

« SCCA »

Station de **C**ommande et de **C**ontrôle
Autonome



Besoins clients / métiers :



- **Télémétrie et automatisation** : gestion d'équipement ou machine, compteurs intelligents, réseau électrique « Smartgrid », supervision d'équipements, panneaux solaires, VMC, climatisation



- **Infrastructure publique ou de collectivités** : traitement de l'eau, stockage réservoir, horodatage, affichage et signalisation route, points d'information public accident, télépéage



- **Sécurité et surveillance** : monitoring de l'environnement, détection d'intrusion, télésurveillance



- **Logistique et traçabilité** : gestion de flotte et de localisation, géo positionnement

Problématique et spécificités des sites isolés :



- Absence de source d'énergie secteur stable ou secteur non secouru
- Environnement climatique difficile
- Absence d'opérateur humain qualifié
- Problème géographique (rivière, montagne, zone rurale)
- Pas d'accès aisé à un réseau cellulaire ou signal réseau faible
- Process de maintenance difficile
- Sites sensibles, autorisation d'accès

Fonctionnalités matériel requises :



- Faible consommation électrique, alimentation par sources d'énergies renouvelables et réserve sur batterie

- Connectivité télécom multiples à prévoir pour dialogue distant

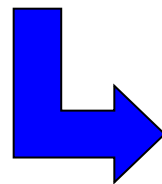


- Equipement en grade industriel et support température à plage élargie

- Matériel sans ventilateur/ sans air pulsé car MTBF faible

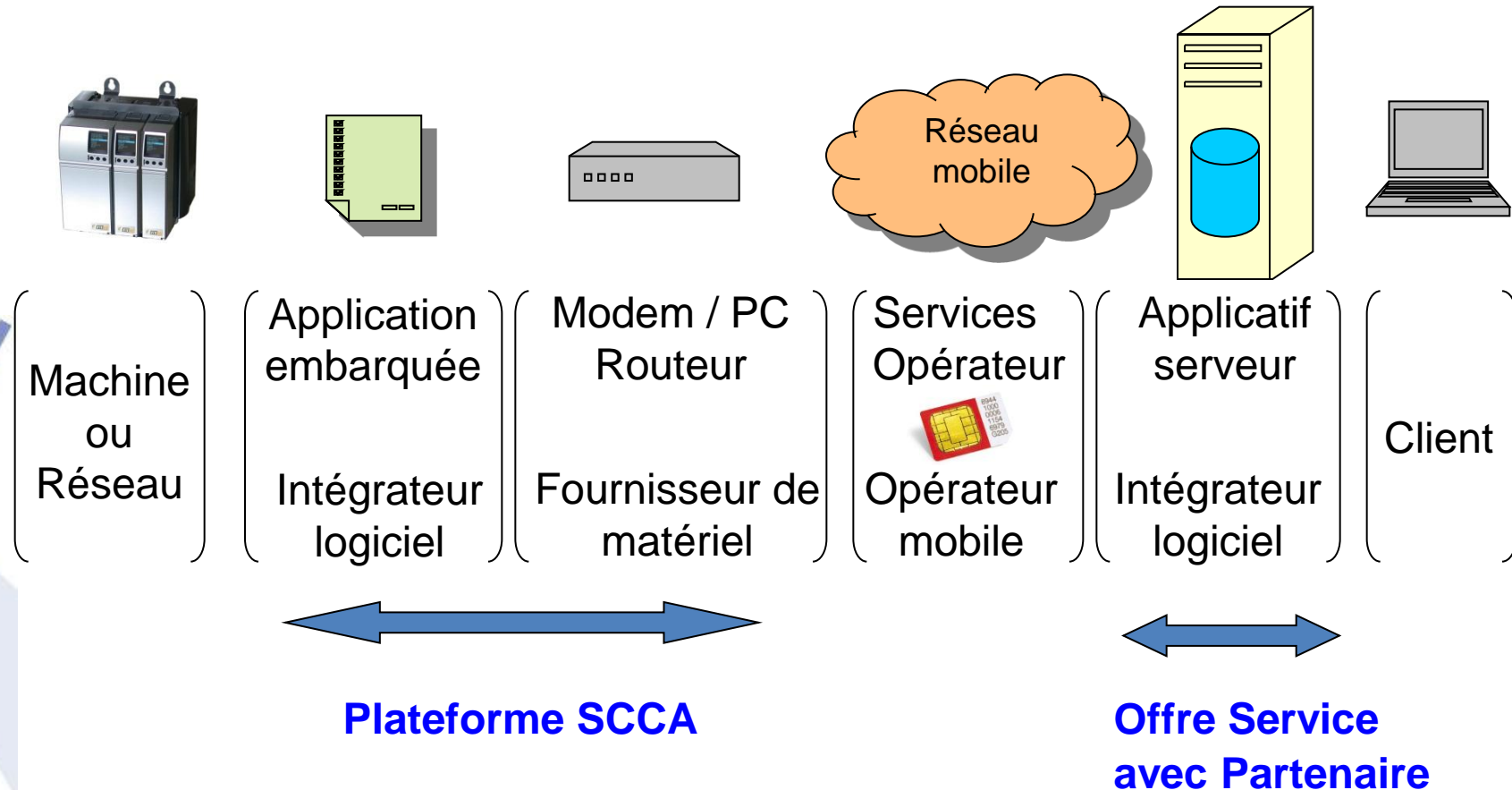


- Entrées/sorties pour commande, tout ou rien, relais, RS232, Ethernet



Notre réponse : le SCCA

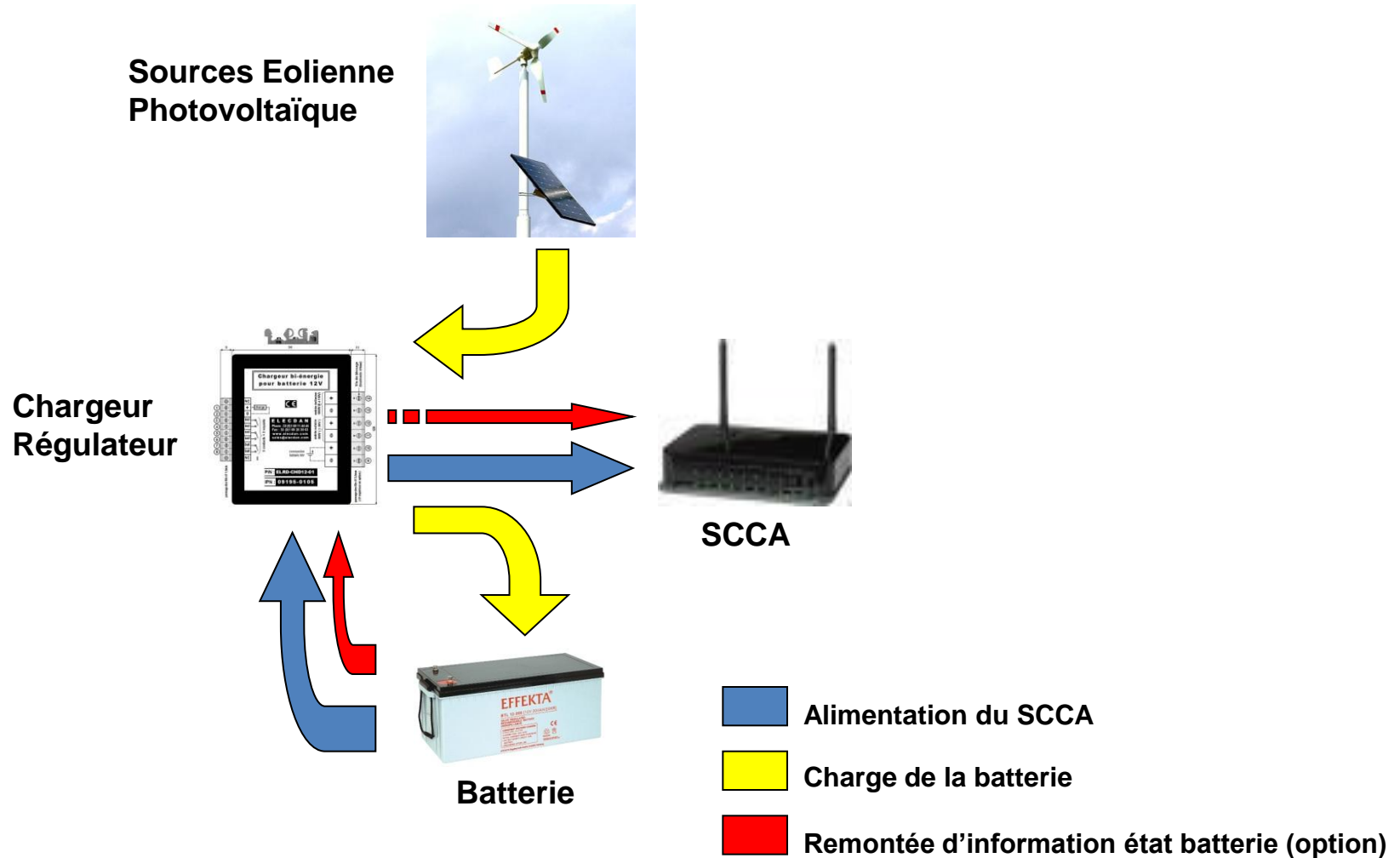
Positionnement de notre offre :



Gestion avancée de l'alimentation :

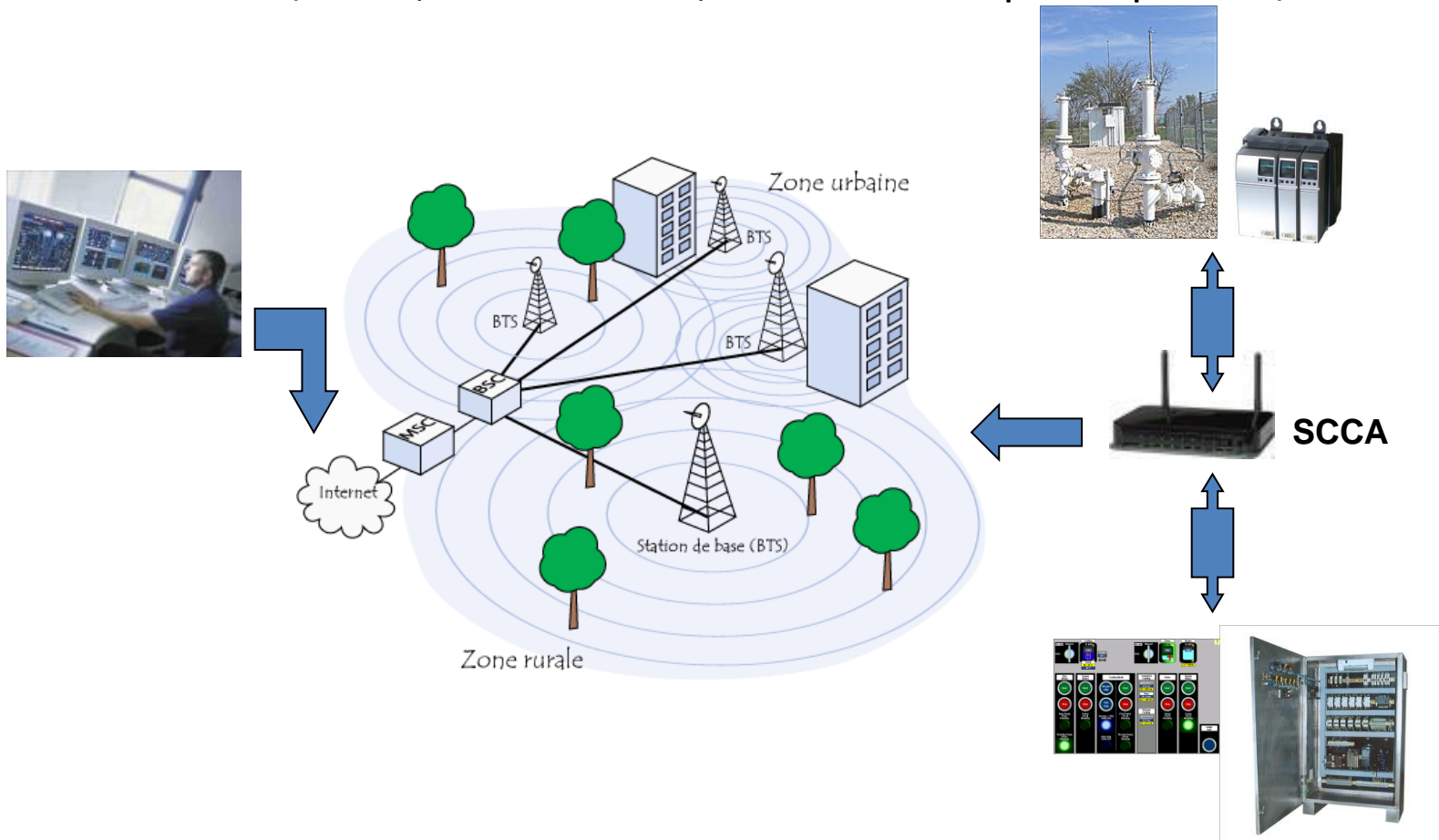
- Large plage d'alimentation : de 7 à 27 VDC, **couplage** avec nos régulateurs/chargeurs intelligents **optimisés bi-énergie photovoltaïques et éolien avec courbe de réponse adaptée**
- Basse consommation: **3W nominal**
- **Mode veille** : permettant d'éteindre le module CPU en cas d'inactivité; pour le réveiller : appel d'un des vecteurs de communication ou activité sur le port série.

Gestion avancée de l'alimentation :



Communication Machine to Machine/ M2M/ plusieurs vecteurs de communication :

- Interconnexions et communication entre machines au travers de réseaux GSM/GPRS, Radio sans fil, ou réseau téléphonique RTC/PSTN

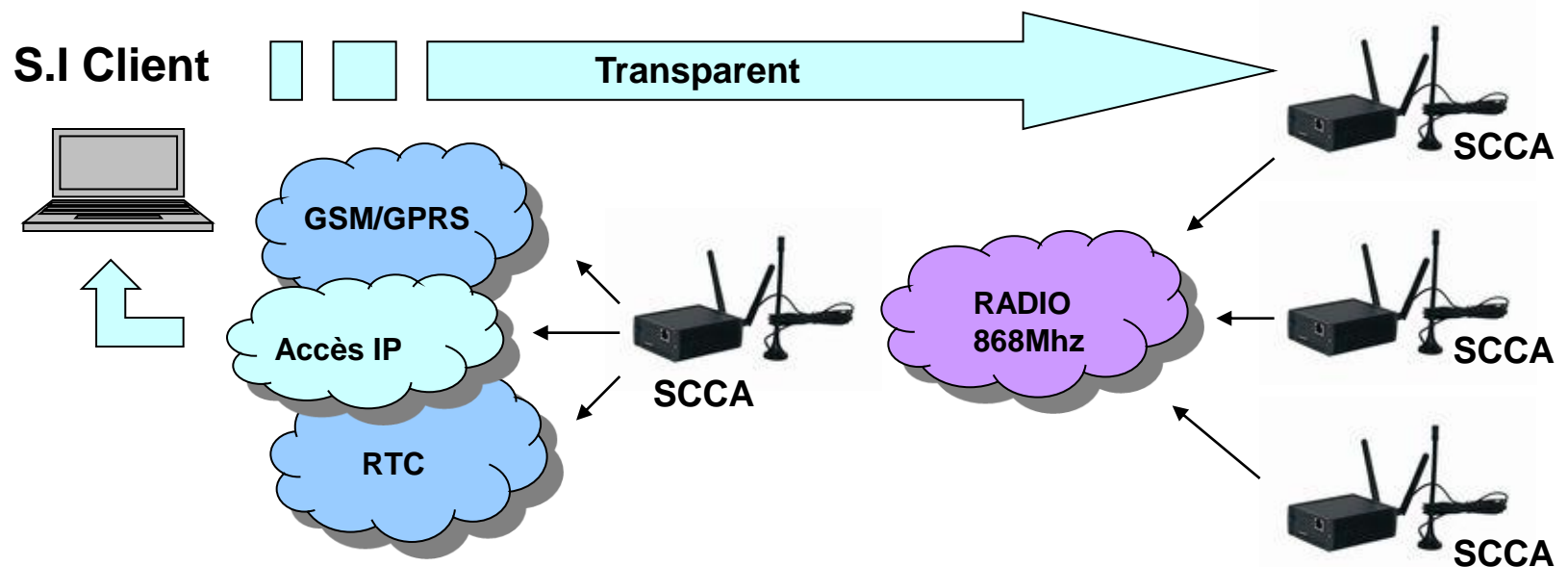


SCCA = 4 vecteurs de communication

- **RTC** (Réseau Téléphonique Commuté)
 - Modem embarqué pour un média très répandu
- **GSM/GPRS ou GPS (option)**
 - Modem embarqué offrant l'accès au réseau mobile data pour une plus grande flexibilité
- **Radio 868 MHz ou 802.15.4 Zigbee (option)**
 - Modem embarqué avec une portée de 40 Km (suivant version module) et des fonctionnalités d'adressage permettant de communiquer avec plusieurs équipements indépendamment
- **Ethernet 10/100 Mbit/s**
 - Switch 4 ports embarqué pour une capacité de communication avec tout équipement IP à proximité.

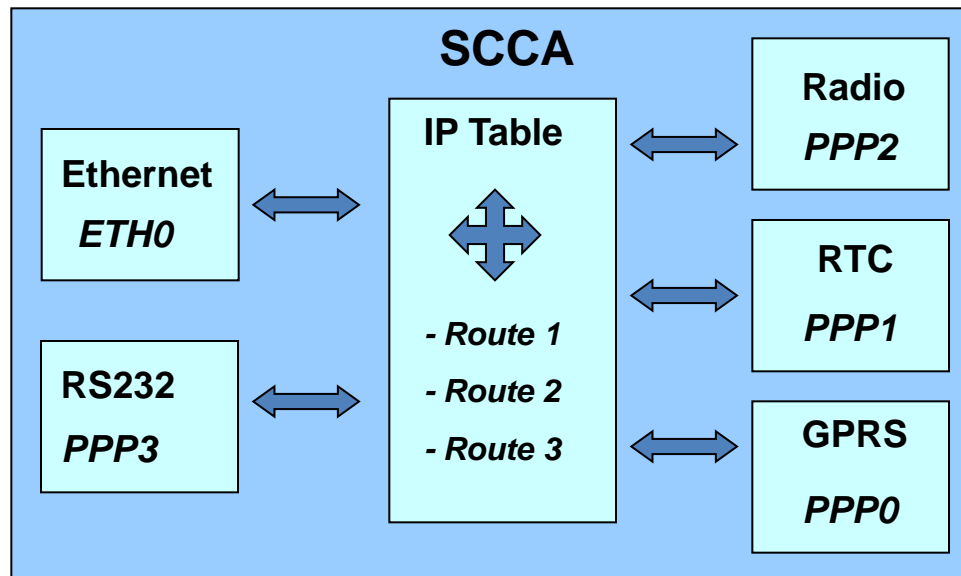
Scénario de communication : Aboutement de média

- 1 SCCA de concentration
- 3 SCCA esclaves aboutés par communication radio



Gestion des flux et interfaces

- Accès par Telnet, SSH, Console série
- Support HTTP, FTP
- Vecteurs de communication = Interfaces au niveau IP
- Gestion des flux de données entre interface: IP Table = Routage entre interface
- Mode Masquerade et Natting (translation d'adresse IP)



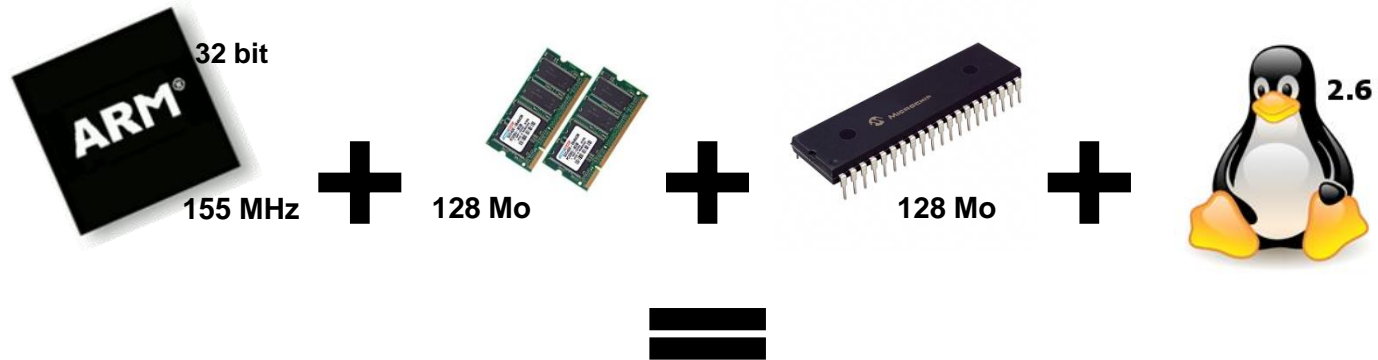
Fonctionnalité de contrôle et de commande :

- **8 entrées ToR** : pour contrôler l'état de paramètres ou d'équipements environnants
- **8 relais** : pour commander/actionner des équipements
 - Possibilité de réagir par rapport à un paramètre détecté sur l'un des ports ToR en agissant sur les relais
 - Possibilité de mettre en marche un équipement pour utiliser ces fonctionnalités à un instant défini (exemple : mise sous tension d'une caméra IP pour récupérer une image donnée puis extinction)

Intelligence et sécurité embarquées :

- Processeur ARM 9 cadencé à 150 MHz
 - Souplesse de fonctionnement
- 64 Mo SDRAM et 128 Mo de Flash
 - Capacité de stockage et de traitement accrue
- Noyau Linux embarqué 
 - Développement simplifié et rationalisé
- WatchDog matériel 
 - Contrôle permanent du fonctionnement
- Puce de cryptographie matérielle 3DES intégrée
 - Sécurisation des échanges

Système d'exploitation Linux



Un système d'exploitation ouvert et répandu sur une plateforme robuste permet d'exploiter tous les outils provenant de l'écosystème Linux sur le serveur d'environnement sans avoir à redévelopper les fonctionnalités

API dédiées

Librairie Système fournie



void SystemReset(void)

Description: Reset the whole system
Input: none
Return: none

int ComponentReset(int module)

Description: Reset a module in the system
Input: module index
0: GSM, RTC
1: Switch
Return: error code

void SystemShutDown(void)

Description: Shut down the system
Input: none
Return: none

int ComponentShutDown(int module)

Description: Shut down a module in the system
Input: module index
0: GSM
1: Switch
Return: error code

int GetComponentStatus(int module)

Description: Get module's status
Input: module index
0: GSM
1: Switch
Return: status code

int EnableGSM(int onoff)

Description: Enable or disable the GSM module
Input: GSM status
0: Disable
1: Enable
Return: error code

int EnableSwitch(int onoff)

Description: Enable or disable the Switch module
Input: Switch status
0: Disable
1: Enable
Return: error code

int Sleep()

Description: Put in low power state
Input: --
Return: error code



```
#include <stdio.h>
int main(void)
{
    printf("Hello World!\n");
    return 0;
}
```

C/C++

API dédiées

Librairie Périphériques fournie



int SetDCMode(int mode, int block)

Description: Set dry contact mode

Input: mode: dry contact mode

- 0. Output
- 1. Isolation
- 2. Input

block: dry contact block number

Return: error code

int GetDCMode(int block)

Description: Get dry contact mode

Input: block: dry contact block number

Return: mode code

int SetDCOutput(int val, int block)

Description: Set output mode dry contact value

Input: val: configuration value (0/1)

block: dry contact block number

Return: input value

int GetDCInput(int block)

Description: Get input/isolate mode dry contact value

Input: block: dry contact block number

Return: input value



```
#include <stdio.h>
int main(void)
{
    printf("Hello World!\n");
    return 0;
}
```

C/C++

API dédiées

Librairie Communication fournie



int SetCodePin(char * pin)

Description: Enter pin code to enable GSM

Input: pin code, a four digital number

Return: error code

int SetCodePuk(char * puk, char *npin)

Description: Set puk code

Input: puk code, a 16 digital number

npin: new pin code, a four digital numbers

Return: error code

int SendSMS(char *num, char*msg)

Description: Send SMS message to a destination

Input: num: destination telephone number

msg: SMS message

Return: error code

char * ReadSMS(int id)

Description: Read SMS in the phone

Input: message index

Return: message

int GetGSMStatus(void)

Description: Read GSM status

Input: none

Return: status code

int GetGPRSStatus(void)

Description: Read GPRS status

Input: none

Return: status code

int GetSimStatus(void)

Description: Read SIM card status

Input: none

Return: status code

char * EnableGPRS()

Description: Connect to the Internet

Input: none

Return: if it's connected, IP address will be returned

int DisableGPRS()

Description: Disconnect the phone with internet.

Input: none

Return: error code

int EnableEth0(int onoff)

Description: Enable/Disable the eth0 interface

Input: eth0 status

0: disable

1: enable

Return: error code

int EnableGPRS_PPP(int onoff)

Description: Enable/Disable gprs ppp interface

Input: ppp status

0: disable

1: enable

Return: error code

int EnableRTC_PPP(int onoff)

Description: Enable/Disable modem ppp interface

Input: ppp status

0: disable

1: enable

Return: error code



Int GetPPP_Status(int num)

Description: get ppp interface status

Input: ppp interface number (0-1)

Return:

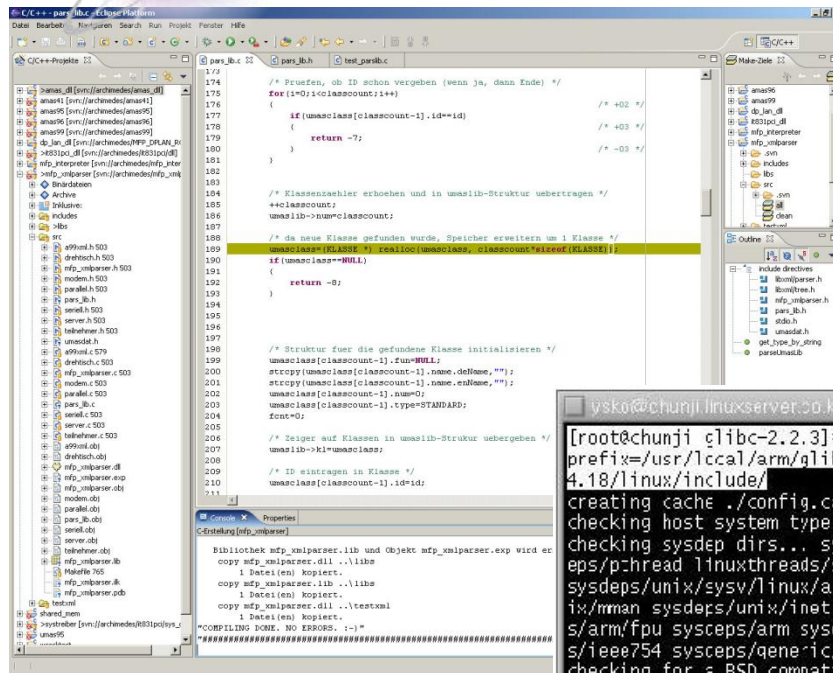
0: disable

1: enable

```
#include <stdio.h>
int main(void)
{
    printf("Hello World!\n");
    return 0;
}
```

C/C++

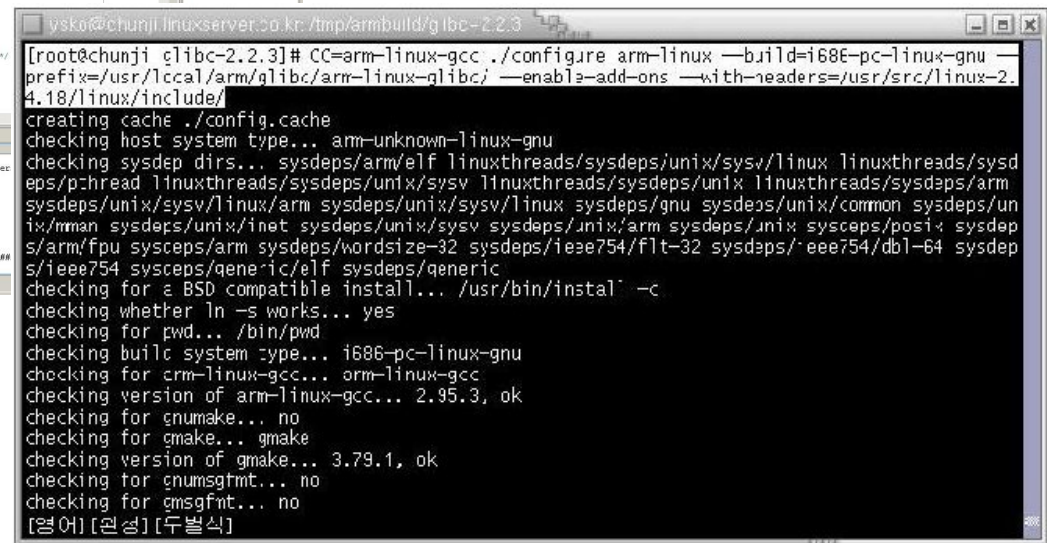
Développement et portage de toute application métier



Linux/ARM



GDB



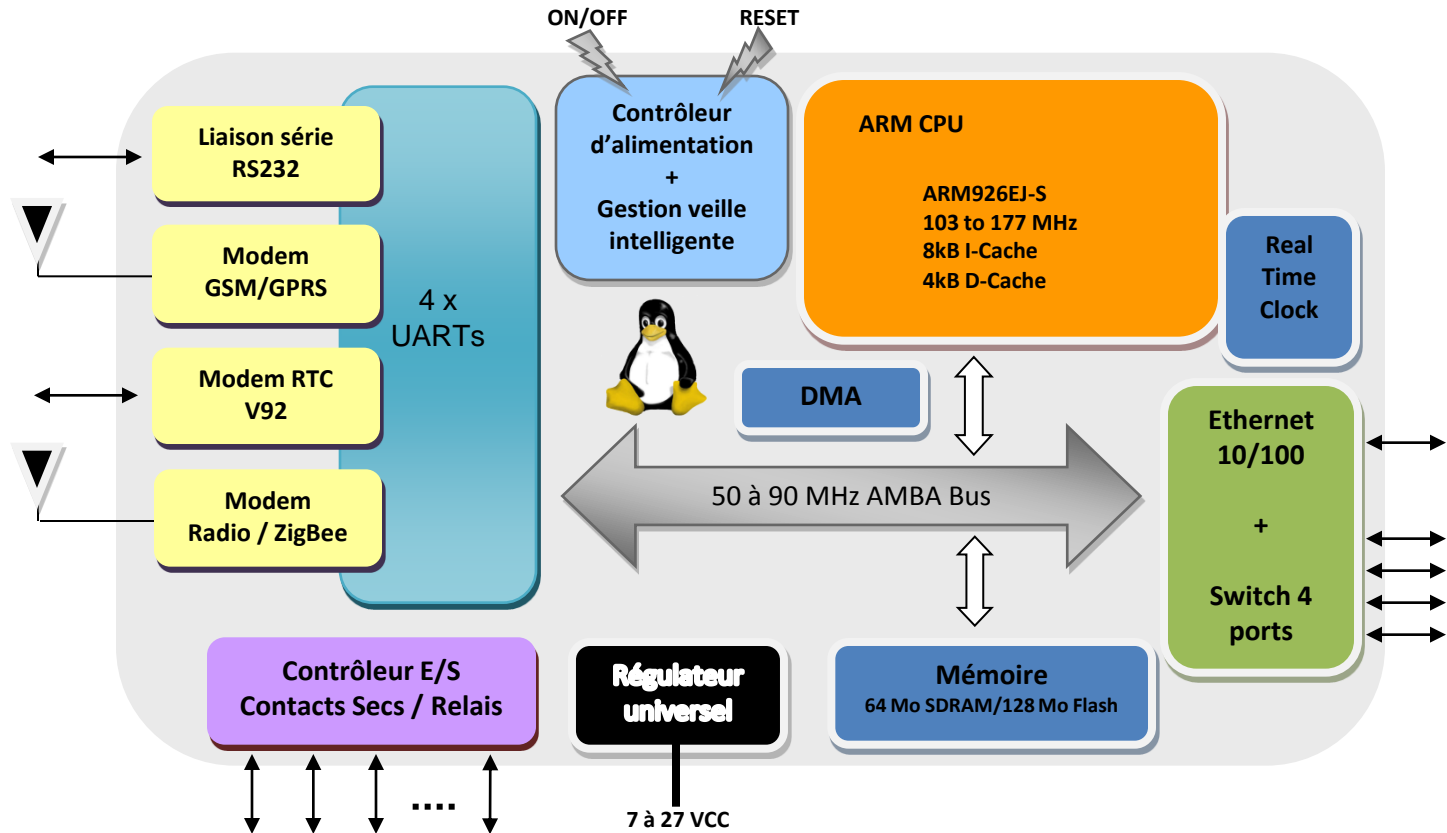
Studio de développement
Eclipse dédié ou en ligne
de commande arm-linux-
gcc.
Débogage avec GDB

Construction robuste, IP 40 minimum :

- Boîtier métallique
- Dissipation thermique passive : pas de ventilateur
- Niveau de protection allant de l'IP40 à l'IP68 à la demande
- Méthodes de fixation multiples (rail Din, paroi, mât, baie 19",...) à la demande
- Possibilité de design et packaging propre à projet client OEM / ajout logo client ou Adaptation.



Architecture ouverte et modulable...



... et évolutive pour le futur.

