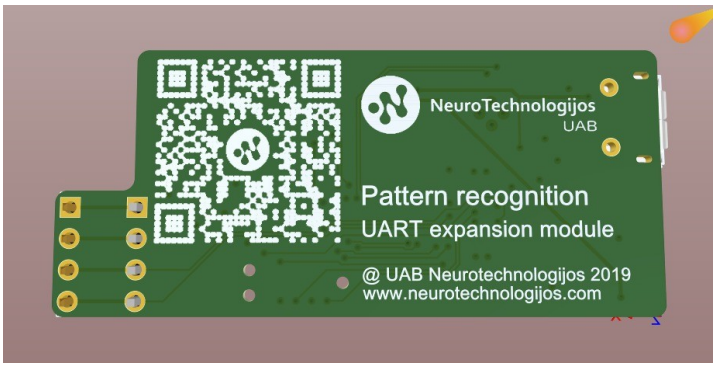
NeuroTechnologijos UAB

Pattern recognition expansion module API

Руководство пользователя

версия 0.1



2019

Введение

Данное руководство предназначено для создания программного обеспечение, которое использует Pattern recognition expansion module под управлением операционных систем Raspbian или Windows.

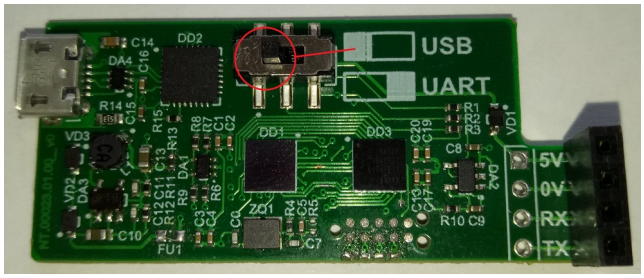
Руководство содержит полное описание состава и функций модуля Pattern recognition expansion module API, а также описание порядка их использования.

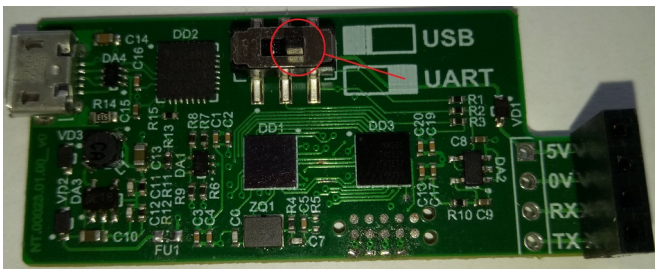
Руководство содержит фрагменты исходного кода примеров использования библиотеки.

Для работы с библиотекой необходимо изучить следующие документы:

1. NM500 Hardware Manual
http://general-vision.com/documentation/TM_NM500_Hardware_Manual.pdf
2. NeuroMem Technology Reference Guide
https://www.general-vision.com/documentation/TM_NeuroMem_Technology_Reference_Guide.pdf

Для работы платы с Virtual COM Port ПК или UART Raspberry Pi установите переключатель на плате в соответствующее положение.





1. Состав модуля Pattern recognition expansion module API

Базовые операции

Функция чтения регистра

register_read(address: int) → int

Функция записи в регистр

register_write(address: int, value: int)

Рабочие операции

Функция обучения вектором

```
learn_vector( metrics: int,
              context: int,
              category: int,
              maxif: int,
              minif: int,
              comps: int,
              vector: bytearray
              )
```

Функция распознавания вектора

```
reco_vector(  metrics: int,
              context: int,
              classificator: int,
              answers: int,
              comps: int,
              vector: bytearray
              )
```

Функция чтения содержимого одного определенного нейрона

```
read_neuron( number: int,
             comps: int
             ) → list
```

Операции сохранения-восстановления базы знаний

Функция сохранения (выгрузки) базы знаний

```
save_base( comps: int
         )
```

Функция восстановления (загрузки) базы знаний

```
load_base(array: list,
          comps: int
          ) → list
```

Вспомогательные функции

Сброс базы знаний

def base_forget() → None:

Получение количества нейронов в сети

def get_amount_neurons() → int

Выгрузка базы знаний из 2D списка в текстовый файл

```
vector_set_to_txt(filename: str,
                  a: list
                 )
```

Загрузка базы знаний из текстового файла в 2D список

```
text_to_vector_set(filename: str
                   ) -> list
```

2. Описание функций, входящих в модуль

2.1 Базовые операции

2.1.1 Функция чтения регистра

register_read(address: int) → int

Функция возвращает значение, содержащееся в регистре с адресом address.

Параметры:

address – адрес регистра.

Возвращаемое значение:

значение, содержащееся в регистре с адресом address.

2.1.2 Функция записи в регистр

register_write(address: int, value: int)

Функция записывает значение data в регистр с адресом address.

Параметры:

address – адрес регистра.

data – значение для записи в регистр.

Возвращаемое значение:

значение, записанное в регистр с адресом address.

2.2 Рабочие операции

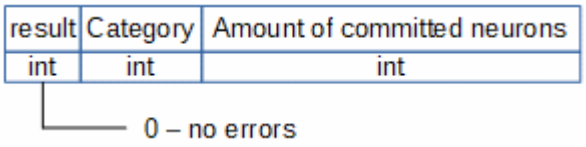
2.2.1 Функция обучения вектором

```
learn_vector( metrics: int,          # метрика для текущего вектора (L1=0,  Lsup=1)
              context: int,          # значение контекста для текущего вектора (1...127)
              category: int,         # значение категории для текущего вектора (0...32767)
              maxif: int,            # значение MAXIF для текущего вектора
              minif: int,            # значение MINIF для текущего вектора
              comps: int,            # количество компонентов в векторе для обучения (1...256)
              vector: bytearray      # значения компонентов вектора для обучения
              )
```

Функция загружает компоненты вектора и параметры обучения в классификатор.

Возвращаемые значения:

список, содержащий результат обучения и имеющий следующую структуру:



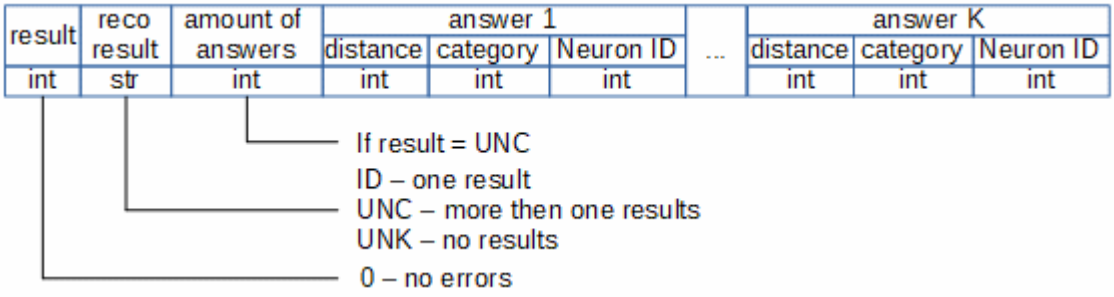
2.2.2 Функция распознавания вектора

```
reco_vector(  metrics: int,          # метрика для текущего вектора (L1=0,  Lsup=1)
              context: int,          # значение контекста для текущего вектора (1...127)
              classificator: int,    # тип классификатора для текущего вектора (RBF=0,  KNN=1)
              answers: int,          # количество результатов, возвращаемое при неоднозначном
                                     #                 результате распознавания
              comps: int,            # количество компонентов в векторе для обучения (1...256)
              vector: bytearray      # значения компонентов вектора для обучения
              )
```

Функция загружает компоненты вектора и параметры распознавания в классификатор.

Возвращаемые значения:

список, содержащий результаты распознавания и имеющий следующую структуру:



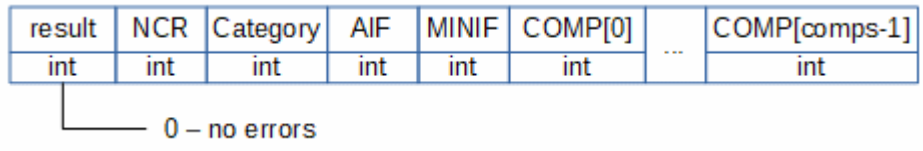
2.2.3 Функция чтения содержимого одного определенного нейрона

read_neuron(number: int, # номер нейрона, который необходимо считать (0...575)
comps: int # к-во компонентов нейрона, которые необходимо считать (1...256)
) → list

Значение index не должно превышать значение номера последнего возбужденного (committed) нейрона.

Возвращаемые значения:

список, содержащий результат чтения компонентов нейрона и имеющий следующую структуру:



2.3 Операции сохранения-восстановления базы знаний

2.3.1 Функция сохранения (выгрузки) базы знаний

save_base(comps: int # к-во компонентов нейронов для считывания (1...256)
)

Функция считывает содержимое возбужденных нейронов (базу знаний) в 2D список.

Возвращаемые значения:

1- результат выполнения операции. Если значение результата равно 0, функция выполнена успешно.

2- 2D список, содержащий базу знаний.

Структура элемента списка:

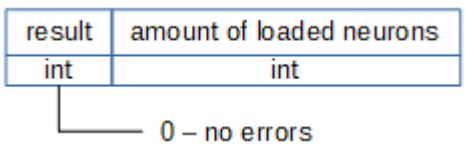
2.3.2 Функция восстановления (загрузки) базы знаний

load_base(array: list, # 2D список, содержащий базу знаний.
comps: int # к-во компонентов нейронов для восстановления (1...256)
) → list

Функция загружает содержимое 2D списка (базу знаний) в нейроны.

Возвращаемые значения:

список, содержащий результат выполнения функции и количество нейронов, загруженных в классификатор.



2.4 Вспомогательные функции

2.4.1 Сброс базы знаний

def base_forget() → None:

Сбрасывает значения категории всех нейронов в 0 и переводит указатель на нейрон с номером 0.

Параметры: нет

Возвращаемое значение: нет.

2.4.2 Получение количества нейронов в сети

def get_amount_neurons() → int

Функция возвращает количество нейронов в нейросети.

Параметры: нет

Возвращаемые значения:

1- результат выполнения функции. Если значение результата равно 0, функция выполнена успешно.

2- количество нейронов в сети.

2.4.3 Выгрузка базы знаний из 2D списка в текстовый файл

```
vector_set_to_txt(filename: str,        # имя файла для сохранения базы знаний
                  a: list               # 2D список, содержащий базу знаний
                 )
```

Возвращаемые значения: нет

Структура строки в текстовом файле:

0		NCR		Category		AIF		MINIF		COMP[0]			COMP[comps-1]		
%5d	" "	%5d	" "	%5d	" "	%5d	" "	%5d	" "	%5d	" "	...	%5d	" "	'\n '

2.4.4 Загрузка базы знаний из текстового файла в 2D список

```
text_to_vector_set(filename: str        # имя файла, содержащего базу знаний
                  ) -> list
```

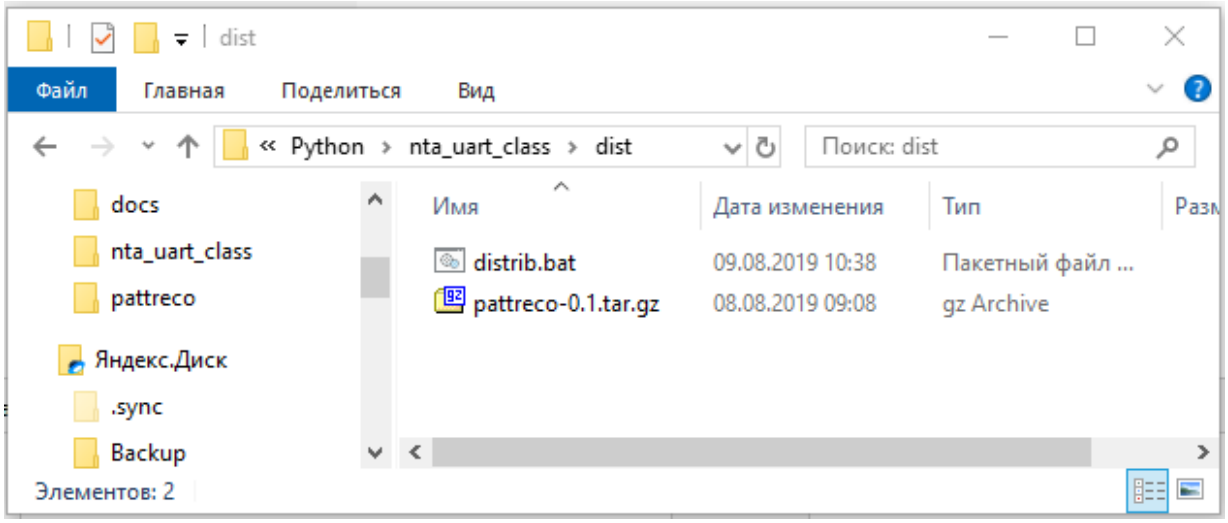
Возвращаемые значения:

2D список, содержащий базу знаний

3. Инсталляция и деинсталляция модуля

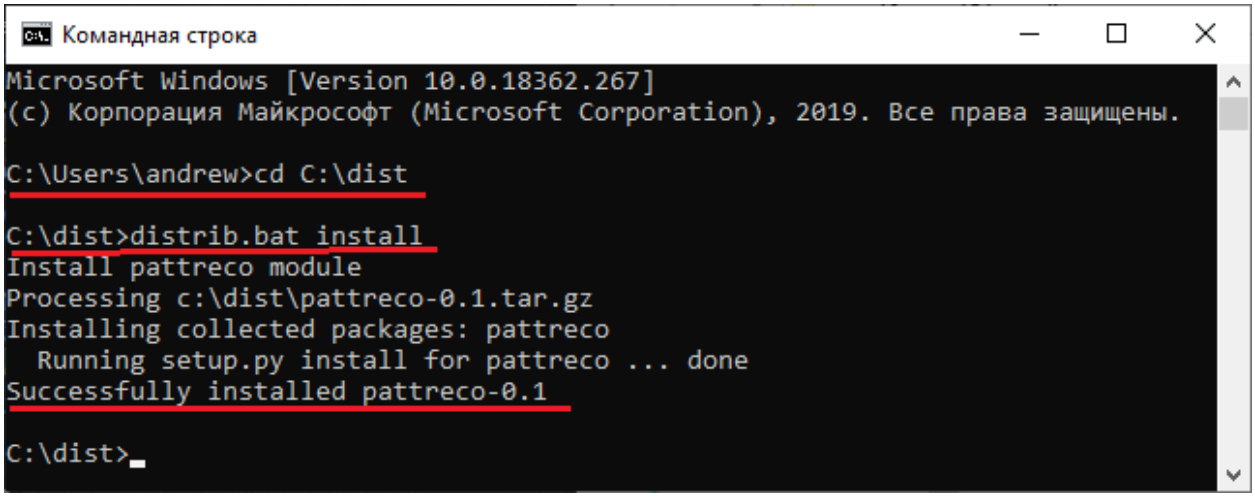
3.1 Windows

Модуль pattreco распространяется в виде папки dist. Содержание папки приведено на рисунке:



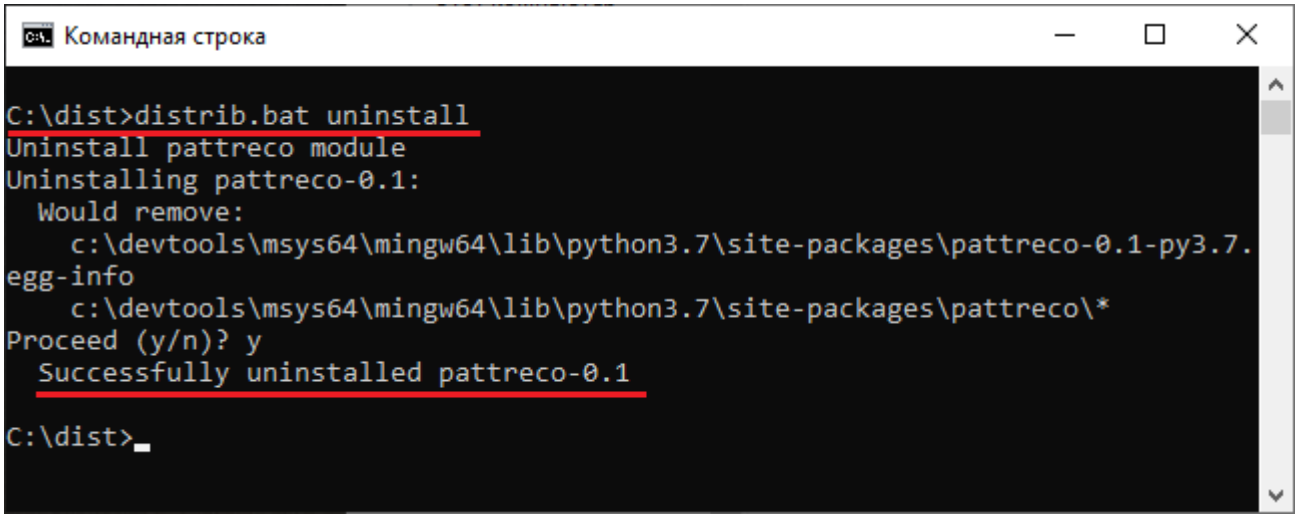
3.1.1 Инсталляция

Для инсталляции модуля необходимо перейти в папку /dist и запустить файл install_distrib.bat.



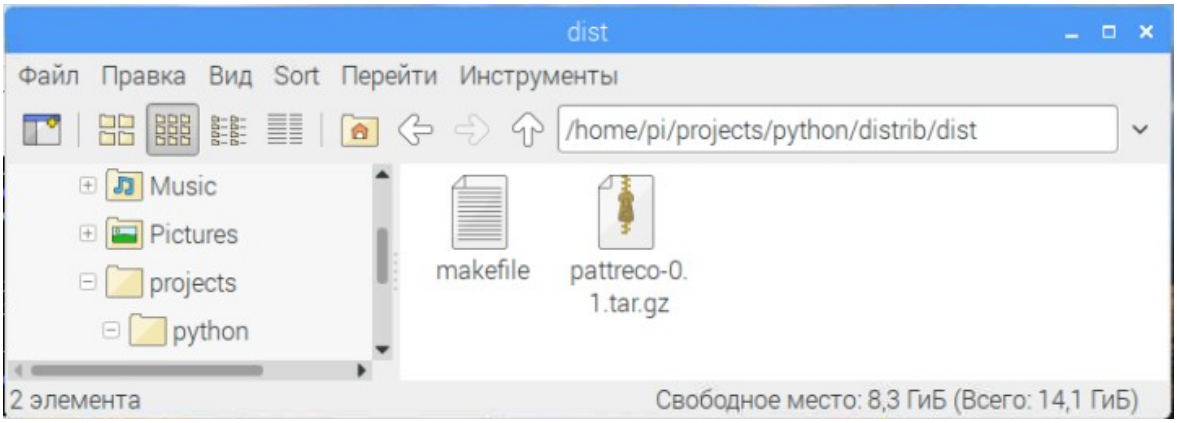
3.1.2 Деинсталляция

Для деинсталляции библиотеки необходимо в командной строке запустить pip uninstall pattreco.

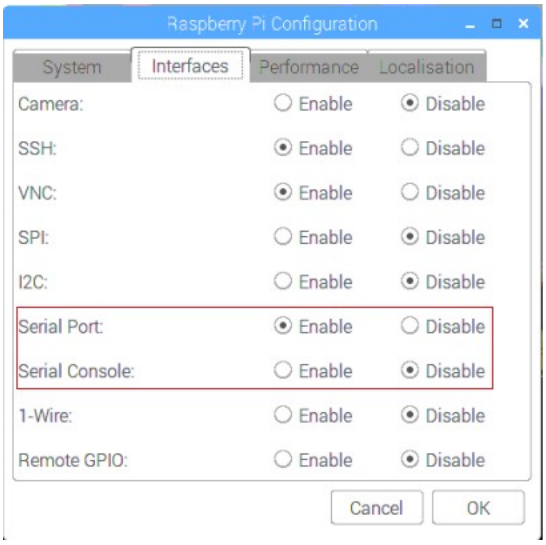


3.2 Linux

Модуль pattreco распространяется в виде папки dist. Содержание папки приведено на рисунке:

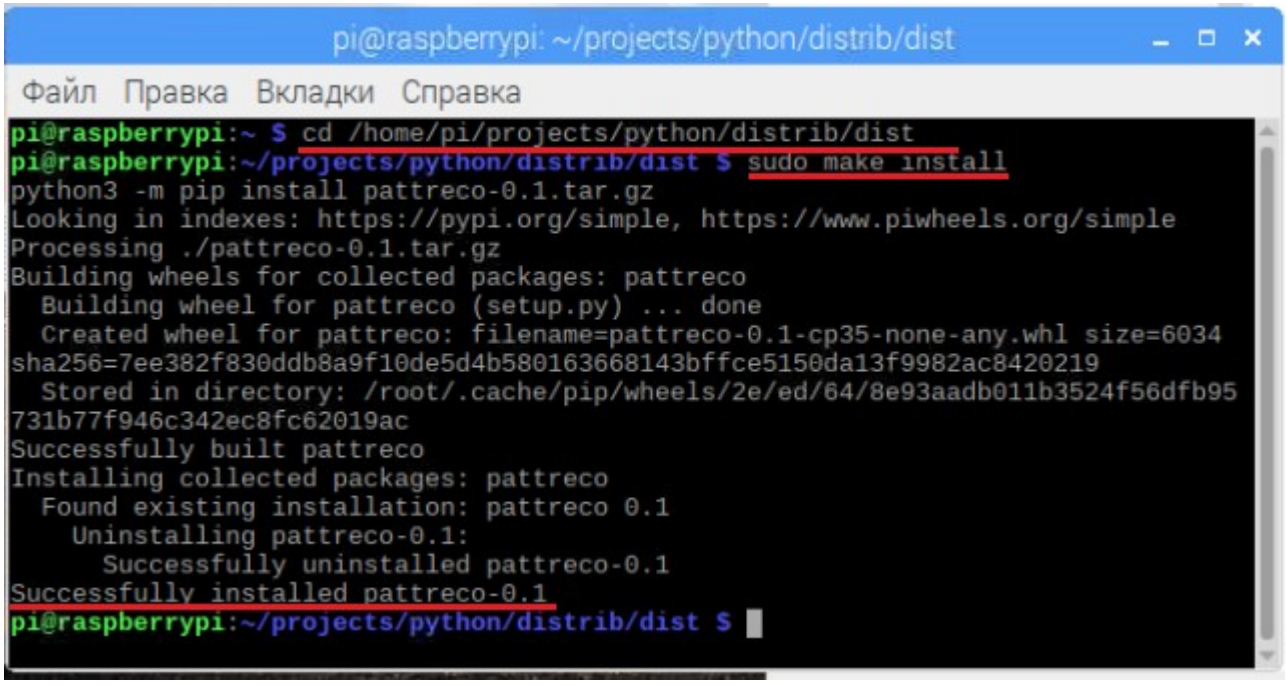


Необходимо включить Serial port:



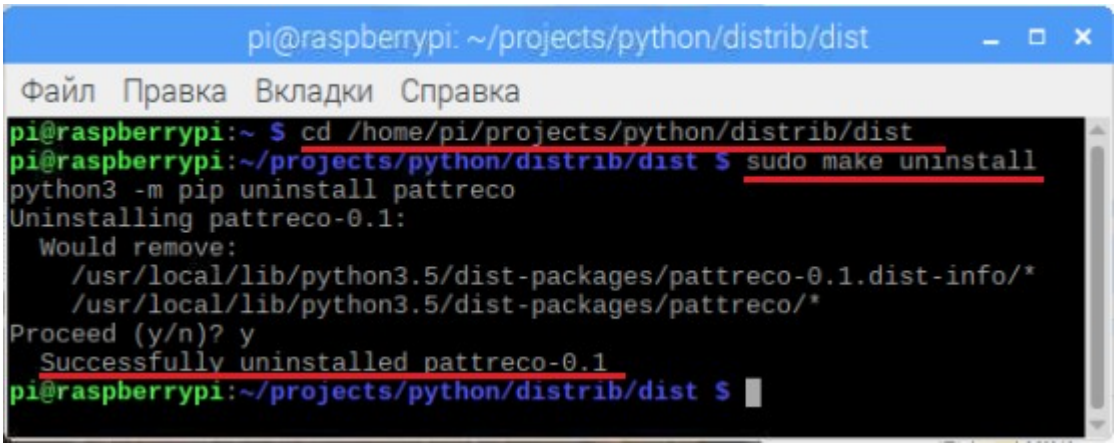
3.2 1Инсталляция

Для инсталляции модуля необходимо в командной строке перейти в папку dist и запустить sudo make install.



3.2.3 Деинсталляция

Для деинсталляции модуля необходимо в командной строке перейти в папку dist и запустить sudo make uninstall.



4. Подключение и использование модуля

4.1 Подключение модуля

```
# -*- coding: utf-8 -*-

import pattreco.class_lib as cl
```

4.2 Использование модуля

4.2.1 Чтение регистра

```
# создаем объект класса nm500
self.nm500 = cl.nm500()
# находим COM-порт, к которому подключена плата
portname = self.nm500.uart.port_find()
# открываем  COM-порт
result = self.nm500.uart.open_port(portname)
# чтение регистра MINIF
value = self.nm500.register_read(6)
# выводим результат
print("value of register %2d = %5d \n" % (value))
# закрываем  COM-порт
self.nm500.uart.close_port()
```

4.2.2 Загрузка обучающего вектора

```
# создаем объект класса nm500
self.nm500 = cl.nm500()
# находим COM-порт, к которому подключена плата
portname = self.nm500.uart.port_find()
# открываем  COM-порт
result = self.nm500.uart.open_port(portname)
# готовим компоненты вектора
vector = bytearray([225, 226, 224, 174, 170, 160, 123, 25, 12, 120])
# загрузка вектора для обучения
pack = learn_vector(    0,              # метрика для текущего вектора (L1)
                        1,              # значение контекста для текущего вектора
                        22,             # значение категории для текущего вектора
                        400,            # значение MAXIF для текущего вектора
                        2,              # значение MINIF для текущего вектора
                        10,             # количество компонентов в векторе для обучения
                        vector          # значения компонентов вектора для обучения
                        )
# выводим результат
print("committed neurons-  ", pack[1])
# закрываем  COM-порт
self.nm500.uart.close_port()
```

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ