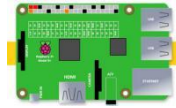
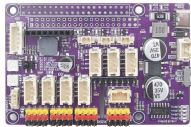


Lesson 22 Control the LED through TCP

22.1 Overview

In this lesson, we will create a system that uses a Raspberry Pi to control an LED. By leveraging the TCP protocol, instructions will be sent from a client device to the Raspberry Pi connected on the server side, and then the Raspberry Pi will control the state of the LED. This aims to enable beginners to understand how to combine TCP-based communication with simple hardware control.

22.2 Required Components

Components	Quantity	Picture
Raspberry Pi	1	
Adept Robot HAT V3.2	1	

22.3 Principle Introduction

In our system, the client will send commands like "LED1 ON" or "LED1 OFF" over the TCP connection. The server - side Raspberry Pi will receive these commands and use the GPIO pins to control the LED's state accordingly.

Instruction	Describe
LED1 ON	Turn on the LED labeled as LED1
LED1 OFF	Turn off the LED labeled as LED1

LED2 ON	Turn on the LED labeled as LED2
LED2 OFF	Turn off the LED labeled as LED2
LED3 ON	Turn on the LED labeled as LED3
LED3 OFF	Turn off the LED labeled as LED3
LEDALL ON	Turn on all the LEDs in the system
LEDALL OFF	Turn off all the LEDs in the system

22.4 Demonstration

1. **Remotely log:** Remote login as a server-side Raspberry Pi terminal.
2. **Navigate to the Program Folder:** Enter the following command in the terminal and press Enter to access the folder where the program is located:

```
cd Adeept_PiCar-Pro/Examples/15_Remote_Control/
```

```
pi@raspberrypi:~ $ cd Adeept_PiCar-Pro/Examples/15_Remote_Control
pi@raspberrypi:~/Adeept_PiCar-Pro/Examples/15_Remote_Control $
```

3. **View Directory Contents:** Type "ls" in the terminal and press Enter. This will display all the files in the current directory, ensuring that the "**LedServer.py**" and "**LedClient.py**" file is present:

```
ls
```

```
pi@raspberrypi:~/Adeept_PiCar-Pro/Examples/15_Remote_Control $ ls
Client.py  LedClient.py  LedServer.py  Server.py
```

4. Enter the command below and press Enter to start the **LedServer.py** program:

```
sudo python3 LedServer.py
```

```
pi@raspberrypi:~/Adeept_PiCar-Pro/Examples/15_Remote_Control $ sudo python3 LedS
erver.py
Server has started and is listening for connections...
Accepted connection from ('192.168.3.31', 35178)
```

5. To run the client program, you need to provide the server's IP address as a parameter. Use the following command, replacing <server_ip> with the actual IP address of the Raspberry Pi running the serve:

```
sudo python3 LedClient.py <server_ip>
```

```
pi@raspberrypi:~/Adeept_PiCar-Pro/Examples/15_Remote_Control $ sudo python3 LedClient.py 192.168.3.31
Connected to server 192.168.3.31:8000
Please enter the message to send (type 'exit' to quit):
```

For example, my Raspberry Pi IP address is "192.168.3.31", and the command to run the client program is as follows:

```
sudo python3 LedClient.py 192.168.3.31
```

On Raspberry Pi servers, the script will prompt you to enter commands. For example, you can input "**LED1 ON**". After entering the command, the client will send it to the client through a TCP connection..

Client - side:

```
pi@raspberrypi:~/Adeept_PiCar-Pro/Examples/15_Remote_Control $ sudo python3 LedClient.py 192.168.3.31
Connected to server 192.168.3.31:8000
Please enter the message to send (type 'exit' to quit): LED1 ON
Please enter the message to send (type 'exit' to quit):
```

On the server-side terminal, you will see the received command printed out and the led1 light will be illuminated.

Server - side:

```
pi@raspberrypi:~/Adeept_PiCar-Pro/Examples/15_Remote_Control $ sudo python3 LedServer.py
Server has started and is listening for connections...
Accepted connection from ('192.168.3.31', 35178)
Received command: LED1 ON
LED1 is turned on
```

Now you can see that the onboard LED is already on. You can also control the on and off states of other LEDs by using the commands in the previous table.

Termination:

When you want to terminate a running program, you can press the "**Ctrl+C**" shortcut key on the keyboard or enter "**exit**" on the keyboard and click "**Enter**".

22.5 Code

Complete code refer to [LedServer.py](#).

```
001  #!/usr/bin/env/python
002  # File name   : LedServer.py
003  # Website    : www.Adeept.com
004  # Author     : Adeept
005  # Date      : 2025/03/11
006  import socket
007  import threading
008  from gpiozero import LED
009
010  # Function to set up the LED objects.
011  # Initializes three LED objects corresponding to different GPIO pins.
012  def switchSetup():
013      global led1, led2, led3
014      led1 = LED(9)
015      led2 = LED(25)
016      led3 = LED(11)
017
018  # Function to control the state of a specific LED.
019  # port: The number of the LED (1, 2, or 3).
020  # status: 1 to turn the LED on, 0 to turn it off.
021  def switch(port, status):
022      if port == 1:
023          if status == 1:
024              led1.on()
025          elif status == 0:
026              led1.off()
027      elif port == 2:
028          if status == 1:
029              led2.on()
030          elif status == 0:
031              led2.off()
032      elif port == 3:
033          if status == 1:
034              led3.on()
035          elif status == 0:
036              led3.off()
037      else:
038          print('Wrong Command: Example--switch(3, 1)->to switch on port3')
039
040  # Function to handle client connections.
041  # client_socket: The socket object used for communication with the client.
042  # client_address: The address of the client.
043  def handle_client(client_socket, client_address):
044      try:
045          while True:
046              data = client_socket.recv(1024)
047              if not data:
048                  break
049              try:
050                  message = data.decode('utf-8')
051                  print(f"Received command: {message}")
```

```
052         if message.startswith("LED"):
053             parts = message.split()
054             if message == "LEDALL ON":
055                 switch(1, 1)
056                 switch(2, 1)
057                 switch(3, 1)
058                 print("All LEDs are turned on")
059             elif message == "LEDALL OFF":
060                 switch(1, 0)
061                 switch(2, 0)
062                 switch(3, 0)
063                 print("All LEDs are turned off")
064             else:
065                 led_num = int(parts[0][3:])
066                 if len(parts) == 2:
067                     if parts[1] == "ON":
068                         switch(led_num, 1)
069                         print(f"LED{led_num} is turned on")
070                     elif parts[1] == "OFF":
071                         switch(led_num, 0)
072                         print(f"LED{led_num} is turned off")
073                     else:
074                         print(f"Invalid command: {message}")
075             else:
076                 print(f"Invalid command: {message}")
077         except UnicodeDecodeError:
078             print(f"Error decoding the received data: {data}")
079     except socket.error as e:
080         print(f"Socket error while communicating with {client_address}: {e}")
081     except Exception as e:
082         print(f"Error handling the client request: {e}")
083     finally:
084         client_socket.close()
085
086 # Create a TCP-based socket object.
087 # AF_INET indicates the IPv4 address family, and SOCK_STREAM indicates the TCP protocol.
088 server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
089
090 # Bind the socket to all available network interfaces and set the port number to 8000.
091 server_address = ('0.0.0.0', 8000)
092 server_socket.bind(server_address)
093
094 # Start listening for incoming connections.
095 # The maximum number of queued connections is set to 5.
096 server_socket.listen(5)
097 print("Server has started and is listening for connections...")
098
099 switchSetup()
100
101 while True:
102     client_socket, client_address = server_socket.accept()
103     print(f"Accepted connection from {client_address}")
104     client_thread = threading.Thread(target=handle_client, args=(client_socket, client_address))
105     client_thread.start()
```

Complete code refer to [LedClient.py](#).

```
01  #!/usr/bin/env/python
02  # File name   : LedClient.py
03  # Website    : www.Adeept.com
04  # Author     : Adeept
05  # Date      : 2025/03/11
06  import socket
07  import sys
08
09  if len(sys.argv)!= 2:
10      print("Please enter the server's IP address when running, for example: python3 client.py
11      192.168.1.100")
12      sys.exit(1)
13
14  server_ip = sys.argv[1]
15  server_port = 8000
16
17  # Create a TCP - based socket object
18  client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
19
20  try:
21      # Connect to the server
22      client_socket.connect((server_ip, server_port))
23      print(f"Connected to server {server_ip}:{server_port}")
24      while True:
25          # Get input from the keyboard
26          message = input("Please enter the message to send (type 'exit' to quit): ")
27          if message.lower() == 'exit':
28              break
29          # Send the message to the server
30          client_socket.send(message.encode('utf-8'))
31  except socket.error as e:
32      print(f"Error connecting to the server: {e}")
33  finally:
34      # Close the client socket
35      client_socket.close()
36
```

Code explanation

[LedServer.py](#)

Import the necessary libraries and initialize three LED objects.

Define a function to control the on and off states of the LEDs.

Create a TCP socket, bind it to an address and port, and start listening.

Enter a loop and wait for client connections.

When there is a connection, create a new thread. In the thread, receive commands from the client, and control the on and off states of the LEDs after parsing the commands.

[LedClient.py](#)

Import the socket and sys libraries, check the running parameters, and obtain the server's IP address.

Create a TCP socket and connect to the server.

Enter a loop to receive user input. If the input is exit, exit the loop; otherwise, encode the message and send it to the server.