ti.com/lit/ds/symlink/lm2596.pdf>> [↑](#footnote-ref-7)
7. Maxim Integrated Products.: In: *MAX17048/MAX17049 3μA 1-Cell/2-Cell Fuel Gauge with ModelGauge* [online]. 2016 [cit. 12. 02. 2025]. Available: <<https://www.analog.com/media/en/technical-documentation/data-sheets/max17048-max17049.pdf>> [↑](#footnote-ref-8)
8. Texas Instruments.: In: *LMR51430 SIMPLE SWITCHER® Power Converter 4.5-V to 36-V, 3-A, Synchronous Buck Converter in a SOT-23 Package* [online]. 2022 [cit. 12. 02. 2025]. Available: <<https://www.ti.com/lit/ds/symlink/lmr51430.pdf?ts=1732795972687>> [↑](#footnote-ref-9)
9. Maxim Integrated Products.: In: *MAX98306 Stereo 3.7W Class D Amplifier* [online]. 2017 [cit. 16. 02. 2025]. Available: < <https://www.analog.com/media/en/technical-documentation/data-sheets/MAX98306.pdf>> [↑](#footnote-ref-10)
10. CEVA.: In: *BNO08X Data Sheet* [online]. 2023 [cit. 12. 02. 2025]. Available: <<https://www.ceva-ip.com/wp-content/uploads/BNO080_085-Datasheet.pdf>> [↑](#footnote-ref-11)
11. Autodesk.: In: *Autodesk Fusion* [online]. 2025 [cit. 12. 02. 2025] . Available: <<https://www.autodesk.com/products/fusion-360>> [↑](#footnote-ref-12)
12. Arduino.: In: Arduino IDE download. 2025 [cit. 12. 02. 2025]. Available: <<https://www.arduino.cc/en/software>> [↑](#footnote-ref-13)
13. Williams, J..: In: Imperial March. 1980 [cit. 16. 02. 2025]. Available: < <https://music.youtube.com/watch?v=s3SZ5sIMY6o&si=uIJZYDY7TUdfV3m1>> [↑](#footnote-ref-14)