Pattern recognition expansion module. Demo application manual.

Introduction.

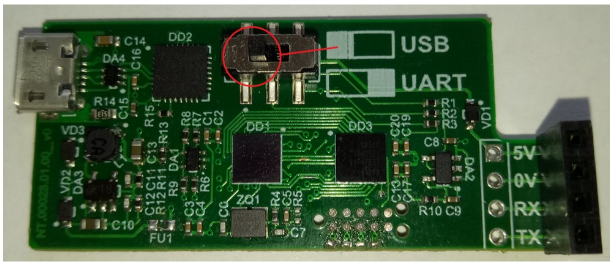
This tutorial is intended to create software that uses the Pattern recognition expansion module under Raspbian or Windows operating systems.

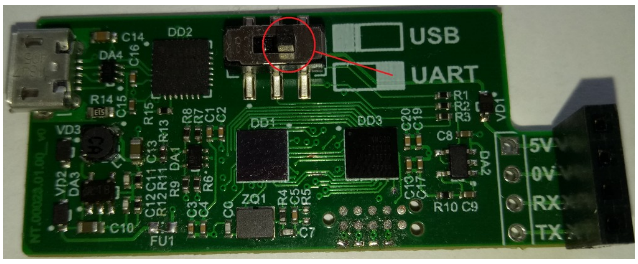
The manual contains a complete description of the composition and functions of the Pattern recognition expansion module API, as well as a description of how to use them.

The manual contains fragments of the source code of examples of using the library.

To work with the library, you must study the following documents:

To operate the board with the Virtual COM Port of a PC or UART Raspberry Pi, set the switch on the board to the appropriate position.





Module composition.

Below is a complete list of library functions.

Register read function:

```
register_read  (address: int) → int
```

Register write function:

```
register_write (address: int,
                value: int
                )
```

Vector learning function:

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

Vector recognition function:

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

Function to read the contents of one specific neuron:

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

The function of saving (unloading) the knowledge base:

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

Knowledge base restore (load) function:

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

Reset knowledge base:

```
def base_forget () → None
```

Getting the number of neurons in the network:

```
        def get_amount_neurons () → int
```

Unloading a knowledge base from a 2D list into a text file:

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

Loading knowledge base from text file into 2D list:

```
        text_to_vector_set (filename: str) → list
```

Description of the functions included in the module

1. Register Read Function

```
        register_read (address: int) → int
```

Description:

the function returns the value contained in the register with the address address.

Parameters:

address **-** register address.

Return value:

the value contained in the register at address.

2. Register write function

```
        register_write (address: int, value: int)
```

Description:

the function writes the data value to the register with the address address.

Parameters:

address **-** register address.

data **-** the value to write to the register.

Return value:

the value written to the register with the address address.

3. Vector learning function

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

Description:

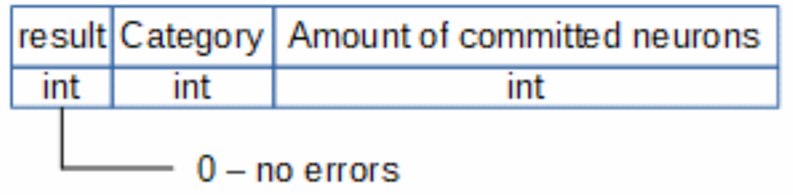
the function loads vector components and training parameters into the classifier.

Parameters:

metrics - metric for the current vector (L1 = 0, Lsup = 1)
context - context value for the current vector (1 ... 127)
category - category value for the current vector (0 ... 32767)
maxif - MAXIF value for the current vector
minif - MINIF value for the current vector
comps - number of components in the vector for training (1 ... 256)
vector - values of vector components to train

Returned values:

a list containing the learning result and having the following structure:



result	Category	Amount of committed neurons
int	int	int

0 – no errors

4. Vector Recognition Function

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

Description:

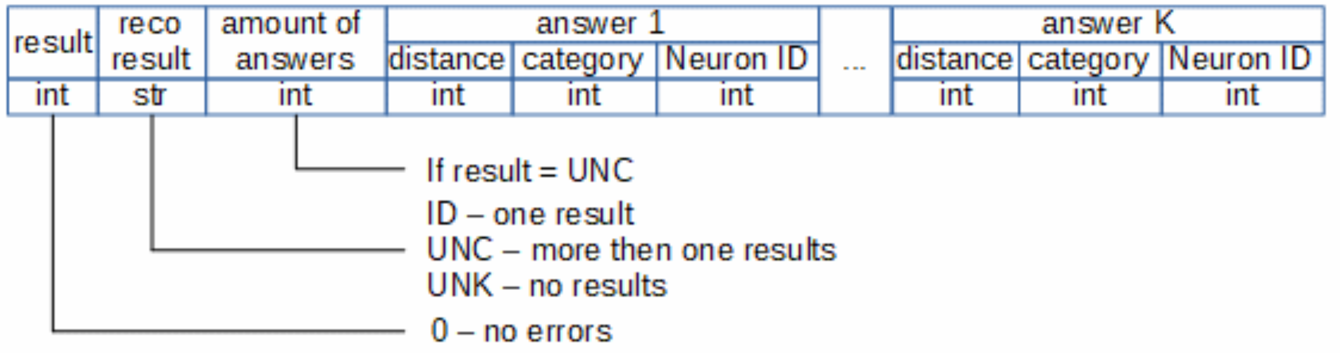
the function loads vector components and recognition parameters into the classifier.

Parameters:

metrics - metric for the current vector (L1 = 0, Lsup = 1)
context - context value for the current vector (1 ... 127)
classificator - classifier type for the current vector (RBF = 0, KNN = 1)
answers - the number of results returned if the recognition result is ambiguous
comps - number of components in the vector for training (1 ... 256)
vector - values of vector components to train

Returned values:

a list containing recognition results and having the following structure:



result	reco result	amount of answers	answer 1			...	answer K		
			distance	category	Neuron ID		distance	category	Neuron ID
int	str	int	int	int	int		int	int	int

If result = UNC

ID – one result
UNC – more then one results
UNK – no results

0 – no errors

5. The function of reading the contents of one specific neuron

read_neuron (number: int, # neuron number to be read (0 ... 575)
comps: int # number of neuron components to be read (1 ... 256)
) → list

Description:

the index value should not exceed the number of the last committed neuron.

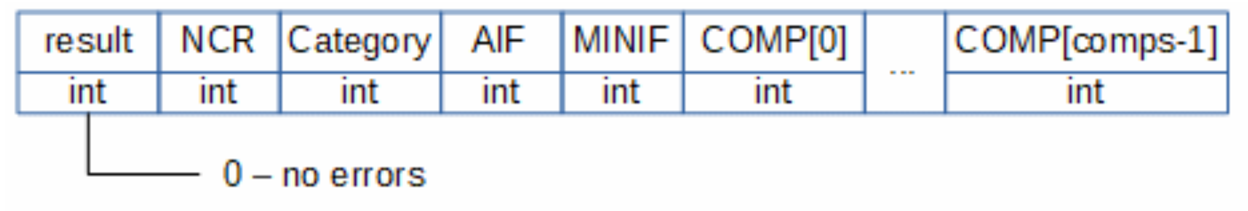
Parameters:

number - neuron number to be read (0 ... 575)
comps - number of neuron components to be read (1 ... 256)

Caution: The index value should not exceed the number of the last committed neuron.

Returned values:

a list containing the result of reading neuron components and having the following structure:



result	NCR	Category	AIF	MINIF	COMP[0]	...	COMP[comps-1]
int	int	int	int	int	int		int

0 – no errors

6. Function of saving (unloading) knowledge base

save_base (comps: int)

Description:

the function reads the contents of the fired neurons (knowledge base) into a 2D list.

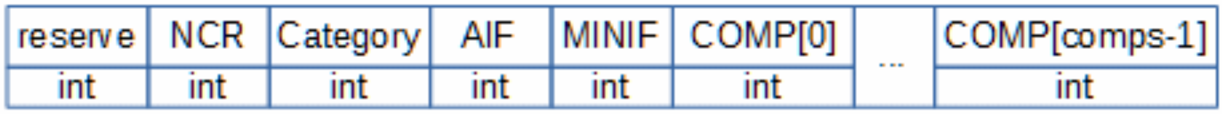
Parameters:

comps - number of neuron components to read (1 ... 256)

Returned values:

1 - result of the operation. If the result is 0, the function succeeded.
2 - 2D list containing knowledge base.

List item structure:



reserve	NCR	Category	AIF	MINIF	COMP[0]	...	COMP[comps-1]
int	int	int	int	int	int		int

7. Knowledge Base Recovery (Boot) Function

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

Description:

the function loads the contents of a 2D list (knowledge base) into neurons.

Parameters:

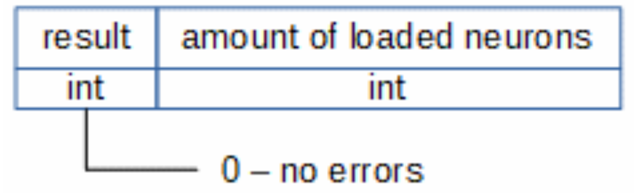
array - 2D list containing the knowledge base.

comps - number of neuron components to recover (1 ... 256)

Returned values:

a list containing the result of the function execution and the number of neurons loaded into the classifier.

List item structure:



result	amount of loaded neurons
int	int

0 – no errors

8. Reset knowledge base

def base_forget () → None:

Description:

resets the values of the category of all neurons to 0 and moves the pointer to the neuron with number 0.

Parameters:

no

Returned value:

none.

9. Getting the number of neurons in the network

def get_amount_neurons () → int

Description:

the function returns the number of neurons in the neural network.

Parameters:

no

Returned values:

1- result of function execution. If the result is 0, the function succeeded.

2- the number of neurons in the network.

10. Unloading a knowledge base

vector_set_to_txt (filename: str,
a: list
)

Description:

unloading a knowledge base from a 2D list into a text file

Parameters:

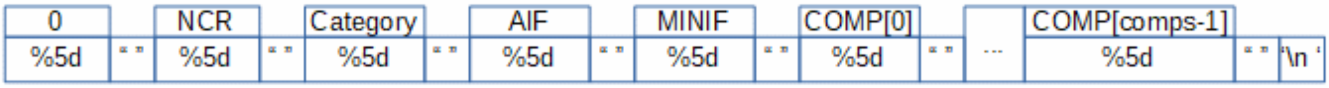
filename - filename to save the knowledge base

a: list - 2D list containing knowledge base

Returned values:

none

Line structure in a text file:



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

11. Loading the knowledge base

text_to_vector_set (filename: str
) → list

Description:

Loading the knowledge base from a text file into a 2D list

Parameters:

filename - name of the file containing the knowledge base

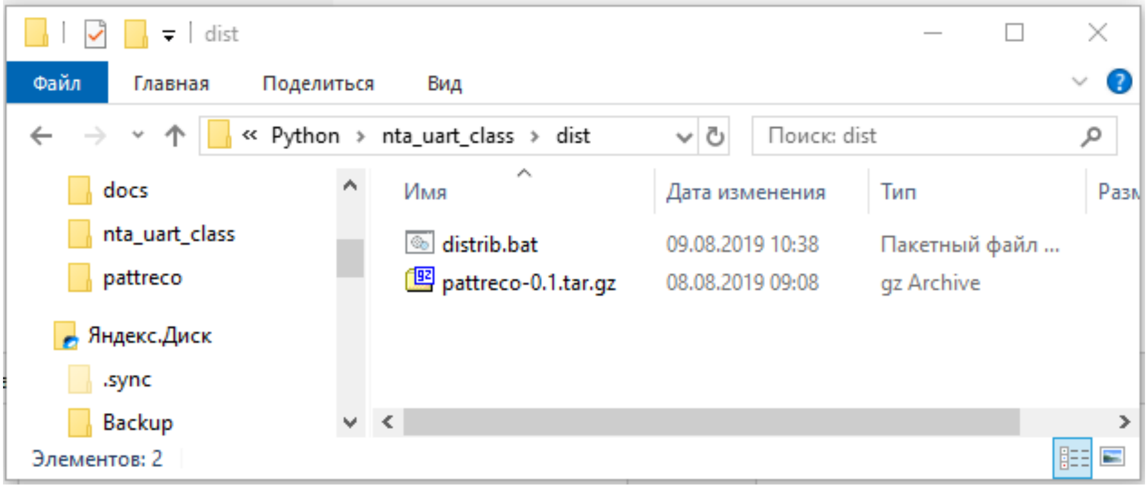
Returned values:

2D list containing knowledge base

Installation and uninstallation of the module

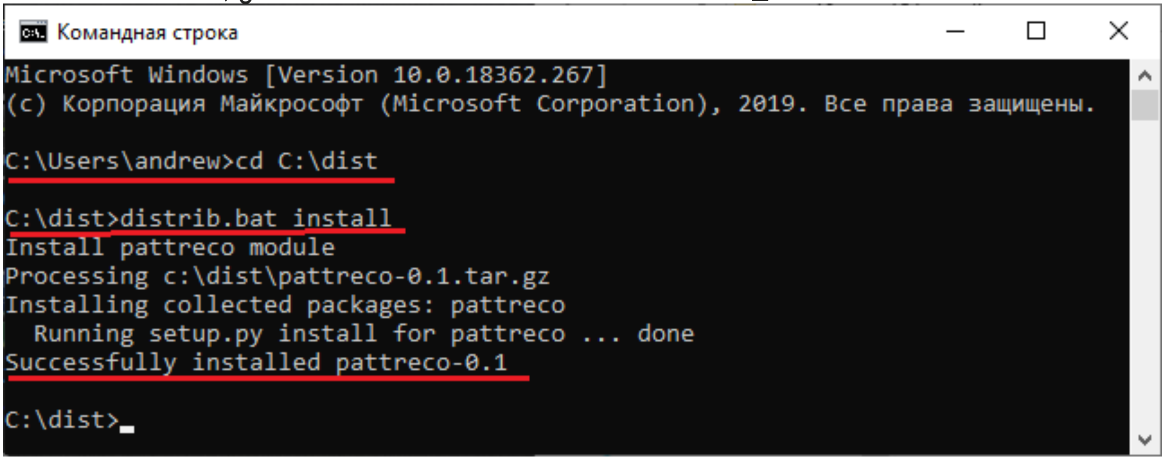
Installation under Windows.

The pattern recognition module (abbreviated like «pattreco») is distributed as a «/dist» folder. The contents of the folder are shown in the figure:



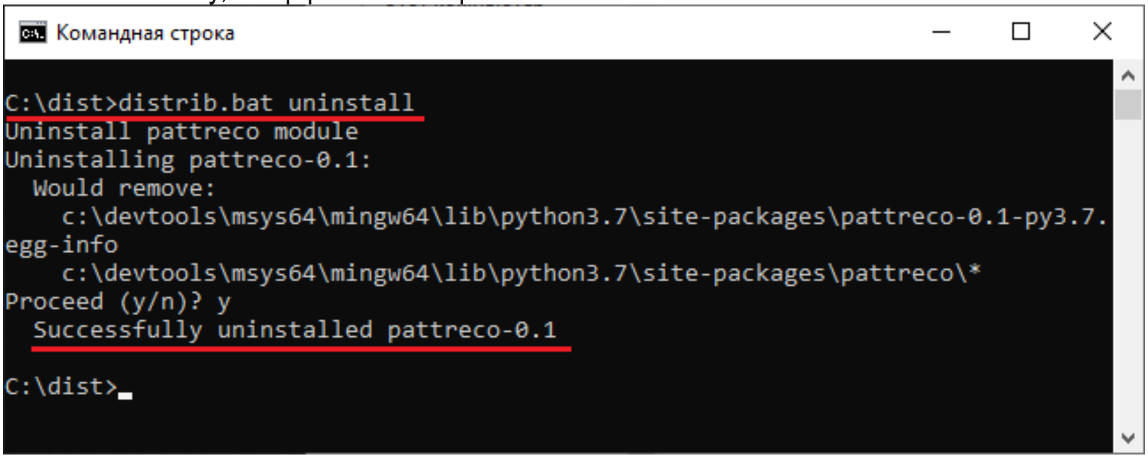
Имя	Дата изменения	Тип	Разм
distrib.bat	09.08.2019 10:38	Пакетный файл ...	
pattreco-0.1.tar.gz	08.08.2019 09:08	gz Archive	

Process of installation.
To install the module, go to the «/dist» folder and run the install_distrib.bat file.



```
Microsoft Windows [Version 10.0.18362.267]
(c) Корпорация Майкрософт (Microsoft Corporation), 2019. Все права защищены.

C:\Users\andrew>cd C:\dist

C:\dist>distrib.bat install
Install pattreco module
Processing c:\dist\pattreco-0.1.tar.gz
Installing collected packages: pattreco
  Running setup.py install for pattreco ... done
Successfully installed pattreco-0.1

C:\dist>_
```

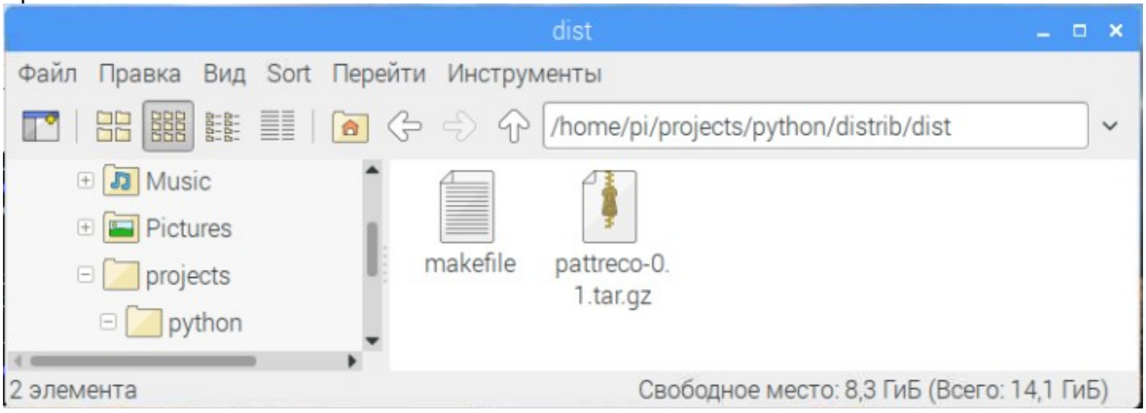
Process of uninstallation
To uninstall the library, run pip uninstall «pattreco» on the command line.



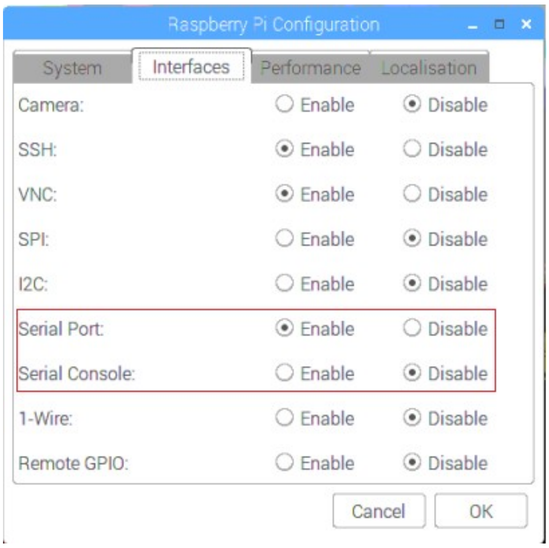
```

C:\dist>distrib.bat uninstall
Uninstall pattreco module
Uninstalling pattreco-0.1:
  Would remove:
    c:\devtools\msys64\mingw64\lib\python3.7\site-packages\pattreco-0.1-py3.7.
egg-info
    c:\devtools\msys64\mingw64\lib\python3.7\site-packages\pattreco\*
Proceed (y/n)? y
  Successfully uninstalled pattreco-0.1

C:\dist>_
```

Installation under Linux

The «pattreco" module is distributed as a «/dist» folder. The contents of the folder are shown in the figure:


dist
Файл Правка Вид Sort Перейти Инструменты
/home/pi/projects/python/distrib/dist
Music
Pictures
projects
python
makefile
pattreco-0.1.tar.gz
2 элемента
Свободное место: 8,3 ГиБ (Всего: 14,1 ГиБ)

Serial port must be enabled:



Camera: ○ Enable ● Disable

SSH: ● Enable ○ Disable

VNC: ● Enable ○ Disable

SPI: ○ Enable ● Disable

I2C: ○ Enable ● Disable

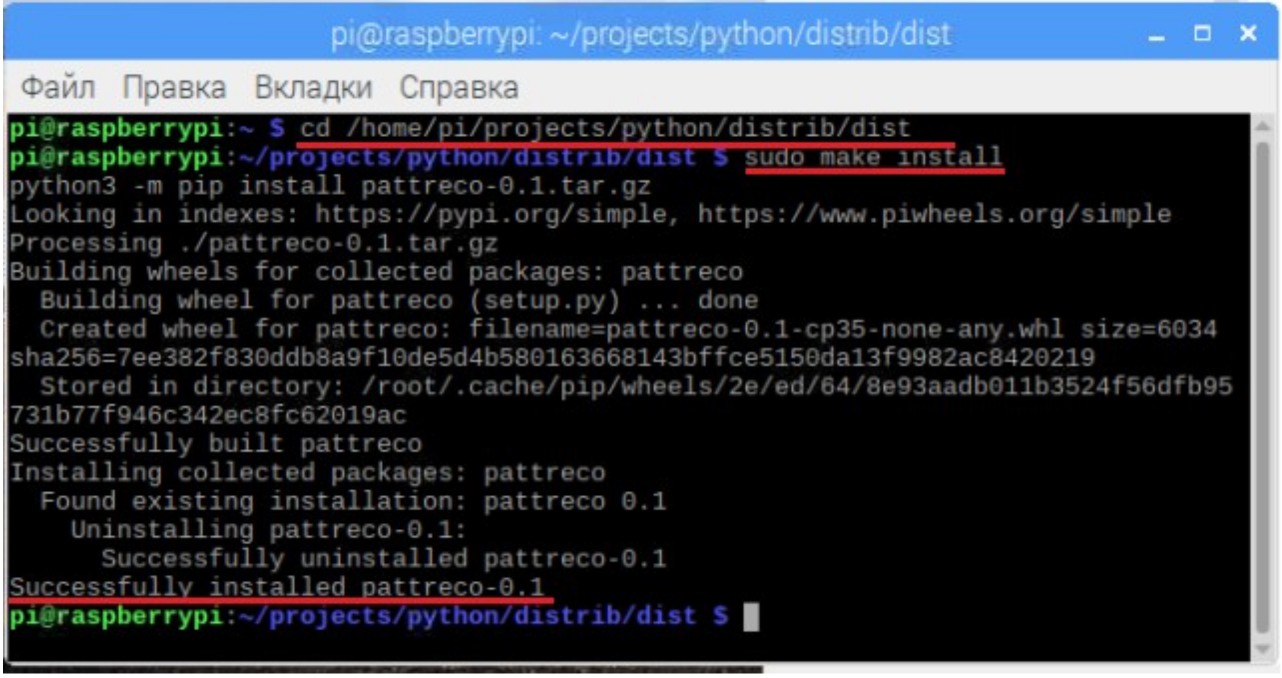
Serial Port: ● Enable ○ Disable

Serial Console: ○ Enable ● Disable

1-Wire: ○ Enable ● Disable

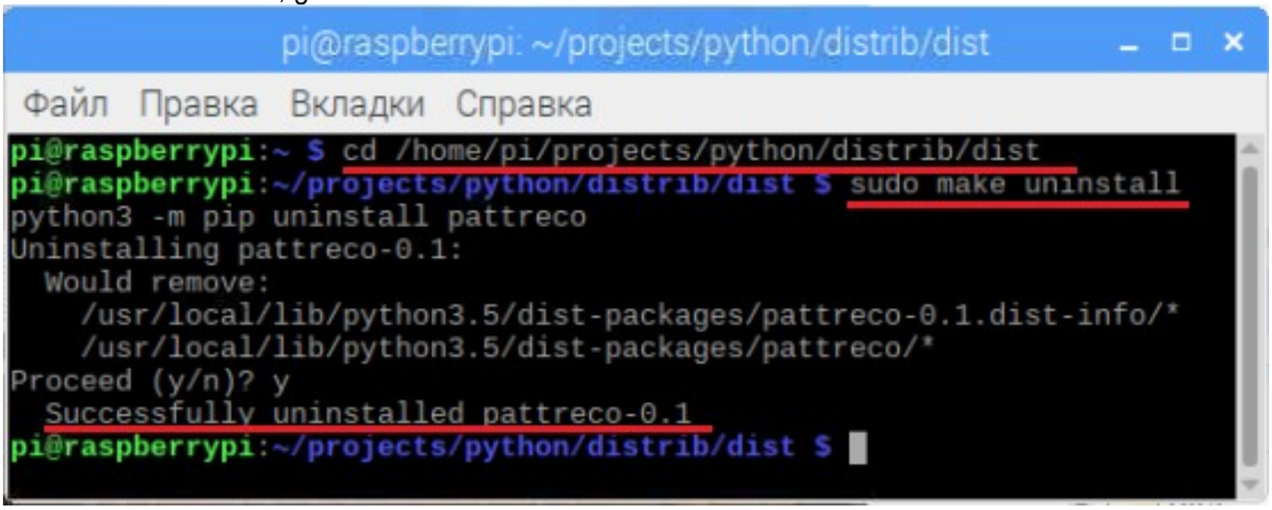
Remote GPIO: ○ Enable ● Disable

Process of installation.
To install the module, go to the «/dist» folder on the command line and run sudo make install.


pi@raspberrypi: ~/projects/python/distrib/dist
Файл Правка Вкладки Справка

```
pi@raspberrypi:~ $ cd /home/pi/projects/python/distrib/dist
pi@raspberrypi:~/projects/python/distrib/dist $ sudo make install
python3 -m pip install pattreco-0.1.tar.gz
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Processing ./pattreco-0.1.tar.gz
Building wheels for collected packages: pattreco
  Building wheel for pattreco (setup.py) ... done
  Created wheel for pattreco: filename=pattreco-0.1-cp35-none-any.whl size=6034
sha256=7ee382f830ddb8a9f10de5d4b580163668143bffce5150da13f9982ac8420219
  Stored in directory: /root/.cache/pip/wheels/2e/ed/64/8e93aadb011b3524f56dfb95
731b77f946c342ec8fc62019ac
Successfully built pattreco
Installing collected packages: pattreco
  Found existing installation: pattreco 0.1
    Uninstalling pattreco-0.1:
      Successfully uninstalled pattreco-0.1
Successfully installed pattreco-0.1
pi@raspberrypi:~/projects/python/distrib/dist $ 
```

Process of uninstallation.
To uninstall the module, go to the dist folder on the command line and run sudo make uninstall.


pi@raspberrypi: ~/projects/python/distrib/dist
Файл Правка Вкладки Справка

```
pi@raspberrypi:~ $ cd /home/pi/projects/python/distrib/dist
pi@raspberrypi:~/projects/python/distrib/dist $ sudo make uninstall
python3 -m pip uninstall pattreco
Uninstalling pattreco-0.1:
  Would remove:
    /usr/local/lib/python3.5/dist-packages/pattreco-0.1.dist-info/*
    /usr/local/lib/python3.5/dist-packages/pattreco/*
Proceed (y/n)? y
  Successfully uninstalled pattreco-0.1
pi@raspberrypi:~/projects/python/distrib/dist $ 
```

Connecting and using the module

Module connection

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

import pattreco.class_lib as cl
```

Using the module

Reading a register

```
# create an object of class nm500
self.nm500 = cl.nm500 ()

# find the COM port to which the board is connected
portname = self.nm500.uart.port_find ()

# open the COM port
result = self.nm500.uart.open_port (portname)

# read MINIF register
value = self.nm500.register_read (6)

# output the result
print (“value of register% 2d =% 5d \ n”% (value))

# close the COM port
self.nm500.uart.close_port ()
```

Loading the training vector

```
# create an object of class nm500
self.nm500 = cl.nm500 ()

# find the COM port to which the board is connected
portname = self.nm500.uart.port_find ()

# open the COM port
result = self.nm500.uart.open_port (portname)

# prepare vector components
vector = bytearray ([225, 226, 224, 174, 170, 160, 123, 25, 12, 120])

# loading vector for training
```

```
pack = learn_vector (0, # metric for the current vector (L1)
                    1, # the context value for the current vector
                    22, # category value for the current vector
                    400, # MAXIF value for the current vector
                    2, # MINIF value for current vector
                    10, # the number of components in the vector for training
                    vector # vector component values for training
)

# output the result
print (“committed neurons-”, pack [1])

# close the COM port
self.nm500.uart.close_port ()
```