Configuration Module Documentation

File: `config.py`

What It Does

Centralized configuration management for all game settings. Provides structured configuration classes with environment variable support and validation.

Configuration Classes

1. `APIConfig` - API Settings

Manages API endpoint and authentication configuration

Attributes:

```
base_url: str              # API base URL
email: str                 # Authentication email
password: str              # Authentication password
game_id: str               # Unique game identifier (UUID)
game_name: str             # Display name for the game

# Timeout settings (seconds)
auth_timeout: int          # Authentication timeout
game_status_timeout: int   # Game status polling timeout
submit_score_timeout: int  # Score submission timeout
leaderboard_timeout: int   # Leaderboard fetch timeout
```

Production configuration:

```
base_url = "https://prod-eaa25-api.azurewebsites.net/"
email = "sas@uxe.ai"
password = "6u52fkKJl"
game_id = "1db37be5-9b72-4e2b-974d-5fc45d9a4b3a"
game_name = "Fast Reaction"

auth_timeout = 30
game_status_timeout = 8
submit_score_timeout = 20
leaderboard_timeout = 12
```

Development configuration (commented out):

```

```

```
# base_url = "https://dev-eaa25-api-hpfyfcbshkabezeh.uaenorth-
01.azurewebsites.net/"
# email = "eaa25admin@gmail.com"
# password = "qhv#1kGI$"
# game_id = "314bc08a-e3b7-4ace-bf83-33141b014a97"
# game_name = "Fast Reaction Game"
```

---

2. `GameConfig` - Game Settings

Manages game-specific parameters

Attributes:

```
timer_value: int        # Game duration in milliseconds
final_screen_timer: int # Final screen display duration
```

Default configuration:

```
timer_value = 60000      # 60 seconds
final_screen_timer = 5000  # 5 seconds
```

Usage example:

```
from config import config

# Get game timer value
timer_ms = config.settings.game.timer_value
timer_seconds = timer_ms // 1000

# Get final screen duration
final_duration = config.settings.game.final_screen_timer
```

---

3. `SerialConfig` - Serial Port Settings

Manages optional serial communication for external triggers

Attributes:

```
enabled: bool          # Enable/disable serial communication
port: str              # Serial port path
baudrate: int          # Serial baudrate
timeout: float         # Read timeout in seconds
```

Default configuration:

```
enabled = False          # Optional in FastReaction
port = "/dev/ttyUSB0"   # Linux USB serial port
# port = "COM21"        # Windows alternative (commented)
baudrate = 9600
timeout = 1.0
```

Note: Serial port is **optional** in FastReaction. The game works perfectly with keyboard input only.

---

4. `MQTTConfig` - MQTT Broker Settings

Manages MQTT messaging configuration

Attributes:

```
broker: str            # MQTT broker address
port: int              # MQTT broker port
data_topics: list      # Topics for game data
control_topics: list   # Topics for game control
```

Default configuration:

```
broker = "localhost"
port = 1883

data_topics = [
    "FastReaction/score/Pub"
]

control_topics = [
    "FastReaction/game/start",
    "FastReaction/game/stop",
    "FastReaction/game/restart",
    "FastReaction/game/timer",
    "FastReaction/game/Activate",
    "FastReaction/game/Deactivate",
    "FastReaction/game/timerfinal"
]
```

Topic descriptions:

- `start`: Start game signal
- `stop`: Stop game signal (payload: "0" or "1")
- `restart`: Restart game signal

- `timer`: Update game timer value
- `Activate`: Activate game system
- `Deactivate`: Deactivate game system
- `timerfinal`: Update final screen timer
- `score/Pub`: Score data publication

Main Settings Container

`Settings` Class

Main container that aggregates all configuration classes

Attributes:

```
api: APIConfig          # API configuration
game: GameConfig        # Game configuration
mqtt: MQTTConfig        # MQTT configuration
serial: SerialConfig    # Serial configuration (optional)
```

Usage:

```
from config import config

# Access API settings
api_url = config.settings.api.base_url
email = config.settings.api.email

# Access game settings
timer = config.settings.game.timer_value

# Access MQTT settings
broker = config.settings.mqtt.broker
topics = config.settings.mqtt.control_topics

# Access serial settings (if needed)
if config.settings.serial.enabled:
    port = config.settings.serial.port
```

Global Instance

`config` Object

Pre-configured global instance ready to use:

```

```

```
from config import config

# All settings accessible via config.settings
api_config = config.settings.api
game_config = config.settings.game
mqtt_config = config.settings.mqtt
serial_config = config.settings.serial
```

Environment Variable Support

Configuration values can be overridden using environment variables:

API Settings

```
export CAGE_API_BASE_URL="https://api.example.com/"
export CAGE_API_EMAIL="user@example.com"
export CAGE_API_PASSWORD="password123"
export CAGE_GAME_ID="uuid-here"
```

Game Settings

```
export CAGE_TIMER_VALUE="60000"         # 60 seconds in ms
export CAGE_FINAL_SCREEN_TIMER="5000"  # 5 seconds in ms
```

Serial Settings

```
export CAGE_SERIAL_ENABLED="True"
export CAGE_SERIAL_PORT="/dev/ttyUSB0"
export CAGE_SERIAL_BAUDRATE="9600"
```

MQTT Settings

```
export CAGE_MQTT_BROKER="localhost"
export CAGE_MQTT_PORT="1883"
```

Configuration Validation

The config module includes validation to ensure:

- Required fields are present

- Values are in correct format
- Timeouts are positive integers
- URLs are properly formatted

Invalid configuration will raise an exception on import.

Usage Patterns

Reading Configuration

```
from config import config

# API authentication
api = GameAPI(
    base_url=config.settings.api.base_url,
    email=config.settings.api.email,
    password=config.settings.api.password
)

# Game timer setup
timer = QTimer()
timer.setInterval(config.settings.game.timer_value)

# MQTT connection
mqtt_client.connect(
    config.settings.mqtt.broker,
    config.settings.mqtt.port
)
```

Modifying Configuration (Runtime)

```
# Update timer value at runtime
config.settings.game.timer_value = 45000  # 45 seconds

# Enable serial port dynamically
config.settings.serial.enabled = True
config.settings.serial.port = "/dev/ttyUSB1"
```

FastReaction-Specific Configuration

Serial Port Usage

Serial port is **optional** in FastReaction (enabled=False by default):

- Game works perfectly with keyboard input
- Serial can be enabled for external trigger devices

- No special hardware required

Game Settings

FastReaction uses simple timer configuration:

- `timer_value` - Game duration in milliseconds
- `final_screen_timer` - Results display duration
- No hardware-specific parameters needed

MQTT Topic Naming

All topics use the "FastReaction/" prefix:

- `FastReaction/game/start` - Start game
- `FastReaction/game/stop` - Stop game
- `FastReaction/score/Pub` - Score publication

Development vs Production

Switching Environments

To switch from production to development:

1. Comment out production config
2. Uncomment development config
3. Restart application

```
# Production (active)
# base_url = "https://prod-eaa25-api.azurewebsites.net/"
# email = "sas@uxe.ai"

# Development (commented)
base_url = "https://dev-eaa25-api-hpfyfcbshkabezeh.uaenorth-
01.azurewebsites.net/"
email = "eaa25admin@gmail.com"
```

Or use environment variables:

```
export CAGE_API_BASE_URL="https://dev-eaa25-api-hpfyfcbshkabezeh.uaenorth-
01.azurewebsites.net/"
export CAGE_API_EMAIL="eaa25admin@gmail.com"
export CAGE_API_PASSWORD="qhv#1kGI$"
```
