



Федеральное государственное автономное образовательное учреждение высшего образования «Национальный Исследовательский Университет ИТМО»

ЛАБОРАТОРНАЯ РАБОТА №3
ПРЕДМЕТ «ПРОГРАММИРОВАНИЕ ПРОМЫШЛЕННЫХ
КОНТРОЛЛЕРОВ»
ТЕМА «ПЕРЕДАЧА ДАННЫХ С ПЛК ПО ПРОТОКОЛУ
MODBUS TCP И УПРАВЛЕНИЕ ЧАТОТНЫМ
ПРЕОБРАЗОВАТЕЛЕМ»

Преподаватель:
Крылова А. А.

Выполнили:
Румянцев А. А.
Дьячихин Д. Н.
Чебаненко Д. А.

Факультет: СУиР
Потоки ПРОГ. ПРОМ.ЛК:
2.2, 2.1, 1.1

Санкт-Петербург
2024

Содержание

1 Введение.	2
1.1 Цель работы.	2
1.2 Задания по работе.	2
2 Установка.	2
2.1 Схема установки.	2
3 Теоретическая часть.	3
4 Экспериментальная часть.	3
4.1 Передача данных с ПЛК по протоколу Modbus TCP.	3
4.2 Управление частотным преобразователем.	5
5 Выводы о проделанной работе.	6

1 Введение.

1.1 Цель работы.

Разработать программу для ПЛК, которая передает в сеть данные с датчиков температуры и влажности по протоколу ТСР. Разработать программу для управления скоростью и направлением вращения асинхронного двигателя с помощью частотного преобразователя.

1.2 Задания по работе.

1. Изучить теоретическую часть работы.
2. Реализовать модуль передачи данных в сеть на языке ST.
3. Реализовать модуль получения, обработки и отображения данных на языке Python.
4. Изучить документацию на частотный преобразователь.
5. Реализовать модуль для управления частотным преобразователем по протоколу Modbus ТСР.

2 Установка.

В данной лабораторной работе мы работали со стендом на основе ПЛК Schneider Electric Modicon M251.

2.1 Схема установки.

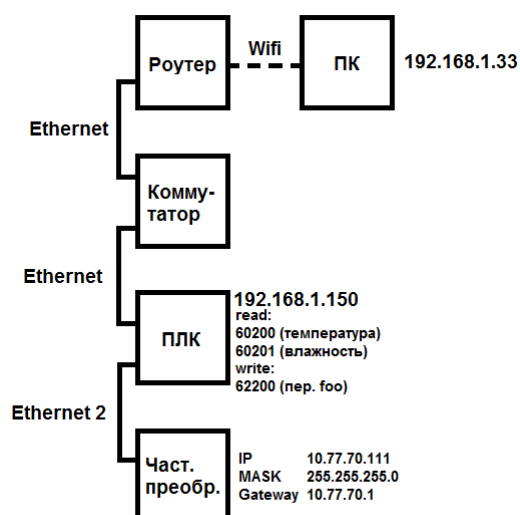


Рис. 1: Схема протоколов стенда.

Так как в данной лабораторной работе мы работаем с тем же стендом, что и в предыдущей работе, то общая схема остается неизменной. Связь ПК и ПЛК происходит через

коммутатор и Wi-Fi роутер. Во время выполнения работы ПК имел IP адрес 192.168.1.33 (было обнаружено с помощью команды `ifconfig`), а ПЛК – 192.168.1.150. 60200 и 60201 – адреса регистров датчика температуры и влажности соответственно, из которых можно прочесть данные с датчиков (для начала нужно подключиться). 62200 – аналогично для переменной *foo* (позже про нее будет написано подробнее), однако с той разницей, что это номер регистра для записи данных.

3 Теоретическая часть.

Modbus TCP – открытый протокол, не зависящий от производителя используемого аппаратного обеспечения. Предназначен для контроля и управления оборудованием в системах несложной промышленной автоматизации. В частности, позволяет реализовать обмен сообщениями Modbus в локальных и глобальных вычислительных сетях с использованием стека протоколов TCP/IP, который часто используется для подключения к общей промышленной сети ПЛК, модулей ввода-вывода и «шлюзов».

Частотный преобразователь – это электронное или электромеханическое устройство, которое преобразует переменный ток (AC) одной частоты в переменный ток другой частоты. Устройство также может изменять напряжение, но это является второстепенным по отношению к его основной цели, поскольку преобразование напряжения переменного тока гораздо проще реализовать, нежели преобразование частоты.

4 Экспериментальная часть.

За основу возьмем проект лабораторной работы №2.

4.1 Передача данных с ПЛК по протоколу Modbus TCP.

В дереве проекта на вкладке *Device tree* в узле *Ethernet_1* создадим объект *ModbusTCP_SLAVE_Device*.

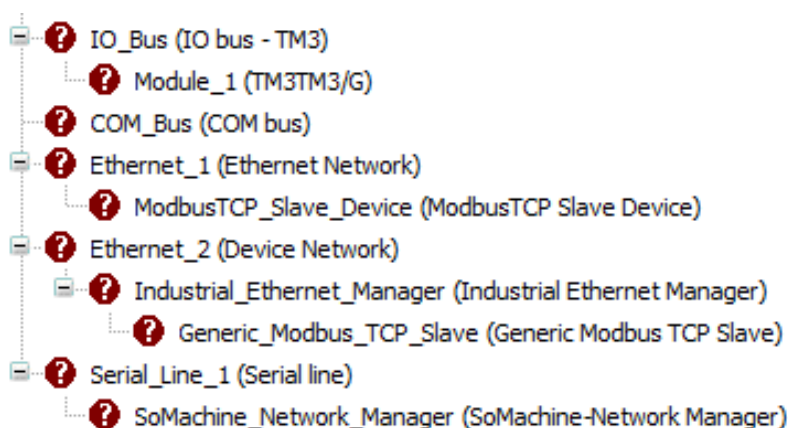


Рис. 2: Созданный объект *ModbusTCP_SLAVE_Device* в дереве проекта.

Создадим новый список глобальных переменных в узле *GVL* на вкладке *Application tree* и добавим туда переменную *foo* целого типа (остальные переменные будут рассмотрены позже).

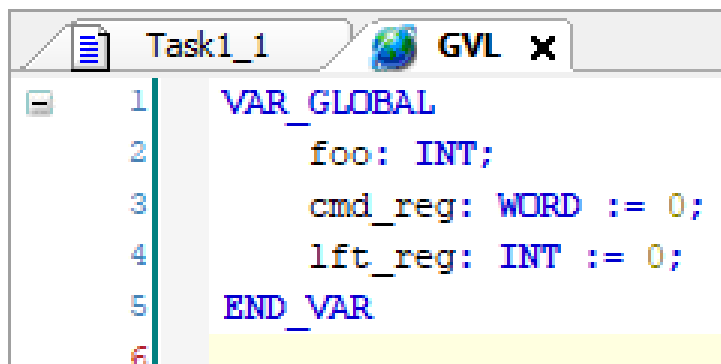


Рис. 3: Список глобальных переменных.

На вкладке *Tools Tree* создадим объект *Relocation Table*. В таблице *Read* создадим две новые пустые переменные *Modbus*. Нажмем на ячейку *Variable* и в открывшемся диалоге найдем переменные для температуры и влажности из предыдущей работы. Создадим пустую переменную в таблице *Write* и ассоциируем ее с глобальной переменной *foo*, которую мы задали ранее.

Далее напишем Python-скрипт, в котором реализуем код, необходимый для работы с ПЛК (в данном случае передача данных на ПЛК).

```

1  import time
2  from pyModbusTCP import utils
3  from pyModbusTCP.client import ModbusClient
4
5  def read_int16(addr):
6      regs = c.read_holding_registers(reg_addr=addr, reg_nb=1)
7      hex_uint = utils.get_list_2comp(regs, 16)
8      return hex_uint[0]
9
10 c = ModbusClient(host='192.168.1.150', port=502, auto_open=True)
11
12 while True:
13     if c.is_open:
14         temp = read_int16(60200) / 10
15         hum = read_int16(60201)
16         val = int(input('type int:'))
17         val = min(24000, max(0, val))
18         if c.write_single_register(62200, val):
19             print("ok")
20             print(temp, hum)
21     else:
22         c.open()
23     time.sleep(1)

```

Листинг 1: Python-скрипт для взаимодействия с ПЛК.

Данный скрипт устанавливает соединение с ПЛК по протоколу ModbusTCP, после чего начинает бесконечный циклический опрос регистров, содержащих значения температуры и влажности, а также запись введенных пользователем в консоль значений (*val*) в регистр, ассоциированный с глобальной переменной проекта. При этом значение переменной не может превысить 24000 или быть ниже 0.

После мы загрузили программу в ПЛК и запустили ее, тем самым убедившись в том, что все скомпилировалось без ошибок, значение глобальной переменной в узле *GVL* изменяется всегда, когда пользователь вводит некоторое значение в консоль, данные температуры и влажности отображаются в консоли.

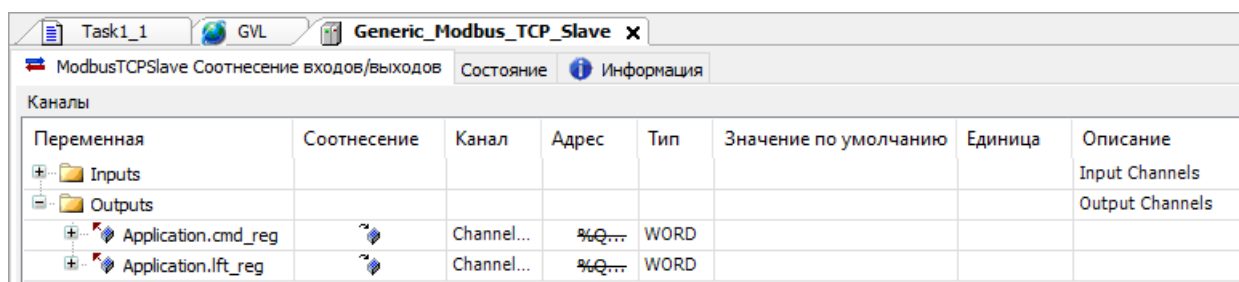
Мы приравнивали переменную *foo* к переменной *motor*, чтобы при передаче данных в первую из них вторая подбирала это значение и передавала на мотор. Программа корректно выполнялась – мотор вращался с той скоростью, которую пользователь ввел в консоль на своем ПК. Подаваемые значения не превышали заданный диапазон значений и не были ниже его.

4.2 Управление частотным преобразователем.

Данное задание является продолжением предыдущего. В дерево устройств добавим устройство *Generic_ModbusTCP_Slave* к узлу *Ethernet_2* (см. рис. 2). После этого зададим IP адрес частотного преобразователя 10.77.70.110 (см. рис. 1).

Далее на вкладке *Modbus TCP Channel Configuration* нажмем кнопку *Add Channel* и в открывшемся диалоговом окне зададим два канала управления для включения частотного преобразователя (регистр CMD, адрес 8501) и частоты (регистр LFR, адрес 8502).

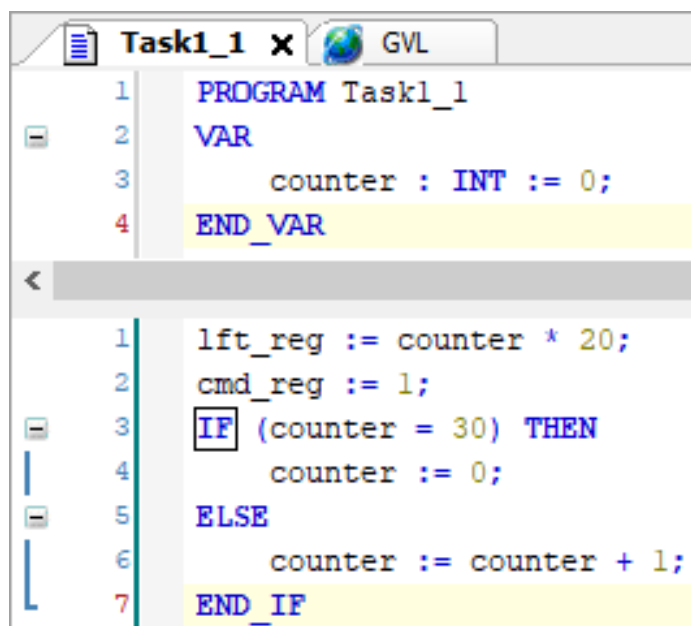
Для описанных ранее регистров в узле *GVL* создадим две новые глобальные переменные – *cmd_reg* и *ifr_reg* (см. рис. 3). После этого ассоциируем их с каналами управления на вкладке *ModbusTCPSlave I/O Mapping*.



Переменная	Соотнесение	Канал	Адрес	Тип	Значение по умолчанию	Единица	Описание
Inputs							Input Channels
Outputs							Output Channels
Application.cmd_reg		Channel...	%Q...	WORD			
Application.lfr_reg		Channel...	%Q...	WORD			

Рис. 4: ModbusTCPSlave I/O Mapping (Outputs).

На вкладке *Application Tree* добавим новую программу с именем *TASK1_1*. Создадим в ней локальную переменную *counter* и напишем программу для управления частотным преобразователем.



```

1 PROGRAM Task1_1
2 VAR
3     counter : INT := 0;
4 END_VAR
5
6 lfr_reg := counter * 20;
7 cmd_reg := 1;
8 IF (counter = 30) THEN
9     counter := 0;
10 ELSE
11     counter := counter + 1;
12 END_IF

```

Рис. 5: Программа для управления частотным преобразователем.

Создадим циклическую задачу в MAST и ассоциируем ее с программой управления скоростью частотного преобразователя. Интервал поставим в одну секунду, чтобы значения не обновлялись слишком часто.

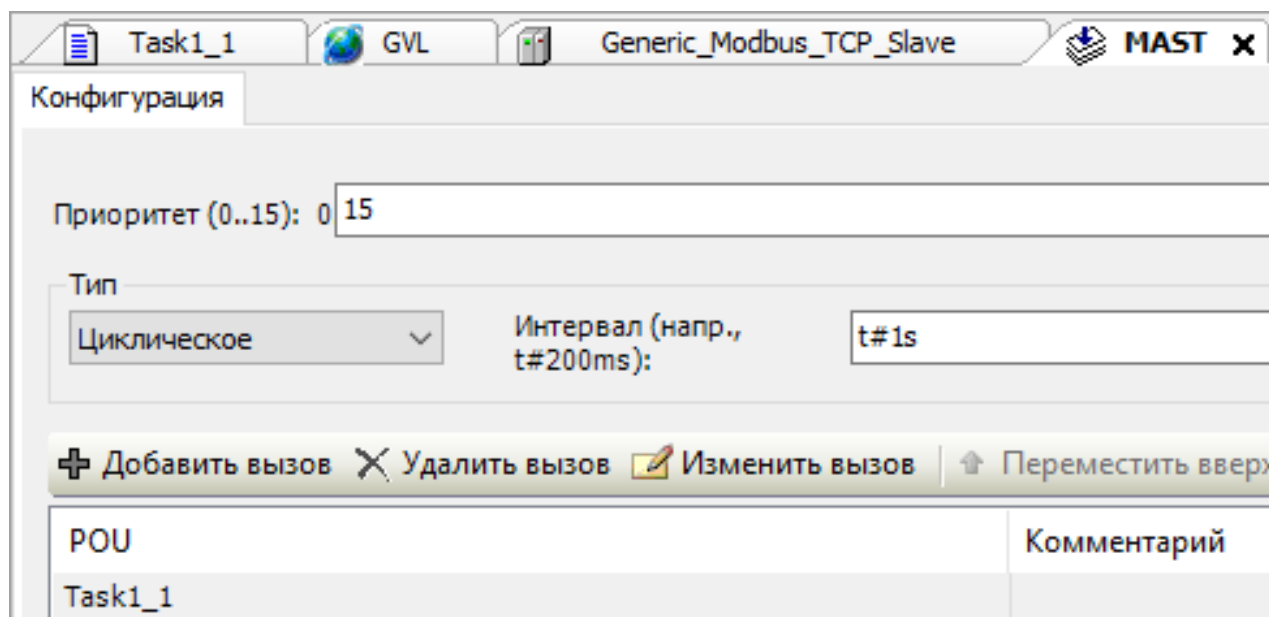


Рис. 6: Циклическая задача в MAST, к которой привязан TASK1_1.

Программа работает следующим образом. Для включения частотного преобразователя необходимо в нулевой бит регистра CMD записать значение 1, то есть в нашем случае просто присвоить значение 1 всей переменной. Изначальная частота вращения двигателя равна 0 (переменная *ifr_reg*). На каждом цикле программы мы берем локальный счётчик, умножаем его значение на 20, а затем записываем в регистр частоты, после чего инкрементируем счётчик, а при достижении значения 30 обнуляем его. Таким образом частота плавно увеличивается от 0 до 600 Гц. При этом скорость вращения двигателя будет зависеть от его паспортных характеристик, например, если на шильдике двигателя написано 50 Гц, 1400 об/мин., частота будет меняться в диапазоне от 0 до 2800 об/мин.

После мы запустили программу на стенде ПЛК и частотный преобразователь выполнял действия в соответствии с ранее описанным алгоритмом. Данные о частоте мы отслеживали с помощью специального пульта, прилагаемого к предоставленному нам ПЛК.

5 Выводы о проделанной работе.

В ходе лабораторной работы мы разработали программу для ПЛК на языке ST, которая передает в сеть данные с датчиков температуры и влажности по протоколу TCP, а также принимает значения для управления скоростью мотора. Также мы написали программу для получения, обработки и отображения данных на языке Python. В конце мы поработали с частотным преобразователем и разработали программу для управления скоростью и направлением вращения асинхронного двигателя с помощью частотного преобразователя по протоколу Modbus TCP.