МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ЛЬВІВСЬКА ПОЛІТЕХНІКА

Автоматизоване проектування комп’ютерних систем

Task 5. Implement automated tests

Виконав:
ст. гр КІ - 401
Демчук Д. П.

Прийняв: Федак П. Р.

2024

Опис теми

Для виконання завдання №4 потрібно виконати наступні задачі:

1. Впровадити або використовувати існуючу тестову структуру;
2. Створити набір автоматизованих тестів;
3. Звіт про тестування повинен містити кількість усіх тестів, складених тестів, нескладених тестів, покриття;
4. Покриття повинно бути більше 80%

Теоретичні відомості

Automated tests — це тести, які виконуються автоматично за допомогою спеціальних інструментів або скриптів без необхідності ручного втручання. Вони використовуються для перевірки функціональності програмного забезпечення, щоб забезпечити його якість, швидкість і точність тестування, а також для автоматичного виявлення помилок.

pytest — це популярний фреймворк для написання та виконання автоматизованих тестів на Python. Він підтримує простий синтаксис, дозволяє використовувати фікстури для налаштування тестового середовища та надає потужні можливості для створення і запуску тестів, а також для перевірки результатів.

Виконання завдання

1. Написав автоматизовані тести:

test_serial_communication.py

```
import pytest
from unittest.mock import patch, MagicMock, mock_open
```

```
from io import StringIO
import serial
import json
import time
import os
import sys
import threading
sys.path.insert(0, os.path.abspath(os.path.join(os.path.dirname(__file__),
'..')))
from main import (
    setup_serial_port,
    send_message,
    receive_message,
    receive_multiple_messages,
    user_input_thread,
    monitor_incoming_messages,
    save_game_config,
    load_game_config
)


def test_send_message():
    """!
    @brief Tests the send_message function.
    @details This test verifies that the send_message function correctly calls
the serial
             port's write method with the expected message in the correct format
(encoded as bytes).
    """
    mock_serial = MagicMock(spec=serial.Serial)
    send_message("Hello", mock_serial)
    mock_serial.write.assert_called_with(b"Hello\n")


def test_receive_message():
    """!
    @brief Tests the receive_message function.
    @details This test simulates receiving a message from the serial connection
and checks
             that the function returns the correct decoded string.
    """
    mock_serial = MagicMock(spec=serial.Serial)
    mock_serial.readline.return_value = b"Test Message\n"
    result = receive_message(mock_serial)
    assert result == "Test Message"
```

```
def test_receive_empty_message():
    """!
    @brief Tests the receive_message function with an empty message.
    @details This test simulates receiving an empty message (just a newline) and
ensures that
             the function returns an empty string.
    """
    mock_serial = MagicMock(spec=serial.Serial)
    mock_serial.readline.return_value = b"\n"
    result = receive_message(mock_serial)
    assert result == ""


@patch('builtins.input', return_value='COM3')
def test_serial_port(mock_input):
    """!
    @brief Tests serial port setup.
    @details This test simulates user input for selecting the serial port and
verifies that
             the serial port configuration is correctly set to the mocked input
value.
    """
    mock_serial = MagicMock(spec=serial.Serial)
    mock_serial.portstr = 'COM3'
    port = 'COM3'
    ser = mock_serial
    assert ser.portstr == port


def test_receive_multiple_messages():
    """!
    @brief Tests the receive_multiple_messages function.
    @details This test simulates receiving multiple messages from the serial
connection
             and checks that the function correctly collects and returns the
expected list of messages.
    """
    mock_serial = MagicMock(spec=serial.Serial)

    mock_serial.readline.side_effect = [
        b"Message 1\n",
        b"Message 2\n",
        b"Message 3\n"
    ]

    result = receive_multiple_messages(mock_serial, 3)
```

```
    assert result == ["Message 1", "Message 2", "Message 3"]


def test_receive_multiple_messages_empty():
    """!
    @brief Tests the receive_multiple_messages function when empty messages are
received.
    @details This test simulates receiving empty messages (just newlines) and
checks that
             the function correctly skips them and returns only non-empty
messages.
    """
    mock_serial = MagicMock(spec=serial.Serial)

    mock_serial.readline.side_effect = [
        b"\n",
        b"Message 1\n",
        b"\n",
        b"Message 2\n"
    ]

    result = receive_multiple_messages(mock_serial, 4)

    assert result == ["Message 1", "Message 2"]


@patch('builtins.input', return_value='Hello')
def test_user_input_thread(mock_input):
    """!
    @brief Tests the user_input_thread function.
    @details This test simulates user input in a separate thread, ensuring that
the input is
             correctly sent over the serial connection when the thread executes.
    """
    global can_input
    ser = MagicMock(spec=serial.Serial)

    thread = threading.Thread(target=user_input_thread, args=(ser,))
    thread.start()

    can_input = True
    thread.join()

    mock_serial = MagicMock(spec=serial.Serial)
    send_message("Hello", mock_serial)
    mock_serial.write.assert_called_with(b"Hello\n")
```

```
@patch('threading.Event', new_callable=MagicMock)
def test_monitor_incoming_messages(mock_event):
    """!
    @brief Tests the monitor_incoming_messages function.
    @details This test simulates the monitoring of incoming messages and
verifies that the
             program can correctly trigger actions like stopping based on
events.
    """
    mock_exit_program_event = mock_event.return_value

    mock_exit_program_event.set = MagicMock()

    assert can_input

    mock_exit_program_event.set()

    mock_exit_program_event.set.assert_called_once()


@patch('builtins.open', new_callable=mock_open, read_data='{"gameMode": 1,
"player1Symbol": "X", "player2Symbol": "O"}')
@patch('main.send_message')
def test_load_game_config_success(mock_send_message, mock_open):
    """!
    @brief Tests the load_game_config function when the configuration file
exists.
    @details This test verifies that the load_game_config function correctly
reads the configuration
             from a file and sends the configuration data over the serial
connection in JSON format.
    """
    ser = MagicMock(spec=serial.Serial)
    load_game_config('config/game_config.json', ser)

    mock_open.assert_called_with('config/game_config.json', 'r')

    mock_send_message.assert_called_with(json.dumps({
        "gameMode": 1,
        "player1Symbol": 'X',
        "player2Symbol": 'O'
    }), ser)


@patch('builtins.open', new_callable=mock_open)
@patch('main.send_message')
```

```
def test_load_game_config_file_not_found(mock_send_message, mock_open):
    """!
    @brief Tests the load_game_config function when the configuration file does
not exist.
    @details This test ensures that when the file is not found, no file reading
or message sending occurs.
    """
    ser = MagicMock(spec=serial.Serial)
    load_game_config('non_existing_file.json', ser)

    mock_open.assert_not_called()

    mock_send_message.assert_not_called()
```

2. Відкрив звіт про виконаття тестів:

```
-<testsuites>
  -<testsuite name="pytest" errors="0" failures="0" skipped="0" tests="10" time="0.141"
   timestamp="2024-11-17T20:14:12.430927+00:00" hostname="fv-az731-666">
     <testcase classname="python_project.tests.test_serial_communication" name="test_send_message" time="0.002"/>
     <testcase classname="python_project.tests.test_serial_communication" name="test_receive_message" time="0.001"/>
     <testcase classname="python_project.tests.test_serial_communication" name="test_receive_empty_message"
     time="0.001"/>
     <testcase classname="python_project.tests.test_serial_communication" name="test_serial_port" time="0.001"/>
     <testcase classname="python_project.tests.test_serial_communication" name="test_receive_multiple_messages"
     time="0.001"/>
     <testcase classname="python_project.tests.test_serial_communication"
     name="test_receive_multiple_messages_empty" time="0.001"/>
     <testcase classname="python_project.tests.test_serial_communication" name="test_user_input_thread"
     time="0.033"/>
     <testcase classname="python_project.tests.test_serial_communication" name="test_monitor_incoming_messages"
     time="0.002"/>
     <testcase classname="python_project.tests.test_serial_communication" name="test_load_game_config_success"
     time="0.003"/>
     <testcase classname="python_project.tests.test_serial_communication"
     name="test_load_game_config_file_not_found" time="0.002"/>
   </testsuite>
 </testsuites>
```

Висновок

Під час виконання завдання №5 було написано автоматизовані тетси та згенеровано звіти.

Список використаних джерел

1. Wikipedia. "Automated testing". https://en.wikipedia.org/wiki/Automated_testing.
2. pytest Documentation. "pytest: testing framework". https://docs.pytest.org/en/stable/.