Requirements: python 3.8, PySide2

Obsługa GUI

Ustawienia:

Ustawienia maszyny szyfrującej znajdują się w pliku config\_gui.json. Nieprawidłowe wprowadzenie danych skutkuje błędem podczas uruchamiania programu. Jeśli program nie wykryje pliku ustawień, sam stworzy plik domyślnych ustawień. Domyślne ustawienia odpowiadają konfiguracji modelowi Enigma I (1930).

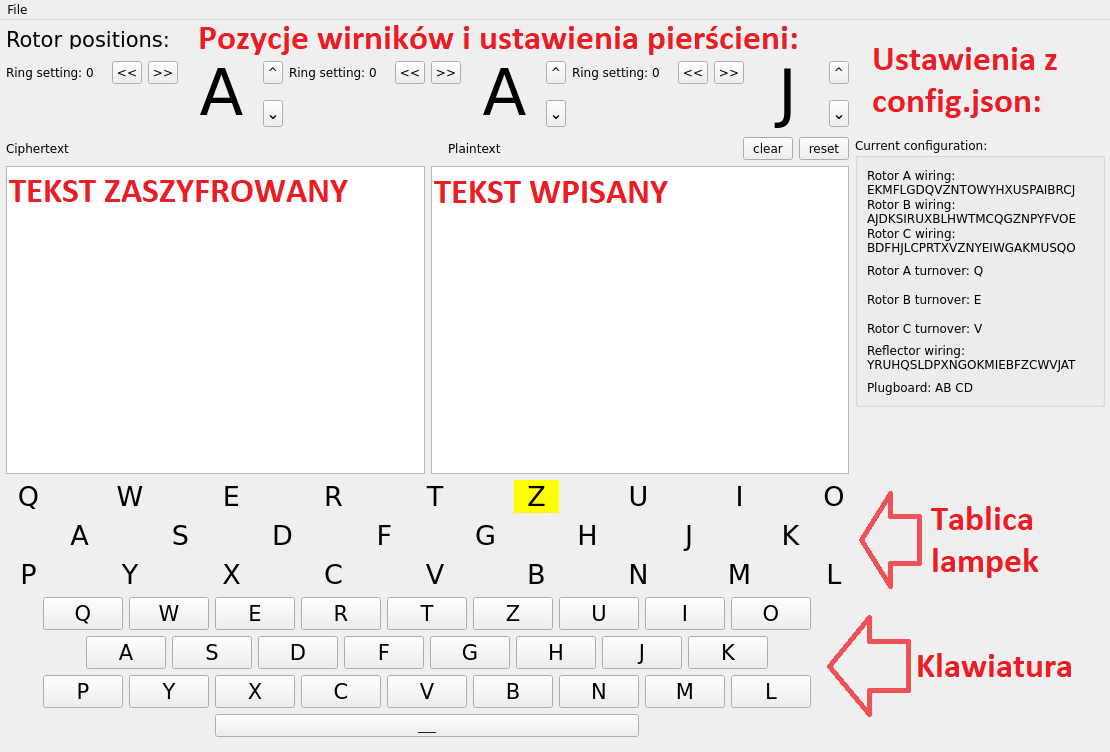
Poszczególne zmienne odpowiadają ustawieniom rzeczywistej maszyny szyfrującej:

rotor(Indeks wirnika)\_wiring – Połączenia między wejściami a wyjściami w wirniku. Litera wskazuje na wyjście połączenia, a pozycja litery, na wejście.   
Np. Dla „EKMFLGDQVZNTOWYHXUSPAIBRCJ” pierwsze połączenie, czyli wejścia 1, ma wyjście w 5, ponieważ litera E jest na 5 miejscu w angielskim alfabecie.  
Analogicznie działa ustawienie połączeń walca odwracającego(reflektora).  
Prawidłowo wprowadzone połączenie musi zawierać 26 wielkich liter angielskiego alfabetu. Litery te nie mogą się powtarzać.

Starting\_letter ustala pozycję w jakiej będzie znajdować się wirnik podczas uruchomienia programu.

Turnover specyfikuje, które litera jest pozycją przeskoku następnego wirnika. W rzeczywistej maszynie enigma, jeśli wirnik znajduje się w tej pozycji, to przy wprowadzeniu litery do maszyny, ten i następny wirnik muszą przejść o jedną pozycję.

Plugs specyfikuje połączenia w łącznicy kablowej. Prawidłowo wprowadzone połączenia muszą mieć postać [„litera1”, „litera2”]. Kolejność liter nie ma znaczenia. Nie można powtarzać tych samych połączeń.   
Przykładowo, prawidłowym połączeniem jest: [[„A”, „B”], [„C”, „D”]], w którym połączone są ze sobą litery A i B, oraz C i D.

Interfejs: 

Litery opisujące pozycję wirników odpowiadają literom ustawienia wirników w maszynie rzeczywistej. Strzałki pionowe pozwalają dowolnie je zmieniać. Ustawienia pierścienia mogą być skonfigurowane dla dowolnego wirnika strzałkami poziomymi.

Klawiatura i tablica lampek dla aplikacji działają analogicznie do tych w maszynie rzeczywistej. Możliwa jest obsługa za pomocą klawiatury komputera, oczywiście pamiętając, że zaszyfrowane mogą być jedynie litery angielskie, więc próba wpisania znaku „ś” przy użyciu alt+s, zakończy się wpisaniem „s”. Spacja wpisuje spację do tekstu.

Aplikacja pozwala zaszyfrować plik przy użyciu opcji File > Open. Wtedy plik tekstowy zostanie zaszyfrowany przy użyciu aktualnych ustawień, tych z config\_gui.json i tymczasowych, obowiązujących w interfejsie. Tekst zostanie sformatowany na możliwy do zaszyfrowania i automatycznie wpisany do maszyny. W ten sposób program może zaszyfrować pliki tekstowe do 5KB. Dla większych tekstów należy skorzystać z szyfrowania bezpośredniego File > Cipher directly.

Opcje save as… I save as… (only cipher) służą do zapisania szyfru i tekstu wpisanego. Opcja save as… zapisuje oba, a save as… (only cipher) tylko szyfr. Przy używaniu tych opcji należy pamiętać, że program nie zapamiętuje ustawień z jakimi zaczęliśmy szyfrować dany tekst.

Cipher directly pozwala bezpośrednio zaszyfrować plik tekstowy i go zapisać w innym pliku. W ten sposób można zaszyfrować do 10MB pliku tekstowego. Program zapisuje ustawienia tymczasowe interfejsu, z jakimi rozpoczęło się szyfrowanie, na końcu pliku. Warto pamiętać, że w enigmie szyfrowanie i odszyfrowanie przeprowadza się w dokładnie ten sam sposób.

Obsługa batch

Wersja wsadowa programu wymaga do szyfrowania ustawień w pliku pod nazwą config\_bat.json w katalogu, w którym program jest używany. Jeśli program nie znajdzie takiego pliku, to stworzy domyśly. Tak jak w przypadku wersji graficznej, będą to ustawienia maszyny Enigma I.

W argumentach należy podać wymagany argument -k KEY, gdzie KEY to klucz, którego chcemy użyć do zaszyfrowania tekstu. Key należy podać w formacie LNNLNNLNN, gdzie L to litera na wirniku i NN to ustawienie pierścienia wirnika. Klucz podaje się od pierwszego wirnika do trzeciego.  
Np. A25Z19C13, to ustawienie w którym pierwszy wirnik jest w pozycji A, a jego ustawienie pierścienia wynosi 25. Drugi wirnik w Z, a ustawienie pierścienia 19 itd..

Tekst do zaszyfrowania można podać na dwa sposoby. Należy użyć -p PATH, gdzie PATH to ścieżka pliku do zaszyfrowania. Alternatywnie można użyć potoku, którym należy wprowadzić tekst do zaszyfrowania. Program priorytezuje wybór ścieżki nad potokiem, więc podanie potoku i ścieżki naraz skutkuje zaszyfrowaniem tylko pliku z ścieżki.

Działanie

Moduły:

Implementacja projektu składa się z następujących modułów:

enigma\_cipher.py – zawiera metody szyfrowania tekstu i sprawdzania danych wejściowych. Dodatkowo przechowuje klasy reprezentujące błędy ustawień wejściowych.  
enigma\_classes.py – klasy przechowujące dane potrzebne do szyfrowania tekstu przy użyciu modułu enigma\_cipher.py. Klasy te zawierają metody zmiany swoich własnych wartości.  
enigma\_config\_io.py – posiada metodę pobierania ustawień z pliku json do programu.  
enigmaCipheringButton.py i enigmaLamp.py – posiadają klasy zaprojektowane do łatwiejszego programowania GUI.  
enigma\_gui.py – moduł główny graficznej wersji programu, zawierający klasę okna aplikacji. Klasa te zawiera funkcje, których implementacja w innych modułach była utrudniona np. metody zapisy do pliku.  
enigma\_bat.py – moduł główny wsadowej wersji programu, po uruchomieniu szyfruje wprowadzony tekst.

Oba moduły enigma\_gui.py enigma\_py.py używają metod z enigma\_cipher.py do zaszyfrowywania tekstu i metod z enigma\_config\_io.py do wczytywania ustawień maszyny z plików konfiguracji.

enigmaCipheringButton.py i enigmaLamp.py zawierają klasy dziedziczące po QPushButton i QLabel, które posiadają metody z których enigma\_gui.py korzysta do obsługi wprowadzania danych.