TP Transmissions Numériques

Série Synchrone

Kimberley Jacquemot & Ayoub Ech Chamali

07/12/2022

I - Système Hydraulique

1) Dessiner le schéma bloc qui représente la carte ARDUINO UNO et son SHIELD ETHERNET version V2.

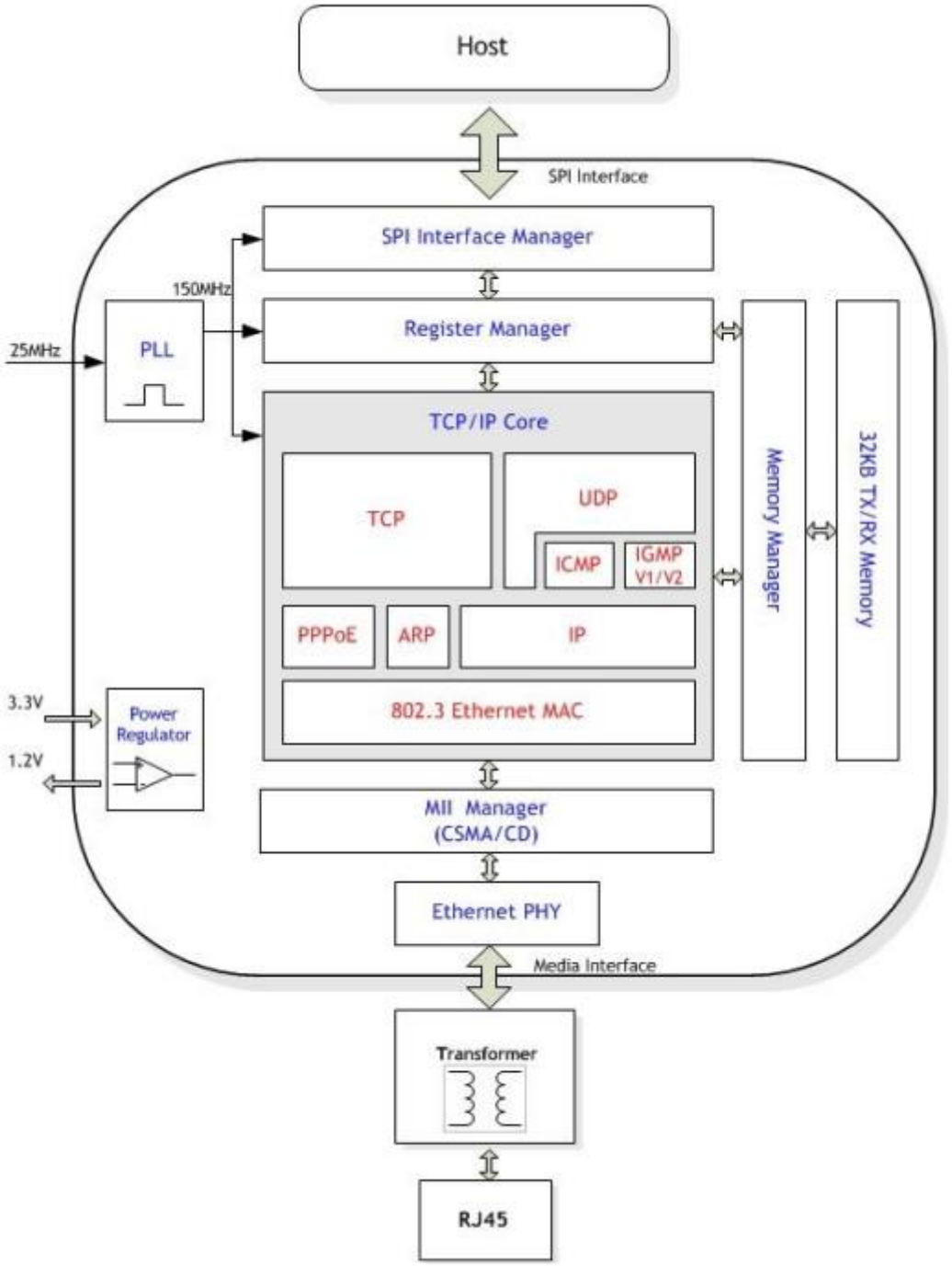


Figure 1 - Schéma relation Arduino et Shield

2) Par quel interface le micro-contrôleur ATMEGA 128P parle au SHIELD?

Le micro-contrôleur ATMEGA 128P peut communiquer avec le SHIELD via une interface SPI (Serial Peripheral Interface).

3) Quels sont les messages qui passent par cette interface ?

Les messages qui passent par cette interface peuvent inclure des données, des commandes et des informations d'état. Les données peuvent être envoyées et reçues entre le micro-contrôle.

4) Quel est le rôle du composant WIZnet 5500 ?

Le composant WIZnet 5500 est un contrôleur Ethernet qui permet à l'ATMEGA 128P de communiquer avec des périphériques réseau. Il fournit une interface Ethernet pour le micro-contrôleur.

5) Quel est le codage utilisé ?

Le codage utilisé est le codage UART (Universal Asynchronous Receiver/Transmitter).

6) A quoi sert le connecteur RJ45, et que contient-il ?

Le connecteur RJ45 est un connecteur Ethernet qui permet à l'ATMEGA 128P de se connecter à des périphériques réseau. Il contient 8 fils qui sont utilisés pour transmettre et recevoir les informations.

7) Où peut t'on observer les commandes passées par le micro-contrôleur avec le composant WIZnet 5500 ?, et comment peut on les observer ?

Les commandes passées par le micro-contrôleur avec le composant WIZnet 5500 peuvent être observées dans le moniteur série. Pour observer ces commandes, vous devez connecter un câble série à l'ATMEGA 128P et utiliser un logiciel de moniteur série pour afficher les données.

8) Où et comment peut on observer le format de la trame 802.3 ?

Le format de la trame 802.3 peut être observé dans le moniteur série ou bien dans wireshark. Pour observer ce format, nous devrons connecter un câble série à l'ATMEGA 128P et nous utiliserons ici Wireshark.

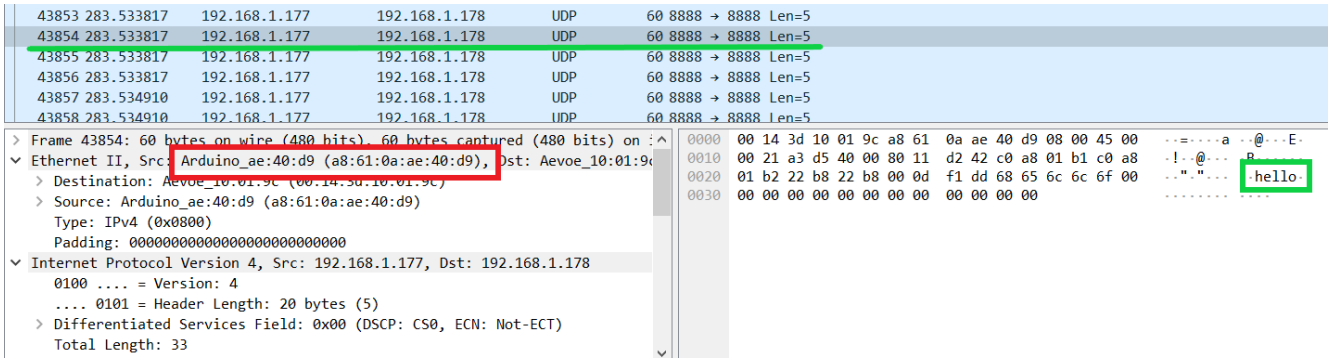


Figure 2 - Lecture de trame Wireshark

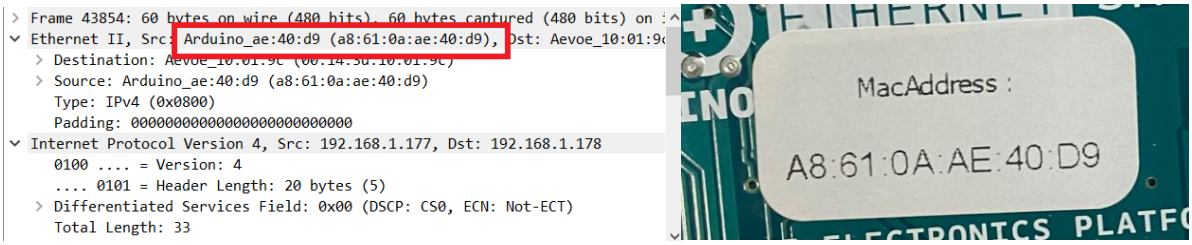
9) Est ce que le composant WIZnet respecte le standard MID/MID-X ?, et quelle en est la conséquence importante ?

Le composant WIZnet respecte le standard MID/MID-X. Cela signifie que les commandes passées par le micro-contrôleur peuvent être utilisées avec les appareils qui supportent ce standard.

10) Quel « instrumenté » et quelle architecture utiliser pour observer les trames échangées par le SHIELD et le réseau auquel il est connecté ?

Le SHIELD de l'Arduino est équipé d'un port Ethernet et d'un port USB. Il est possible d'utiliser un logiciel de surveillance des trames, comme Wireshark, pour observer les échanges entre le SHIELD et le réseau.

11) Vérifier que l'adresse MAC qui a été programmée est bien celle utilisée lors des échanges sur ETHERNET.



On a bien les mêmes adresses que ce soit sur les requêtes ou sur le sticker.

12) Où sont stockées les données qui sont échangées sur ETHERNET.

Elle est situé après l'adresse mac dans la trame envoyée

13) Le composant WIZnet fonctionne-t-il en mode Half ou Full duplex ?

WIZnet est un composant Half duplex.

14) Combien de communications simultanées le composant WIZnet 5500 est il capable de gérer ?

Le composant WIZnet 5500 est capable de gérer jusqu'à 128 communications simultanées.

15) Quel est le volume des tampons proposés par le composant WIZnet 5500 ? Qu'en pensez-vous ?

Le composant WIZnet 5500 propose un volume de tampons de 10 tampons.

16) Quels sont les registres « principaux » utilisés par le circuit WIZnet (nom et fonction) Adresse MAC de la carte SHIELD, Adresse MAC destinataire, données à transmettre en protocole UDP (User Datagram Protocol)
    - Le registre « nom » contient le nom de la carte SHIELD et sa fonction.
    - Le registre « adresse MAC » contient la MAC de la carte SHIELD.
    - Le registre « protocole UDP » contient les données à transmettre en protocole UDP.
17) Mettre en évidence par une mesure que l'isolation galvanique est bien réalisée par la prise RJ45.

On pourra observer au multimètre que l'on a une résistance infinie, ce qui montre que la carte est bien isolée.

18) Ecrire dans l'IDE ARDUINO deux programmes :
    a. un programme qui permette d'envoyer un paquet UDP contenant le texte « Hello word », et vérifier grâce à Wireshark que le paquet est conforme au protocole UDP. (Adresses MAC, IP, Taille, texte,….)
    b. un programme qui affiche les informations du paquet reçu.

Ethernet.ino

```ino
#include <SPI.h>
#include <Ethernet.h>
#include <EthernetUdp.h>

// Enter a MAC address and IP address for your controller below.
// The IP address will be dependent on your local network:
byte mac[] = { 0xA8, 0x61, 0x0A, 0xAE, 0x40, 0xD9 };
IPAddress ip(192, 168, 1, 177);
IPAddress ipReceiver(192, 168, 1, 178);
unsigned int localPort = 8888;      // local port to listen on
// An EthernetUDP instance to let us send and receive packets over UDP
EthernetUDP Udp;

void setup() {
  // start the Ethernet and UDP:
  Ethernet.begin(mac,ip);
  Udp.begin(localPort);
}

void loop() {
  Udp.beginPacket(ipReceiver, localPort);
    Udp.write("hello");
  Udp.endPacket();
}
```

Figure 3 - Envoi du message "hello"

EthernetRead.ino

```ino
#include <SPI.h>
#include <Ethernet.h>
#include <EthernetUdp.h>

byte mac[] = { 0xA8, 0x61, 0x0A, 0xAE, 0x40, 0xD9 };
IPAddress ip(192, 168, 1, 177);
IPAddress ipReceiver(192, 168, 1, 178);
unsigned int localPort = 8888;      // local port
EthernetUDP Udp;
char packetBuffer[UDP_TX_PACKET_MAX_SIZE];
char ReplyBuffer[] = "okidoki";       //ACK


void setup() {
  //Connexion
  Ethernet.begin(mac,ip);
  Udp.begin(localPort);
}
```

Figure 4 - Première partie de réception de données

```
23  void loop() {
24
25    int packetSize = Udp.parsePacket();
26    //on affiche les données du pacquet, s'il y en a un.
27    if (packetSize) {
28      Serial.print("Received packet of size ");
29      Serial.println(packetSize);
30      Serial.print("From ");
31      IPAddress remote = Udp.remoteIP();
32      for (int i=0; i < 4; i++) {
33        Serial.print(remote[i], DEC);
34        if (i < 3) {
35          Serial.print(".");
36        }
37      }
38      Serial.print(", port ");
39      Serial.println(Udp.remotePort());
40
41      // On lit les données grâce à un packet Buffer
42      Udp.read(packetBuffer, UDP_TX_PACKET_MAX_SIZE);
43      Serial.println("Contents:");
44      Serial.println(packetBuffer);
45
46      //On confirme à l'emetteur que nous avons bien reçu le paquet.
47      Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());
48      Udp.write(ReplyBuffer);
49      Udp.endPacket();
50    }
51    delay(10);
52  }
```

Figure 5 - Code pour réception d'un paquet

```
EthernetUDP Udp;
char packetBuffer[UDP_TX_PACKET_MAX_SIZE];
char ReplyBuffer[] = "okidoki";        //ACK
```

Figure 6 - Nouvelles valeurs initialisées

19) Qui gère le calcule, le contrôle du CRC et effectue les éventuelles corrections ?

Avec ce mécanisme de détection d'erreur, un polynôme prédéfini est connu de l'émetteur et du récepteur. L'émetteur effectue un algorithme sur les bits de la trame qu'il va envoyer afin de générer un CRC (notre polynôme), et de transmettre ces éléments au récepteur. Le récepteur va ensuite effectuer les mêmes afin de vérifier que les éléments partagés sont les mêmes.

20) Ecrire dans l'IDE ARDUINO deux programmes qui permettent d'envoyer et de recevoir une matrice 5X5 de nombre flottant codés sur 32 bits. La matrice est initialisée par des nombres de 1.0 -> 25.0.