Python加CH9329模块实现云顶之弈自动刷局数

原创

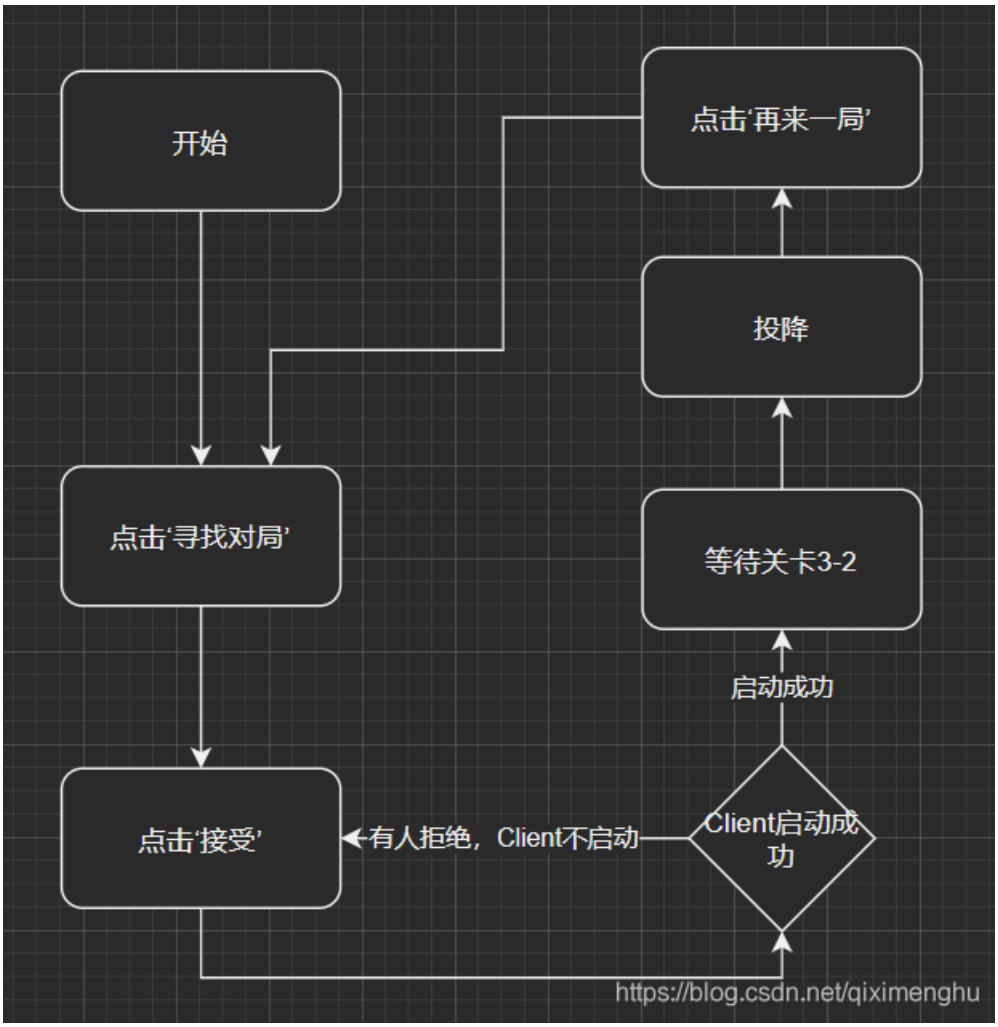
七夕猛虎 于 2021-03-07 21:34:44 发布 2337 收藏 26
分类专栏： python工具 文章标签： python 游戏

前言

2.0版本脚本，不需要硬件模块。

现在的LOL活动是又氪又肝，各种代币宝典之类的太费肝了。所以我就想搞个云顶自动刷局数的脚本。

流程大概如下：



一开始我打算直接用pyautogui这个模块控制鼠标和键盘实现这个功能:

Welcome to PyAutoGUI's documentation!

PyAutoGUI lets your Python scripts control the mouse and keyboard to automate interactions witl other applications. The API is designed to be as simple. PyAutoGUI works on Windows, macOS, ar Linux, and runs on Python 2 and 3.

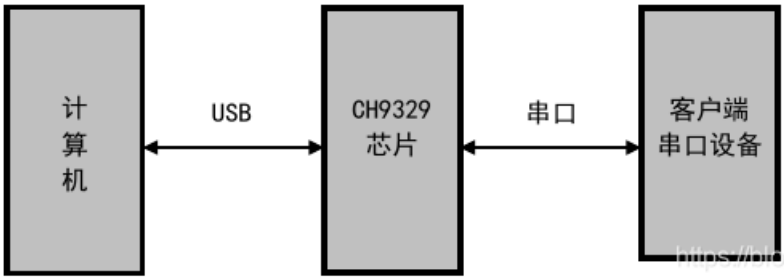
但是LOL在这方面好像有防备，在客户端上面可以控制鼠标移动，但是控制鼠标点击的时候，它没反应。而且到了游戏内时候，直接无法用pyautogui模块控制鼠标移动，也无法控制鼠标点击。这也就是为什么要用CH9329这个硬件模块了（某宝加邮费总共23块不到，外加一个CH340 USB转TTL模块，如果自己没有，买一个大概7块钱，买的时候会送杜邦线）。我觉得LOL总不能检测到我用了个假鼠标和键盘吧。

串口转 HID 键盘鼠标芯片 CH9329

中文手册
版本：V1.1
http://wch.cn

1、概述

CH9329 是一款串口转标准 USB HID 设备(键盘、鼠标、自定义 HID)芯片，根据不同的工作模式，在电脑上可被识别为标准的 USB 键盘设备、USB 鼠标设备或自定义 HID 类设备。该芯片接收客户端发送过来的串口数据，并按照 HID 类设备规范，将数据先进行打包再通过 USB 口上传给计算机。通过提供的上位机软件，用户也可自行配置芯片工作模式、串口通信模式、串口通信波特率、多种超时时间、VID、PID，以及各种 USB 字符串描述符。下图为其一般应用框图。



虽然，pyautogui的一部分功能不能使用了，但是它还有一些比较有用的功能，比如在屏幕上定位一幅图片的位置(pyautogui.locateOnScreen)，比如我们可以将上面流程图里面的按钮都截图，然后使用pyautogui去定位这些按钮的位置，然后通过CH9329模块进行硬件鼠标左键单击。pyautogui还可以通过界面的标题来获取到windows下面的hwnd信息(pyautogui.getWindowsWithTitle)，这样我们就可以用来检测游戏是否启动或者结束了。

... you can call the `locateOnScreen('calc7key.png')` function to get the screen coordinates. The retur value is a 4-integer tuple: (left, top, width, height). This tuple can be passed to `center()` to get the and Y coordinates at the center of this region. If the image can't be found on the screen, `locateOnScreen()` raises `ImageNotFoundException`.

CH9329模块是通过串口接收指令，然后自己上报鼠标或者键盘事件。它支持鼠标的相对移动，绝对移动，键盘按键。

以下是这个脚本所需要的Python库环境：

Python 3.7

pyautogui,PIL,serial,opencv-python

注意：虽然没有直接调用opencv的功能，但是pyautogui模块在模糊定位图片时候，需要用到opencv库。

控制CH9239模块

CH9239芯片通信协议的资料：

链接：某度网盘 提取码：rarl

首先根据CH9239的通信协议，实现控制鼠标移动点击函数，实现键盘按键功能，把pyautogui模块不可用的功能用我们的硬件模块代替。

2.2.5、CMD_SEND_MS_REL_DATA

通过该命令向芯片发送相对鼠标数据包,模拟相对鼠标相关动作(包括左中右键按下与释放、滚轮上下滚动、上下左右移动)。

外围串口设备→芯片：

帧头	地址码	命令码	后续数据长度	后续数据	累加和
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57、0xAB	0x00	0x05	5	5 个字节数据	0x??

该命令带 5 个字节后续数据，后续数据为 USB 相对鼠标的数据包，依次为：

(1)、第 1 个字节：必须为 0x01；

(2)、第 2 个字节：1 个字节的鼠标按键值，最低 3 位每位表示 1 个按键，具体如下：

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
0	0	0	0	0	中键	右键	左键

BIT2---BIT0：为 1 表示该键按下，为 0 表示该键释放或未按下。

(3)、第 3 个字节：1 个字节 X 方向(纵坐标，上下方向)移动距离；

A、不动：字节 3 = 0x00，则表示在 X 轴方向不移动；

B、向右移动：0x01 <= 字节 3 <= 0x7F； 移动像素点 = 字节 3；

C、向左移动：0x80 <= 字节 3 <= 0xFF； 移动像素点 = 0x00 - 字节 3；

(4)、第 4 个字节：1 个字节 Y 方向(纵坐标，上下方向)移动距离；

A、不动：字节 4 = 0x00，则表示在 Y 轴方向不移动；

B、向右移动：0x01 <= 字节 4 <= 0x7F； 移动像素点 = 字节 4；

C、向左移动：0x80 <= 字节 4 <= 0xFF； 移动像素点 = 0x00 - 字节 4；

(5)、第 5 个字节： 1 个字节滚轮滚动齿数，

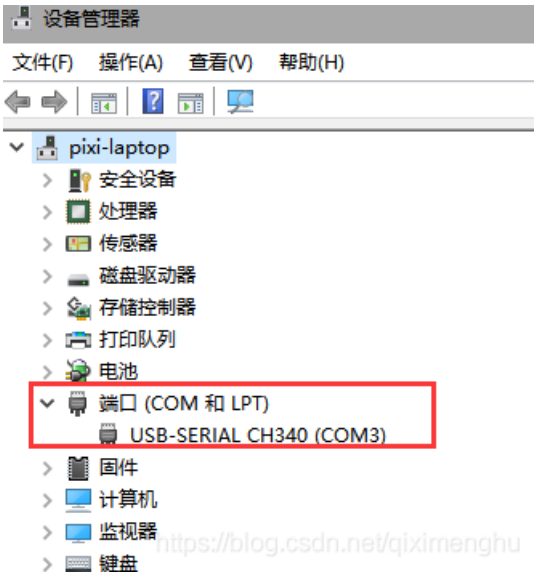
0x01---0x7F，表示屏幕向上滚动，单位：齿数；

0x81---0xFF，表示屏幕向下滚动，单位：齿数；

鼠标绝对移动并单击按键（模仿pyautogui模块的API写的，后来发现用不到多次点击的功能）：

```
mserial = serial.Serial("COM3", 115200, timeout=1)
def hard_click(x, y, clicks=1, interval=0.0, button=pg.LEFT, duration=0.0):
    global mserial
    nx = int(x * 4096 / 1920)
    ny = int(y * 4096 / 1080)
    # 鼠标绝对坐标命令
    cmd = [0x57, 0xAB, 0x00, 0x04, 0x07, 0x02]
    button_val = 1 << get_button_val(button)
    low_x = nx & 0xFF
    high_x = (nx >> 8) & 0xFF
    low_y = ny & 0xFF
    high_y = (ny >> 8) & 0xFF
    scroll = 0x00
    data = [button_val, low_x, high_x, low_y, high_y, scroll]
    sum_val = (sum(cmd) + sum(data)) & 0xFF
    data.append(sum_val)
    # 按下
    press = cmd + data
    # 释放
    release = press.copy()
    release[6] = 0x0
    sum_val = sum(release[:-1]) & 0xFF
    release[len(release) - 1] = sum_val

    while clicks > 0:
        # 移动并按下键
        mserial.write(bytes(press))
        # 延时50ms
        time.sleep(50 / 1000)
        # mserial.readall()
        # 释放键
        mserial.write(bytes(release))
        time.sleep(50 / 1000)
        # mserial.readall()
        clicks -= 1
        if clicks > 0:
            time.sleep(interval)
    # 忽略CH9239回复的消息
    mserial.flushInput()

    return True
```

注意：其中串口的名称根据设备管理器里面的COM口实际名称来定。一般插上CH340 USB转TTL模块，设备管理器中就能看到。如果没有这一项的话，需要装一个CH340的驱动，这个驱动很常用，很容易找到安装的教程，此处不做赘述。波特率设置可以使用CH9239模块资料中的设置工具设置你想要的波特率。



鼠标绝对移动，不点击：

```
def hard_moveTo(x, y):
    global mserial
    nx = int(x * 4096 / 1920)
    ny = int(y * 4096 / 1080)

    # 鼠标绝对坐标命令
    cmd = [0x57, 0xAB, 0x00, 0x04, 0x07, 0x02]
    button_val = 0
    low_x = nx & 0xFF
    high_x = (nx >> 8) & 0xFF
    low_y = ny & 0xFF
    high_y = (ny >> 8) & 0xFF
    scroll = 0x00
    data = [button_val, low_x, high_x, low_y, high_y, scroll]
    sum_val = (sum(cmd) + sum(data)) & 0xFF
    data.append(sum_val)

    cmd_move = cmd + data
    mserial.write(bytes(cmd_move))
    mserial.flushInput()

    return True
```

然后是按键，因为我们这个脚本只用到了'D','F','ESC'这三个按键，所以其他的按键暂时不管：

2.2.2、CMD_SEND_KB_GENERAL_DATA

通过该命令向芯片发送普通键盘数据包，模拟普通按键按下或释放动作。支持全键盘、组合键操作，可支持 8+6 个无冲突按键，其中 8 为 8 个控制键(左 Ctrl、右 Ctrl、左 Shift、右 Shift、左 Windows、右 Windows、左 Alt 和右 Alt)，6 为 6 个控制键之外的普通按键。

外围串口设备→芯片：

帧头	地址码	命令码	后续数据长度	后续数据	累加和
HEAD	ADDR	CMD	LEN	DATA	SUM
0x57、0xAB	0x00	0x02	8	8 个字节数据	0x??

该命令带 8 个字节的后续数据，后续数据为 USB 键盘普通按键的键值。

依次为：

(1)、第 1 个字节：1 个字节的控制键，每个位表示 1 个按键，具体如下：

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
右 Windows 键	右 Alt 键	右 Shift 键	右 Ctrl 键	左 Windows 键	左 Alt 键	左 Shift 键	左 Ctrl 键

(2)、第 2 个字节：1 个字节 0x00，该字节必须为 0x00；

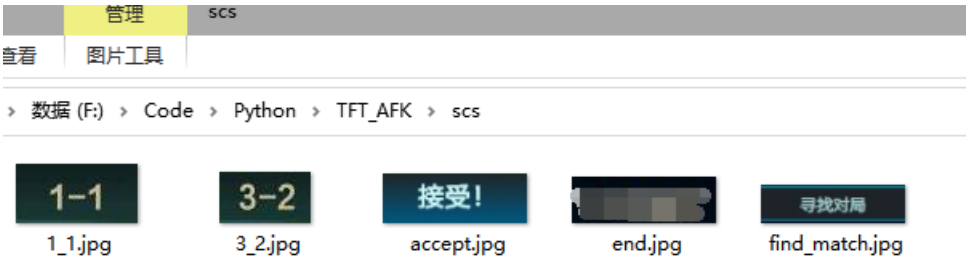
(3)、第 3-8 个字节：6 个字节普通按键值，最多可以表示 6 个按键按下，如果无按键按下则填写 0x00；

具体键盘普通按键及对应的键码见附录 1-“CH9329 键码表”。

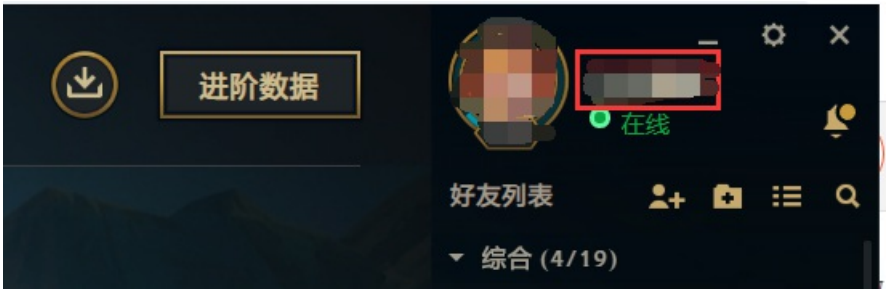
```
# ch9329 键值对
key_map = {
    'A': 0x4,
    'D': 0x7,
    'F': 0x9,
    'ESC':0x29,
}
# ch9329 控制键offset
control_key_map = {
    'ctrl': 0,
    'shift': 1,
    'alt': 2,
    'win': 3,
}

def hard_key_write(keys):
    global key_map, control_key_map
    l_keys = keys.split('+')
    v = []
    c = []
    for k in l_keys:
        k = k.upper()
        if k in key_map:
            v.append(key_map[k])
        if k in control_key_map:
            c.append(control_key_map[k])

    cmd = [0x57, 0xab, 0x00, 0x02, 0x08]
    data = []
    ctl_byte = 0x0
    for i in c:
        ctl_byte += 1 << i
    data.append(ctl_byte)   # 1
    data.append(0x00)       # 2
    data += v[:6]            # 3-8
    data += [0] * (8 - len(data))

    press = cmd + data
    press.append(sum(press) & 0xFF)

    release = [0x57, 0xab, 0x00, 0x02, 0x08] + [0x00] * 8 + [0x0c]

    mserial.write(bytes(press))
    # 延时50ms
    time.sleep(50 / 1000)
    # 释放键
    mserial.write(bytes(release))
    time.sleep(50 / 1000)

    mserial.flushInput()

    return True
```

截图准备

首先我们需要去截图，把我们要点击的按钮，要检测的图片先截图保存下来：



end.jpg是对局完成后，结算界面右上角的你自己的召唤师名字。为什么不使用“再来一局”按钮的截图作为检测图片呢？因为LOL这个客户端结算界面，如果鼠标移动到了好友的头像上面，它的下半部分会卡成黑条，导致我们一直检测不到按钮图片，脚本卡在这里动不了。打个码，免得被封号（手动滑稽）。



正常情况下，石头人打完就能点投降了，也就是3-1回合。这里截3-2回合是因为，3-1回合刚开始有时是不能点投降的，得等到战斗开始一阵后才能点投降的，所以直接采用3-2回合，确保投降能一次点成功。

所以现在流程就变成了这样：检测find_match.jpg图片，如果检测到了，调用hard_click去点击检测到位置的中心。然后一边等待一边检测accept.jpg图片，如果检测到了，点击中心。然后我们需要一边等待，一边检测LOL的游戏进程是不是启动了，同时还要检测accept.jpg这张图片，因为如果有人拒绝了，需要再次点击接受按钮。游戏进程启动后，我们一边等待一边检测1_1.jpg这张图片，检测到后，我们就开始当演员，比如控制鼠标在棋盘上右键随便点几个点，防止被系统认为我们在挂机，然后控制键盘按F给小小英雄升个等级之类的。然后就是一边当演员，一边检测3_2.jpg这张图片，检测到后，直接控制鼠标点小地图上面的设置按钮，然后点“发起投降”，然后点“确认退出”，三连告辞。然后等待游戏进程结束。然后去检测end.jpg，检测到后，直接根据再来一局和LOL客户端的相对位置，直接控制鼠标在那个位置点击（“再来一局”按钮，不管这个按钮是否被黑条挡住了）。然后重复以上流程。

注意：每次点击完按钮后，都需要将鼠标移开到非检测区域，防止影响后面的检测。

流程代码实现

检测和接受对局：

```
def pg_find_match(lol_hwnd):
    global thread_stage
    thread_stage = ''
    pic_find_match = Image.open(r'scs\find_match.jpg')
    pic_accept = Image.open(r'scs\accept.jpg')

    while True:
        # 点击寻找对局
        box = pg.locateOnScreen(pic_find_match, confidence=0.9)
        if box:
            p = pg.center(box)
            hard_click(p[0], p[1])
            # 将鼠标移动到非检测区域，防止遮挡按钮，影响后面的检测
            move_to_empty_area(lol_hwnd)
        else:
            time.sleep(1)
            continue
        client_title = 'League of Legends (TM) Client'
        # 队列中
        while True:
            # 接受对局
            box3 = pg.locateOnScreen(pic_accept, confidence = 0.9)
            if box3:
                p3 = pg.center(box3)
                hard_click(p3[0], p3[1])
                move_to_empty_area(lol_hwnd)
                time.sleep(5)
            if pg.getWindowsWithTitle(client_title) != []:
                return True
            time.sleep(1)

    return False
```

等待游戏进程启动和加载界面结束：

```
def pg_wait_loading(lol_hwnd):
    pic = Image.open(r'scs\1_1.jpg')

    while True:
        # pyautogui 模块检测关卡
        box = pg.locateOnScreen(pic, confidence=0.9)
        if box:
            return True

    return True
```

在对局内当演员：

```
def hang_out():
    global box_stage_3_2

    # 在棋盘小范围内随机游荡，防止鼠标指针挡住关卡图片导致检测不到3-2关卡
    x = random.randint(400, 1500)
    y = random.randint(100, 700)

    # 随机点击2-5下鼠标右键
```

```
    for i in range(random.randint(2,5)):
        hard_click(x, y, button=pg.RIGHT)
        time.sleep(0.1)

    return True

# 商店5个怪的边框位置（1920x1080分辨率下，其他分辨率情况下需要自己截图计算出大概位置）
champ_box = [
    (480, 930, 670, 1070),
    (680, 930, 870, 1070),
    (880, 930, 1070, 1070),
    (1080, 930, 1270, 1070),
    (1290, 930, 1480, 1070),
]

# 买一个怪
def buy_single_champ(index):
    global champ_box

    if index > 4:
        index = 4
    if index < 0:
        index = 0

    box = champ_box[index]
    center = get_box_center(box)
    hard_click(center[0], center[1], button=pg.LEFT)

    return True

# 刷新商店
def refresh_shop():
    hard_key_write('D')
    return True
# 买经验
def upgrade_champ():
    hard_key_write('F')

# 一边演，一边等待3-2回合
def pg_wait_stage_3_2():
    pic = Image.open(r'scs\3_2.jpg')

    while True:
        # pyautogui 模块检测关卡
        box = pg.locateOnScreen(pic, confidence=0.9)
        if box:
            return True

        case = random.randint(1,100)
        # %40 概率游荡
        if case < 40:
            hang_out()
        case = random.randint(1,100)
        # %30 概率买1个英雄
        if case >= 20 and case < 50:
            buy_single_champ(random.randint(0, 4))

        case = random.randint(1,100)
        # %1 概率刷新商店
```

```
        if case == 50:
            refresh_shop()

        case = random.randint(1,100)
        # %10 概率升级
        if case >= 80 and case < 90:
            upgrade_champ()

        time.sleep(2)
```

三连告辞（投降）：

```
def get_box_center(box):
    x = int((box[2] - box[0]) / 2) + box[0]
    y = int((box[3] - box[1]) / 2) + box[1]

    return (x, y)


def surrender():
    global box_settings, box_surrender, box_confirm, thread_check_flag
    center = get_box_center(box_settings)
    hard_moveTo(center[0], center[1])  # 点击设置
    time.sleep(0.3)
    hard_click(center[0], center[1])
    # hard_key_write('esc')
    time.sleep(1)
    center = get_box_center(box_surrender)
    hard_moveTo(center[0], center[1])  # 发起投降
    time.sleep(0.3)
    hard_click(center[0], center[1])
    time.sleep(0.5)
    center = get_box_center(box_confirm)
    hard_moveTo(center[0], center[1])  # 确定离开
    time.sleep(0.3)
    hard_click(center[0], center[1])

    thread_check_flag = 0

    return True
```

注意：LOL的这些按钮，直接hard_click点击按钮，可能不成功，界面不会显示出来。所以先把鼠标移到图标上，稍微延迟一点时间再进行点击，这样确保点击成功。

等待游戏进程结束：

```
def pg_wait_surrender_finish():
    client_title = 'League of Legends (TM) Client'
    times = 0
    while True:
        time.sleep(1)
        times += 1
        if pg.getWindowsWithTitle(client_title) == []:
            break

        if times % 10 == 0:
            print("pg_wait_surrender_finish %d sec, still wait client over!"%times)

    # 检测结算界面是否打开
    while True:
        pic = Image.open(r'scs\end.jpg')
        box = pg.locateOnScreen(pic, confidence = 0.9)
        if box:
            break

        time.sleep(1)

    return True
```

再玩一次：

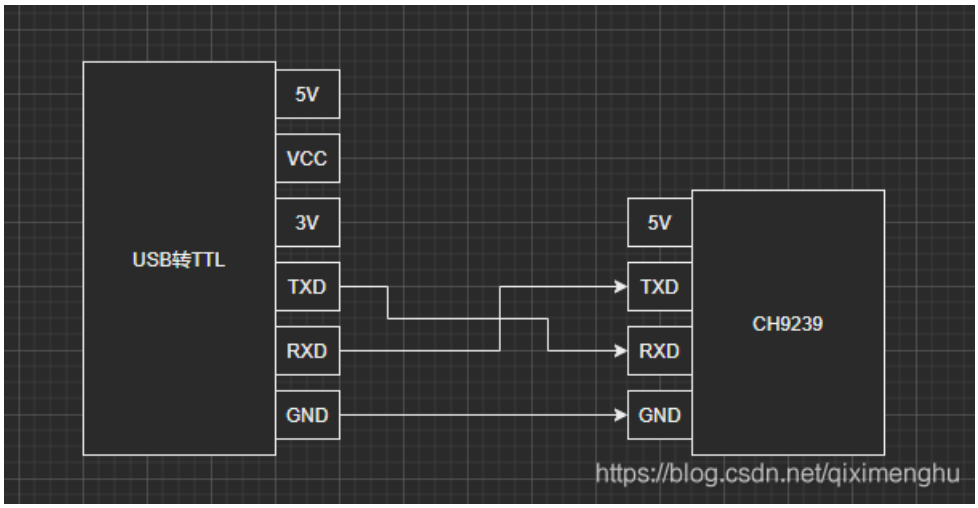
```
def pg_play_again(lol_hwnd):
    global box_play_again, box_empty_area
    center = convert_lol_to_destop_point(get_box_center(box_play_again), lol_hwnd)
    hard_moveTo(center[0], center[1])
    time.sleep(0.3)
    hard_click(center[0], center[1])  # 点击再玩一次
    move_to_empty_area(lol_hwnd)
    return True
```

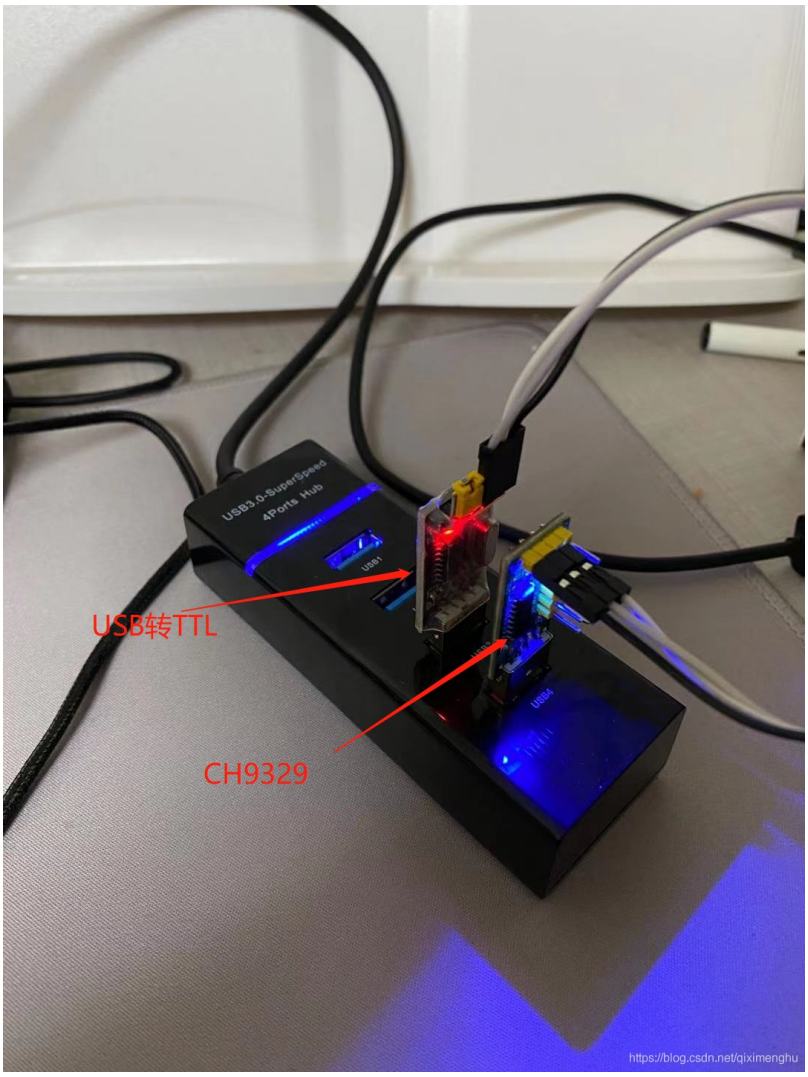
注意：这部分代码有可能出现点击‘再玩一次’按钮失败的情况，然后导致脚本不能正常执行。可以考虑检测是否出现‘寻找对局’按钮来检测是否点击成功：

```
def pg_play_again(lol_hwnd):
    global box_play_again, box_empty_area
    center = convert_lol_to_destop_point(get_box_center(box_play_again), lol_hwnd)
    hard_moveTo(center[0], center[1])
    time.sleep(0.5)
    hard_click(center[0], center[1])  # 点击再玩一次
    move_to_empty_area(lol_hwnd)

    # 检测寻找对局按钮
    time.sleep(1)
    pic_find_match = Image.open(r'scs\find_match.jpg')
    box = pg.locateOnScreen(pic_find_match)

    while not box:
        hard_moveTo(center[0], center[1])
        time.sleep(0.5)
        hard_click(center[0], center[1])  # 点击再玩一次
        move_to_empty_area(lol_hwnd)

        time.sleep(1)
        box = pg.locateOnScreen(pic_find_match)

    return True
```

硬件连接

只要电脑上有两个USB口就行，通过杜邦线连接USB转TTL和CH9239模块，如果没有，再买个USB转hub的器件：





完整代码

```
import time
import pyautogui as pg
from PIL import Image
import serial
import serial.tools.list_ports
import random

mserial = None
AutoPalyFlag = True
# 0:未进入对局 1:第一阶段 2:剩下的阶段

# 设置 图标 box
box_settings = (1890, 870, 1920, 900)
# 发起投降 按钮 box
box_surrender = (700, 820, 840, 860)
# 确认投降 按钮 box
box_confirm = (720, 460, 940, 520)
# 空白区域
box_empty_area = (700, 0, 800, 10)
# 再玩一次
box_play_again = (530, 830, 780, 880)

# 商店5个怪的box
champ_box = [
    (480, 930, 670, 1070),
```

```
    (680, 930, 870, 1070),
    (880, 930, 1070, 1070),
    (1080, 930, 1270, 1070),
    (1290, 930, 1480, 1070),
]

# ch9329 键值对
key_map = {
    'A': 0x4,
    'D': 0x7,
    'F': 0x9,
    'ESC':0x29,
}
# ch9329 控制键offset
control_key_map = {
    'ctrl': 0,
    'shift': 1,
    'alt': 2,
    'win': 3,
}

def get_button_val(button):
    if button == pg.LEFT:
        return 0
    if button == pg.RIGHT:
        return 1
    if button == pg.MIDDLE:
        return 2

def hard_click(x, y, clicks=1, interval=0.0, button=pg.LEFT, duration=0.0):
    global mserial
    nx = int(x * 4096 / 1920)
    ny = int(y * 4096 / 1080)
    # 鼠标绝对坐标命令
    cmd = [0x57, 0xAB, 0x00, 0x04, 0x07, 0x02]
    button_val = 1 << get_button_val(button)
    low_x = nx & 0xFF
    high_x = (nx >> 8) & 0xFF
    low_y = ny & 0xFF
    high_y = (ny >> 8) & 0xFF
    scroll = 0x00
    data = [button_val, low_x, high_x, low_y, high_y, scroll]
    sum_val = (sum(cmd) + sum(data)) & 0xFF
    data.append(sum_val)
    # 按下
    press = cmd + data
    # 释放
    release = press.copy()
    release[6] = 0x0
    sum_val = sum(release[:-1]) & 0xFF
    release[len(release) - 1] = sum_val

    while clicks > 0:
        # 移动并按下键
        mserial.write(bytes(press))
        # 延时50ms
        time.sleep(50 / 1000)
        # mserial.readall()
        # 释放键
        mserial.write(bytes(release))
```

```
        time.sleep(50 / 1000)
        # mserial.readall()
        clicks -= 1
        if clicks > 0:
            time.sleep(interval)

    mserial.flushInput()

    return True

def hard_moveTo(x, y):
    global mserial
    nx = int(x * 4096 / 1920)
    ny = int(y * 4096 / 1080)

    # 鼠标绝对坐标命令
    cmd = [0x57, 0xAB, 0x00, 0x04, 0x07, 0x02]
    button_val = 0
    low_x = nx & 0xFF
    high_x = (nx >> 8) & 0xFF
    low_y = ny & 0xFF
    high_y = (ny >> 8) & 0xFF
    scroll = 0x00
    data = [button_val, low_x, high_x, low_y, high_y, scroll]
    sum_val = (sum(cmd) + sum(data)) & 0xFF
    data.append(sum_val)

    cmd_move = cmd + data
    mserial.write(bytes(cmd_move))
    mserial.flushInput()

    return True

def hard_key_write(keys):
    global key_map, control_key_map
    l_keys = keys.split('+')
    v = []
    c = []
    for k in l_keys:
        k = k.upper()
        if k in key_map:
            v.append(key_map[k])
        if k in control_key_map:
            c.append(control_key_map[k])

    cmd = [0x57, 0xab, 0x00, 0x02, 0x08]
    data = []
    ctl_byte = 0x0
    for i in c:
        ctl_byte += 1 << i
    data.append(ctl_byte)   # 1
    data.append(0x00)       # 2
    data += v[:6]            # 3-8
    data += [0] * (8 - len(data))

    press = cmd + data
    press.append(sum(press) & 0xFF)

    release = [0x57, 0xab, 0x00, 0x02, 0x08] + [0x00] * 8 + [0x0c]
```

```
    mserial.write(bytes(press))
    # 延时50ms
    time.sleep(50 / 1000)
    # 释放键
    mserial.write(bytes(release))
    time.sleep(50 / 1000)

    mserial.flushInput()

    return True

def hang_out():
    # 在棋盘小范围内随机游荡，防止鼠标指针挡住关卡图片导致检测不到3-2关卡
    x = random.randint(400, 1500)
    y = random.randint(100, 700)

    # 随机点击2-5下鼠标右键
    for i in range(random.randint(2,5)):
        hard_click(x, y, button=pg.RIGHT)
        time.sleep(0.1)

    return True

# 买一个怪
def buy_single_champ(index):
    global champ_box

    if index > 4:
        index = 4
    if index < 0:
        index = 0

    box = champ_box[index]
    center = get_box_center(box)
    hard_click(center[0], center[1], button=pg.LEFT)

    return True

# 刷新商店
def refresh_shop():
    hard_key_write('D')
    return True

# 提升等级
def upgrade_champ():
    hard_key_write('F')
    return True

def get_lol_hwnd():
    title = 'League of Legends'
    w = pg.getWindowsWithTitle(title)
    for i in w:
        if i.title == title:
            w = i
            break

    return w

def get_available_serial():
```

```
def get_available_serial():
    l = list(serial.tools.list_ports.comports())
    if len(l) < 1:
        return False
    port_list = list(l[0])
    port = port_list[0]
    return port

def open_serial():
    global mserial
    port = get_available_serial()
    if not port:
        return False
    # mserial = serial.Serial("COM3", 115200, timeout=1)
    mserial = serial.Serial(port, 115200, timeout=1)
    return True

def get_box_center(box):
    x = int((box[2] - box[0]) / 2) + box[0]
    y = int((box[3] - box[1]) / 2) + box[1]

    return (x, y)

def move_to_empty_area(lol_hwnd):
    global  box_empty_area

    center = convert_lol_to_destop_point(get_box_center(box_empty_area), lol_hwnd)
    hard_moveTo(center[0], center[1])

    return True

# 将LOL客户端相对坐标根据LOL客户端位置转换为实际桌面的绝对坐标
def convert_lol_to_destop_point(p, lol_hwnd):
    x = lol_hwnd.topleft[0] + p[0]
    y = lol_hwnd.topleft[1] + p[1]

    return (x, y)

def surrender():
    global box_settings, box_surrender, box_confirm
    center = get_box_center(box_settings)
    hard_moveTo(center[0], center[1])  # 点击设置
    time.sleep(0.3)
    hard_click(center[0], center[1])
    # hard_key_write('esc')
    time.sleep(1)
    center = get_box_center(box_surrender)
    hard_moveTo(center[0], center[1])  # 发起投降
    time.sleep(0.3)
    hard_click(center[0], center[1])
    time.sleep(0.5)
    center = get_box_center(box_confirm)
    hard_moveTo(center[0], center[1])  # 确定离开
    time.sleep(0.3)
    hard_click(center[0], center[1])

    return True

# 寻找对局 -> 队列中 -> 接受 -> 开始加载
# 寻找对局 -> 队列中 -> 接受 -> 有人拒绝 -> 队列中
```

```
def pg_find_match(lol_hwnd):
    pic_find_match = Image.open(r'scs\find_match.jpg')
    pic_accept = Image.open(r'scs\accept.jpg')

    while True:
        # 点击寻找对局
        box = pg.locateOnScreen(pic_find_match, confidence=0.9)
        if box:
            p = pg.center(box)
            hard_click(p[0], p[1])
            move_to_empty_area(lol_hwnd)
        else:
            time.sleep(1)
            continue
        client_title = 'League of Legends (TM) Client'
        # 队列中
        while True:
            # 接受对局
            box3 = pg.locateOnScreen(pic_accept, confidence = 0.9)
            if box3:
                p3 = pg.center(box3)
                hard_click(p3[0], p3[1])
                move_to_empty_area(lol_hwnd)
                time.sleep(5)
            if pg.getWindowsWithTitle(client_title) != []:
                return True
            time.sleep(1)

    return False

def pg_wait_loading(lol_hwnd):
    pic = Image.open(r'scs\1_1.jpg')

    while True:
        # pyautogui 模块检测关卡
        box = pg.locateOnScreen(pic, confidence=0.9)
        if box:
            return True
        time.sleep(2)
    return True

def pg_wait_stage_3_2():
    pic = Image.open(r'scs\3_2.jpg')

    while True:
        # pyautogui 模块检测关卡
        box = pg.locateOnScreen(pic, confidence=0.9)
        if box:
            return True

        case = random.randint(1,100)
        # %40 概率游荡
        if case < 40:
            hang_out()
        case = random.randint(1,100)
        # %30 概率买1个英雄
        if case >= 20 and case < 50:
            buy_single_champ(random.randint(0, 4))

        case = random.randint(1,100)
```

```
        case = random.randint(1,100)
        # %1 概率刷新商店
        if case == 50:
            refresh_shop()

        case = random.randint(1,100)
        # %10 概率升级
        if case >= 80 and case < 90:
            upgrade_champ()

        time.sleep(2)

def pg_wait_surrender_finish():
    client_title = 'League of Legends (TM) Client'
    times = 0
    while True:
        time.sleep(1)
        times += 1
        if pg.getWindowsWithTitle(client_title) == []:
            break

        if times % 10 == 0:
            print("pg_wait_surrender_finish %d sec, still wait client over!"%times)

    # 检测结算界面是否打开
    while True:
        pic = Image.open(r'scs\end.jpg')
        box = pg.locateOnScreen(pic, confidence = 0.9)
        if box:
            break

        time.sleep(1)

    return True

def pg_play_again(lol_hwnd):
    global box_play_again, box_empty_area
    center = convert_lol_to_destop_point(get_box_center(box_play_again), lol_hwnd)
    hard_moveTo(center[0], center[1])
    time.sleep(0.5)
    hard_click(center[0], center[1])  # 点击再玩一次
    move_to_empty_area(lol_hwnd)

    # 检测寻找对局按钮
    time.sleep(1)
    pic_find_match = Image.open(r'scs\find_match.jpg')
    box = pg.locateOnScreen(pic_find_match)

    # 未找到，就重新点击再来一局按钮，然后继续找
    while not box:
        hard_moveTo(center[0], center[1])
        time.sleep(0.5)
        hard_click(center[0], center[1])  # 点击再玩一次
        move_to_empty_area(lol_hwnd)

        time.sleep(1)
        box = pg.locateOnScreen(pic_find_match)

    return True
```

```
def pg_main():
    global mserial, AutoPalyFlag

    open_serial()

    w = get_lol_hwnd()

    while AutoPalyFlag:
        print("[%s] pg_find_match"%time.asctime())
        pg_find_match(w)
        print("[%s] pg_wait_loading"%time.asctime())
        pg_wait_loading(w)
        print("[%s] pg_wait_stage_3_2"%time.asctime())
        pg_wait_stage_3_2()
        print("[%s] surrender"%time.asctime())
        surrender()
        print("[%s] pg_wait_surrender_finish"%time.asctime())
        pg_wait_surrender_finish()
        print("[%s] pg_play_again"%time.asctime())
        pg_play_again(w)

    mserial.close()
    print('Over')

    return True

if __name__ == '__main__':
    pg_main()
```

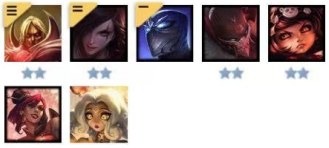
注意：其中的坐标都是LOL客户端分辨率1600x900，LOL游戏分辨率1980x1080时测试得到的，如果你的电脑设置了不同的分辨率，需要重新截图计算实际的坐标。

实际效果

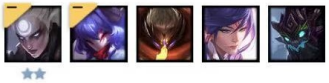
基本15分钟左右一把，一小时能刷4把，24x4 = 96把，一把两个币，一天192个币。护肝。

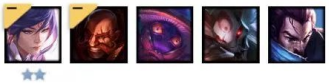
全部战绩

#8  匹配赛 7分钟前

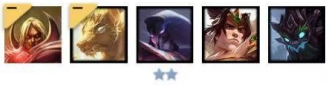
#8  匹配赛 24分钟前

#8  匹配赛 41分钟前

#8  匹配赛 13分钟前

#8  匹配赛 28分钟前

#8  匹配赛 40分钟前

#8  匹配赛 54分钟前


https://blog.csdn.net/qiximenghu