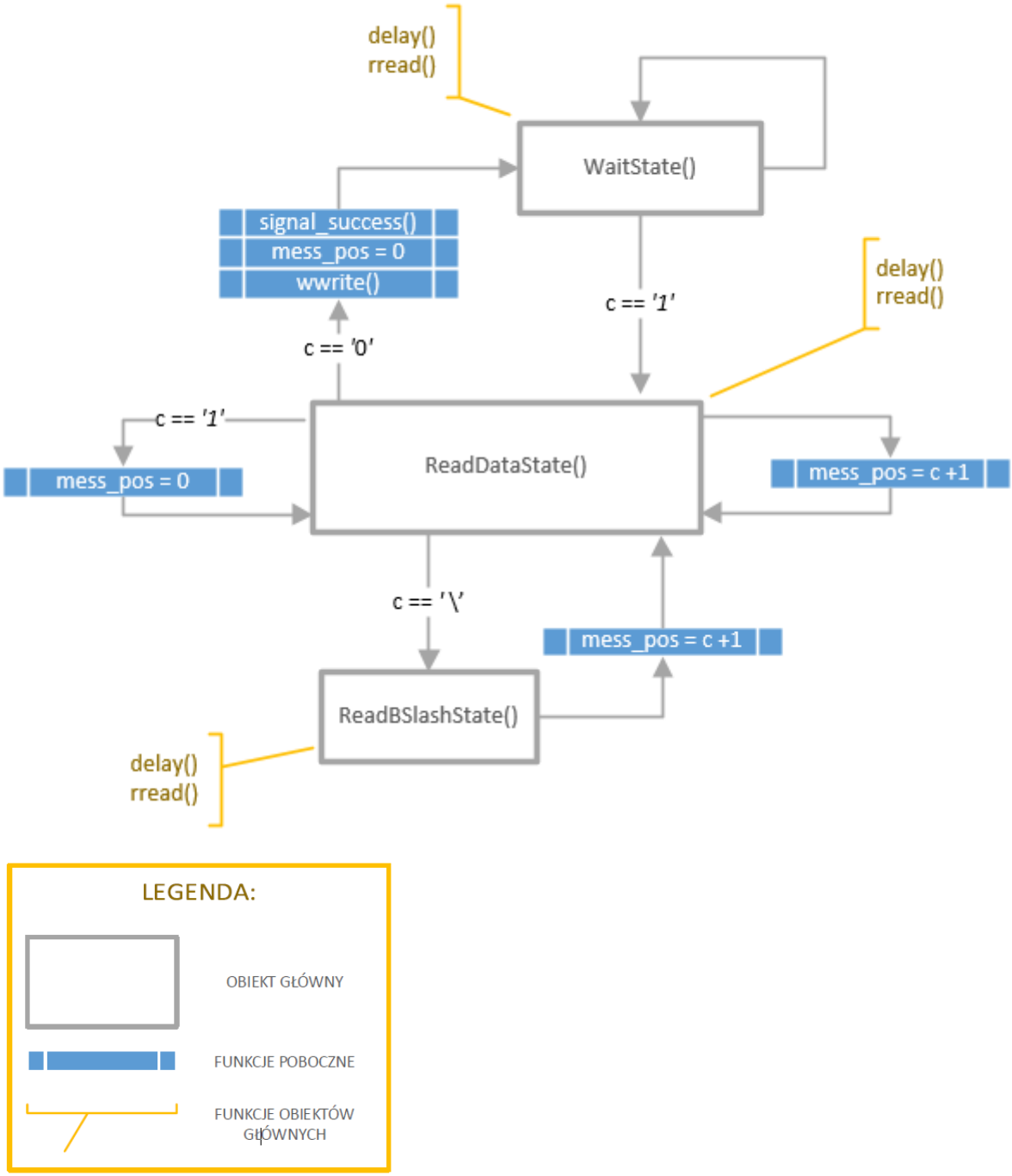
Zadanie - Python	Nr grupy	Termin zajęć	Imię i Nazwisko	Nr albumu
2	**-**	**zajęcia zdalne**	**Grzegorz Dobroń**	**236 260**

1. Zadanie 1. Schemat blokowy automatu stanów



Rys. 1. Diagram stanów automatu

Zaprojektowany automat składa się z trzech obiektów głównych tj.

- WaitState()
- ReadDataState()
- ReadBSlashState()

Stanem początkowym automatu jest WaitState() (CodeBlock 1).

CodeBlock 1. Incicjalizacja stanu poczatkowego

```
def __init__(self):
    self.currentState = ChannelStateMachine.waitState
    while (1):
        self.currentState = self.currentState.run()
```

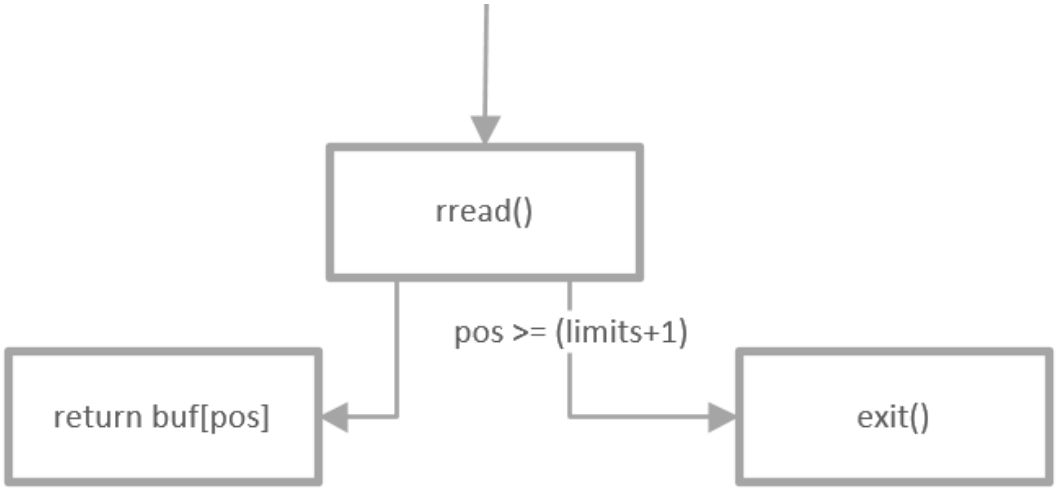
Klasa *WaitState* czeka (*delay()*), a następnie odczytuje (*rread()*) znak do niej przekazany. Gdy czytany znak jest inny niż ‘1’ funkcja się zapętla i czeka (*delay()*) na odczyt kolejnego znaku. W przypadku uzyskania ‘1’ funkcja przechodzi do kolejnego obiektu jakim jest *ReadDataState()*, ‘1’ jest to początek ramki danych.

Wejście do *ReadDataState()* rozpoczyna się od *delay()* oraz odczytu (*rread()*) przekazywanego znaku:

- w przypadku uzyskania ‘0’ następuje przejście do obiektu *WaitState()*, poprzedzonego wypisaniem znaku/znaków, przesunięcie kursora odczytu na pozycje 0 oraz zakomunikowaniem o pomyślnym odczycie danych;
- w przypadku gdy czytany znak to ‘1’ następuje przypisaniem pozycji zerowej do bieżącego pozycji kursora odczytu;
- w przypadku odczytu ‘\’ następuje przejście do obiektu *ReadBSlashState()*, w którym następuje przesunięcie kursora jedną pozycje do przodu oraz bezwarunkowy powrót do *ReadDataState()*;
- w przypadku, gdy czytany jest inny znak, następuje przesunięcie kursora jedną pozycję do przodu i powrót do *ReadDataState()*.

2. Zadanie 2.

Odporność maszyny na pakiety dłuższe niż 16 znaków, zostało zrealizowane dodając warunek pozycji kursora (*pos*) odczytu w funkcji *rread()*. Gdy pozycja kursora odczytu będzie się charakteryzować wartością wyższą niż *limits+1* program zakończy swoje działania. *Limits* to zdefiniowana długość pakietu, *1* wynika z faktu wliczania początku ramki danych jako znak. Schematyczne działanie zmodyfikowanej funkcji *rread()* zostało przedstawione
na Rys. 2



Rys. 2. Schemat zmodyfikowanej funkcji rread()

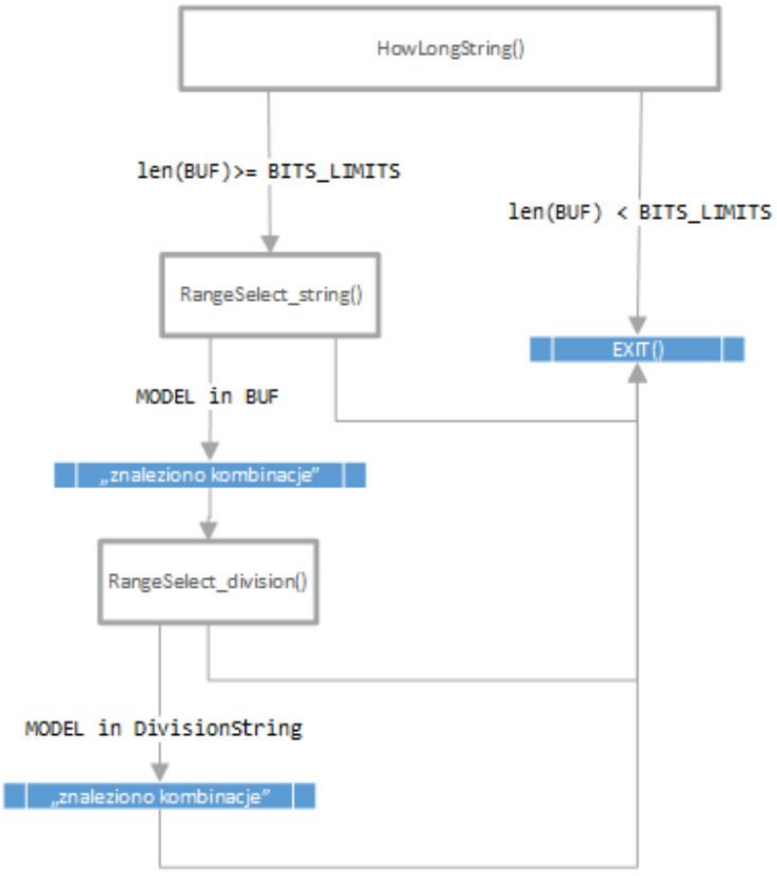
3. Zadanie 3. Detektor sekwencji 5-bitowej

Zdefiniowany bufor danych wejsiowych *BUF* zostaje przeanalizowany przez metodę klasy *HowLongString()*, której wynikiem jest informacja o tym, jaka jest liczna znaków (*len(BUF)*) w zdefiniowanym *BUF*,

- w przypadku, gdy liczba znaków (*len(BUF)*) jest krótsza niż 5-bitowa długość ramki (*BITS_LIMITS = 5*), wynikiem jest informacja jest informacja, że liczba podanych znaków w *BUF* jest za mała;
- w przypadku, gdy liczba znaków (*len(BUF)*) jest dłuższa lub równa niż 5-bitowa długość ramki (*BITS_LIMITS = 5*), program przechodzi do metody klasy *RangeSelect_string().*

Klasa *RangeSelect_division()* odpowiedzialna jest za znalezienie *MODEL*u w całym ciągu znaków *BUF*. W sytuacji, gdy *MODEL* zostaje znaleziony, następnym krokiem jest przejście do *RangeSelece_division(),* jeśli *MODEL* nie został znaleziony program kończy działanie.

Klasa *RangeSelect_division()* w pierwszej kolejności dzieli *BUF* na pakiety 5-bitowe, które zebrane są w formie listy, a następnie wśród pakietów szuka *MODEL*u. Jest to ostatni człon automatu, wobec czego niezależnie od wyniku program kończy działanie.



Rys. 3. Diagram stanów detektora sekwencji 5-ciu bitów