Electronics for IoT

Autonomous Remote Operation

Bernhard E. Boser
University of California, Berkeley
boser@eecs.berkeley.edu

Outline

- Last time:
  - Connect to WiFi
  - mDNS
- Today:
  - Programming via WiFi
  - Fetch time from Internet
  - Autonomous operation
  - Example app:
    - Wireless LED control

Connect to WiFi

```
# Establish Internet connection
from network import WLAN, STA_IF
from network import mDNS
import time

wlan = WLAN(STA_IF)
wlan.active(True)

wlan.connect('EECS-PSK', 'Thequickbrown', 5000)

while not wlan.isconnected():
    print("Waiting for wlan connection")
    time.sleep(1)

print("WiFi connected at", wlan.ifconfig()[0])
```

Download code from course webpage

mDNS: Advertise Hostname

```
# Advertise as 'hostname', alternative to IP address
try:
    hostname = 'ee49'
    mdns = mDNS(wlan)
    mdns.start(hostname, "MicroPython REPL")
    mdns.addService('_repl', '_tcp', 23, hostname)
    print("Advertised locally as {}.local".format(hostname))
except OSError:
    print("Failed starting mDNS server - already started?")
```

Works only on “local net”

- Typically ESP32 and host computer connected to same WiFi

Telnet Server: Remote Login

```
# start telnet server for remote login
from network import telnet

print("start telnet server")
telnet.start(user='micro', password='python')
```

- Security:
  - Choose different user/password!

Wireless Connection

In shell49:

1. Disconnect serial port (USB)
   - Only one connection (USB or Wireless) at a time
   - Shell49:
     - `disconnect`

2. Connect telnet
   - `connect telnet <ip or mDNS> <user> <password>`
   - Same user/password as in telnet instruction
   - E.g.
     - `connect telnet ee49.local micro python`
     - `connect telnet 192.168.0.100 micro python`

Fetch Internet Time

```
# fetch NTP time
from machine import RTC

print("inquire RTC time")
rtc = RTC()
rtc.ntp_sync(server="pool.ntp.org")

timeout = 5
for _ in range(5):
    if rtc.synced():
        break
    time.sleep(1)

if rtc.synced():
    print(time.strftime("%c", time.localtime()))
else:
    print("cannot get NTP time")
```

- Optional
- Requires “global” internet connection

Automatically Connect to Internet

- Copy code to ESP32
  - File `/flash/boot.py` executed
    - On power up
    - After pressing reset button

- Shell49
  - cp <src-file> <dst-file>
    - `cp ee49.py /flash/boot.py`
  - Other useful shell49 commands:
    - `ls /flash`
    - `cat /flash/boot.py`
    - `help ls`
    - `help`

Autonomous Operation

- Power from battery (or USB brick)
- Connect via telnet
  - Operate as usual, e.g.
    - `run hello.py`
- Install app as `main.py`
  - Automatically executed after `boot.py`

Example App: Webserver

```
from machine import Pin
from board import LED
import socket
import time

# html response

html_response = """<!DOCTYPE html>
<html>
  <head><title>ESP32 LED ON/OFF</title></head>
  <body>
    <h2>MicroPython Web Server</h2>
    <form>
      LED:
      <button name="LED" value="ON"  type="submit">LED ON</button>
      <button name="LED" value="OFF" type="submit">LED OFF</button>
    </form>
    <br/>
    {}
  </body>
</html>
"""
```

...

```
led = Pin(LED, Pin.OUT)

# WebServer

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('', 80))
s.listen(3)

while True:
    print("Waiting for connection ...")
    conn, addr = s.accept()
    print("Got a connection from %s" % str(addr))
    request = conn.recv(1024)
    print("Content = %s" % str(request))
    request = str(request)
    if 'GET /?LED=ON' in request:
        print('Turn led on')
        led(1)
    if 'GET /?LED=OFF' in request:
        print('Turn lef off')
        led(0)
    t = time.strftime("%c", time.localtime())
    conn.send(html_response.format(t))
    conn.close()
```

Summary

- WiFi connection
- Autonomous operation
  - `boot.py, main.py`
  - Battery power