Cách thêm thư viện vào pycharm:

Chuẩn bị file:

__init__.py

Code:

```
from .fxplc import FXPLC, RegisterDef, NotSupportedCommandError,
NoResponseError, ResponseMalformedError
```

fxplc.py

Code:

```
import binascii
import enum
import logging
import struct
from typing import Tuple, Union

import serial

logger = logging.getLogger("fxplc")


class NotSupportedCommandError(Exception):
    pass


class NoResponseError(Exception):
    pass


class ResponseMalformedError(Exception):
    pass


STX = b"\x02"  # Start of text
ETX = b'\x03'  # End of text
EOT = b'\x04'  # End of transmission
ENQ = b'\x05'  # Enquiry
ACK = b'\x06'  # Acknowledge
LF = b'\x0A'  # Line Feed
CL = b'\x0C'  # Clear
CR = b'\x0D'  # Carrier Return
NAK = b'\x15'  # Not Acknowledge

registers_map = {
    "S": 0x0000,
    "X": 0x0080,
    "Y": 0x00a0,
    "T": 0x00c0,
```

```
    "M": 0x0100,
    "D": 0x1000,
}

registers_map_counter = {
    "T": 0x0800,
}

registers_map_bits = {
    "S": 0x0000,
    "X": 0x0400,
    "Y": 0x0500,
    "T": 0x0600,
    "M": 0x0800,
}


class RegisterType(enum.Enum):
    State = "S"
    Input = "X"
    Output = "Y"
    Timer = "T"
    Memory = "M"
    Data = "D"


class RegisterDef:
    def __init__(self, type: RegisterType, num: int):
        self.type = type
        self.num = num

    def __str__(self):
        return f"{self.type.value}{self.num}"

    def get_bit_image_address(self) -> Tuple[int, int]:
        top_address = registers_map[self.type.value]
        return top_address + self.num // 8, self.num % 8

    @staticmethod
    def parse(definition: str) -> 'RegisterDef':
        return RegisterDef(type=RegisterType(definition[0]),
num=int(definition[1:]))


def calc_checksum(payload):
    return bytes(f"{sum(payload):02X}"[-2:].encode("ascii"))


class FXPLC:
    def __init__(self, port: str):
        self.serial = serial.Serial(port=port, timeout=1,
                                    baudrate=9600, bytesize=serial.SEVENBITS,
parity=serial.PARITY_EVEN, stopbits=serial.STOPBITS_ONE)

    def close(self):
        self.serial.close()
```

```
    def read_bit(self, register: Union[RegisterDef, str]) -> bool:
        if not isinstance(register, RegisterDef):
            register = RegisterDef.parse(register)
        addr, bit = register.get_bit_image_address()

        resp = self.read_bytes(addr, 1)
        return (resp[0] & (1 << bit)) != 0

    def write_bit(self, register: Union[RegisterDef, str], value: bool):
        if not isinstance(register, RegisterDef):
            register = RegisterDef.parse(register)
        top_address = registers_map_bits[register.type.value]
        addr = top_address + register.num

        self._send_command(7 if value else 8, struct.pack("<H", addr))

    def read_counter(self, register: Union[RegisterDef, str]) -> int:
        if not isinstance(register, RegisterDef):
            register = RegisterDef.parse(register)
        addr = registers_map_counter[register.type.value] + register.num * 2

        resp = self.read_bytes(addr, 2)

        value = struct.unpack("<H", resp)[0]
        return value

    def read_bytes(self, addr: int, count: int = 1) -> bytes:
        req = struct.pack(">HB", addr, count)
        resp = self._send_command(0, req)
        return resp

    def write_bytes(self, addr: int, values: bytes):
        req = struct.pack(">HB", addr, len(values)) + values
        self._send_command(1, req)

    def _send_command(self, cmd: int, data: bytes) -> bytes:
        cmd_hex = bytes([ord("0") + cmd])
        payload_hex = binascii.hexlify(data).upper()
        logger.debug("TX [cmd | payload]: " + cmd_hex.decode("ascii") + " | "
+ payload_hex.decode("ascii"))
        payload = cmd_hex + payload_hex

        frame = STX + payload + ETX + calc_checksum(payload + ETX)

        self.serial.flushOutput()
        self.serial.flushInput()
        self.serial.write(frame)

        return self._read_response()

    def _read_response(self):
        def format_code(_code):
            return f"RX [code]: {binascii.hexlify(_code).decode('ascii')}"

        def format_code_data(_code, _data):
            return f"RX [code | payload]:
{binascii.hexlify(_code).decode('ascii')} | {_data.decode('ascii')}"
```

```
        code = self.serial.read(1)
        if code == STX:
            data = b""
            while True:
                d = self.serial.read(1)
                if len(d) == 0:
                    logger.error(f"Invalid response - {format_code_data(code,
data)}")
                    raise ResponseMalformedError()

                if d == ETX:
                    break
                data += d

            logger.debug(format_code_data(code, data))

            checksum = self.serial.read(2)
            if len(checksum) != 2:
                logger.error(f"Invalid response - {format_code_data(code,
data)}")
                raise ResponseMalformedError()

            if calc_checksum(data + ETX) != checksum:
                logger.error(f"Wrong checksum - {format_code_data(code,
data)}")
                raise ResponseMalformedError()

            return binascii.unhexlify(data)
        elif code == NAK:
            raise NotSupportedCommandError()
        elif code == ACK:
            logger.debug(f"{format_code(code)} (ACK)")
        else:
            raise NoResponseError()


__all__ = [
    "NotSupportedCommandError",
    "NoResponseError",
    "ResponseMalformedError",
    "RegisterType",
    "RegisterDef",
    "FXPLC",
]
```

setup.py

Code:

```
from setuptools import setup

setup(
    name='fxplc',  # Tên thư viện
    version='1.0.0',
```

```
    description='A library for interfacing with FXPLC devices',
    author='Your Name',
    author_email='your_email@example.com',
    packages=['fxplc_lib'],  # Tên package (thư mục)
    install_requires=['pyserial'],  # Các thư viện phụ thuộc
)
```

cấu trúc thư viện:

[tên project]-[fxplc_lib]-[__init__.py]

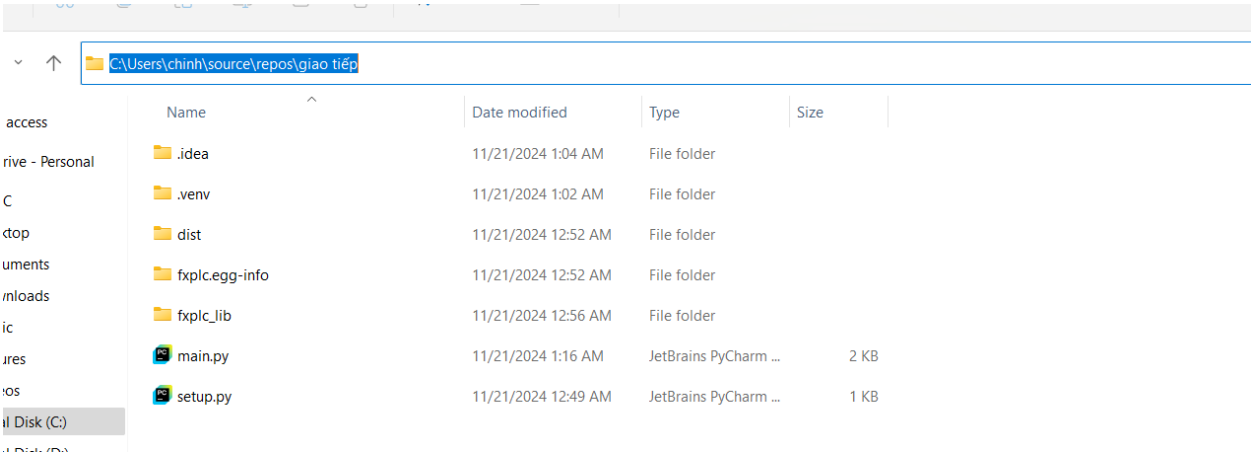
-[fxplc.py]

-[setup.py]

-[test]

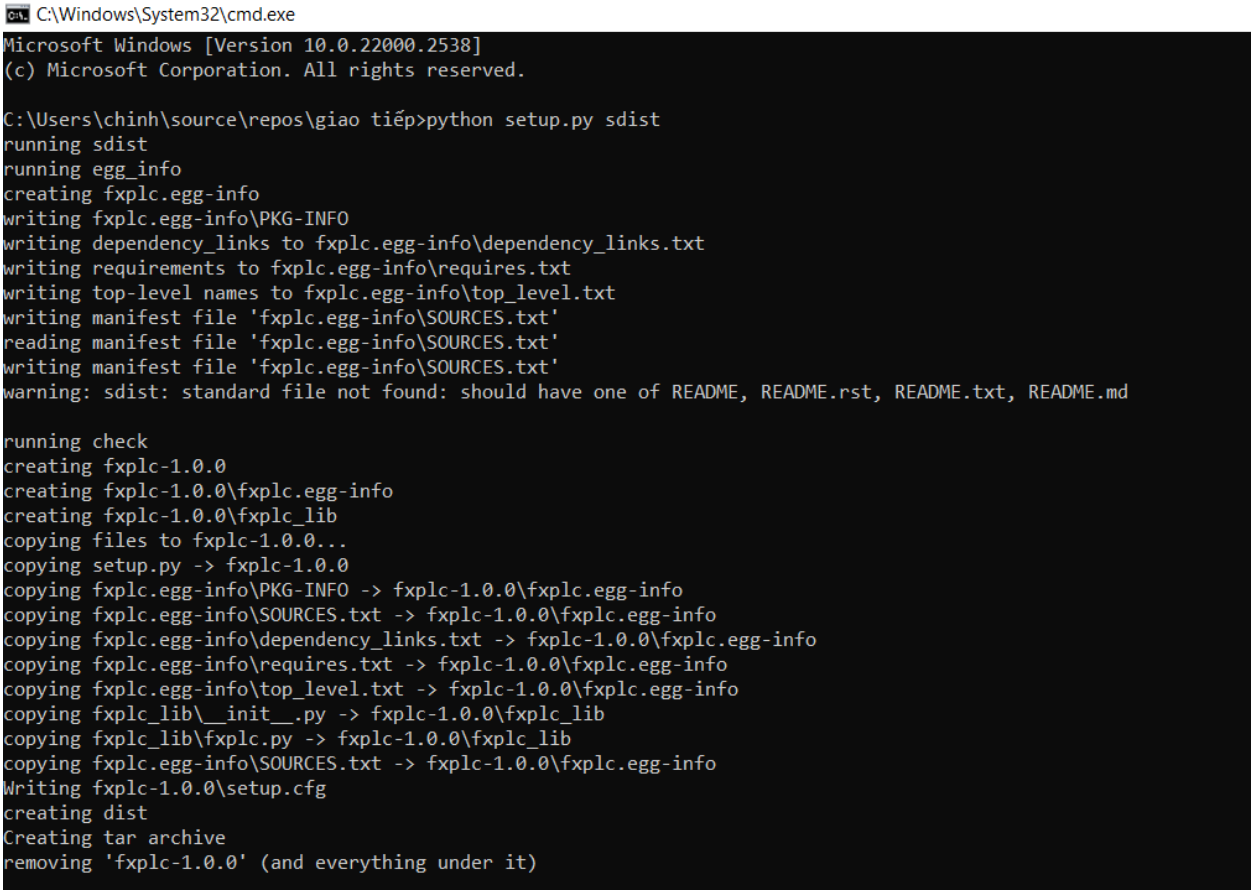
Cách thực hiện:

- mở cmd nơi chứa các file thư viện nơi chứa file setup.py :



chạy dòng : python setup.py sdist

Để đóng gói thư viện , trường hợp xuất xuất hiện báo lỗi thiếu file .md thì kệ!



- **Cài đặt thư viện vào môi trường ảo (virtual environment)**:

- Mở terminal trong PyCharm.
- Chạy lệnh:

```
pip install /path/to/your/fxplc_lib/dist/fxplc-1.0.0.tar.gz
```

```
và sử dụng bình thường!
```