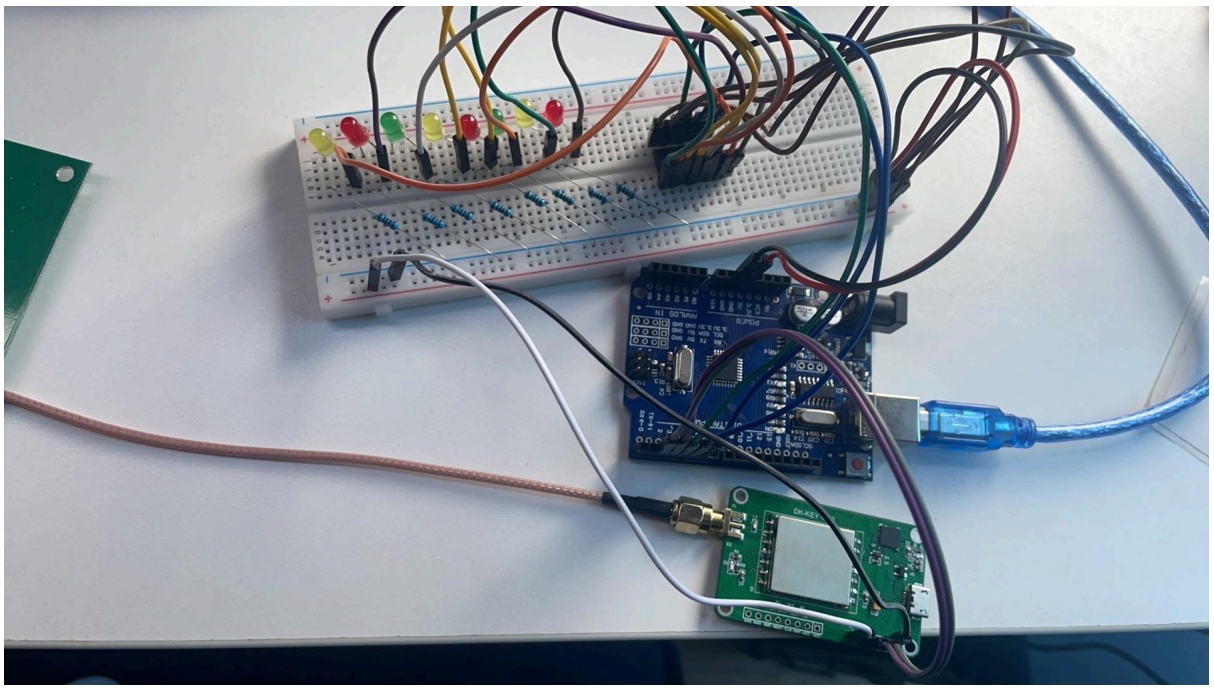
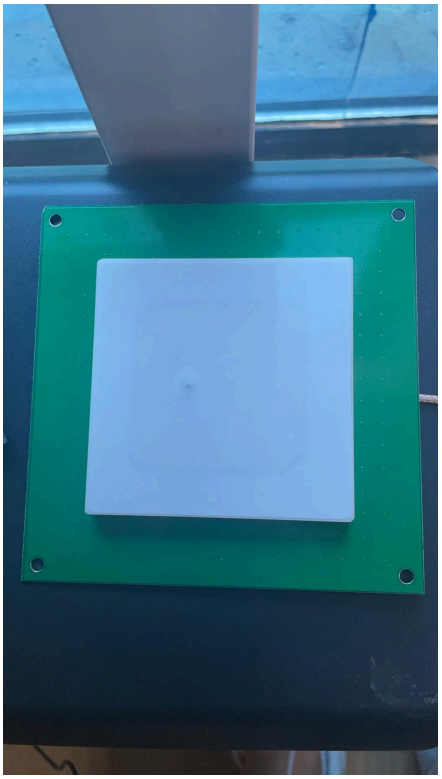


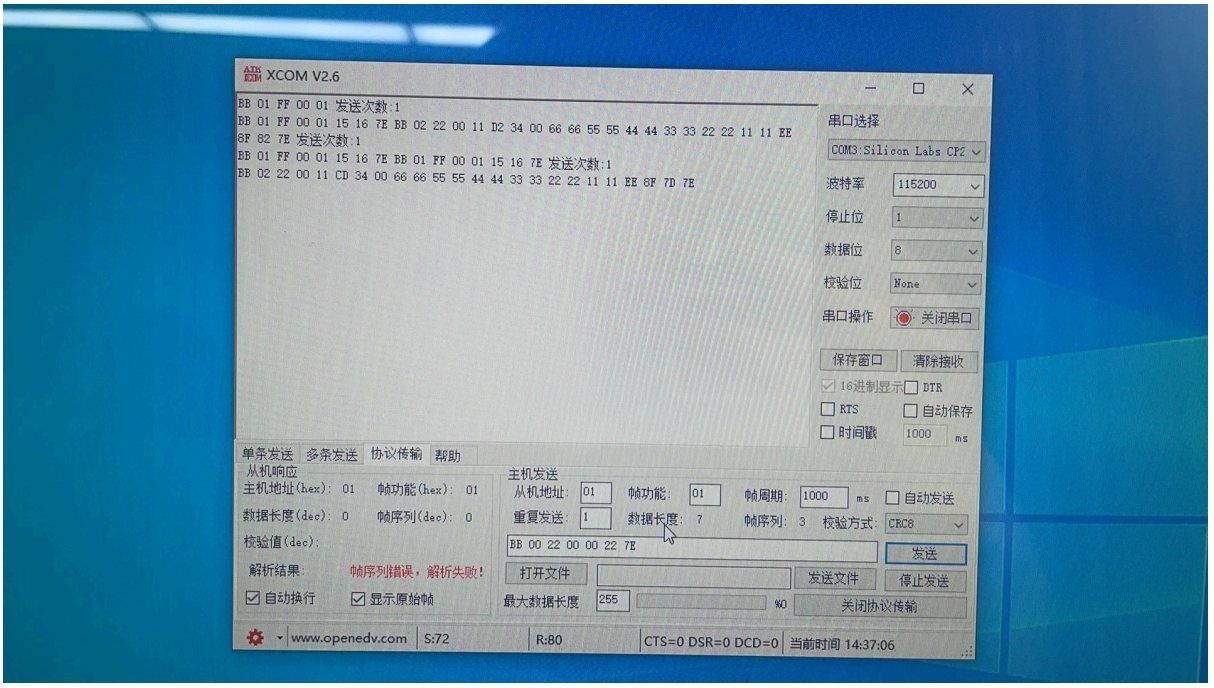
RFID-R200 PR2026 Windows debug development log：

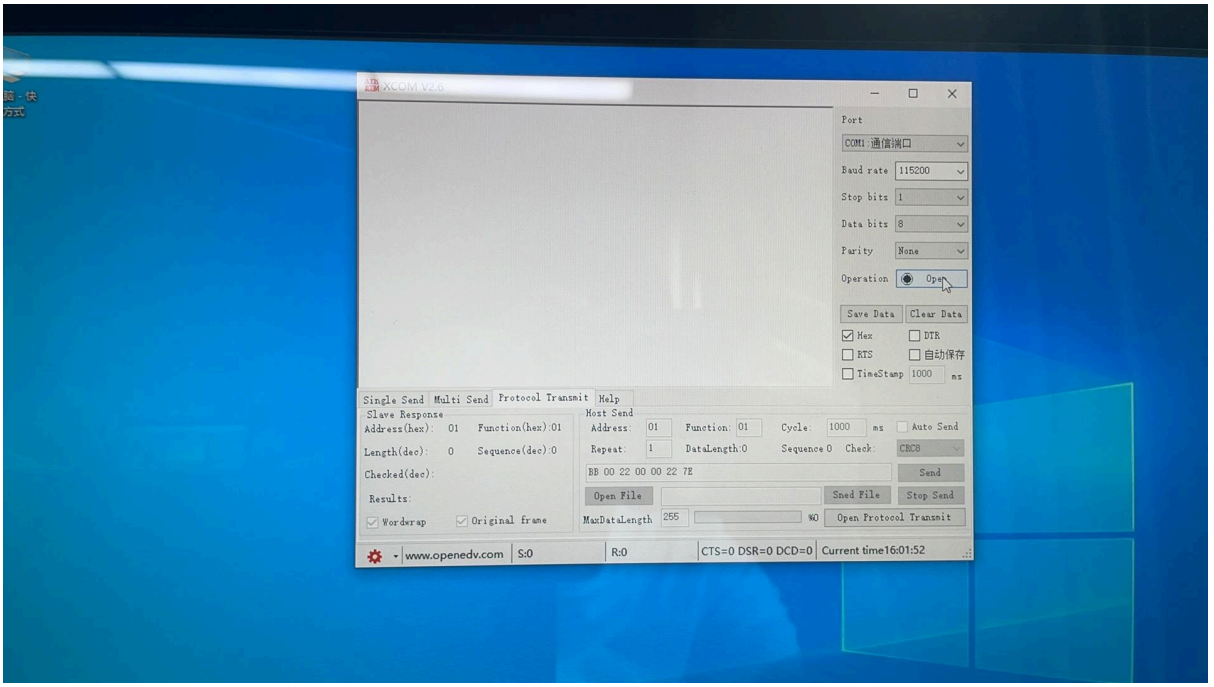
1. Install CP210x USB to Serial Driver on Windows to identify the plug in RFID module. RFID module has to be used with the coil extender, otherwise RFID can not read the Radio frequency information sent by the Ultra High Frequency cards or electronic stickers. The above diagram did not display the coil extender, it appears like the picture below



2. Download ATKXCOM serial communicator app. Open the app then select the corresponding CP210x Serial COM port, 115200 Baud rate, stopping index 1, data index 8, parity none. Then choose the protocol transmission, open protocol transmit ,

then enter repeat send 5 if using the module for the first time. Open serial port communication on the right side by clicking operation. Now there should be a yellow light emitting on the module indicating that the port has been established successfully with the Windows PC. The picture below is a module with modified sending&receiving efficiency setting, this will be covered later in the log.





3. The app has a buggy line changing system but does not affect visualizing the response reading from the module
    a. BB 00 22 00 00 22 7E is a single read command, the module attempts to read the UHF card info from the surrounding
    b. BB 01 FF 00 01 15 16 7E is a response representing fail to read
    c. BB 02 22 00 11 CD 34 00 66 66 55 55 44 44 33 33 22 22 11 11 EE 8F 7D 7E is a response representing reading success

The above command and responses are all expressed in Hex digit code
BB is Header
00 is the type of command：0x00 is a command from the PC to the PR2026 board
0x01 is a response type from PR2026 to the PC
0x02 is a notification type from PR2026 to the PC
22 is command
00 11 or 0x11 is command length PL
CD is RSSI/singal strength（0-255）
34 00 or 0x3400 is PC
66 66 55 55 44 44 33 33 22 22 11 11 is the card EPC Number in 12 byte, mutable
EE 8F or 0xEE8F is the CRC number of the individual card
7D is checksum, the sum of bytes from the second byte to the byte before the checksum, then take the last byte of the sum, which is, in other word, the last 8 bits of the sum.
7E is the message ending indicator

4. The module has built-in multi read function but we can also loop the single read and turn it into a multi read function
5. BB 00 B6 00 02 07 D0 8F 7E is the command for setting maximum receiving range
   BB is Header
   00 is the type of command：0x00 is a command from the PC to the PR2026 board
   0x01 is a response type from PR2026 to the PC
   0x02 is a notification type from PR2026 to the PC

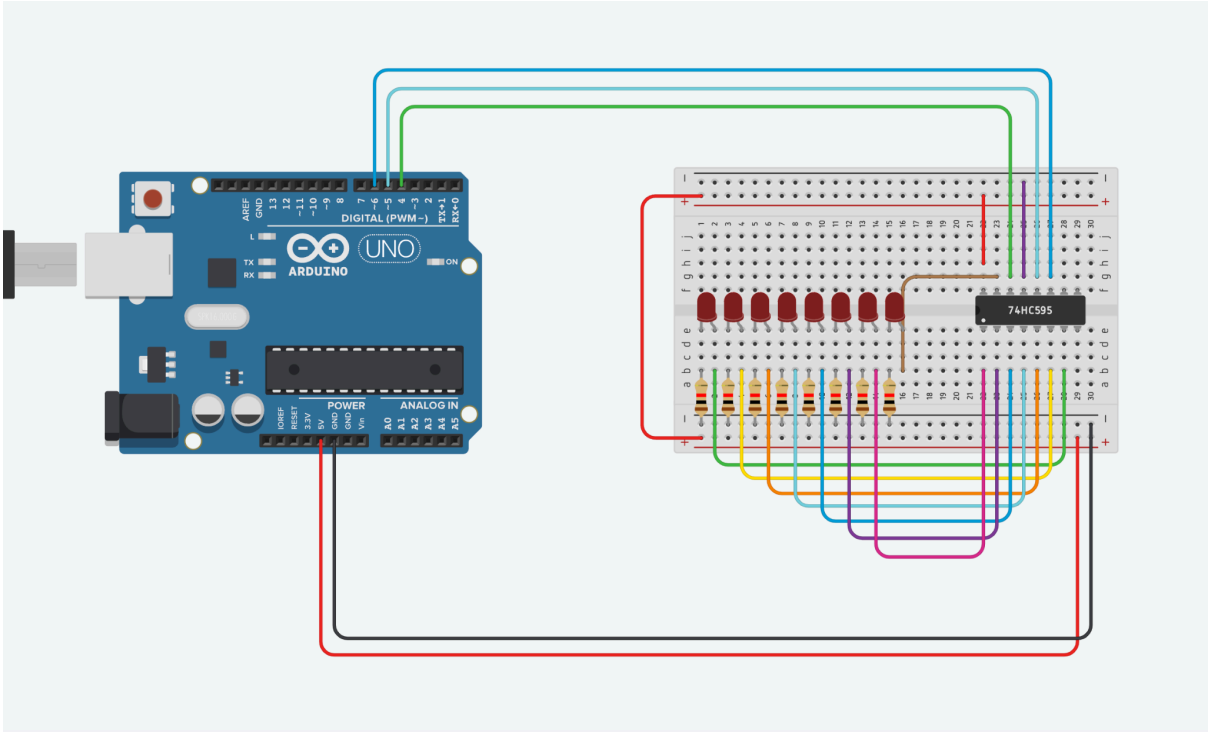
   B6 is command
   00 02 or 0x0002 is command length PL
   07 D0 or 0x07D0 is the efficiency coefficient, in decimal which is = 2000, means 20dBm
   8F is checksum, the sum of bytes from the second byte to the byte before the checksum, then take the last byte of the sum, which is, in other word, the last 8 bits of the sum.

   7E is the message ending indicator
   a. if received response like BB 01 B6 00 01 00 B8 7E, then it means set up successful
6. Other command prompt and the XCOM app are all in the github repository.
7. https://github.com/JasonTheWanger/RFID

RFID-R200 PR2026 Arduino development log：

1. Wiring diagram

Due to the website does not contain the RFID R200 PR2026 module; therfore, it will not be displayed below.



This picture illustrate the wiring diagram for the 73HC595. The VCC and GND pins of the RFID need to be connect to the corresponding VCC and GND port on the Arduino. The TX port needs to be connect to Arduino digital 2, and RX port needs to be Arduino digital 3.

2. Goal：Identify the EPC number of the UHF card. If detected an EPC number in the list, then lit up the corresponding LED.
3. Since the demo file only contains the #C source code, this development can add C/C++ Arduino code to the original file directory
4. Arduino Code：RFID_R200_EPCIdentifier
    a. Current Problem: Everytime sending a single read command to the RFID module, there's a chance that RFID returns half or multiple card read response if there are more than 2 cards present nearby. Currently the code algorithm already filtered most of the noises, and the identifier algorithm will still work even if there are noises at the end of the response. However, there are still some unhandle noises. The main problem is that due the difference range of UHF transmission, the response from the RFID module will have the strongest UHF signal handled more often. According to the current algorithm, eliminating the other response noise followed by the first response, will cause some UHF corresponding LED blinking less often than the other even when they are close to each other. The good news is that the Arduino does not use a specific sorting algorithm when outputting multiple response in one, that is, using our algorithm to cut off the other noise response in a multi-singleline response, the LED blinking frequency will not be affected by our algorithm. Therefore, the frequency of LED blinking is primarily based on the strength of the received signal. For example, the electronic sticker frequency is slightly lower than the PVC white card frequency, so the corresponding LED for the electronic sticker does not blink as frequent as the other LEDs.

b. Future possible modification: We can setup a constant frequency bar. If the detect card appears same or more frequent than the bar, then the LED can be set to be always on instead of blinking. Currently, the LED will turn on and shut off in 50ms on every iteration of the controlled loop, that why it is blinking. If we made it to always on when above certain frequency, we can avoid the previous reading response problem in some extent. If the detection frequency is lower than the bar frequency, change it blinking mode. In that case we need to use changeLED(int, int) instead of updateLEDs, to control the state of a single LED. Currently, electronic sticker and PVC cards are all written under a same type of struct. Electronic sticker has a USR customized information section, able to store 64 bytes of data; however, it is not tested yet. Later on we can add one or two more member in the struct to identify whether if the this UHF card has USR sections or not.

Below is the tested PVC card and electronic sticker's EPC number. Corresponding LEDs from left to right are: pvc1 (yellow), pvc2 (red), electronic sticker (green).

EPC Cards	EPC number
PVC white card #1:	E2 00 47 01 72 A0 60 21 A2 0F 01 12
PVC white card #2:	E2 00 47 18 CF 20 64 26 47 9F 01 0B
Electronic sticker #1:	66 66 55 55 44 44 33 33 22 22 11 11

c. Future update: Adding more 74HC595 to display more UHF card, or adding a LCD to display the captured information from the detected cards. If wanting to add more identifiable UHF cards' EPC number then go to Cards_EPC_List.h and add new card struct and assign the new card struct with the corresponding cards' 12 byte EPC number. Utilize the XCOM serial communicator app to check the EPC and enter it in the hear file by hand. Then, add the new card struct to the cardList struct array for the algorithm to be able to identify the EPC number. Currently this code can only use to detect EPC number, use the RFID_R200_SingleRapidRead code to rapidly send other command. New command can be added in the RFID_R200.h header file. Start by creating a new command struct. Each struct need an 8 bits unsigned char array for store hex code command, an empty array, and the size of by the unsigned char array. Remember to create an pointer to the struct just created. Then enter SerialSendCmd in the loop, and put struct pointer -> member for every member in the command struct (usually 3) as the argument of the SerialSendCmd function.
d. All the C/C++ .ino and .h file has comment on almost every line of code. Can be used with the instruction together.
e. This code sketch implemented the Arduino built-in SoftwareSerial library.