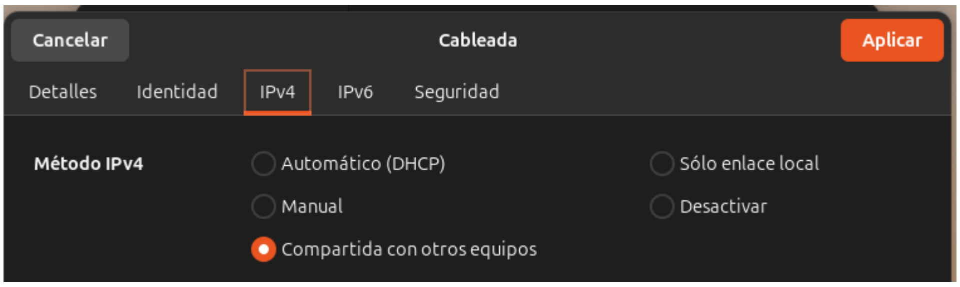
Ivan Rodriguez Canals

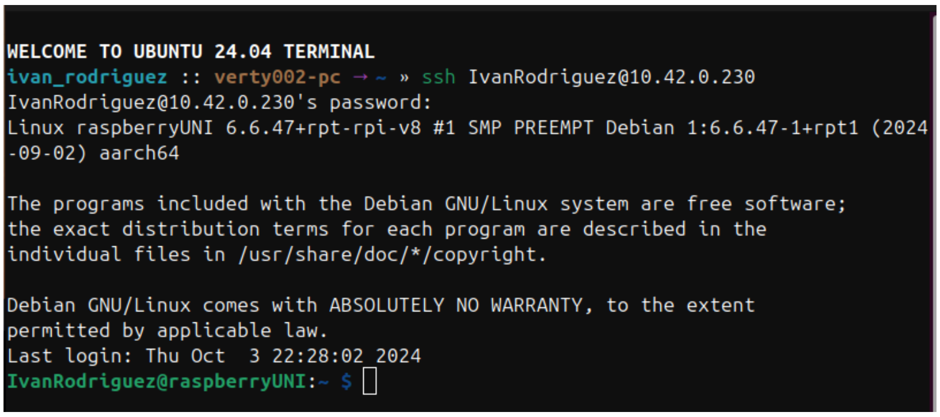
PBE FIRST PUZZLE REPORT

Internet Connections

I had some issues regarding internet connection, at first I tried to create a static IP between the computer and the **Raspberry Pi 5** using an Ethernet cable, after being able to connect with the Raspberry Pi using that method I encountered some problems like wireless internet connection. I was not able to use internet in the Raspberry Pi while being connected to the computer. After sending a Gmail to the professor, he advised me to use a dynamic IP and that solved the problem. I use Ubuntu so that was easy, I just had to activate the connection sharing and the ssh of the Raspberry using an external screen and look for my computer IP and Raspberry Pi IP (10.42.0.230).



Método IPv4: ( ) Automático (DHCP) ( ) Sólo enlace local ( ) Manual ( ) Desactivar [✓ Compartida con otros equipos]



```
WELCOME TO UBUNTU 24.04 TERMINAL
ivan_rodriguez :: verty002-pc → ~ » ssh IvanRodriguez@10.42.0.230
IvanRodriguez@10.42.0.230's password:
Linux raspberryUNI 6.6.47+rpt-rpi-v8 #1 SMP PREEMPT Debian 1:6.6.47-1+rpt1 (2024
-09-02) aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Oct  3 22:28:02 2024
IvanRodriguez@raspberryUNI:~ $
```

After being able to connect to internet, I installed and enabled (using $raspi - config$ and then Interface Options) the VNC app in my Raspberry and computer and configured everything using the Raspberry IP address in order to see the screen of the Raspberry Pi in my computer.

To activate this function, I have to type the command in the terminal: $vncserver - virtual$

Python 3 installation + libraries (in a virtual environment)

To install Python, I used the commands $sudo\ apt\ install\ python3$ and $sudo\ apt\ install\ python3 - pip$ for the libraries. I also had to create a virtual environment in order to install all the libraries so every time I want to use them I have to type the next command $source\ myenv/bin/activate$ if I do not activate this the program won't work.

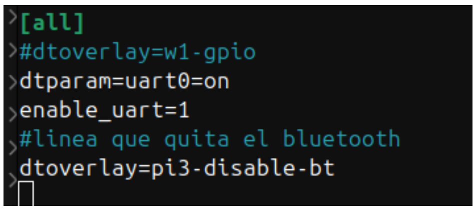
This is the library that I ended up using ⇓, but I also tried **Adafruit_Python_PN532** library.

GitHub NFC PN532 libraries + examples
(https://github.com/gassajor000/pn532pi/tree/master)

Example of the assembly and more Raspberry configurations

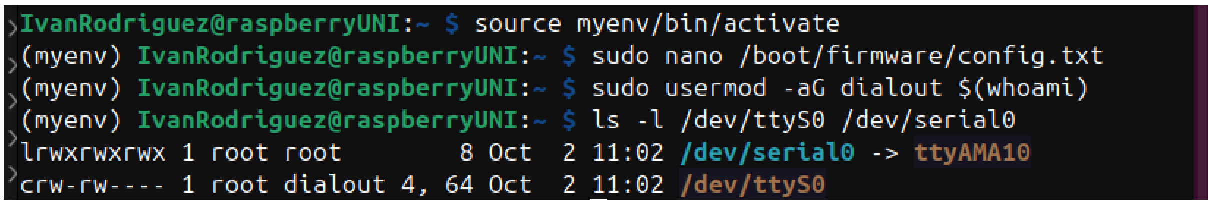
As you can see, I am using I2C instead of HSU. In order to do this, I had to set the switch in the channel 1 to '1' or ON. I did this because my Raspberry Pi was not recognizing UART.

I tried a lot of different things, I checked several times if UART was enabled and the connections between the Raspberry Pi and the NFT PN532 were right, but I still haven't found the problem. The only thing I know is that the problem is not from the libraries because this libraries have HSU support and the teacher Francesc tested the libraries.

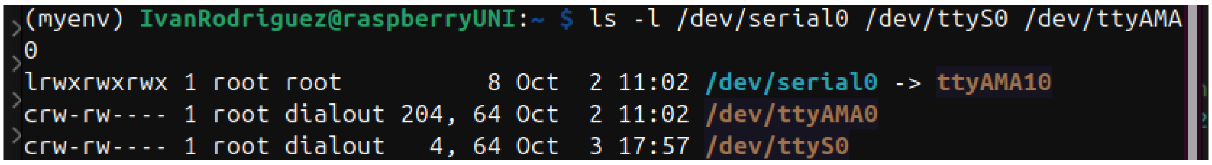


```
[all]
#dtoverlay=w1-gpio
dtparam=uart0=on
enable_uart=1
#linea que quita el bluetooth
dtoverlay=pi3-disable-bt
```

These captures should prove that the UART is active, but I can not find the problem.

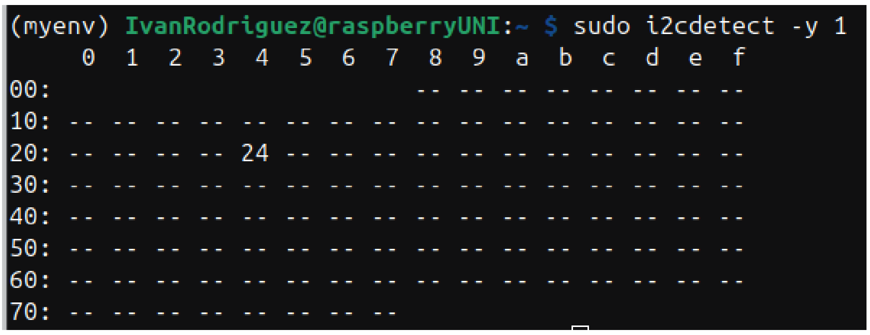


```
IvanRodriguez@raspberryUNI:~ $ source myenv/bin/activate
(myenv) IvanRodriguez@raspberryUNI:~ $ sudo nano /boot/firmware/config.txt
(myenv) IvanRodriguez@raspberryUNI:~ $ sudo usermod -aG dialout $(whoami)
(myenv) IvanRodriguez@raspberryUNI:~ $ ls -l /dev/ttyS0 /dev/serial0
lrwxrwxrwx 1 root root       8 Oct  2 11:02 /dev/serial0 -> ttyAMA10
crw-rw---- 1 root dialout 4, 64 Oct  2 11:02 /dev/ttyS0
```



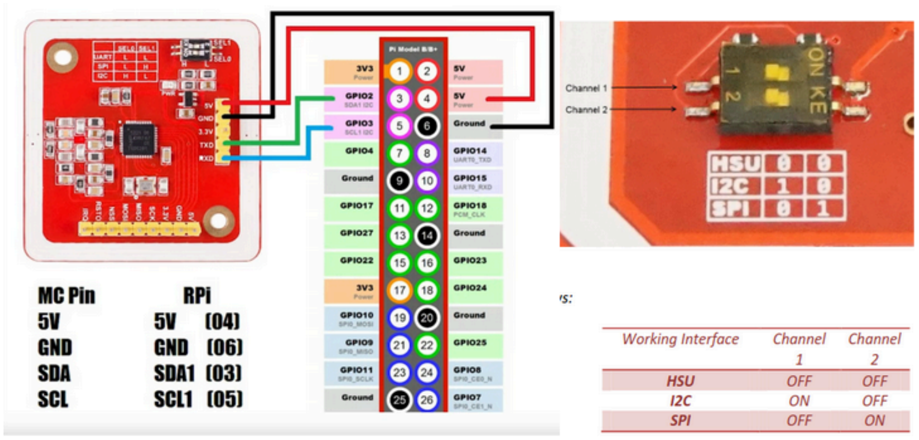
```
(myenv) IvanRodriguez@raspberryUNI:~ $ ls -l /dev/serial0 /dev/ttyS0 /dev/ttyAMA
0
lrwxrwxrwx 1 root root         8 Oct  2 11:02 /dev/serial0 -> ttyAMA10
crw-rw---- 1 root dialout 204, 64 Oct  2 11:02 /dev/ttyAMA0
crw-rw---- 1 root dialout   4, 64 Oct  3 17:57 /dev/ttyS0
```

Instead of UART, I enabled I2C with the command sudo $raspi - config$ ⇒ Interface options and enable. Then, I checked that the Raspberry Pi was recognizing the NFC PN532 using the command $sudo\ i2cdetect\ - y\ 1$.



```
(myenv) IvanRodriguez@raspberryUNI:~ $ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                         -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- 24 -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

This is a scheme of the configuration of the pins using I2C:

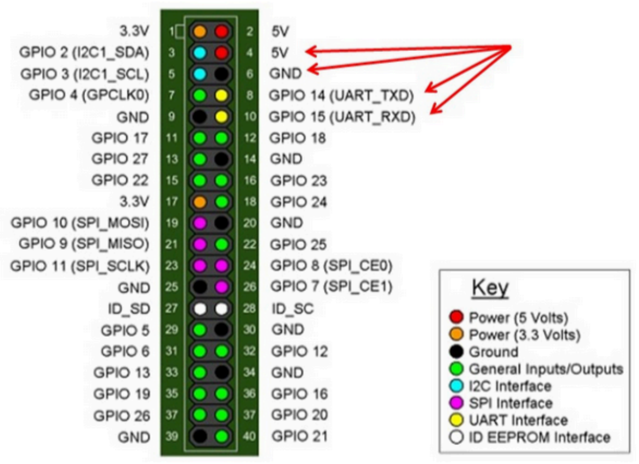


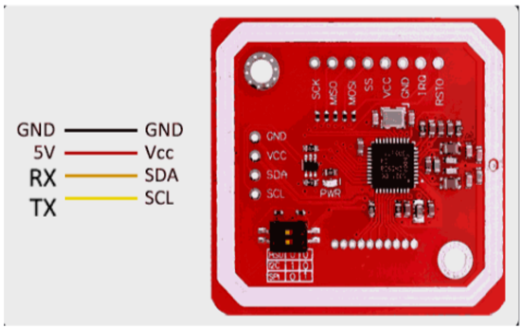
MC Pin	RPi
5V	5V (04)
GND	GND (06)
SDA	SDA1 (03)
SCL	SCL1 (05)

	HSU	I2C	SPI
	0 0	1 0	0 1

Working Interface	Channel 1	Channel 2
HSU	OFF	OFF
I2C	ON	OFF
SPI	OFF	ON

This is a scheme of the configuration of the pins using UART:





After checking that everything was okay, I started programming the puzzle using the next structure:

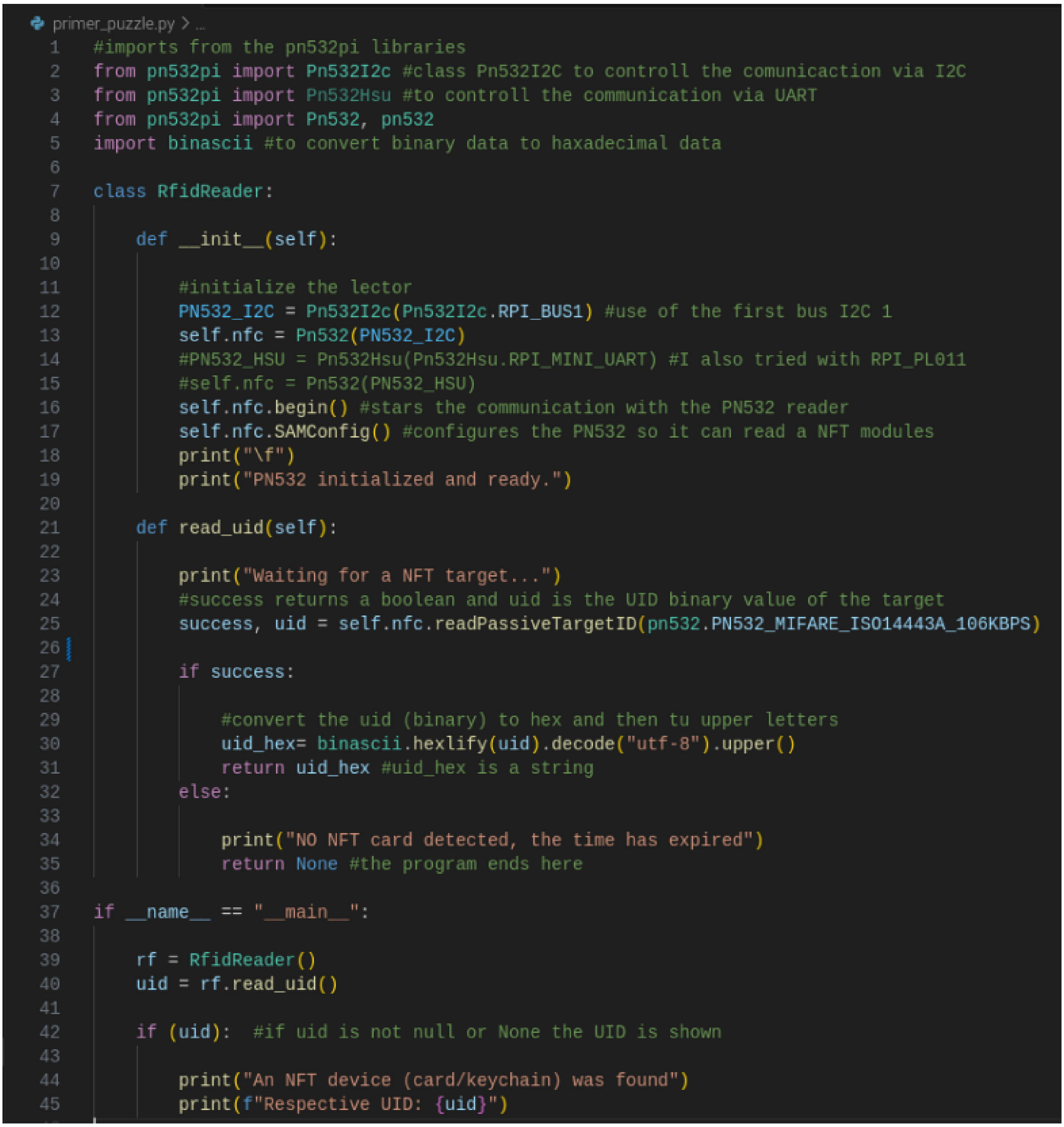
Listing 1: python

```python
class Rfid...:
    ...
    # return uid in hexa str
    def read_uid(self):
        ...

if __name__ == "__main__":
    rf = Rfid...()
    uid = rf.read_uid()
    print(uid)
```

In order to program the imports, initialization and some lines to extract the binary information from the target and convert it to hexadecimal I used the GitHub examples mainly, apart from other internet tools and forums (NFT PN532 EXAMPLES)

Ivan Rodriguez Canals

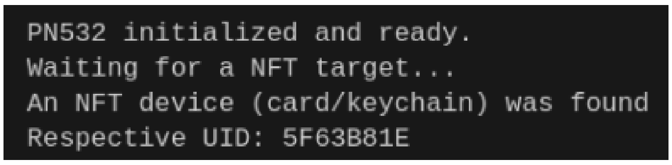
CODE (with both initializations I2C and #UART) editable version on GitHub



primer_puzzle.py > ...

```python
#imports from the pn532pi libraries
from pn532pi import Pn532I2c #class Pn532I2C to controll the comunicaction via I2C
from pn532pi import Pn532Hsu #to controll the communication via UART
from pn532pi import Pn532, pn532
import binascii #to convert binary data to haxadecimal data

class RfidReader:

    def __init__(self):

        #initialize the lector
        PN532_I2C = Pn532I2c(Pn532I2c.RPI_BUS1) #use of the first bus I2C 1
        self.nfc = Pn532(PN532_I2C)
        #PN532_HSU = Pn532Hsu(Pn532Hsu.RPI_MINI_UART) #I also tried with RPI_PL011
        #self.nfc = Pn532(PN532_HSU)
        self.nfc.begin() #stars the communication with the PN532 reader
        self.nfc.SAMConfig() #configures the PN532 so it can read a NFT modules
        print("\f")
        print("PN532 initialized and ready.")

    def read_uid(self):

        print("Waiting for a NFT target...")
        #success returns a boolean and uid is the UID binary value of the target
        success, uid = self.nfc.readPassiveTargetID(pn532.PN532_MIFARE_ISO14443A_106KBPS)

        if success:

            #convert the uid (binary) to hex and then tu upper letters
            uid_hex= binascii.hexlify(uid).decode("utf-8").upper()
            return uid_hex #uid_hex is a string
        else:

            print("NO NFT card detected, the time has expired")
            return None #the program ends here

if __name__ == "__main__":

    rf = RfidReader()
    uid = rf.read_uid()

    if (uid):  #if uid is not null or None the UID is shown

        print("An NFT device (card/keychain) was found")
        print(f"Respective UID: {uid}")
```

Output example if the card is found:



```
PN532 initialized and ready.
Waiting for a NFT target...
An NFT device (card/keychain) was found
Respective UID: 5F63B81E
```