

Paramétrisation vidéo, instrumentation spatiale

Zynq SoC - Linux vs Bare-Metal

Camille Lefort
M2 MISTRE – Projet Intégrateur
30 Janvier 2026



Comment mettre en place une communication Ethernet, permettant la transmission d'images ?

- ◀ Contexte : Paramétrisation capteur optique
 - ◀ Contraintes : Rapidité, efficacité, justesse.
 - ◀ Enjeu : Meilleure observation
-

◀ 1. Présentation Sujet / CSUG

◀ 2. Méthodes utilisées

2.1 Implémentation Bare-Metal

2.2 Implementation Linux embarqué

◀ 3. Résultats et analyses

3.1 Résultats obtenus

3.2 Analyse Bare-Metal vs Linux

◀ 4. Conclusion

1 Présentation Sujet

1.1 Contexte : Projet QlevErSat

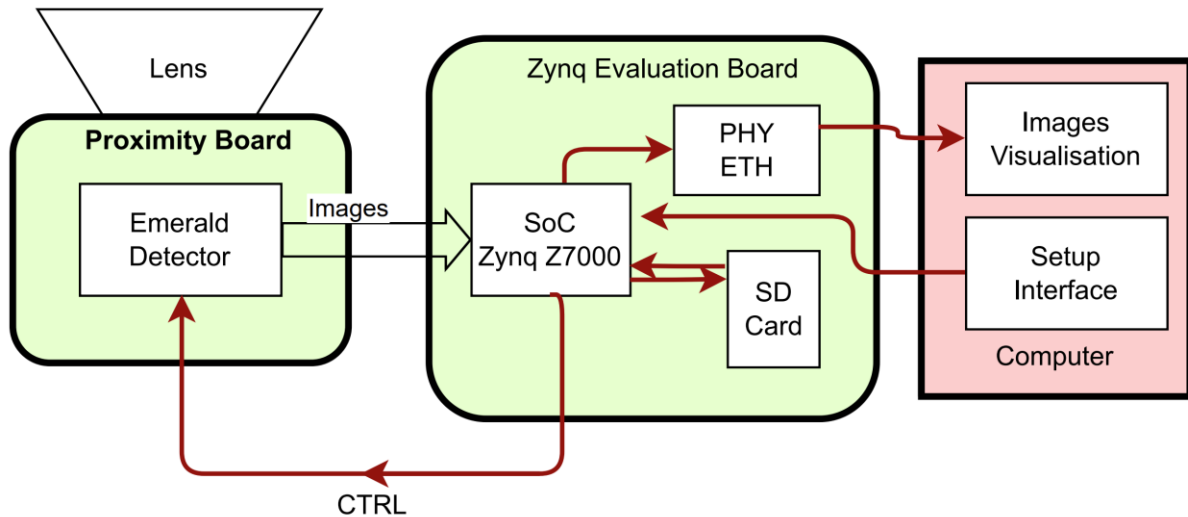
Paramétrisation capteur optique au sol :

- ◀ Campagne de tests du nano satellite
- ◀ Nécessité de modifier de nombreux paramètres
- ◀ Transfert d'image de la carte vers un PC

Choix de la plateforme :

- ◀ Zuno-7000 (ARM Cortex M9 + FPGA)
- ◀ Ethernet entre Carte et PC

Architecture globale du système



2. Méthodes utilisées

2.1 Implémentation Bare-Metal

Utilisation de lwIP

IP statique

ARP, ICMP (ping)

21	20.584268	Xilinx_01:02:03	Broadcast	ARP	60 ARP Announcement for 192.168.1.50
22	20.592261	Xilinx_01:02:03	Broadcast	ARP	60 Who has 192.168.1.100? Tell 192.168.1.50
23	20.592285	HP_ba:d3:f5	Xilinx_01:02:03	ARP	42 192.168.1.100 is at d0:ad:08:ba:d3:f5
24	20.796751	HP_ba:d3:f5	Broadcast	ARP	42 Who has 192.168.1.100? (ARP Probe)

Résultat :

ping fonctionnel

ARP visible sous Wireshark

Difficultés rencontrées :

initialisation EMAC

gestion des timeouts lwIP

2.2 Implementation Linux embarqué

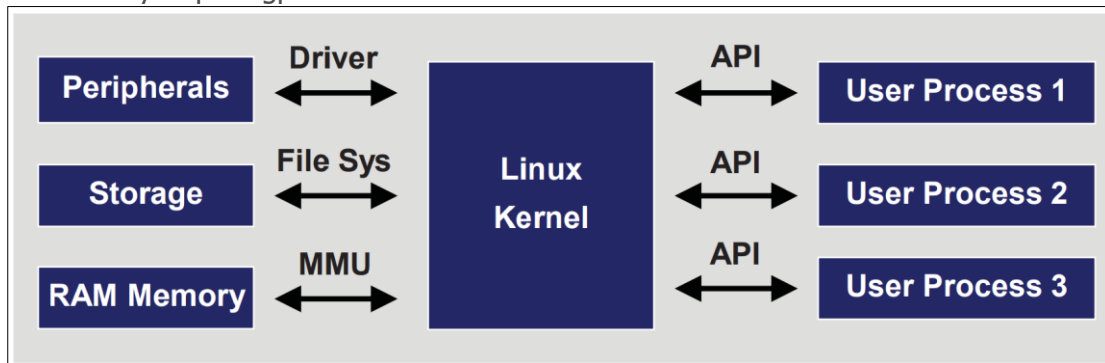
Pourquoi Linux ?

- Drivers
- Sockets réseau standard
- sysfs pour gpio



Validation rapide :

- contrôle LED via sysfs
- LED commandée depuis un PC via UDP



3. Résultats et analyses

3.1 Résultats obtenus

Transmission d'images par UDP

Problème :

MTU Ethernet = 1500 octets
UDP non fiable (ordre, perte)



Solution mise en place :

Découpage en chunks de 1400 octets
Création du protocole avec header

```
typedef struct {  
    uint16_t magic;  
    uint8_t  version;  
    uint8_t  flags;  
    uint32_t frame_id;  
    uint16_t chunk_id;  
    uint16_t total_chunks;  
    uint16_t payload_len;  
} UdpImgHdr;
```

Field	Description
<i>magic</i>	Signature to identify packets
<i>version</i>	Protocol version number
<i>flags</i>	Reserved (for future use)
<i>frame_id</i>	ID for the image/file being sent
<i>chunk_id</i>	Sequence number of this chunk
<i>total_chunks</i>	Total number of chunks for the file
<i>payload_len</i>	Bytes of actual image data in this packet



3.2 Analyse calcul chunks

```
int total_chunks = (int)((size + CHUNK_SIZE - 1) / CHUNK_SIZE);
```

Avec :

- total_chunks : nombre de morceaux à transmettre
- size : la taille du fichier à envoyer
- chunk_size : fixée à 1400 octets

En C la division était arrondie vers le bas. Ajout de $\text{CHUNK_SIZE} - 1$, force l'arrondi vers le haut.

3.2 Analyse trame UDP

```
// Construire packet = header + payload
memcpy(packet, &hdr, sizeof(hdr));
memcpy(packet + sizeof(hdr), payload, r);

// Envoyer paquet
sendto(sock, packet, sizeof(hdr) + r, 0,
       (struct sockaddr *)&dst, sizeof(dst));

usleep(1000); // 1 ms
```

sock : la socket UDP

packet : le buffer complet qu'on vient de construire

sizeof(hdr) + r : la taille à envoyer = header + données

dst : l'adresse IP + port du récepteur

4. Conclusion

Conclusion

Bilan

communication Ethernet fonctionnelle
transmission d'images validée
architecture robuste

Limites actuelles :

pas encore connecté au vrai capteur

Perspectives :

Communication SPI avec le capteur optique

Questions ?

<https://github.com/camillelefort1-creator/zynq-udp-linux-embedded>

-
- ◀ A. Dunkels, Design and Implementation of the lwIP TCP/IP Stack. Swedish Institute of Computer Science, Tech. Report, 2001.
 - ◀ Xilinx Inc., Zynq-7000 SoC Technical Reference Manual, UG585(v1.11.1), 2018.
 - ◀ Xilinx Inc., PetaLinux 2014.2 Board Bring-up Guide, UG980 (v2014.2), 2014.
 - ◀ D.C. Plummer, “An Ethernet Address Resolution Protocol,” IETF RFC826, Nov. 1982
-