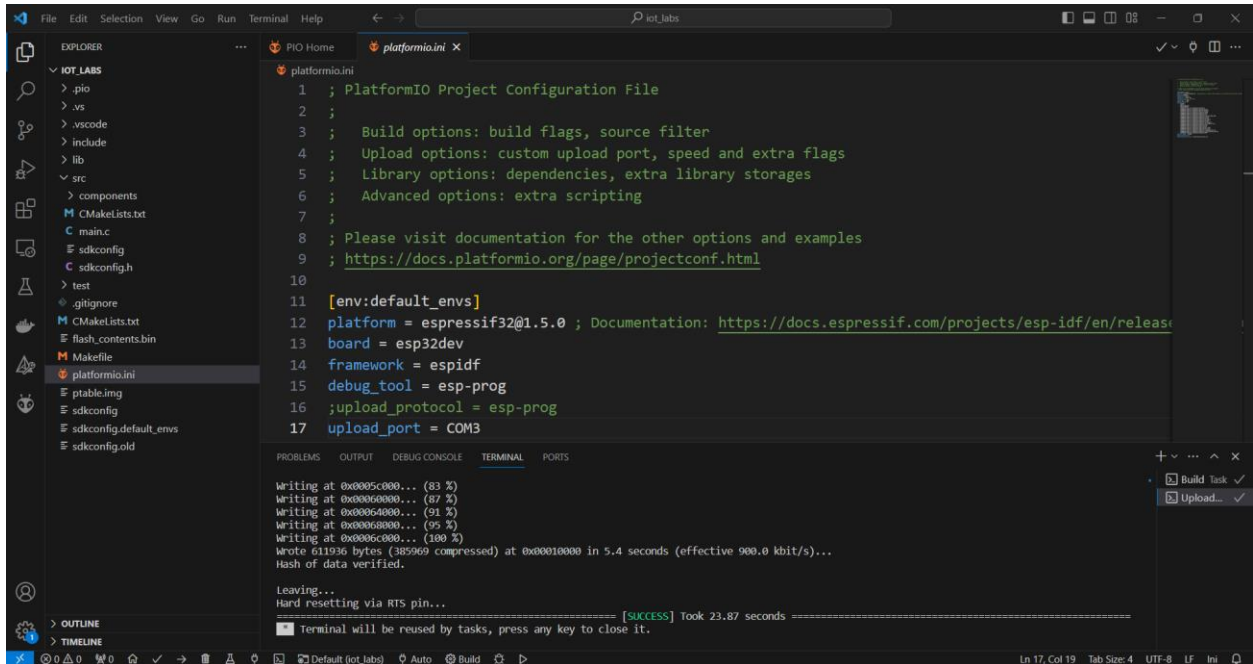


Lab 1: Attacking a UART Console

Uploading the Code (10 Pts):

1. Download the code for Lab 1 from Canvas under files.
2. Extract the code and open it through PlatformIO.
3. Flash the board with the new code.



Determining Connection Parameters (30 Pts):

4. Make sure the serial monitor is closed. Every time you run the code you must hit the enable(EN) button on the ESP32 board for each attempt to connect.
5. Try to open a serial console connected to the board, does the serial port look properly configured? Why or why not?
6. What is the proper baud rate and parity of the port? (Hint: use the script we made in class)

Q4: The serial port can only be used by one application at a time. Ensure that before successfully communicating with the ESP32 from a script. If the serial monitor is not closed:

```
PS D:\School\uf\IoT\Lab1> python .\LLab1_Serial_Connection.py
9600 N
Traceback (most recent call last):
  File "D:\School\uf\IoT\Lab1\LLab1_Serial_Connection.py", line 17, in <module>
    serial_conn = serial.Serial('COM3', baudrate=baudrate_for_loop, parity=parity_for_loop, timeout=0.1)
    ~~~~~^~~~~~
  File "C:\Users\40761\AppData\Roaming\Python\Python311\site-packages\serial\serialwin32.py", line 33, in __init__
    super(Serial, self).__init__(*args, **kwargs)
  File "C:\Users\40761\AppData\Roaming\Python\Python311\site-packages\serial\serialutil.py", line 244, in __init__
    self.open()
  File "C:\Users\40761\AppData\Roaming\Python\Python311\site-packages\serial\serialwin32.py", line 64, in open
    raise SerialException("could not open port {}: {}".format(self.portstr, ctypes.WinError()))
serial.serialutil.SerialException: could not open port 'COM3': PermissionError(13, '拒绝访问。', None, 5)
```

Q6:What is the proper baud rate and parity of the port? 19200 parity even,odd

```

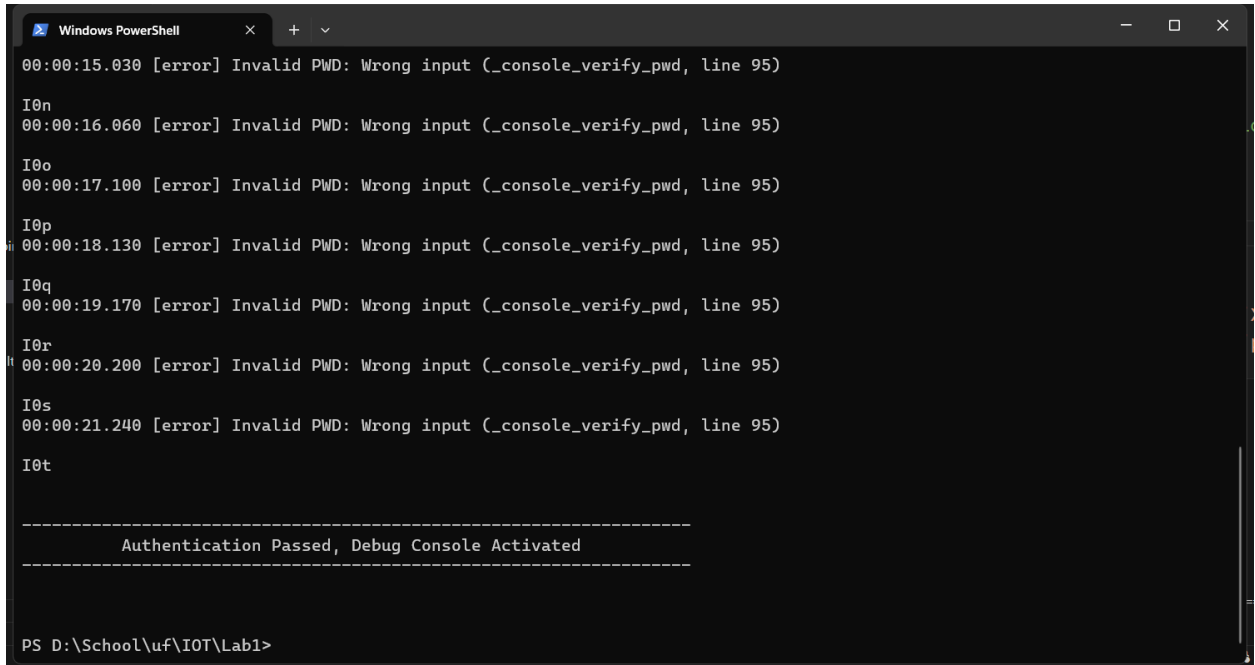
19200 o
b' \x1c<5\x01\x001\x85\x12\xe0\x00\x00\x00\x00\x11=M\x00\xa7\x00\x00\x00\x11f\xf0\x01\x05\x81\x01\x11xX\x00v\x00\x80 "\xf05\x00\x19\x00!\x03 \x81'\x011\x00xc1\x02 \xa1\x1f\x85\x92\x91\x04\x0c\x02\x19\x95F\x8c\x80\xb0\x11,\x81&0\xfc0\x80\xb0a\xcb\x98'6!\"n'
b' \x02\x12\x18f&\x00a!\x02\x02\x12\x812' \x02m\x84\x08$H\x822a\xc16!\x08\x04#&\x802bf\xc0'
b' \x10\x80\x00\x81\xcb'\xc36!\x001\x80\x0012' \x00xb1\x08\x11\x04kb\x02M\x006Ac\x01P\xc80\x11x\x190J\x890\x10\x05\x11xad\" \x00$f'n'
b' \xae8R\x01\x81\x04P202y\x89\x00a'; \x00\x80\x11\xa0e\x08\x02X\x04Pr\x04' X\x84\x00\x05\x04\xc0\xb1\x84\xa0%\x10\x06\x10\xb0\x11n'
b' \x02\x1a\x14#\xe0\x08R\x11P\x04\x08\x01Vh\xaa\xbb66\x01\xbb66m\b6\x04\x06\b6\xdam\x06\b6\x4a0\xa66\x92m\xd0\x91ImmmF\b6\xdaI\t4hm\xbb6\xdb5\x0a\x06\x0b\bh\x04\xdaM\xa6\b6\x06\x06\xda6666\x12mp\x016\b5\xdaI\x02di\x92l\x00\xdaMf\b6\b6\xda2m6\b6\b6\xdb6\xdb6\xda2\x91\xda1f\b6\b6\x00\x12p \x000!\x18\x84\x03\b6\xac\r'b'
b'
n'
Let's Get Started ~~~~~~\r\r\n"
b'\r\r\n'
b'\r\r\n'
b'* Enter the password to enable debug console.\r\r\n'
b'00:00:00.110 [info] ESP_WIFI_MODE_STA\r\r\n'
b'I (357) wifi: wifi driver task: 3ffd119c, prio:23, stack:3584, core=0\r\r\n'
b'I (397) wifi: wifi firmware version: b2c9a19\r\r\n'
b'I (417) wifi: config NVS flash: enabled\r\r\n'
b'I (447) wifi: config nano formatting: disabled\r\r\n'
b'I (487) wifi: Init dynamic tx buffer num: 32\r\r\n'
b'I (497) wifi: Init data frame dynamic rx buffer num: 32\r\r\n'
b'I (527) wifi: Init management frame dynamic rx buffer num: 32\r\r\n'
b'I (567) wifi: Init static rx buffer size: 1600\r\r\n'
b'I (597) wifi: Init static rx buffer num: 10\r\r\n'
b'I (617) wifi: Init dynamic rx buffer num: 32\r\r\n'
b'\xb1b[0.32mi (717) phy: phy version: 4000, b6198fa, Sep 3 2018, 15:11:06, 0, 0\x1b[m\r\r\n'
b'I (717) wifi: mode : sta (30:a:e:a4:97:fa:a8)\r\r\n'
b'00:00:00.480 [info] wifi init_sta finished.\r\r\n'
b'00:00:00.510 [info] connect to ap SSID:DummysSSID\r\r\n'
b'00:00:00.540 [debug] Get MAC address: 30:a:e:a4:97:fa:a8\r\r\n'
b'00:00:00.570 [debug] Client http init start\r\r\n'
b'00:00:00.600 [debug] Client http thread start00:00:00.600 [debug] Client State: IDLE\r\r\n'
b'\r\r\n'
b'00:00:00.650 [debug] Client udp get ctrl msg=0, argc=0\r\r\n'
b'00:00:00.680 [debug] Client State: INIT\r\r\n'
b'00:00:00.700 [debug] opening\r\r\n'
b'00:00:00.720 [info] Client UDP activated.\r\r\n'
b'00:00:00.600 [debug] set msg type=0, arg_len=0\r\r\n'

```

7. What is the password to the evaluated access level? To simplify and shorten this attack, you can assume you know the following: It is 3 characters. The first character is an uppercase letter (A-Z), the second the numbers (0-9), and the third a lowercase letter (a-z). (Hint: modify the script we made in class)

8. Using elevated access, recover the sensitive Wi-Fi information from the device and list it here.

Q7: What is the password to the evaluated access level? IoT



```
Windows PowerShell
00:00:15.030 [error] Invalid PWD: Wrong input (_console_verify_pwd, line 95)
IoTn
00:00:16.060 [error] Invalid PWD: Wrong input (_console_verify_pwd, line 95)
IoTn
00:00:17.100 [error] Invalid PWD: Wrong input (_console_verify_pwd, line 95)
IoTn
00:00:18.130 [error] Invalid PWD: Wrong input (_console_verify_pwd, line 95)
IoTn
00:00:19.170 [error] Invalid PWD: Wrong input (_console_verify_pwd, line 95)
IoTn
00:00:20.200 [error] Invalid PWD: Wrong input (_console_verify_pwd, line 95)
IoTn
00:00:21.240 [error] Invalid PWD: Wrong input (_console_verify_pwd, line 95)
IoTn

-----
Authentication Passed, Debug Console Activated
-----

PS D:\School\uf\IoT\Lab1>
```

Q8: recover the sensitive Wi-Fi information from the device and list it here:

BSSID: 00:00:00:00:00:00

SSID: DummySSID

Channel: 0

RSSI (Received Signal Strength Indicator): 0

Password: DummyPass

```

b"----- Let's Get Started -----\r\r\n"
b'\r\r\n'
b'\r\r\n'
b'* Enter the password to enable debug console.\r\r\n'
b'00:00:00.120 [info] ESP_WIFI_MODE_STA\r\r\n'
b'I (357) wifi: wifi driver task: 3fffd1118, prio:23, stack:3584, core=0\r\r\n'
b'I (397) wifi: wifi firmware version: b2c9a19\r\r\n'
b'I (417) wifi: config NVS flash: enabled\r\r\n'
b'I (447) wifi: config nano formatting: disabled\r\r\n'
b'I (487) wifi: Init dynamic tx buffer num: 32\r\r\n'
b'I (497) wifi: Init data frame dynamic rx buffer num: 32\r\r\n'
b'I (527) wifi: Init management frame dynamic rx buffer num: 32\r\r\n'
b'I (567) wifi: Init static rx buffer size: 1600\r\r\n'
b'I (597) wifi: Init static rx buffer num: 10\r\r\n'
b'I (617) wifi: Init dynamic rx buffer num: 32\r\r\n'
b'\x1b[0;32mI (757) phy: phy_version: 4000, b6198fa, Sep  3 2018, 15:11:06, 0, 0\x1b[0m\r\r\n'
b'I (757) wifi: mode : sta (30:ae:a4:97:fa:a8)\r\r\n'
b'00:00:00.530 [info] wifi_init_sta finished.\r\r\n'
b'00:00:00.550 [info] connect to ap SSID:DummySSID\r\r\n'
b'00:00:00.580 [debug] Get MAC address: 30:ae:a4:97:fa:a8\r\r\n'
b'00:00:00.610 [debug] Client http init start\r\r\n'
b'00:00:00.640 [debug] Client http thread start00:00:00.640 \r\r\n'
b'[debug] Client State: IDLE\r\r\n'
b'00:00:00.690 [debug] Client udp get ctrl msg=0, argc=0\r\r\n'
b'00:00:00.720 [debug] Client State: INIT\r\r\n'
b'00:00:00.750 [debug] opening\r\r\n'
b'00:00:00.770 00:00:00.770 [info] Client UDP activated.\r\r\n'
b'[debug] set msg type=0, arg_len=0\r\r\n'

```

```

b'-----\r\r\n'
b'          Authentication Passed, Debug Console Activated\r\r\n'
b'-----\r\r\n'
b'\r\r\n'
b'\r\r\n'
b''
b''
b''
b'wifi_info>>\r\r\n'
b'BSSID: 00:00:00:00:00:00\r\r\n'
b'SSID: \r\r\n'
b'SSID: DummySSID\r\r\n'
b'Channel: 0\r\r\n'
b'RSSI: 0\r\r\n'
b>Password: DummyPass\r\r\n'
b'\r\r\n'
b'\r\r\n'
b''

```

Closing Thoughts (20 Pts):

9. Describe a method to brute force a password of n characters, in which each character can be any valid ASCII character. **You do NOT need to code this in Python.**
10. Describe a method to brute force a password of *unknown* length. **You do NOT need to code this in Python.**
11. Describe one method in which the UART connection could be better secured.

Q9: In order to brute-force crack a password of length n , where each character can be any valid ASCII character, one can first define a character set containing all possible ASCII characters, which usually includes numbers, uppercase letters, lowercase letters, special characters, etc. Use recursive or iterative methods to generate all possible password combinations of length n . This can be accomplished by iterating over each possible character, trying every character in the character set for each position of the password. For each password combination generated, attempts are made to log in or unlock the target system. The response to the attempted login or unlock operation is used to determine if the password is correct. If the attempt fails, it continues to try the next password combination; if it succeeds, the correct password is recorded and further attempts are stopped.

Q10: The method for cracking unknown length passwords is slightly more complex and is the same as for cracking known length passwords, starting with defining a character set containing all possible ASCII characters. Starting with a length of 1, the length of the attempted password is gradually increased. For each given length, all possible combinations of passwords are generated and tried. For each given length, use a nested loop or recursive function to generate all possible password combinations and try them. Each time a password is attempted, check to see if it was successfully unlocked. If successful, the password is recorded and the program is terminated; if unsuccessful, the password length is increased and the attempt continues.

Q11: Encrypted Communication. Both the transmitting and receiving devices need to implement the chosen encryption and decryption algorithms. The data sent over UART is encrypted by the sender and then decrypted by the receiver.