گزارش پروژه نهایی درس سیستم های نهفته

WoWLAN

استاد محترم: دکتر حسینی منزه

اعضای گروه: شهرزاد آذری، روزبه غزوی، پریا فصاحت

شرح پروژه

در این پروژه قصد داریم به کمک ارسال بسته Wake on Lan از طریق اینترنت، وسیله برقی مورد نظر را که از این قابلیت پشتیبانی نمی کند از راه دور کنترل کنیم.

ابتدا به این می پردازیم که قابلیت Wake on Lan به طور کلی چیست؟

Wake-on-LAN (WoL or WOL) is an Ethernet or Token Ring computer networking standard that allows a computer to be turned on or awakened by a network message.

The message is usually sent to the target computer by a program executed on device connected to the same local area network

به بسته ای که به منظور Wake on Lan فرستاده می شود Magic Packet می گویند که مشخصات زیر را دارد:

The magic packet is a frame that is most often sent as a broadcast and that contains anywhere within its payload 6 bytes of all 255 (FF FF FF FF FF FF in hexadecimal), followed by sixteen repetitions of the target computer's 48-bit MAC address, for a total of 102 bytes.

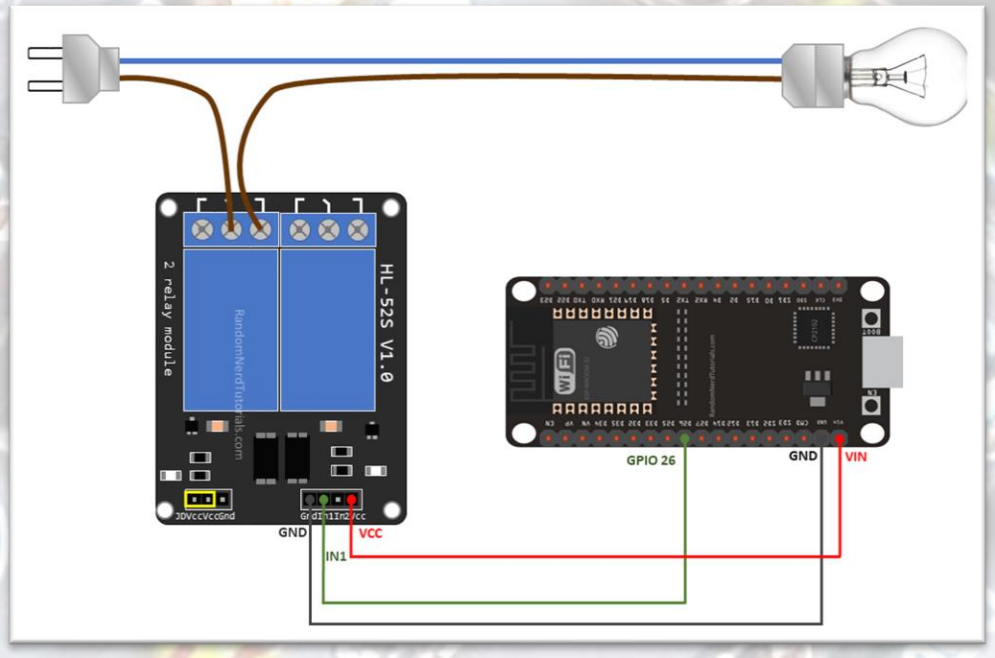
Since the magic packet is only scanned for the string above, and not actually parsed by a full protocol stack, it could be sent as payload of any network- and transport-layer protocol, although it is typically sent as a UDP datagram to port 0 (reserved port number),7 (Echo Protocol) or 9 (Discard Protocol), or directly over Ethernet. A connection-oriented transport-layer protocol like TCP is less suited for this task as it requires establishing an active connection before sending user data.

وسایل مورد نیاز

- برد توسعه ESP32
- ماژول رله 5 ولت
- سیم جامپر مادگی به مادگی
- دو شاخه
- مقداری سیم برای سیم کشی رله و دو شاخه
- منبع تغذیه (باتری) برای روشن کردن برد

نمای کلی مدار پروژه

ابتدا مدار خود را مانند شکل زیر می بندیم.



برای Power supply میتوان از باتری یا پورت Micro-USB کمک گرفت.

ورودی کانال رله را به هر GPIO دلخواه می توان وصل کرد که در اینجا به پین 26 وصل کرده ایم.

برای مثال از لامپ با سرپیچ استفاده شده اما هر وسیله برقی را می توان جای لامپ قرار داد.

همچنین با توجه به نیاز می توان از ماژول رله 5 ولت با تعداد کانال بیشتر بهره برد.(هر کانال برای روشن کردن یک دستگاه برقی استفاده می شود)

پس از بستن مدار به سراغ برنامه نویسی ESP32 به کمک محیط Arduino ide می رویم تا بتوانیم به کمک بسته WOL ماژول رله را کنترل کنیم.

برنامه نویسی

برای نوشتن برنامه این پروژه نیاز به دو برنامه داریم.

یک برنامه برای ESP32 که پکت WOL را از طریق اینترنت دریافت کرده و آن را تشخیص دهد و با توجه به دستورات برنامه، رله را کنترل کند.

همچنین یک برنامه دیگر برای ارسال بسته WOL که با زبان پایتون نوشته می شود.

در ادامه به بررسی کد این دو برنامه می پردازیم.

کد ESP32

همانطور که در شرح پروژه ذکر شد پروتکل TCP زیاد مناسب فرستادن Magic Packet نیست زیرا نیاز به Handshake دارد و وقت گیر تر است. ازین رو از UDP استفاده میکنیم.

در این قسمت کتاب خانه های مورد نیاز خود را Include می کنیم. از Wifi.h برای متصل شدن میکروکنترلر به Wifi و از AsyncUDP برای بالا اوردن سرور UDP استفاده می کنیم.

```
#include "WiFi.h"
#include "AsyncUDP.h"
```

سپس ssid و پسورد WIFI مورد نظر را وارد می کنیم. و در خط بعد شماره پین که رله به آن متصل است را مشخص میکنیم.

```
const char* ssid = "";
const char* pass = "";
const int relay = 26;
```

در مرحله بعد در Setup برنامه، ابتدا Baud rate سریال مانیتور را تعیین می کنیم سپس پینی که رله به آن متصل است را خروجی قرار میدهیم و در خطوط بعد میکرو را به وایفای متصل می کنیم و مک آدرس دستگاه را در سریال مانیتور چاپ می کنیم.

```
  Serial.begin(115200);
  pinMode(relay, OUTPUT);
  //turn it off
  digitalWrite(relay, HIGH);
  WiFi.disconnect(true);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, pass);
  Serial.println(WiFi.macAddress());

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print("connecting...");
  }
```

در مرحله بعد سرور UDP را بالا اورده و روی پورت هفت Listen میکنیم تا پکت های ارسال شده به این پورت را دریافت کنیم. همچنین مشخصات پکت از قبیل سایز و طول و آی پی ارسال کننده و ... را چاپ می کنیم.

```
  if (udp.listen(7)) {
    Serial.print(" ");
    Serial.print("UDP Listening on IP: ");
    Serial.println(WiFi.localIP());
    udp.onPacket([](AsyncUDPPacket packet) {
      Serial.print("UDP Packet Type: ");
      Serial.print(packet.isBroadcast() ? "Broadcast" : packet.isMulticast() ?
"Multicast" : "Unicast");
      Serial.print(", From: ");
      Serial.print(packet.remoteIP());
      Serial.print(":");
      Serial.print(packet.remotePort());
      Serial.print(", To: ");
      Serial.print(packet.localIP());
      Serial.print(":");
      Serial.print(packet.localPort());
      Serial.print(", Length: ");
      Serial.print(packet.length());
      Serial.print(", Data: ");
      Serial.write(packet.data(), packet.length());
      Serial.println();
```

```
String myString = (const char*)packet.data();
packet.printf("Got %u bytes of data", packet.length());
```

در مرحله آخر با کمک گرفتن از طول و سایز پکت دریافت شده تشخیص می دهیم که آیا پکت دریافتی از نوع Magic Packet هست یا خیر. (برای Latch مود بسته ارسالی را تغییر می دهیم)

```
//Normal mode
if (packet.length() == 102)
{
  if(count % 2 != 0)
  {
  digitalWrite(relay, LOW);
  }
  else
  {
    digitalWrite(relay, HIGH);
  }
  count++;
}

//Latch mode
if (packet.length() == 96) {
  digitalWrite(relay,LOW);
  delay(1000);
  digitalWrite(relay,HIGH);
}
```

کد Magic packet sender

ابتدا با توجه به مشخصات Magic Packet آن را می سازیم.

برای Latch Mode یک عدد از Mac address ها را کم میکنیم و طول بسته به 96 کاهش می یابد.

```
#This function checks the MAC address format and create magic packet
def create_magic_packet(macaddress: str, mode: str) -> bytes:

    if len(macaddress) == 17:
        sep = macaddress[2]
        macaddress = macaddress.replace(sep, "")
    elif len(macaddress) == 14:
        sep = macaddress[4]
        macaddress = macaddress.replace(sep, "")
    if len(macaddress) != 12:
        raise ValueError("Incorrect MAC address format")

    #Magic Packet Structure
    if(mode == 'Normal'):
        return bytes.fromhex("F" * 12 + macaddress * 16 )

    if(mode == 'Latch'):
        return bytes.fromhex("F" * 12 + macaddress * 15 )
```

پس از ساختن پکت آن را به کمک Socket Programming به پورت هفت UDP Server در حال اجرا بر روی میکروکنترل میفرستیم.

```
#This function sends the magic packet to a specified Host.
def send_magic_packet(
    *macs: str,
    ip_address: str = BROADCAST_IP,
    port: int = DEFAULT_PORT,
    interface: Optional[str] = None,
    mode: str = "Normal"
) -> None:
    if(mode == 'Latch'):
        packets = [create_magic_packet(mac,'Latch') for mac in macs]
    else:
        packets = [create_magic_packet(mac,'Normal') for mac in macs]
```

```
    with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as sock:
        if interface is not None:
            sock.bind((interface, 0))
        sock.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
        sock.connect((ip_address, port))
        for packet in packets:
            sock.send(packet)
```

در مرحله آخر در فانکشن زیر با اطلاعات دریافتی از کاربر که شامل آی پی و مک آدرس میکرو کنترلر می باشد پکت را ارسال می کنیم.

```
    #This function sends the magic packet to specified ip & mac address
    def Send_Packet(self, instance):
        self.alertlabel.color='00FF00'
        self.alertlabel.text = "Magic Packet sent successfully!"

        #both empty
        if(len(self.mac.text)==0 and len(self.ip.text)==0):
            self.alertlabel.color='FF0000'
            self.alertlabel.text = "Both fields cannot be empty!"

        #both filled
        elif(len(self.mac.text)!=0 and len(self.ip.text)!=0):
            if(self.latch_mode=="ON"):
                send_magic_packet(self.mac.text,ip_address=self.ip.text,port=7,mo
de='Latch')
            else:
                send_magic_packet(self.mac.text,ip_address=self.ip.text,port=7)

        #empty mac address
        else:
            if(len(self.mac.text)==0):
                self.alertlabel.color='FF0000'
                self.alertlabel.text = "Mac address field cannot be empty!"

        #empty ip address
            if(len(self.ip.text)==0):
                if(self.latch_mode=="ON"):
                    send_magic_packet(self.mac.text,port=7,mode='Latch')
                else:
                    send_magic_packet(self.mac.text,port=7)
```

پایان