EMBEDDED AND UBIQUITOUS SYSTEMS

TECHNICAL DOCUMENTATION: THE GAME

José Ramon Ariza Pérez

Mihaela Alexandra Buturuga

Eric López Seguí

Bachelor's Degree in Computer Engineering

10th January 2025

University of Lleida

Course 2024 - 2025

Index

Overview

This document provides the necessary information to replicate the hardware and software setup for the project, including:

1. **Wiring schematics** for all components.
2. **MQTT topic tree structure** for software communication.

Wiring Schematics

Control Base (ESP32)

Hardware Components:

1. **ESP32 Microcontroller**: responsible for player interaction and feedback.
2. **16x2 LCD with I2C adapter**: for displaying game information to players.
3. **Pushbutton**: to receive player input.
4. **Buzzer**: for sound feedback.
5. **LED (with resistor)**: for visual feedback when it's the player's turn.

Connections:

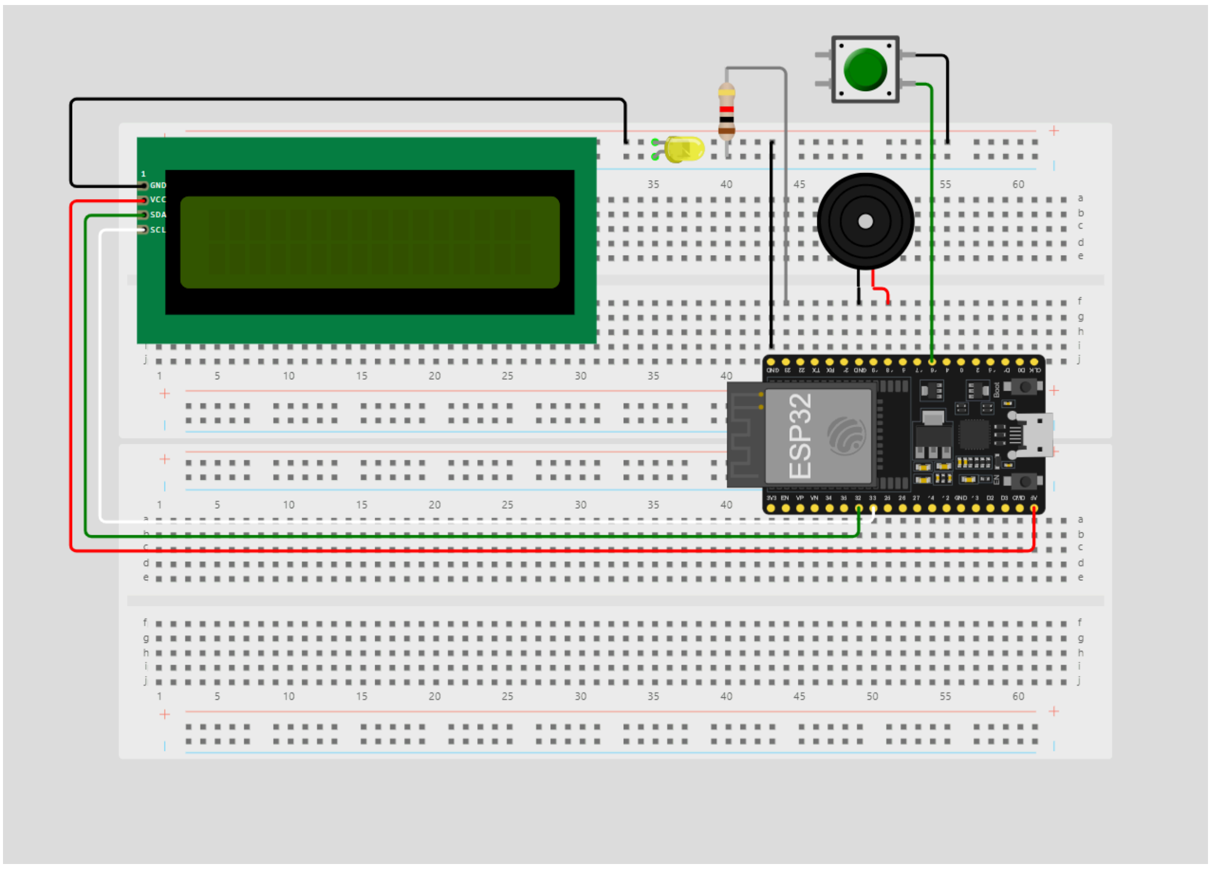
- **LCD (via I2C Adapter)**
  - **GND (Ground):** Connected to the GND rail on the breadboard (from GND pin of the ESP32).
  - **VCC (Power):** Connected to the 5V pin on the ESP32 to power the display.
  - **SDA (Data Line):** Connected to GPIO 32 on the ESP32.
  - **SCL (Clock Line)**: Connected to GPIO 33 on the ESP32.

- **Buzzer**
  - **Positive Pin (+):** Connected to GPIO 18 on the ESP32.
  - **Negative Pin (-)**: Connected to the GND pin of the ESP32.

- **LED**
  - **Anode (+)**: This corresponds to the long leg of the LED. Connected to GPIO 23 of the ESP32 through a resistor.
  - **Cathode (-)**: The other leg of the LED. Connected to the GND rail on the breadboard.

- **Pushbutton**
  - One pin connected to GPIO 16 of the ESP32.
  - The other pin connected to the GND rail on the breadboard.

The following diagram describes the wiring for a single Control Base. Duplicate this setup for each player. Note that LCD I2C connections should be performed **through an I2C adapter which is not displayed on Wokwi**.



Meeple (ESP01)

Hardware Components:

1. **ESP01 Microcontroller:** used to simulate meeple.
2. **A3144 Hall Sensor:** for detecting meeple movement.
3. **LED** (with resistor): for visual feedback when it's the player's turn.

Connections:

ESP32-Powered Setup

For the most part of the project we didn't have a proper power supply for the ESP01, so we powered it by connecting it to the **ESP32** through the 3.3V pin. These are the connections needed for this specific setup:

- **ESP01**
  - **CH_PD and VCC:** both connected to the 3.3V pin on the ESP32.
  - **GND:** connected to the GND rail on the breadboard.

- **Hall Sensor**
  - **VCC (Power Pin):** connected to the 5V pin on the ESP32.
  - **GND (Ground Pin):** connected to the GND rail on the breadboard.
  - **Signal (Digital Output Pin):** connected to the GPIO 0 of the ESP01.

- **LED**
  - **Anode (+)**: This corresponds to the long leg of the LED. Connected to GPIO 2 of the ESP01 through a resistor.
  - **Cathode (-)**: The other leg of the LED. Connected to the GND rail on the breadboard.

Battery-Powered Setup

For a movable setup, we use a smaller breadboard with **power (positive)** and **ground (negative)** rails to distribute power from a 3.7V Li-ion battery. These are the connections for this other setup:

- **ESP01**

    - **CH_PD and VCC:** both connected to the positive rail of the breadboard, which is powered by the battery's positive terminal.
    - **GND:** connected to the negative rail of the breadboard, which is connected to the battery's negative terminal.

- **Hall Sensor**
    - **VCC (Power Pin):** connected to the positive rail of the breadboard.
    - **GND (Ground Pin):** connected to the negative rail of the breadboard.
    - **Signal (Digital Output Pin):** connected to the GPIO 0 of the ESP01.

- **LED**
    - **Anode (+)**: This corresponds to the long leg of the LED. Connected to GPIO 2 of the ESP01 through a resistor.
    - **Cathode (-)**: The other leg of the LED. Connected to the negative rail of the breadboard.

MQTT Topic Tree Structure

Detailed Description

1. Game Control Topics

These topics are used by the game controller to manage the overall game flow. They have the following structure:

```
game/players/<player_id>/<action>
```

- **`game/players/<player_id>/connection`**:
  - **Purpose:** Used to notify the game controller when a control base connects to the network.
  - **Direction:** Control Base → Game Controller.
  - **Payload:** Ignored, any message on this topic can be used to notify connection. The control base sends `"1"` when connected.

- **`game/players/<player_id>/turn`**:
  - **Purpose:** Used to signal to a player that their turn has started or finished.
  - **Direction:** Game Controller → Control Base.
  - **Payload:** `"1"` when the player turn starts, `"0"` when the player turn ends.

- **`game/players/<player_id>/movement`**:
  - **Purpose:** Used to notify the game controller when a player's meeple has moved from one position to another on the board.
  - **Direction:** Control Base → Game Controller.
  - **Payload:** Payload is ignored. Any message on this topic notifies movement. The control base sends `"1"` when the meeple has moved.

2. Component Topics

These topics are used by the game controller to control the components of the control base (LCD screen, buzzer, button). They have the following structure:

```
game/players/<player_id>/components/<component>
```

- **`game/players/<player_id>/components/lcd`**:
  - **Purpose:** Used to send messages to the LCD screen of a specific player's control base.
  - **Direction:** Game Controller → Control Base.

    - **Payload:** JSON object with `top` and `down` fields for the two LCD rows. An optional `time` field can specify display duration in milliseconds.
    - **Example:**
```
{
    "top": "Your Turn",
    "down": "Press Button",
    "time": 3000
}
```

- **`game/players/<player_id>/components/buzzer`**:
    - **Purpose:** Used to send tone sequences to the buzzer of a specific player's control base.
    - **Direction:** Game Controller → Control Base.
    - **Payload:** JSON string including `tones` (list of frequencies) and `duration` (list of durations) in milliseconds for each tone.
    - **Example:**
```
{
    "tones": [262, 392, 0],
    "duration": [200, 200]
}
```

- **`game/players/<player_id>/components/button`**:
    - **Purpose:** This topic is published by a control base when the button is pressed.
    - **Direction:** Control Base → Game Controller.
    - **Payload:** JSON string indicating button press `type` ("short" or "long") and `duration` in milliseconds
    - **Example:**
```
{
    "type": "short",
    "duration": 250
}
```

3. Meeple Topics

These topics manage messages related to the meeple. They have the following structure:

`meeple/<player_id>/<component/action>`

- **`meeple/<player_id>/connection`**

  - **Purpose:** This topic is used to notify the control base that the meeple has connected to MQTT broker and is ready.
  - **Direction:** Meeple → Control Base.
  - **Payload:** The meeple sends `"1"` when connected.

- **`meeple/<player_id>/led`**

  - **Purpose:** This topic is used to control the LED of a specific meeple.
  - **Direction:** Control Base → Meeple.
  - **Payload:**
    - `"1"`: Turn ON the meeple's led.
    - `"0"`: Turn OFF the meeple's led.

- **`meeple/<player_id>/hall_sensor`**:
  - **Purpose:** Used by the meeple device to send hall sensor events to its control base.
  - **Direction:** Meeple → Control Base.
  - **Payload:** The meeple sends `"1"` when hall sensor readings show the meeple has moved.

- **`meeple/<player_id>/debug`**:
  - **Purpose:** Used by the meeple device to send debug logs indicating what is happening on the device. This replaces a traditional console log since the device is not connected to a console.
  - **Direction:** No other component subscribes to this topic. Used externally by developers for debugging.
  - **Payload:** String with the debug message.

Summary Table

Game Control Topics

Topic	Purpose	Direction	Payload
`game/players/<player_id>/connection`	Notify when control base connects.	Control Base → Game Controller	Any (e.g., "1").
`game/players/<player_id>/turn`	Signal turn start or end.	Game Controller → Control Base	"1" (start), "0" (end).
`game/players/<player_id>/movement`	Notify when meeple moves.	Control Base → Game Controller	Any (e.g., "1").

Component Topics

Topic	Purpose	Direction	Payload
`game/players/<player_id>/components/lcd`	Send messages to LCD screen.	Game Controller → Control Base	JSON `{"top": "Your Turn", "time": 3000}`
`game/players/<player_id>/components/buzzer`	Send tone sequences to buzzer.	Game Controller → Control Base	JSON `{"tones": [262, 0], "duration": [200]}`
`game/players/<player_id>/components/button`	Notify when button is pressed.	Control Base → Game Controller	JSON `{"type": "short", "duration": 250}`

Meeple Topics

Topic	Purpose	Direction	Payload
`meeple/<player_id>/connection`	Notify when meeple connects.	Meeple → Control Base	"1".
`meeple/<player_id>/led`	Control meeple LED.	Control Base → Meeple	"1" (ON), "0" (OFF).

Topic	Purpose	Direction	Payload
`meeple/<player_id>/hall_sensor`	Send hall sensor events.	Meeple → Control Base	“1”
`meeple/<player_id>/debug`	Send debug logs for development.	Meeple → External	String with debug messages.