**📌 ESP32-C3 Zero Pin Connections for the Digital Chess Clock**

|  |  |  |
| --- | --- | --- |
| Component | ESP32-C3 Zero Pin | Description |
| TM1637 (Player 1 Display) | GPIO 6 (CLK) | Clock signal for Player 1 display |
|  | GPIO 7 (DIO) | Data signal for Player 1 display |
| TM1637 (Player 2 Display) | GPIO 4 (CLK) | Clock signal for Player 2 display |
|  | GPIO 5 (DIO) | Data signal for Player 2 display |
| Player 1 Button | GPIO 2 | Switches turn to Player 2 |
| Player 2 Button | GPIO 3 | Switches turn to Player 1 |
| Set Time Button | GPIO 10 | Increases initial time in 5-minute increments |
| Buzzer | GPIO 8 | Sounds when time runs out |
| Power (3.3V) | 3.3V | Powers the ESP32-C3 Zero |
| Power (5V for TM1637) | 5V | Powers the TM1637 displays |
| Ground (GND) | GND | Common ground for all components |

**Notes for ESP32-C3 Zero**

* The **ESP32-C3 Zero has fewer GPIOs**, so pins are assigned accordingly.
* The **TM1637 modules** still use only **two GPIOs each** for communication.
* **Push buttons** should be connected with **10kΩ pull-down resistors**.
* The **buzzer** can be either **active (on/off)** or **passive (PWM-controlled)**.

Would you like a wiring diagram specifically for **ESP32-C3 Zero**? 😊♟️

**ESP32 Pin Connections for the Digital Chess Clock**

|  |  |  |
| --- | --- | --- |
| Component | ESP32 Pin | Description |
| TM1637 (Player 1 Display) | GPIO 22 (CLK) | Clock signal for Player 1 display |
|  | GPIO 21 (DIO) | Data signal for Player 1 display |
| TM1637 (Player 2 Display) | GPIO 19 (CLK) | Clock signal for Player 2 display |
|  | GPIO 18 (DIO) | Data signal for Player 2 display |
| Player 1 Button | GPIO 14 | Switches turn to Player 2 |
| Player 2 Button | GPIO 27 | Switches turn to Player 1 |
| Set Time Button | GPIO 12 | Increases initial time in 5-minute increments |
| Buzzer | GPIO 32 | Sounds when time runs out |
| Power (3.3V) | 3.3V | Powers the ESP32 |
| Power (5V for TM1637) | 5V | Powers the TM1637 displays |
| Ground (GND) | GND | Common ground for all components |

**📌 Notes**

* The **TM1637 display modules** use only **two GPIOs each** for communication.
* **Push buttons** should be connected with **10kΩ pull-down resistors**.
* The **buzzer** can be either **active (on/off)** or **passive (PWM-controlled)**.

Let me know if you need any modifications! 😊♟️

### ****Chess Clock with ESP32 & TM1637 7-Segment Display****

You'll build a **digital chess clock** with the ESP32, where:  
✅ Each player has a **countdown timer**.  
✅ Pressing a button **switches turns** and the other clock continues counting down.  
✅ A **setup button** allows **setting the initial time** in **5-minute increments**.  
✅ Uses **TM1637 7-segment displays** for the countdown timers.

## ****Components Needed****

* **ESP32**
* **2x TM1637 7-segment display** (one for each player)
* **2x Push buttons** (to switch turns)
* **1x Push button** (to set the initial time)
* **Resistors (10kΩ for pull-downs)**
* **Power supply (ESP32 runs on 3.3V, TM1637 needs 5V but works with 3.3V logic)**

## ****Wiring ESP32 + TM1637 + Buttons****

|  |  |
| --- | --- |
| **Component** | **ESP32 Pin** |
| **Player 1 Button** | GPIO 14 |
| **Player 2 Button** | GPIO 27 |
| **Set Time Button** | GPIO 12 |
| **TM1637 CLK (Player 1)** | GPIO 22 |
| **TM1637 DIO (Player 1)** | GPIO 21 |
| **TM1637 CLK (Player 2)** | GPIO 19 |
| **TM1637 DIO (Player 2)** | GPIO 18 |

### ****🖼️ Circuit Diagram****

I can generate a **wiring diagram** if you'd like. Let me know! 😊

## ****📜 MicroPython Code for Chess Clock****

from machine import Pin

from time import sleep

import tm1637

# Setup TM1637 displays

tm1 = tm1637.TM1637(clk=Pin(22), dio=Pin(21)) # Player 1

tm2 = tm1637.TM1637(clk=Pin(19), dio=Pin(18)) # Player 2

# Setup buttons

btn1 = Pin(14, Pin.IN, Pin.PULL\_DOWN) # Player 1 Button

btn2 = Pin(27, Pin.IN, Pin.PULL\_DOWN) # Player 2 Button

btn\_set = Pin(12, Pin.IN, Pin.PULL\_DOWN) # Set Time Button

# Default countdown time (5 minutes = 300 seconds)

initial\_time = 300

p1\_time = initial\_time

p2\_time = initial\_time

current\_player = 1 # Player 1 starts

def display\_time(tm, seconds):

"""Display time in MM:SS format on TM1637."""

minutes = seconds // 60

secs = seconds % 60

tm.show("{:02d}{:02d}".format(minutes, secs))

# Initial Display

display\_time(tm1, p1\_time)

display\_time(tm2, p2\_time)

def switch\_turn():

global current\_player

if current\_player == 1:

current\_player = 2

else:

current\_player = 1

# Main Loop

while True:

if btn\_set.value(): # If set button is pressed

initial\_time += 300 # Increment by 5 minutes

p1\_time = initial\_time

p2\_time = initial\_time

display\_time(tm1, p1\_time)

display\_time(tm2, p2\_time)

sleep(0.5)

if btn1.value(): # Player 1 finishes move

switch\_turn()

sleep(0.2)

if btn2.value(): # Player 2 finishes move

switch\_turn()

sleep(0.2)

if current\_player == 1 and p1\_time > 0:

p1\_time -= 1

display\_time(tm1, p1\_time)

elif current\_player == 2 and p2\_time > 0:

p2\_time -= 1

display\_time(tm2, p2\_time)

sleep(1) # Countdown step

## ****🔍 How It Works****

✅ **Setting Initial Time** → Pressing the set button increases time by **5 minutes**.  
✅ **Switching Turns** → Players press their button to switch the countdown.  
✅ **Countdown Timer** → The clock for the current player decreases every second.

## ****🚀 Next Steps****

🔹 Do you need **a circuit diagram**?  
🔹 Would you like to **add a buzzer** when time runs out?

Let me know how you'd like to improve it! 😊

**🔔 Adding a Buzzer to the Chess Clock**

A **buzzer** will sound when a player's time runs out. We'll use a **passive buzzer (PWM control)** or an **active buzzer (on/off control)**.

**Wiring the Buzzer to ESP32**

|  |  |  |
| --- | --- | --- |
| Buzzer Type | ESP32 Pin | Wiring |
| Active Buzzer (ON/OFF) | GPIO 32 | **VCC to 5V, GND to GND, Signal to GPIO 32** |
| Passive Buzzer (PWM) | GPIO 32 | **VCC to 5V, GND to GND, Signal to GPIO 32** |

**Updating the Code for Buzzer**

Add this to your MicroPython script:

from machine import Pin, PWM

from time import sleep

# Setup buzzer on GPIO 32

buzzer = PWM(Pin(32))

buzzer.freq(1000) # Set initial frequency

def beep(duration=0.5):

"""Plays a beep sound for a given duration (seconds)."""

buzzer.duty(512) # Set volume (0-1023)

sleep(duration)

buzzer.duty(0) # Turn off sound

# Modify the main loop to check for time running out

while True:

if p1\_time == 0: # Player 1 loses

for \_ in range(3): # 3 beeps

beep()

sleep(0.3)

break # Stop game

if p2\_time == 0: # Player 2 loses

for \_ in range(3): # 3 beeps

beep()

sleep(0.3)

break # Stop game

**📌 What This Does**

✅ **Buzzer beeps 3 times** when a player’s time runs out.  
✅ Works with both **active** (on/off) and **passive** (PWM-controlled) buzzers.

**🔍 Next Steps**

🔹 Want a **different buzzer sound pattern**?  
🔹 Need a **circuit diagram update** with the buzzer?

Let me know how you'd like to tweak it! 😊