



INTERFACING WITH THE RASPBERRY PI

By:
AAYUSH ANSHU



SSH

- Firewall: block some internet ports
- Secure Shell (SSH): Program that allow you to access machine remotely
- Telnet: SSH not secure
 - *Linus has an SSH client*
 - *RPie has an SSH server*
- `ssh userName@IPAddress` #Of server machine #Run SSH Client
- SSHDeamon: A process that waits a client to request a connection (not enabled by default)
- `ifconfig` #Get RPie IP address (wlan0) #Because it doesn't have a DN
- PS: IP address changes every time we reboot the RPie

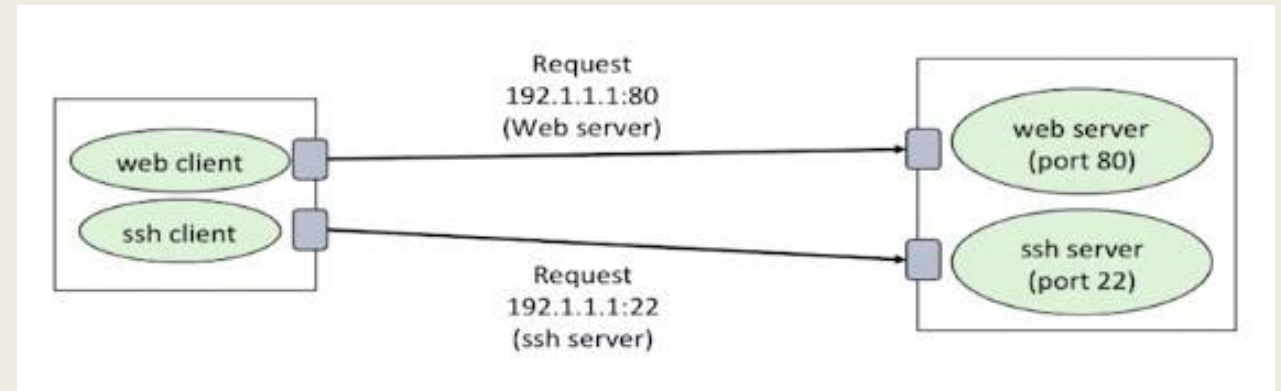
Protocols

- Protocol: Set of rules defining:
 - *How data should be transferred*
 - *What data contained in each packet*
- Every packet contains a header (destination, source) and payload (data)
 - *IP: Internet Protocol (Host to host)*
 - *UDP: Unreliable Datagram Protocol (Complicated, process to process)*
 - *TCP: Transmission Control Protocol (Simple)*

PS: Port + IP = Internet

- IPv6 (128 bit) vs IPv4 (32 bit)
- Ports: (Example: HTTP-80; SSH-22)
- nslookup domainName #Give IP@ of domainName
- Domain Name System: Determine between host name (@IP) and domain name (URL)

Sockets (1)



- Socket: End point of a connection
- Socket interface: Programming interface to perform network connection → Based on C/S programming
- Socket client:
 - `import socket` #Socket python library
 - `ms = socket.socket(sock.AF_INET,socket.SOCK_STREAM)` #Create a socket #Arg1: Internet, Arg2: TCP
 - `host = socket.gethostbyname("URL")` #Convert URL to IP@
 - `ms.connect((host,80))` #Create connection to host on port 80
 - `message = "GET / HTTP/1.1\r\n\r\n"` #/: Directory, 1.1: Version, \r\n\r\n: End of session
 - `ms.sendall(message)` #Send data
 - `data = ms.recv(maxNumberOfBytes)` #Recieve HTML data on a blocking wait
 - `ms.close()`

Sockets (2)

■PS: Errors are common with sockets

try:

 /* code with possible error*/

except errorType:

 /* executed when error occurs */

socket.error #Error type that contains all socket errors

gaierror #Get address info error (when DNS operation occurs)

Import sys; sys.exit() #To quit the program in case of error

Sockets (3)

- Socket server: Server need to wait for a request to come in
- `ms.bind(“”,port) #“”` to receive from any host, port=1234 for example
- `ms.listen(backLog=numberOfClientsWaitingInLine)` #Get ready to accept requests
- `conn, IPaddr = ms.accept()` #Accept request to receive and send data
- `conn.sendall() ; conn.recv() ; conn.close() ; ms.close()`
- `resp = conn.getresponse(); content = resp.read()` #Read content
- Live server: Server that continues its life (after a request)
- PS: if `data==b'message'` #Convert to byte array because data's type is a byte array
- PuTTY: SSH program for Windows
- NetCat: Program that sends and receives requests
- `nc IP@ Port` #Send a request
- `nc -l Port` #Listen for a request

Sockets (4)

```
import socket
import sys

mysock = socket.socket(socket.AF_INET,
                        socket.SOCK_STREAM)

try:
    mysock.bind("", 1234)
except socket.error:
    print("Failed to bind")
    sys.exit()
mysock.listen(5)
while True:
    conn, addr = mysock.accept()
    data = conn.recv(1000)
    if not data:
        break
    conn.sendall(data)

conn.close()
mysock.close()
```

```
import socket
import sys

try:
    mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
except socket.error:
    print("Failed to create socket")
    sys.exit()

try:
    host = socket.gethostbyname("www.google.com")
except socket.gaierror:
    print("Failed to get host")
    sys.exit()

mysock.connect(host, 80)
message = "GET / HTTP/1.1\r\n\r\n"
try:
    mysock.sendall(message)
except socket.error:
    print("Failed to send")
    sys.exit()

data = mysock.recv(1000)
print(data)
mysock.close()
```

Protocol-Specific Libraries

- Socket library → Low level library → Good for ad hoc communication (Must know protocols)
- PS: RPi can't handle some operations → Need of online servers and clouds
- `import http.client`
- `conn = httplib.HTTPConnection("URL")`
- `conn.request("GET", "/")` #GET=Operation; /=Directory
- Web-based services = Services in the cloud → "HTTP" messages: Remote server x Client request
- API: Application Programming Interface → Format of messages between client and server (Basic-HTTP)
- SDK: Software Development Kit → Set of tools to support the use of an API (library functions)

Twython

- `sudo apt-get update`
- `sudo apt-get install python -pip` #PIP: Python installer for packages
- `sudo pip install twython`

→ 4 keys for authentication

- `from Twython import Twython`
- #We do initialization for the 4 keys
- `api=Twython(key1,key2,key3,key4)`
- `api.update_status(status="messageToSend")` #Post a tweet
- `class myStreamer (TwythonStreamer)` #Extend a class and change what you want
- `statuses.filter()` #Search for text in streams
- `stream.status_filter(track='wordToTrack')` #Search for a word in stream
- `execfile("filename.extension")` #Python command to execute a file

```
class MyStreamer(TwythonStreamer):  
    def on_success(self, data):  
        if 'text' in data:  
            print("Found it.")
```

Camera

- Camera: by USB vs by CSI (Camera Serial Interface)
- `sudo raspi-config` → Enable camera → Reboot Pi
- `sudo apt-get install python3-picamera`
- `import picamera`
- `camera = picamera.PiCamera()` #Create a camera object
- `camera.capture("nameOfImage.jpg")` #Capture an image
- `camera.start_preview()` ; `camera.stop_preview()` ; #Take a preview
- `import time; camera.start_recording("nameOfVideo.h264")`
- `time.sleep(timeInSeconds); camera.stop_recording()` #Create a video
- `for filename in camera.capture_continuous(): time.sleep(timeInSeconds)` #Take picture every sec

```
mysocket = socket.socket()
mysocket.connect(('aserver', 8000))
conn = mysocket.makefile('wb')
camera.capture(conn, 'jpeg')
```

Servo

- DC: Control the speed vs Servo: Control the angle
- 1 ms width = 0 degrees → 2 ms width = 180 degrees
 - For motors, we use external power supply because they consume a lot of power
 - We want all the grounds to be related → Common ground (Battery, RPi, Servo)
 - Only pin 12 & 24 are PWM
 - Resistor is needed between the servo and the pin for protection (1kΩ)
- `pwm=GPIO.PWM(pinNumber, Frequency=50)` #Set pin as PWM
- `pwm.start(0)` #Set duty cycle to low all the time
- `pwm.ChangeDutyCycle(valueOfDutyCycle)`
- PS: Any object that is related to pin, we do a PULL UP for it