UNIVERSITÀ POLITECNICA DELLE MARCHE
FACOLTÀ DI INGEGNERIA

Progettazione di una On Board Unit per la comunicazione interveicolare

Relatore:
Prof. Aldo Franco Dragoni

Tesi di:
Riccardo Pelliccia

Correlatori:
Dott. Paolo Pagano

Prof. Massimo Conti

Anno Accademico 2009/2010

Indice

1 Introduzione **1**
1.1 Scenari applicativi . . . 2
1.1.1 Previsione della collisione . . . 3
1.1.2 Avvertimento pre-collisione . . . 4
1.1.3 Segnalazione di zone pericolose . . . 5
1.1.4 Gestione del traffico . . . 5
1.1.5 DSRCs Dedicated Short Range Comunications . . . 6
1.1.5.1 Canale radio per DSRC-ITS . . . 7
1.1.5.2 Protocollo per DSRC . . . 8
1.1.5.3 Funzioni del livello MAC . . . 10
1.1.5.4 Funzioni del livello fisico . . . 13
1.1.6 Conclusioni . . . 13
1.2 Intelligent Trasportation System . . . 14
1.3 On Board Unit (OBU) . . . 16
1.3.1 Il protocollo CAN BUS . . . 17
1.3.1.1 Struttura dei messaggi . . . 18
1.3.1.2 Rilevazione e segnalazione degli errori . . . 20
1.3.1.3 Livello fisico . . . 21
1.4 Related Work . . . 24

2 Architettura del sistema **26**
2.1 Specifiche di Progetto . . . 26
2.2 CPU . . . 27
2.2.1 Microcontrollore . . . 27
2.2.2 ARM . . . 28
2.2.3 MCU vs ARM . . . 29
2.2.4 Analisi dei costi . . . 32

3 OBU su architettura ARM **33**
3.1 Wireless adapter . . . 33
3.2 CAN Bus . . . 36
3.3 Storage . . . 38
3.4 Piattaforma ARM . . . 39
3.5 Schema elettronico . . . 40

3.5.1 Stadio Alimentazione 40
3.5.2 Interfacce 42
3.5.3 Processore 46

4 Piattaforma Software **48**
4.1 Sistema Operativo 48
4.2 OpenWRT 49
4.2.1 Configurazione del sistema e dei pacchetti 49
4.2.2 Compilazione 50
4.3 Cross-compilazione dei programmi 54
4.4 Esempi di applicazioni 54

5 Conclusioni **57**

Capitolo 1

Introduzione

L'incremento di motorizzazione, urbanizzazione e crescita demografica ha come ripercussione un costante aumento del traffico stradale. In Italia nelle maggiori città, il problema del traffico, soprattutto nei centri cittadini, è una realtà di tutti i giorni. In questa ottica il nostro paese detiene il record del numero di auto per abitante (60 ogni 100 abitanti). Questo numero di auto mette a dura prova la nostra rete stradale ed autostradale; provocando a volte disagi dovuti a congestioni del traffico, perlopiù dovuto ad incidenti, con conseguenti code spesso inevitabili. E' cosa nota che gli incidenti sono causati principalmente da comportamenti scorretti da parte dei conducenti delle vetture ed in minore misura da altri fattori come avarie meccaniche o lo stato delle infrastrutture viarie. Una ventina d'anni fa si iniziò ad immaginare una soluzione per far fronte al problema. Nel 1991 il congresso degli Stati Uniti d'America nell'Intermodal Surface Transportation Efficiency Act (ISTEA), creò un programma chiamato Intelligent Vehicle Highway Systems (IVHS), chiamati al giorno d'oggi ITS (Intelligent Trasportation System) i cui obbiettivi principali erano quelli di aumentare la sicurezza stradale, evitare le congestioni del traffico e ridurre l'inquinamento per salvaguardare i combustibili fossili, destinati all'esaurimento. Fin dall'inizio, venne intuito che le comunicazioni wireless erano il fondamento su cui si potevano basare la maggior parte dei servizi di tipo ITS. All'epoca le poche applicazioni esistenti, come ad esempio il pagamento dei pedaggi automatizzato (Telepass), utilizzavano uno spettro tra 902MHz e 928MHz; purtroppo questa porzione di spettro

era troppo stretta e affollata per poterla sfruttare con servizi di comunicazioni ITS. Di conseguenza nel 1997 i vari enti interessati allo sviluppo fecero una petizione alla Federal Communication Commission (FCC) per ottenere 75MHz di banda allocati attorno a 5.9GHz con l'obbiettivo di utilizzare questo spettro per le Dedicated Short Range Communication (DSRC) negli ITS. La FCC accettò la richiesta nell'ottobre del 1999 assegnando 75MHz di banda tra 5.85GHz e 5.925GHz per DSRC-ITS. Nel luglio del 2002, i vari enti influenzarono la commissione dell'FCC per quanto riguardavano le licenze, le norme e le possibili applicazioni dei servizi che potevano fornire le comunicazioni DSRC-ITS. Venne consigliato l'adozione di uno standard unico per quanto riguardava il livello fisico (PHY) e il livello MAC e si propose uno sviluppo in collaborazione con la American Society for Testing and Materials (ASTM) sul protocollo IEEE 802.11. La FCC accettò ufficialmente la proposta negli anni tra il 2003-2004. Nel 2004 un gruppo di ricercatori di IEEE prese in mano il progetto iniziato e cominciò a sviluppare una versione dello standard 802.11 contenente nozioni sulle applicazioni veicolari. Il documento è conosciuto come 802.11p. Un altro gruppo di ricercatori di IEEE (conosciuto come gruppo 1609) era impegnato a sviluppare specifiche per completare anche gli altri livelli del protocollo. Oggi lo standard IEEE1609 è composto da quattro documenti IEEE 1609.1, IEEE 1609.2, IEEE 1609.3 e IEEE 1609.4. Nel complesso i protocolli: IEEE 802.11p e IEEE 1609.1-2-3-4 che ricoprono l'ambito delle comunicazioni interveicolari (o VANETs Vehicular AdHoc NETworks), sono conosciuti come Wireless Access in Vehicular Environments (WAVE).

1.1 Scenari applicativi

L'implementazione di un sistema di comunicazione interveicolare richiede che sia nei veicoli (siano essi automobili o mezzi di trasporto pubblici), sia lungo le strade, siano montati dei dispositivi necessari alla comunicazione. Le prime chiamate On Board Unit (OBU), sono responsabili delle comunicazioni che avvengono tra due o più veicoli, Vehicle-to-Vehicle communication (V2V) e delle comunicazioni con le stazioni radio ai lati della strada (RSU, Road Side Unit), Vehicle-to-Infrastructure communication

(V2I). Queste unità devono essere dotate di almeno un dispositivo di rete per le comunicazioni a corto raggio DSRC che viene utilizzato per inviare, ricevere e inoltrare informazioni di sicurezza ad altri veicoli. Le Road Side Unit possono essere installate in un qualsiasi posto utile come ristoranti, distributori di carburante, semafori. Sono dotate di un dispositivo di rete per supportare comunicazioni DSRC basato sullo standard 802.11p e di altri dispositivi per la comunicazione con la rete di infrastrutture (Ethernet, IEEE 802.11a/b/g/n). Fra le principali funzioni di una RSU possono essere identificate:

- estendere la comunicazione V2V inoltrando i dati ad altri veicoli che sopraggiungono;
- inoltrare a tutti i veicoli nel range di copertura informazioni di sicurezza;
- cooperare con le altre RSU per scambio di informazioni di sicurezza;

Andremo ora ad analizzare le principali applicazioni che portano all'utilizzo di questi sistemi a partire dal problema della sicurezza attiva, per passare infine all'utilizzo nella gestione del traffico.

1.1.1 Previsione della collisione

Le cause tipiche di tamponamenti tra veicoli sono la distrazione o la frenata improvvisa del veicolo che ci precede. Questi inconvenienti possono essere evitati quando siamo in presenza di veicoli che durante la marcia comunicano tra loro scambiandosi informazioni sulla propria posizione, velocità e direzione. Ogni veicolo tiene monitorato il comportamento del proprio conducente, la propria posizione e il comportamento di tutti i veicoli nelle vicinanze. Quando il veicolo rileva che ci potrebbe essere un pericolo imminente avverte il guidatore con un segnale visivo e uditivo così il conducente avrà abbastanza tempo per poter intervenire. In aggiunta a questo sistema che prevede comunicazioni wireless possiamo affiancare l'utilizzo di sensori per identificare veicoli che non sono dotati di sistemi di comunicazioni interveicolari.

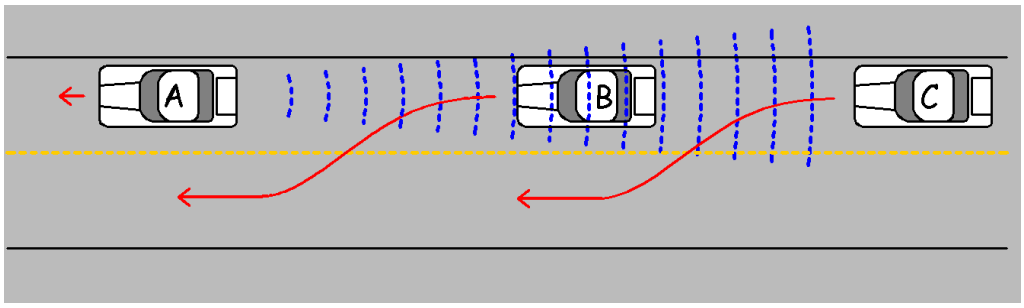


Figura 1.1: Il veicolo A procede lentamente e segnala la sua presenza in broadcast; i conducenti dei veicoli B e C che sopraggiungo velocemente sono avvertiti, tempestivamente, della presenza del veicolo A che verrà sorpassato con una rapida manovra

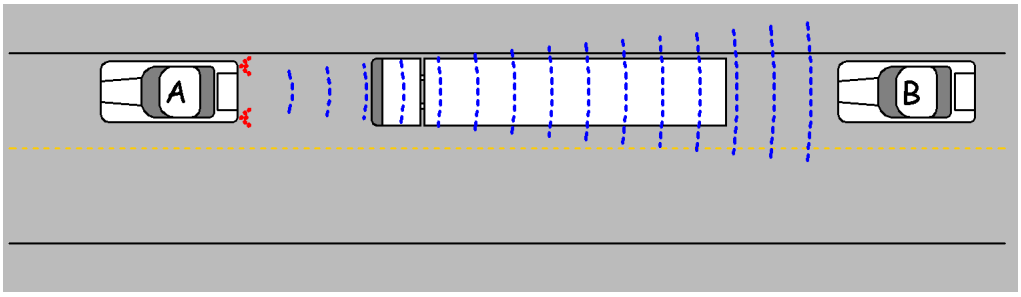


Figura 1.2: Il veicolo A frena bruscamente e trasmette l'evento in broadcast; il conducente del veicolo B viene avvertito dell'evento anche se a la visuale ostruita

1.1.2 Avvertimento pre-collisione

L'avvertimento pre-crash è un altro possibile utilizzo delle comunicazioni veicolari; serve per ottimizzare l'utilizzo dei sistemi di protezione dei passeggeri in caso di collisione imminente e inevitabile. Quando uno o più veicoli si trovano in una situazione in cui la prevenzione non è sufficiente ad evitare lo scontro si passa ad una fase successiva di utilizzo ottimale delle risorse di sicurezza passiva presenti nel veicolo. I mezzi che stanno per essere coinvolti nello essere coinvolti nello scontro scambiano informazioni in modo veloce e affidabile sulla posizione del veicolo rispetto agli altri e sulle sue dimensioni. Questo scambio di informazioni supplementari permette di utilizzare al meglio air bag, pre-tensionatori delle cinture di sicurezza e paraurti allungabili.

1.1.3 Segnalazione di zone pericolose

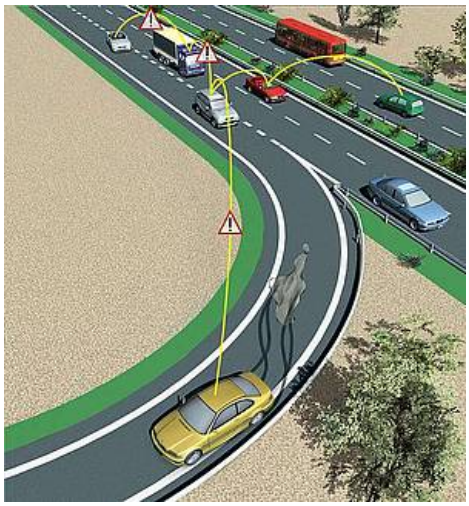


Figura 1.3: Esempio di segnalazione di zona pericolosa

Le notifiche che avvertono la presenza di elementi pericolosi sulla strada sfrutta la rete formata dai veicoli presenti nei dintorni. I veicoli scambiano informazioni utili tra loro come la presenza di manto stradale scivoloso, o di qualche ostacolo. Questo scambio di messaggi tra veicoli è reso possibile grazie all'utilizzo di sensori già esistenti su molti veicoli, come l'Elettronic Stability Program meglio conosciuto come ESP. Tutti i veicoli che ricevono le informazioni possono avvertire il proprio conducente o ottimizzare automaticamente l'assetto del veicolo. Le informazioni utili possono essere inoltrate di veicolo in veicolo per avvertire anche quelli che sopraggiungeranno nel tratto di strada pericoloso. Per poter fruire dei sensori di cui sono dotati i veicoli però è necessario che l'OBU implementi anche il protocollo di comunicazione con cui tutti i sensori interni al veicolo: il protocollo CAN (Controller Area Network).

1.1.4 Gestione del traffico

Gli utilizzi delle comunicazioni veicolari per la gestione del traffico hanno lo scopo di migliorare l'efficienza della rete di trasposto automobilistica con il duplice vantaggio di fornire informazioni utili sia ai veicoli che vi transitano che ai proprietari della rete (ad esempio nelle autostrade). Un sistema di trasporto più efficiente riduce i ritardi negli spostamenti ed evita congestioni del traffico grazie al costante scambio V2V e V2I. Tutti questi dati raccolti (che possono interessare anche una vasta area geografica) vengono poi elaborati e ritrasmessi dalle RSUs poste ai bordi delle carreggiate verso le OBUs informando il conducente sulle condizioni del traffico e sugli eventuali percorsi alternativi da seguire. Un'altra applicazione molto utile per gestire il traffico

in città consiste nell'utilizzare questo tipo di comunicazioni per informare i conducenti dei veicoli della velocità ottimale da tenere per avere un'alta probabilità che il semaforo che si trova lungo percorso sia verde al nostro attraversamento. Questo ha un'immediata conseguenza nel favorire lo smaltimento del traffico e ridurre il più possibile il consumo di ogni veicolo.

1.1.5 DSRCs Dedicated Short Range Comunications

Sono comunicazioni a corto o medio raggio, studiate appositamente per l'utilizzo in ambito trasportistico; tuttora ne esistono due tipologie discriminate della frequenza e dal tipo di funzionamento:

- la prima con frequenza di funzionamento a 915 MHz è utilizzata per la comunicazione del veicolo con reti infrastrutturali: ha un range limitato (circa 30m), un data rate basso (0.5 Mbps) ed inoltre dispone di un solo canale di comunicazione;

  Il sistema era stato inizialmente studiato per sistemi elettronici di pagamento siano essi parcheggi o pedaggi autostradali (del genere del Telepass); per poi poter essere utilizzato in altre applicazioni del genere: dalle lanterne semaforiche 'dinamiche' le quali modificano i tempi di fermata in funzione del numero di veicoli in coda. I problemi sorgono per l'elevato costo della rete fissa di un hardware 'customizato' anch'esso costoso il quale non ovvia hai problemi ci corto range e bassa velocità di trasmissione.

- la seconda, con una frequenza centrale di funzionamento di 5.9 GHz (e ben 7 canali di comunicazione separati tra di loro da 75MHz), può essere utilizzata oltre che per la comunicazione dei veicoli con reti infrastrutturali (siano esse RSU o MAN), con per quella tra due o più veicoli; hanno un range di 1Km e un data rate che può arrivare a 27 Mbps. L'hardware utilizzato per i dispositivi di bordo e per la rete fissa è general purpose quindi economico .

Queste ultima tipologia di reti illustrate sono quelle sulle quale baseremo lo sviluppo della nostra On Board Unit e che vengono anche chiamate DRSC-ITS.

1.1.5.1 Canale radio per DSRC-ITS

Le applicazioni appena descritte utilizzano lo spettro DSRC che va dai 5.85GHz ai 5.925GHz con centro banda a 5.9GHz.

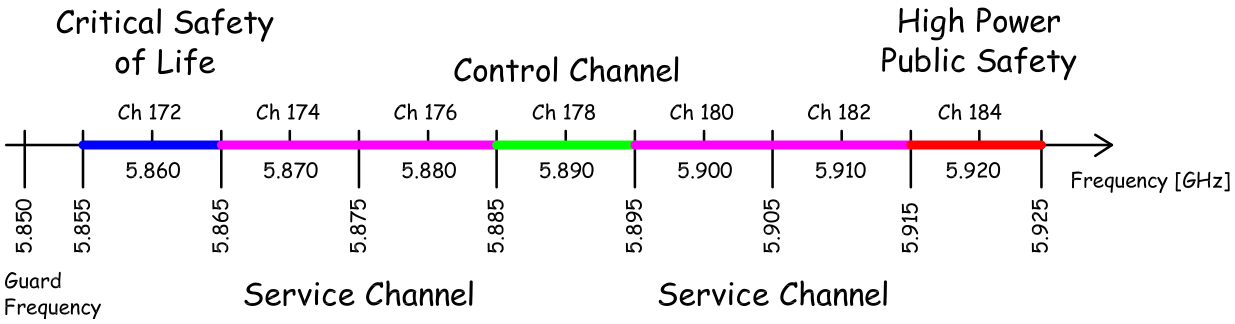


Figura 1.4: Spettro di frequenza DSRC

Questi 75MHz sono suddivisi in sette sottocanali da 10MHz l'uno. Il canale 178 (5.885 - 5.895 GHz) è il Control Channel (CCH) riservato esclusivamente alle comunicazioni di sicurezza. Il CCH è l'unico canale condiviso tra tutti i devices WAVE ed è utilizzato per il controllo delle comunicazioni. I canali ai margini dello spettro sono riservati per future applicazione che sono: la prevenzione avanzata degli incidenti (Ch 172) e per utilizzi di pubblica sicurezza (Ch 184). I canali restanti denominati Service Channel (SCHs) i quali possono essser utilizzati per le restanti applicazioni oltre che per le normali comunicazioni. La banda DSRC-ITS è con licenza e non deve essere confusa con la banda senza licenze a 900MHz, 2.4GHz e 5GHz. Queste bande libere da licenze hanno reso possibile lo sviluppo di sistemi come WiFi e Bluetooth. La banda DSRC rispetto alle altre prive di licenza è regolata da diverse norme come l'utilizzo di un determinato standard. In Europa si stanno accordando per allocare 30MHz di banda alla frequenza di 5GHz per le comunicazioni interveicolari, come è già stato fatto negli USA.

1.1.5.2 Protocollo per DSRC

Lo standard per DSRC è essenzialmente un aggiustamento ad hoc dello standard 802.11 per supportare le comunicazioni tra veicoli in movimento, con una velocità di punta di 200 Km/h e un raggio di comunicazione fino a 1.000 metri. Infatti le attuali comunicazioni Wi-Fi, che fanno parte della stessa famiglia (802.11b/g/n), perdono di efficacia appena la velocità relativa tra i terminali supera i 10 Km/h. La scrittura del nuovo standard è anche finalizzato a ridurre il più possibile le dimensioni del pacchetto da inviare tramite il collegamento radio.

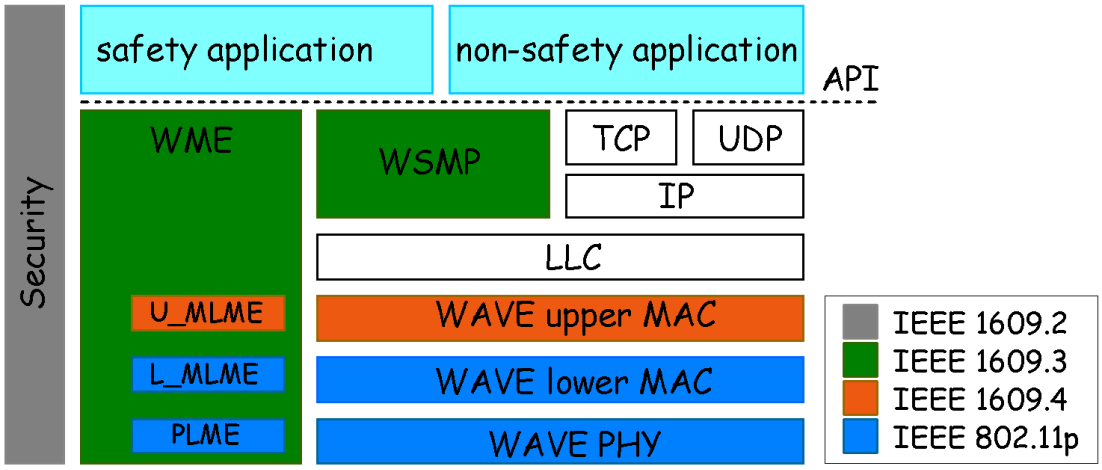


Figura 1.5: Stack del protocollo DSRC

Questo standard si occupa di descrivere le funzioni e i servizi richiesti per operare in rapidità scambiando messaggi senza avere a disposizione un Basic Service Set come avviene utilizzando lo standard IEEE802.11. Inoltre definisce le tecniche di segnalazione e le funzioni dello standard IEEE802.11 che sono controllate dal livello MAC. Una rete basata sullo standard IEEE 802.11 è chiamata Basic Service Set (BSS) e si compone di un certo numero di stazioni wireless e di un Access Point (AP). La pila protocollare che definisce standard per le comunicazioni interveicolari è composto da diversi protocolli oltre a 802.11p dei quali verrà fornita solo una breve descrizione.

IEEE1609.1 è lo standard che definisce:

- i componenti di WAVE
- le interfacce e le risorse
- i formati dei messaggi e di memorizzazione
- i dispositivi che equipaggiano gli OBU

IEEE1609.2 è lo standard che definisce:

- i formati sicuri di messaggio e la loro elaborazione
- le circostanze in cui usare gli scambi sicuri di messaggi

IEEE1609.3 è lo standard che definisce:

- i servizi offerti dagli strati di rete e di trasporto, inclusi indirizzamento e instradamento, in supporto agli scambi sicuri
- il WAVE Short Messages Protocol (WSMP), una efficiente alternativa ad IP che permette alle applicazioni di controllare canale e potenza di trasmissione

IEEE1609.4 è lo standard che definisce:

- gli arricchimenti al MAC 802.11 per operare in modalità WAVE, permettendo il coordinamento e la gestione dei diversi canali
- offre un collegamento logico agli strati protocollari superiori senza preoccuparsi di come avviene lo scambio di informazioni a livello fisico.

IEEE802.11p è lo standard che definisce:

- le funzioni e i servizi richiesti dalle stazioni WAVE per operare in un ambiente che varia spesso e rapidamente scambiando messaggi senza la necessità di creare una BSS
- le tecniche di segnalazione e le funzioni di interfaccia che sono controllate dal livello MAC

1.1.5.3 Funzioni del livello MAC

Come abbiamo detto nel paragrafo precedente lo standard IEEE 802.11p è basato sul già esistente e collaudato IEEE 802.11, con alcune modifiche per essere applicato alle comunicazioni veicolari. Nella strato protocollare MAC di 802.11 il principio di funzionamento si basa sullo scambio di informazioni preliminari per stabilire e mantenere una connessione sicura. L'idea proposta da IEEE 802.11p è quello di riuscire a creare una comunicazione molto efficiente senza il pesante overhead tipicamente presente in 802.11 MAC a vantaggio della velocità di trasmissione per l'utilizzo nelle reti veicolari. Per stabilire la connessione è necessario effettuare l'autenticazione e l'associazione; tutto questo comporta lo scambio di messaggi denominati Beacon. Una BSS (definizione a pagina precedente) è riconosciuta dall'utente grazie al Service Set IDentification (SSID) corrispondente al nome visualizzato quando andiamo a connettere il computer alla rete. Il SSID non deve essere confuso con il Basic SSID (BSSID) che corrisponde al nome con cui la rete è riconosciuta a livello MAC. Ogni BSS ha un unico BSSID a livello mondiale che, nelle reti basate sullo standard IEEE 802.11, è l'indirizzo MAC dell'AP. Nelle reti 'ad hoc' l'indirizzo IBSS viene fornito

dall'amministratore locale di quella rete, ad esempio da uno dei computer. Il filtraggio degli indirizzi BSSID a livello MAC è il metodo per riconoscere mittente e destinatario in ogni BSS. il formato di un pacchetto dati dello standard 802.11, che è composto da quattro campi di indirizzo, rispettivamente Source Address (SA), Destination Address (DA), Transmitting Station Address (TA), Receiving Station Address (RA). Le operazioni di autenticazione e associazione sono troppo onerose per essere adottate in IEEE 802.11p. Le comunicazioni tra veicoli richiedono, soprattutto in caso di comunicazioni di sicurezza, la capacità di scambiare dati rapidamente, il che rende inefficiente la procedura di adesione utilizzata in 802.11. Pertanto è essenziale per tutti i dispositivi basati su 802.11p siano settati sullo stesso canale e configurati con lo stesso indirizzo MAC (BSSID). La novità introdotta da IEEE802.11p WAVE è proprio il termine WAVE che sta ad indicare la configurazione della stazione radio in 'modalità WAVE'. Questa permette alla stazione di trasmettere e ricevere pacchetti di dati senza la necessità di appartenere ad una determinata BSS. Ugualmente, per le comunicazioni non legate alla sicurezza, l'overhead di una tradizionale BSS è troppo dispendioso; per questo in 802.11p viene introdotto un nuovo concetto di BSS: il Wave BSS (WBSS). Una stazione forma una WBSS utilizzando un pacchetto di richiesta denominato Beacon; a sua volta la stazione ricevente diffonde il messaggio di beacon a stazioni radio vicine in modo tale da espandere la WBSS. La richiesta di creare una WBSS viene dagli stati protocollari superiori e contiene tutte le informazioni necessarie alle stazioni riceventi per capire il tipo di servizio offerto dalla WBSS così da poter decidere se aderire o meno. In caso di adesione, nel pacchetto di richiesta sono contenute anche tutte le informazioni necessarie per auto configurarsi come stazione della WBSS. Notiamo che questa procedura di interazione è molto veloce perché permette di creare una connessione stabile tra più unità semplicemente con lo scambio di un pacchetto (beacon) di richiesta e non c'è bisogno di ulteriori interazioni per completare il processo di adesione. Questo approccio offre overhead estremamente ridotti per la creazione di una rete tra veicoli evitando processi di associazione e autenticazione. Per far funzionare correttamente la rete servono ulteriori meccanismi gestiti a livelli superiori, ad esempio garantire la sicurezza della trasmissione, ma questi esulano dallo

scopo della trattazione dello standard 802.11p. Nelle applicazioni per la sicurezza è supportato da tutte le stazioni l'utilizzo dell'indirizzo speciale BSSID (wildcard BSSID). Questo significa che se un veicolo facente parte di una WSSB deve trasmettere una segnalazione per la sicurezza, può inviare il pacchetto con l'indirizzo speciale in modo che questo sia ricevuto da tutti i veicoli nei dintorni, anche se non compresi nella sua WSSB. Allo stesso modo veicoli che ricevono pacchetti con l'indirizzo speciale BSSID possono utilizzarlo anche se non facenti parte delle WSSB della stazione trasmittente. I pacchetti che possono essere trasmessi anche alle unità al di fuori della propria WBSS vengono ricevuti solo se contengono l'indirizzo speciale BSSID perché, in genere, ogni stazione a livello MAC filtra i pacchetti ricevuti conservando solamente quelli provenienti dalla stessa WSSB a cui appartiene la stazione ricevente. Le innovazioni principali a livello MAC sono:

- Una stazione in modalità WAVE può inviare e ricevere pacchetti dati con indirizzo speciale BSSID indipendentemente dal fatto di essere o meno membro della stessa WBSS.

- Una WBSS è un tipo di BSS che consiste in una serie di stazioni (veicoli) che cooperano tra loro in modalità WAVE e che comunicano utilizzando uno stesso indirizzo BSSID. Una WBSS è inizializzata quando una stazione in modalità WAVE invia un 'WAVE beacon' nel quale include tutte le informazioni necessarie perché il ricevitore si configuri alla WBSS.

- Un stazione entra a far parte di una WBSS quando è configurata per inviare e ricevere pacchetti con indirizzo BSSID definiti per quella WBSS. Viceversa cessa di far parte di una WBSS quando il livello MAC interrompe la ricezione e l'invio di pacchetti dati contenenti l'indirizzo BSSID di quella WBSS.

- Una stazione non può essere membro di più di una WBSS contemporaneamente.

- Una WBSS cessa di esistere quando non ci sono membri che la compongono. Inoltre la prima unità che richiede la creazione della WBSS invia un messaggio che è uguale a quello che utilizzano poi tutti gli altri veicoli quindi quando il

primo membro che ha generato la richiesta di formare la WBSS si dissocia, quest'ultima continua ad esistere.

1.1.5.4 Funzioni del livello fisico

A livello fisico la filosofia di 802.11p è quella di apportare cambiamenti minimi rispetto alla già esistente 802.11 PHY in modo da ottimizzare quest'ultima per l'utilizzo nelle reti veicolari. Questo approccio è fattibile perché IEEE 802.11a opera alla frequenza di 5 GHz e non è difficile configurarlo per renderlo operativo a 5.9 GHz. Mentre gli aggiornamenti a livello MAC sono relativamente semplici, richiedendo solo un aggiornamento software, gli upgrade a livello fisico sono più complicati per cui si è cercato il più possibile di sfruttare 802.11a per evitare di creare una nuova tecnologia di collegamento radio. IEEE 802.11p è essenzialmente basato sulla tecnica di modulazione OFDM con una larghezza di banda del canale di 10MHz anziché i 20MHz di 802.11a. Il motivo principale di questa scelta è ridurre gli effetti di rumore introdotti dal canale radio.

1.1.6 Conclusioni

Quanto discusso nelle sezioni precedenti ci fa comprendere come il sistema delle VANET permette lo sviluppo di sitemi più articolati e complessi per il controllo del sistema dei trasporti inteso come movimentazione di merci, flussi di persone, controllo dei veicoli; questi sistemi sono chiamati per l'appunto sistemi intelligenti di trasporto o **ITS (Intelligent Trasportation System)**.

1.2 Intelligent Trasportation System

Un sistema intelligente di trasporto nasce dall'applicazione delle tecnologie informatiche e delle telecomunicazioni (ICT) al settore dei trasporti. In genere ci si riferisce a tutti i tipi di trasporto: terrestre (strade, ferrovie), aereo e marittimo; nonché ai servizi per i viaggiatori e per le merci trasportate. Apparati elettronici come computer (elaborazione dati), sensori (di flusso, telecamere, ecc) e satelliti (GPS) svolgono un ruolo fondamentale in questi sistemi; i quali acquisiscono, elaborano, gestiscono e trasmettono i dati relativi ai veicoli, allo stato delle infrastrutture e agli utenti integrandoli tra loro.

Proprio questa integrazione consente di affrontare i problemi legati alla mobilità

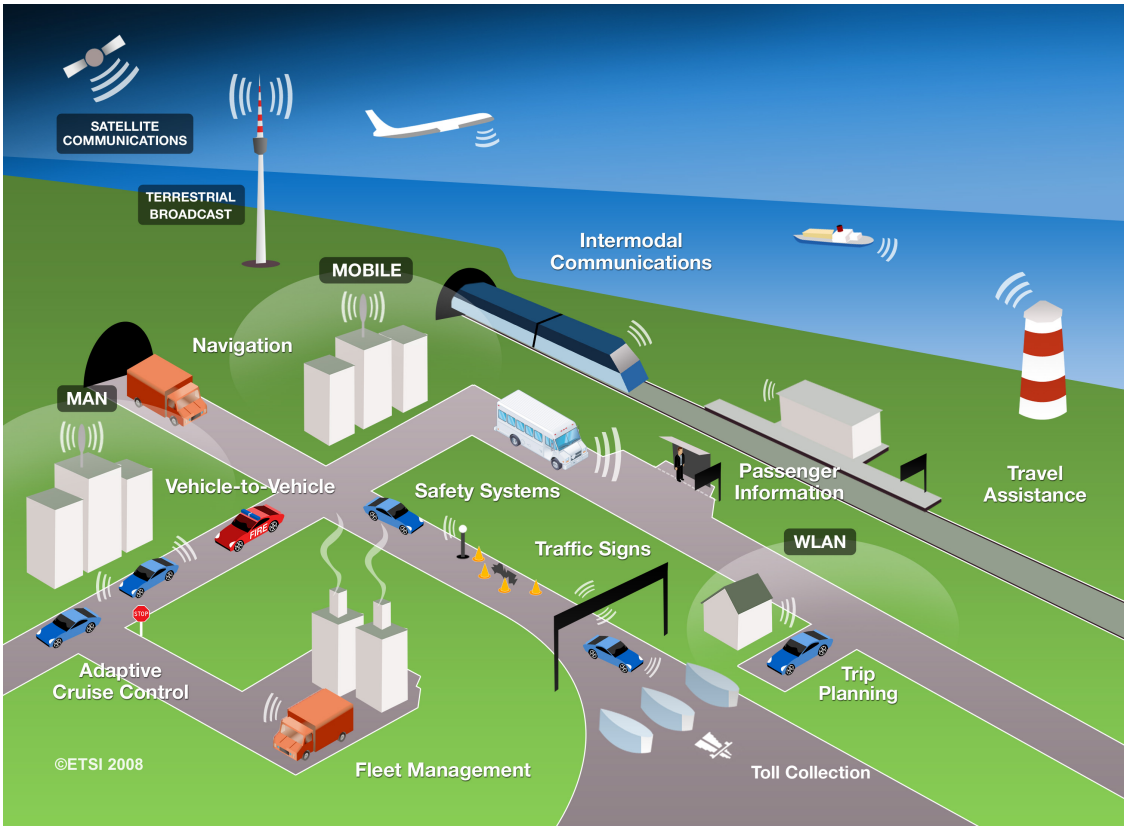


Figura 1.6: Esempio di ITS

in modo semplice, incrementando la sicurezza, l'efficienza e l'efficacia del trasporto, riducendo allo stesso tempo l'impatto ambientale e migliorando la qualità della vita

degli utenti.
Queste tecnologie trasformano i trasporti in un sistema nel quale il traffico delle merci vieni distribuito in modo equilibrato tra le varie tipologie di trasporto (gomma, treni, navi o aerei), ottenendo una maggiore produttività ed efficenza, nonché una maggiore sicurezza del trasporto. Fra i maggiori obiettivi di un sitema intelligente di traporti possono essere trovati: la riduzione del traffico (travel planner), l'organizzazione del carico dei veicoli, l'incremento della sicurezza dei viaggiatori (safety system), la diminuizione tempo di viaggio e del consumo di carburante. Questi effetti, comportando, la riduzione del traffico del tempo di viaggio, del consumo di carburante e dell'inquinamento hanno ripercussioni sui costi dei prodotti, i quali a fronte di una diffusione della tecnologia diminuiranno.
Quanto appena detto può essere implementato a patto che sia le infrastrutture sia i veicoli siano in qualche modo 'intelligenti'; nei veicoli possono essere installati dispositivi che permettano di evitare le collisioni, assistere il guidatore e, se necessario, lanciare un allarme in caso di incidente; le infrastrutture oltre a recepire tali allarmi, devono consentire la gestione degli eventi. Il caso di un tamponamento tra due veicoli permette di introdurre uno scenario di utilizzo per queste reti: una volta che le due automobili si sono tamponate, le loro centraline inviano sul canale wireless un segnale di allarme; i veicoli che sopraggiungono saranno avvertiti del pericolo, allo stesso modo i veicoli di soccorso possono essere informati della precisa localizzazione dello scontro, nonche i tabelloni presenti sulla rete avvertiranno gli automobilisti attraverso un sistema intelligente di gestione degli eventi senza la necessità del coinvolgimento diretto dell'uomo nel protocollo di gestione.

1.3 On Board Unit (OBU)

Per rendere possibile quanto appena detto, è necessario che le vetture possano comunicare tra loro e con le appostite reti fisse (infrastructure networks). Tutto questo può essere reso possibile dotando, quanti più veicoli possibili, di un'unità da inserire al suo interno che possa acquisire, elaborare, memorizzare e trasmettere i dati relativi al veicolo. Questa unità di solito chiamata OBU (On Board Unit), al fine di implementare i servizi descritti nella sezione precedente dovrà soddisfare alcuni requisiti di base in termini di interfacce e potenza di calcolo. Facendo riferimento alla Figura 1.7, questi requisiti possono essere sintetizzati nei seguenti punti:

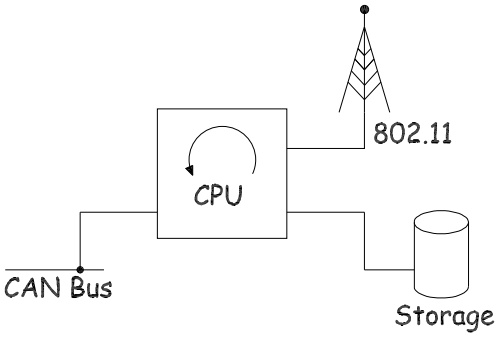


Figura 1.7: Architettura base

- **Transceiver IEEE 802.11p** (o 802.11a per la compatibilità in frequenza) per realizzare la connessione wireless, sia con le reti infrastrutturali sia per quelle interveicolari;

- **CAN bus:** per interfacciarsi con cui tutti i sensori del veicolo che comunicano con la centralina. Attraverso questa periferica si trasferiscono sia dati quali la temperatura dell'acqua, pressione dell'olio, ecc sia eventuali segnalazioni di malfunzionamenti (e.g. rottura dei sensori stessi o di qualche dispositivo meccanico)

- **Storage:** la board deve funzionare come la scatola nera presente in tutti gli aereoplani e nei treni; registrando tutti i dati provenienti dal CAN Bus (velocità del veicolo, integrità degli apparati, manovre del guidatore, ecc)

- **Capacità computazionali:** tutte le funzioni appena descritte necessitano di una CPU che sia in grado di elaborare e trasmettere i dati, il tutto supportato da un sistema operativo embedded.

- **Costi contenuti:** Come obiettivo aggiunto si vogliono mantere bassi i costi di produzione in modo che questi sistemi non siano riservati a veicoli di fascia alta, ma accessibili alla fasce media e medio/bassa. Tutto questo può essere possibile implementando il sistema con componenti general purpose invece che con altri sistemi molto più costosi del tipo ASIC (Application Specific Integrated Circuit).

- **Sistema Operativo:** Un'ulteriore obbiettivo è quello di riuscire ad utilizzare un sistema operativo Open Source (OpenWRT, EmDebian,ecc) del tipo Linux embedded .

Quelli elencati come gia visto sono considerati i requisiti minimi a cui dovrà rispondere la nostra board. Come vedremo in seguito questi saranno ampliati, sia per implementare nuove funzionalità (GNSS, eventuale monitor, ecc) sia con interfacce necessarie allo sviluppo del disposivo stesso (porte seriali, Ethernet, led di segnalazione, ecc).

1.3.1 Il protocollo CAN BUS

Il Controller Area Network (CAN) è un protocollo di comunicazione seriale sviluppato nei primi anni ottanta dalla Bosch per la realizzazione sistemi di controllo ditribuiti ed in tempo reale. Nel 1993 è divenuto uno standard internazionale ISO ed oggi è ampiamente utilizzato nel settore industriale, ed in particolare in campo automobilistico per la connessione dei dispositivi che presiedono al controllo dei veicoli. Il protocollo consente la realizzazione di reti con gerarchie di tipo multimaster, in cui ciascun dispositivo connesso all'unico bus può assumere il controllo del canale di comunicazione e trasmettere i propri dati secondo un modello di trasmissione broadcast orientata al messaggio. Un modello di questo tipo prevede che siano definiti i contenuti dei messaggi invece che i nodi e gli indirizzi dei nodi che compongono la

rete. In quest'ottica ciascun messaggio è etichettato da un identificatore univoco che descrive il significato dei dati contenuti, e fissa una priorità statica necessaria a risolvere le situazioni di contesa del bus da parte di due o più nodi trasmittenti. Il modello di tipo broadcast comporta che ciascun dispositivo connesso alla rete è in grado di ricevere tutti i messaggi inviati sul bus: è compito dei singoli nodi decidere sulla base dell'identificatore se il messaggio è di propria competenza o vada invece ignorato. Il risultato di tali scelte ha portato ad una architettura che permette la progettazione di sistemi modulari ed altamente configurabili. L'aggiunta, ad esempio, alla rete di dispositivi riceventi non comporta alcuna modifica software ed hardware ai dispositivi esistenti, mentre la trasmissione in modalità broadcast consente la ricezione multipla dei dati e la sincronizzazione di processi distribuiti.

1.3.1.1 Struttura dei messaggi

L'intero meccanismo di trasferimento dati del protocollo CAN si basa su 4 tipi di messaggi con struttura fissa a lunghezza variabile ma comunque limitata:

- messaggi dati: sono utilizzati per la trasmissione di dati fra due o più nodi della rete.
- messaggi remoti: sono utilizzati per la richiesta di trasmissione di un messaggio dati con lo stesso identificatore del messaggio remoto trasmesso.
- messaggi di errore: sono trasmessi dai dispositivi della rete al rilevamento di un errore sul bus.
- messaggi di ritardo: introducono un ritardo tra un messaggio di richiesta remota ed un messaggio dati a fini di sincronizzazione.

Vediamo in dettaglio la stuttura di un messaggio dati schematizzata in figura Figura 1.8

SOF	ID	RTR	IDE	DLC	DATA	CRC	ACK	EOF

Figura 1.8: Struttura di un messaggio dati nel protocollo CAN.

SOF (Start Of Frame) è un singolo bit che contrassegna l'inizio del messaggio; dal momento che un dispositivo può trasmettere solamente in assenza di attività sul bus, il bit SOF svolge anche un ruolo di sincronizzazione fra il nodo che inizia una trasmissione e gli altri nodi della rete.

ID (Identifier) é il codice di identificazione che consente di differenziare i messaggi in base al contenuto; consente inoltre di fissare una gerarchia di priorità statiche dei messaggi da utilizzare in caso di conflitto durante la fase di trasmissione: ai valori di identificazione più bassi corrispondono priorità più elevate. Il campo ID è di 11 bit per i messaggi in formato base, 29 per quelli in formato esteso introdotti con la specifiche 2.0 del protocollo.

RTR è il bit per la richiesta di trasmissione remota che consente di distinguere un messaggio di trasmissione dati da uno di richiesta ad un altro nodo della rete.

IDE (IDentifier Extension) è il bit utilizzato per distinguere i messaggi in formato base da quelli estesi.

DLC (Data Legth Code) campo formato da 6 bit dei quali 4 sono utilizzati per indicare il numero di byte di dati trasmessi o richiesti dal messaggio, 2 bit sono invece riservati per usi futuri.

DATA campo contenente i dati che il messaggio sta trasportando, il limite massimo è di 8 byte.

CRC (Cyclic Redundant Check) campo contenente una sequenza di controllo calcolata per verificare l'integrità dei dati ricevuti.

ACK campo di due bit utilizzati dal dispositivo trasmittente e ricevente per la conferma di una corretta ricezione di un messaggio.

EOF (End Of Frame) campo di 7 bit che segnala la fine del messaggio.

1.3.1.2 Rilevazione e segnalazione degli errori

Il protocollo CAN prevede che eventuali errori nella trasmissione siano segnalati non appena ne venga rilevata la presenza. A livello di messaggio sono disponibili tre meccanismi per l'individuazione e la segnalazione degli errori:

- Sequenza CRC: il dispositivo trasmittente calcola la sequenza CRC e la spedisce assieme al messaggio. Il nodo ricevente ricalcola la sequenza e la confronta con quella ricevuta: se le due sequenze non coincidono viene segnalato che si è verificato un errore nella fase di trasmissione.

- Controllo del formato: ogni dispositivo prima di inviare un messaggio sul bus verifica che la struttura dei campi sia conforme per formato e dimensione al modello previsto dal protocollo.

- Segnale di riconoscimento: il nodo che riceve un messaggio di propria competenza deve informare il dispositivo trasmittente dell'avvenuta consegna. In caso di mancata segnalazione il nodo trasmittente segnala la presenza di un errore.

Ai tre meccanismi appena visti se ne aggiungono due operanti a livello di bit:

- Parallelamente alla trasmissione dei dati, ciascun dispositivo esegue un'analisi del segnale del bus in modo da poter rilevare la differeanza fra i bit spediti e quelli ricevuti; in questo modo si possono inoltre distinguere gli errori causati dal dispositivo da quelli introdotti dal resto della rete.

- Durante la codifica dei campi SOF, ID, DATA e CRC ad ogni cinque bit consecutivi identici viene accodato un bit di valore complementare. Il dispositivo ricevente verifica che non esistano sequenze di sei bit uguali e consecutivi: in caso di responso positivo viene rimosso il sesto bit e si procede con la decodifica dei dati, in caso contrario viene generato un messaggio di errore.

Nel momento in cui viene rilevata una condizione di errore, il nodo interessato annulla la trasmissione corrente ed invia un messaggio di errore evitando che gli altri nodi ricevano dati corrotti, mantenendo così il sistema in uno stato consistente; subito dopo il dispositivo ripete automaticamente la trasmissione che aveva provocato l'errore.

1.3.1.3 Livello fisico

Le specifiche del protocollo CAN non si limitano al solo livello data link, ma definiscono in parte anche quelle che sono le caratteristiche del livello fisico del modello OSI. A livello di segnale i singoli bit vengono rappresentati tramite una codifica del tipo NRZ (Non-Return-to-Zero), in cui il livello del segnale rimane costante per l'intera durata del periodo di bit.

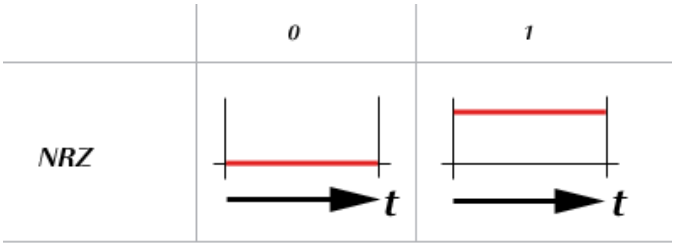


Figura 1.9: Codifica NRZ

A livello di bit la trasmissione avviene in modalità sincrona: se da un lato questo consente di incrementare le capacità trasmissive, dall'altro rende necessario l'uso di tecniche per mantenere la sincronizzazione.
A tale scopo ciascun periodo di bit può essere pensato come la composizione di quattro intervalli temporali non sovrapposti

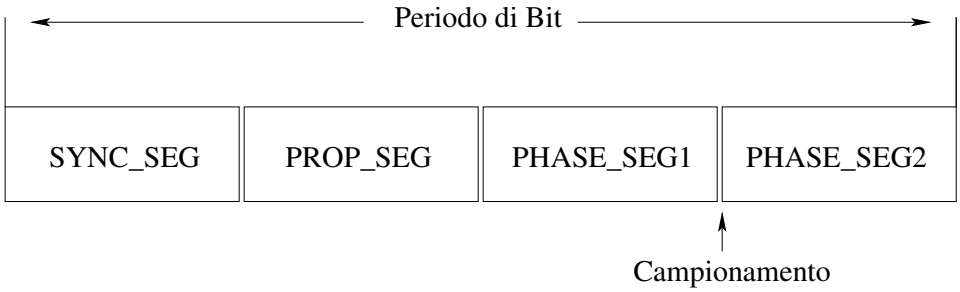


Figura 1.10: Scomposizione del periodo di bit.

Riferendoci alla Figura 1.10

SYNC_SEG: è il segmento temporale del periodo di bit utlizzato per sincronizzare i nodi della rete; la sua durata è pari ad un periodo del segnale di clock della rete.

PROP_SEG: questo segmento è utilizzato per compensare il ritardo del segnale sul mezzo fisico; il suo valore è pari al doppio della somma del tempo di propagazione sulla rete, e dei tempi di trasmissione e ricezione dei dispositivi.

PHASE_SEG1, PHASE_SEG2: questi due segmenti sono utilizzati per compensare eventuali ritardi o anticipi di fase rispetto al segnale di clock della rete. I loro valori sono programmabili e vengono variati in fase di risincronizzazione.

Al termine del segmento PHASE_SEG1 avviene la lettura del livello del segnale sul bus e l'interpretazione del valore del bit corrispondente.

Esistono due modalità di sincronizzazione:

Sincronizzazione forzata: è realizzata quando in assenza di attività sul bus un nodo riceve il bit SOF di un messaggio appena inviato da un altro dispositivo. In seguito a tale ricezione il nodo inizializza un nuovo periodo di bit in modo che il segnale che ha indotto la sincronizzazione si venga a trovare all'interno del segmento SYNC_SEG del nuovo periodo di bit.

Risincronizzazione: è realizzata nella fase di ricezione dei messaggi variando il valore dei segmenti SYNC_SEG1, SYNC_SEG in modo da variare l'istante di campionamento

in accordo al messaggio trasmesso.

Dal momento che il periodo di bit dipende dal tempo di propagazione del segnale sul mezzo fisico, la massima frequenza di trasmissione su una rete è strettamente correlata alla lunghezza del bus: quanto più è grande la distanza massima fra due nodi, tanto minore è la frequenza massima di trasmissione utilizzabile. I due casi estremi previsti dal protocollo prevedono un bit rate massimo di 1 Mbps su bus da 40 metri, per arrivare a bus lunghi 1000 metri con velocità di trasmissione di 40 Kbps.

1.4 Related Work

La maggior sicurezza e fruibilità dei trasporti interessa da una parte le amministrazioni pubbliche, per la ricaduta economica che gli incidenti hanno sulle finanze pubbliche; dall'altra gli esperti di logistica, per via dei tempi di percorrenza delle merci sulle strade; e tutti i fruitori della arterie viarie per evitare di incappare in lunghe code.
Possiamo affermare che non sono stati trovati molti lavori relativi alla teconologia costruttiva di una On Board Unit per comunicazioni V2V e V2I, simili a quello che è il nostro obiettivo finale; andremo comunque ad elelencare dei lavori che a nostro avviso ci sono sembrati degni di nota.

An Integrated Embedded Solution for Vehicle Communication & Control[**?**] Viene proposto il progetto di una soluzione embedded per la comunicazione con il veicolo e il suo controllo in modo da fornire servizi ah-hoc per conducente e passeggeri del veicolo. Il cuore del progetto è costituito da due board separate: la prima l'unità di elaborazione è costituita da un Single Board Computer (SBC) il *Soekris Net4801* (processore x86) a cui vanno collegati: la scheda wireless, il modem GPRS e la scheda di espansione per la comunicazione CAN; la quale costituisce la seconda board; che risulta equipaggiata con due microcontrollori Microchip utilizzati per le comunicazioni con la main board, il controller CAN (MCP2515) e un trasceiver. Il tutto è gestito da un sistema operativo Linux (Geento) sul quale vengono lanciati gli applicativi per la gestione delle interfaccie. Attraverso questo insieme di schede, le informazioni possono essere cambiate tra veicoli, infrastrutture e telefoni cellulari. L'insieme del software sull' SBC e i firmware dei microcontrollori implementano un AODV (Ad-hoc On-Demand Distance Vector, un algoritmo di routing per reti ad-hoc mobili che supporta sia l'instradamento unicast che multicast).

Verification of On-Line Vehicle Collision Avoidance Warning System using DSRC[**?**] Questo lavoro presenta la realizzazione di un sistema di allarme per evitare le collisioni con i veicoli. Il sistema integra le Dedicated Short Range Comunication

(DSRC) con quella del Global Navigation Satellite System (GNSS) utilizzando un ricevitore GPS per stabilire la posizione; il tutto gestito da un sistema embedded. L'unità di elaborazione è costituita da un processore x86 (con sistema operativo Linux) su cui è stata integrata , oltre alle sudette tecnologie, anche un controller CAN (connesso utilizzando la porta seriale). Il sistema trasmette sulla rete WAVE la posizione (GPS) e le informazioni di controllo provenienti dal bus CAN (percentuale di accelerazione o di frenata, angolo di sterzo e velocità delle ruote) costituendo un vero e proprio radar per veicoli con la possibilità di evitare le collisioni tra di essi. Il sistema interfacciandosi con un comune laptop provvede a fornire una pianta in cui sono inseriti i veicoli presenti nel raggio di azione del transceiver radio inserendo in sudetta pianta le informazioni dei veicoli stessi (accelerazione, posizione, ecc). Questo prototipo fornisce un valido sistema di ausilio al conducente, minimizzando il riscio di collisioni.

Extending Vehicular CAN Fieldbuses With Delay-Tolerant Networks[**?**] Questo articolo risulta molto interessante non per l'utilizzo di tecnologia DSRC o WAVE ma poichè si effettua un calcolo approfondito dei ritardi per estendere la rete CAN utilizzando le reti Delay Tollerant Networks (DTNs).

Multi-Role WAVE Platform for Dedicated Short Range Communication (DSRC)[**?**] Arada System (specializzati in soluzioni WiFi) ha lanciato, in occasione del ITS world congress 20010 (BEXCO, Busan, Corea); una On Board Unit (OBU), capace di comunicazioni V2V e V2I (o con RSU),attraverso il protocollo WAVE, con interfacce verso: CAN Bus, Bluetooth, GPS e permettendo anche il pagamento elettronico dei pedaggi. Il sistema si basa su un processore x86, ma non vengono dati dettagli sul sistema operativo o sulla connessione delle interfaccie.

Capitolo 2

Architettura del sistema

Partendo dai requisiti di sistema, descritti al capitolo uno, siamo andati a definire le specifiche di progetto della nostra On Board Unit.

2.1 Specifiche di Progetto

Interfaccie necessarie

- Bus PCI: per connettere il transceiver wireless; inoltre dato che tale interfaccia permette di collegare più dispositivi, il nostro fine ultimo sarebbe quello di creare un bridge tra IEEE802.11p e le reti IEEE802.11b/g/n. Questo tipo di bus necessita di un data rate da parte del processore di 133 MBps poiché deve gestire istruzioni a 32 bits con una velocità di clock di 33.3 MHz;

- Porta Ethernet: risulta strategico la connessione anche con la rete cablata, potremmo così realizzare un solo tipo di hardware sia per l'OBU sia ad esempio per dei nodi fissi (tale scelta risulta utile nella fase di protoripizzazione del codice).

- Porte seriali, SPI, I^2C queste porte sono necessarie nella fase iniziale di debug, inoltre serviranno per comunicare: con il CAN bus, il GPS ed in futuro con 802.15.4;

- Slot per memoria (Compact Flash, SD-Card, ecc) il sistema deve riuscire a memorizzare le principali attività eseguite: dai dati provenienti dal CAN Bus (stato del veicolo, manovre eseguite, ecc) alle informazioni di sicurezza provenienti dalla rete infrastrutturale; esattamente come succede negli aerei con la scatola nera.

Capacità computazionali

Le capacità computazionali erano state inserite all'interno dei requisiti; in questo sistema il tipo di architettura (Microcontrollore o ARM) deve essere studiata per gestire un insieme molto vasto di dati: dalle connessioni senza fili passando alla comunicazione CAN ed arrivando alla scrittura dei dati nelle varie memorie (per citare le principali operazioni). Per il suo dimensionamento ci siamo prefissi di far lavorare, la CPU che andremo a scegliere, non oltre il 50% delle capacità totali; per rendere così possibili, in un prossimo futuro, possibili upgrade del sistema.

2.2 CPU

Le nostre ricerche, sono partite con la ricerca di quale architettura quella con Microcontrollore (MCU Micro Controller Unit) o con processore ARM si adattasse alle nostre esigenze computazionali; tenendo sempre presente, la velocità con cui i dati devono essere elaborati e la necessità di poter fara espansione o upgrade del sistema.

2.2.1 Microcontrollore

Il microcontrollore è un dispositivo elettronico caratterizzato dal fatto che tutti i componenti necessari al suo funzionamento (CPU, RAM e memorie) sono integrati all'interno di un unico chip. L'interfacciamento del micro con il resto del circuito avviene mediante pin specializzati e convigurabili; gestiti tramite un programma chiamato *firmware*.

Le capacità computazionali si aggirano intorno agli 80 MHz, hanno un irrisorio consumo di potenza, numerosi moduli per l'interconnessione di moltissime periferiche

(ADC, SPI, I²C e UART) il tuttu unito ad un basso costo ed una facilità di programmazione dovuta ai numerosi tool di sviluppo disponibili. Questi fattori hanno contribuito sin dagli inizi (nella metà degli anni '70 con l'Intel 8048) ad un loro larghissimo sviluppo; dato il basso costo e le discrete prestazioni sono divenuti ben presto processori utilizzati in moltissime applicazioni sia in ambito industriale che amatoriale. Alcuni di loro recentemente incorporano al loro interno dei moduli altamente specializzati chiamati *DSP (Digital Signal Processor)* che si occupano della trattazione digitale dei segnali analogici.
Negli ultimi anni si sono sviluppati interi sistemi operativi che data la loro leggerezza posso essere implementati su queste architetture (è il caso del sistema operativo real time E.R.I.K.A.[1]).

2.2.2 ARM

Con architettura ARM sono indicati una categoria di microprocessori a tecnologia *RISC* sviluppata a partire dalla metà degli anni '80 che oggi equipaggiano moltissimi sistemi: partendo dai router wireless, passando per gli smart-phone ed arrivando alle applicazioni industriali più avanzate. Le capacità computazionali sono molto elevate, la velocità si aggira intorno ai 400 MHz con un basso consumo di potenza, tanto da non essere dotati di dissipatore. Come tutti i processori, necessitano di essere supportati da un hardware esterno: memorie, alimentazioni multi-tensione, ecc per questo il suo costo lievita notevolmente rimuovendoli dalla fascie del mercato amatoriale; questo ci fà intuire però quanto versatili siano le loro capacità ad esempio per effettuare un upgrade di memoria è sufficente sostituire la stessa con un di capacità maggiore. Inoltre dato l'elevato numero di linee in ingresso ed uscita dispongono di moltissime interfaccie integrate: bus SD-Card e Compact Flash, SPI, I²C UART, USB, Ethernet ecc. Naturalmente per poter funzionare devono disporre di un sistema operativo (EmDebian, OpenWRT, ecc) studiato per applicazioni embedded; su cui poi gireranno i vari programmi.

[1] http://erika.tuxfamily.org/

2.2.3 MCU vs ARM

Come anticipato ci siamo concentrati su due tipi di architetture: la prima basata su micro-controllore e l'altra basata su processori di tipo ARM. Quella a micro-controllore risulta essere più leggera ed economica con lo svantaggio di avere prestazioni molto limitate e memoria non espandibile; mentre quella con processore di tipo ARM, più veloce, al tempo stesso più complessa in fase progettuale e costosa, ma espandibile a piacimento. Andremo ora ad analizzare le due architetture partendo da un confronto tra le caratteristiche (per la compilazione della tabella sono stati presi come esempio per il micro controllore la famiglia PIC32 di Microchip e per l' ARM la serie AT91SAM9 di Atmel):

	PIC	ARM
CPU speed [MHz]	80	da 170 a 700
Max Data Rate[1][Mbps]	22	n.d.
Program (Flash) memory [KB]	512	no in chip
RAM [KB]	128	2 * 16
I/O pins	85	180
SPI	4 (mux[2])	2 (con quattro Cable Selected)
I^2C	5 porte (mux)	2
Ethernet	no	1
USB	no	2 host
ADC	8 (12 bits)	2 (10bits)
Image Sensor	no	1
Possibili upgrade	solo memorie EEProm su bus I^2C	in teoria illimitati

[1]Numero di bits che vengono trasmessi o elaborati per unità di tempo.

[2]Multiplexer, mux o selettore è utilizzato per avere più interfaccie disponibili su medesimi piedini ad esempio SPI, I^2C o UART; i quali possono essere selezionati via software.

Sulla base delle considerazione fatte fin'ora, si è scelti di basare lo sviluppo del prototipo su un'architettura di tipo ARM, anche se questo, come visto, comporta un maggior costo e un maggior tempo di sviluppo (dovuto più che altro al circuito stampato); ma saremo ripagati dalle maggiori performance e dalla possibilità di upgrade. In commercio esistono moltissimi produttori di processori ARM, fra le maggiori si possono trovare: Atmel, Intel e Texas Instruments le cui caratteristiche, in generale, si assomigliano. Si doveva quindi cercare il giusto trade-off tra velocità di computazione, relativa semplicità implementativa e la disponibilità di risorse (driver, tool di sviluppo e tutorial). I processori di casa Atmel raggiungono il punto di incontro della nostra ricerca; infatti hanno un'ottima velocità (400MHz), la casa produttrice fornisce un'ottima documentazione (datasheet, application notes, schemi di collegamento con le varie periferiche, ecc) ed inoltre, dato che equipaggiano molte schede di sviluppo, da terze parti è possibile reperire molto materiale aggiuntivo (correzioni di eventuali bugs, driver, ecc).
Per queste motivazioni è stato scelto il chipset **AT91SAM9G20** [2].

[2] http://www.atmel.com/dyn/resources/prod_documents/doc6384.pdf

2.2.4 Analisi dei costi

Un'analisi preliminare è stata condotta per trovare il giusto trade-off tra costi e tempo di sviluppo e tra costi e performance dei componenti.

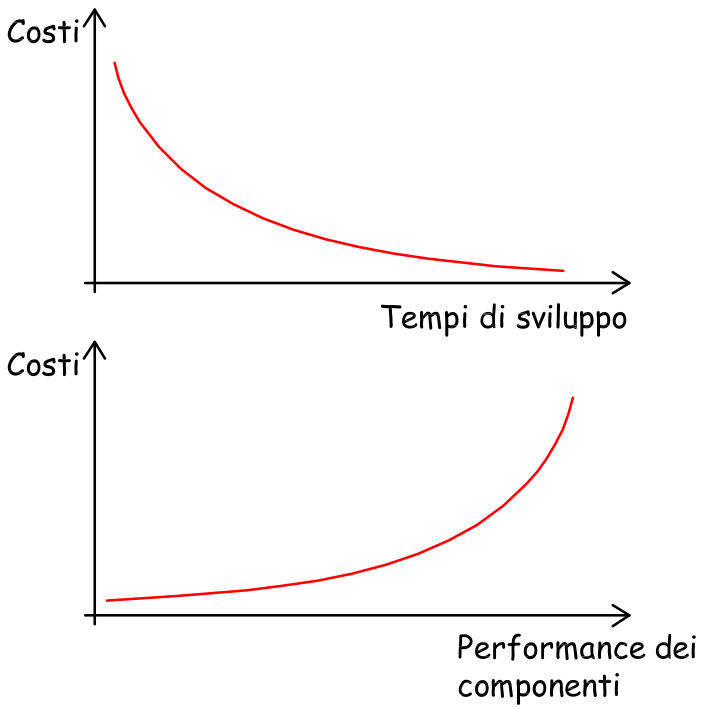


Figura 2.1: Costo di prototipizzazioni in riferimento al tempo di sviluppo e alle performance dei componenti

Riferendoci alla Figura 2.1 risulta semplice capire che un rapido sviluppo, comportando l'occupazione di più unità lavorative possa far lievitare in maniera significati i costi; come pure le performance dei componenti. Infatti lo stesso chip è possibile reperirlo sul mercato con differenti caratteristiche: una, ad esempio, è la temperatura di funzionamento, infatti un chip che può funzionare da -40°C fino a 80°C (che magari risponde a direttive di tipo militari o industrial) è molto più costoso rispetto ad uno il cui range di funzionamento è compreso tra 0°C e 60°C (di tipo consumer). Per queste ragioni per questa prima fase di sviluppo sono stati scelti componenti con caratteristiche di tipo consumer.

Capitolo 3

OBU su architettura ARM

3.1 Wireless adapter

Nelle specifiche di progetto, l'adatattore wireless doveva essere collegato al processore mediante il bus PCI. Questo per riuscire a costruire, con il medesimo hardware, oltre all'OBU anche un brigde che ci permettesse lo scambio di informazioni, tra le reti veicolari (802.11p) e le reti WiFi (802.11a/b/g/n); questa importante soluzione permetterebbe quindi di realizzare le RSU (Road Site Unit) senza la necessità di alcun tipo cablaggio, ottenendo un notevole risparmio sopratutto nello start-up del sistema. Purtroppo però i processori ARM, da noi presi in considerazione, non implementano tale interfaccia e risulta impossibile utilizare componenti del tipo di Cypress CY7C09449PVA-AC, (definiti *bridge PCI*), per via della struttura dei bus del processore stesso poichè, per sfuttare le potenzialità del bus PCI, occorre che i bus dati e address del processore possano indirizzare parole di 4 Byte. Questo non avviene quindi si dovrà riconsiderare questa specifica. La necessità del bus PCI era dovuto al fatto che le interfacce che implementano il protocollo IEEE 802.11p al momento disponibili sono tutte su bus miniPCI (stessa piedinatura del bus PCI ma con dimensioni meccaniche molto ridotte). Pur essendo

Figura 3.1: SX-SDWAG 802.11a/b/g SDIO Card.

uno svantaggio molto invalidante (per via della citata possibilità di rapida sostituzione del protocollo) si è notato che queste prime interfacce implementano solo il layer fisico del protocollo 802.11p mentre il layer MAC risulta essere ancora quello generico di 801.11 non ancora adattato per le comunicazioni veicolari. Date le considerazioni fatte, abbiamo deciso di utilizzare un transceiver IEEE 802.11a per mantenere perlomeno la compatibilità in frequenza. Ricordando che l'interfaccia che dobbiamo utilizzare debba avere i driver per sistemi operativi Unix, sono state prese in considerazione numerose alternative prima di arrivare alla soluzione finale. Una delle prime è stata quella di collegare direttamente il chip per la gestione della comunicazione wireless (ad esempio il AR6001 di Atheros), idea scartata per le difficoltà che si verrebbero a creare per la realizzazione del circuito stampato. Un'altra possibilità è quella di utilizzare una scheda WiFi 802.11a (visibile in Figura 3.1) che utilizza il bus di comunicazione SDIO (Secure Digital Input Output); queste interfaccie venivano utilizzate agli albori dei collegamenti senza fili per dotate i palmari (sprovvisti di transceiver WiFi) di questa tencnologia, purtroppo sono divenute obsolete ed andate quindi fuori produzione.

Figura 3.2: Adattatore WUBR-506N.

La soluzione definitiva, è arrivata dall taiwanese SparkLAN la quale produce e commercializza il WUBR-506N (visibile a lato), un modulo USB wireless dual band che lavora su reti 802.11a/b/g con frequenza di 2.4 e 5 GHz garantendo la sicurezza dei dati attraverso la crittrografia WPA, WPA2 ed AES. Tutto questo può essere connesso al sistema tramite un'interfaccia a 4 pin (USB); per facilitare l'integrazione anche in dispositivi embedded. Il modulo utilizza due potenti chip di casa Ralink l'RT2870 e l'RT2850: il primo è un transceiver per il collegamento in radio frequenza ed il secondo costituisce l'intefaccia USB. Dato la grande diffusioni di queste interfacce la casa prodruttrice fornisce sia i driver per Windows Embedded sia per distribuzioni basate su Unix (EmDebian, OpenWRT, ecc). In figura Figura 3.1 è mostrato lo schema a blocchi di questa interfaccia.

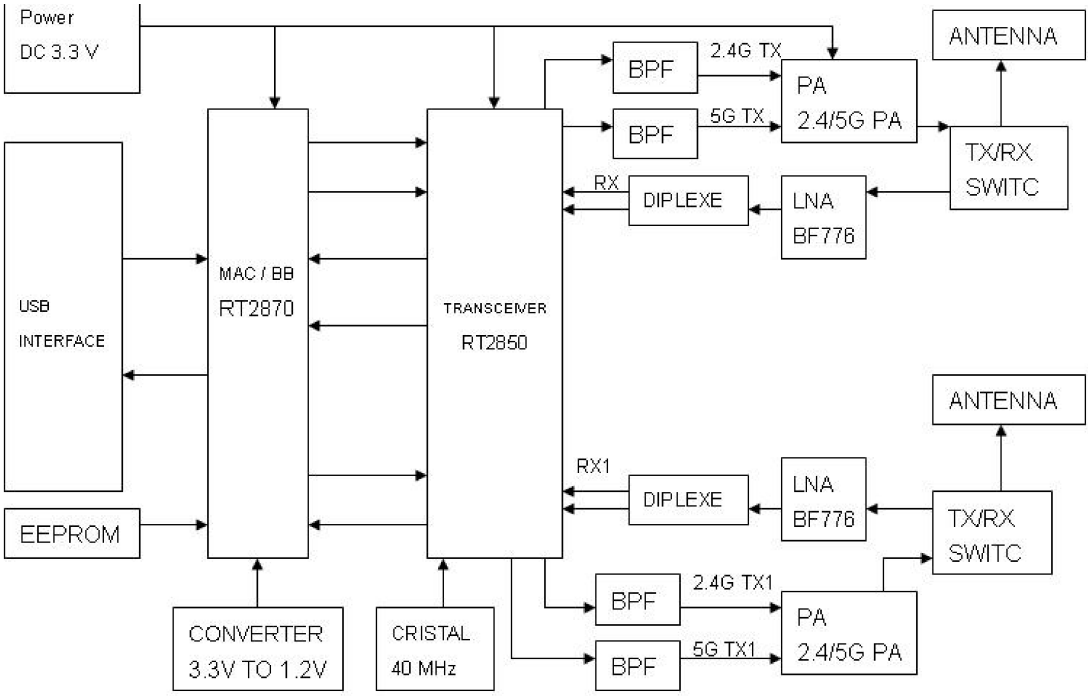


Figura 3.3: Schema a blocchi dell' adattatore WUBR-506N.

3.2 CAN Bus

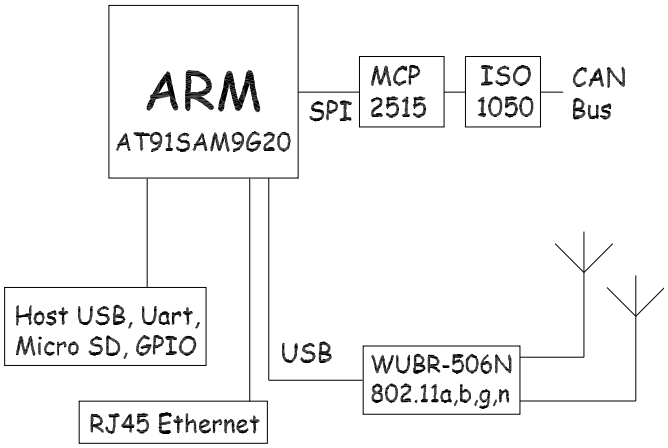


Figura 3.4: Schema di collegamento della cascata processore MCP2515 e ISO1050.

Il Controller Area Network (CAN) noto anche come CAN-bus, è uno standard seriale per bus di campo (principalmente in ambiente automotive), di tipo multicast, introdotto negli anni ottanta dalla Bosch, per collegare diverse unità di controllo elettronico ECU (le comuni centraline per autovetture). Il CAN è stato espressamente progettato per funzionare senza problemi anche in ambienti fortemente disturbati dalla presenza di onde elettromagnetiche. L'immunità ai disturbi EMC può essere ulteriormente aumentata utilizzando cavi di tipo twisted pair (doppino intrecciato). Nel 1993 è divenuto uno standard internazionale ISO ed oggi è ampiamente utilizzato sia nel settore industriale, sia in particolare in campo automobilistico per la connessione dei dispositivi che presiedono al controllo dei veicoli. Il CAN trasmette dati con un modello basato su bit dominanti e recessivi, in cui i bit dominanti sono gli 0 logici ed i bit recessivi sono gli 1 logici. Se un nodo trasmette un bit dominante ed un altro un bit recessivo, allora il bit dominante vince fra i due (realizzando una combinazione AND logico). Con questa tecnica, quando viene trasmesso un bit recessivo, e contemporaneamente un altro dispositivo trasmette un bit dominante, si ha una collisione, e solo il bit dominante è visibile in rete (tutte le altre collisioni sono invisibili). In pratica avviene che un bit dominante è asserito dalla generazione di una tensione fra i conduttori, mentre un bit recessivo è semplicemente ignorato. Si è così sicuri che ogni volta che si impone una differenza di potenziale, tutta la rete la rileva, e quindi è in grado di riconoscere che si tratta di un bit dominante. Per riuscire ad interfacciarci abbiamo optato per una soluzione che

prevede l'utilizzo di due componenti posti in cascata tra loro: l'MCP2515 e l'ISO1050. L'MCP2515 é un controller stand-alone utilizzato per semplificare le applicazioni dove è richiesto l'interfacciamento su un bus CAN; in comunicazione con l'unità di controllo attraverso il protocollo SPI. Mentre l'ISO1050 è un transceiver CAN isolato galvanicamente; questo dispositivo offre la capacità di trasmissione e ricezione differenziale ad un controller CAN potendo sopportare trasmissioni con velocità sino a 1 Mbps. L'isolamento galvanico è reso necessario dall'ambiente dove dobbiamo andare ad operare, pensiamo infatti se una scarica elettrostatica o un picco di tensione va a colpire il nostro bus, per via dei collegamenti elettrici con tutta la scheda comporterebbe sicuramente danni ingenti. Il funzionamento di questi due componenti semplifica moltissimo il controllo della comunicazione, infatti dato che l'MCP2515 è dotato di un pin su cui trasmettere eventuali interrupt, quando questo avrà ricevuto un messaggio dal bus CAN ritenuto valido, invierà il segnale di IRQ al processore; il quale lo andra a leggere dall'interfaccia SPI per la ricezione.

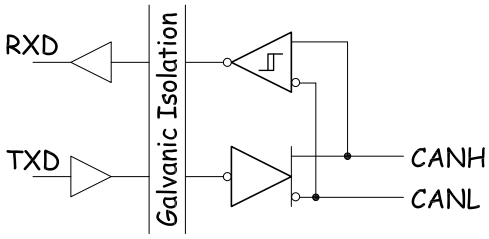


Figura 3.5: Schema dell'ISO1050.

3.3 Storage

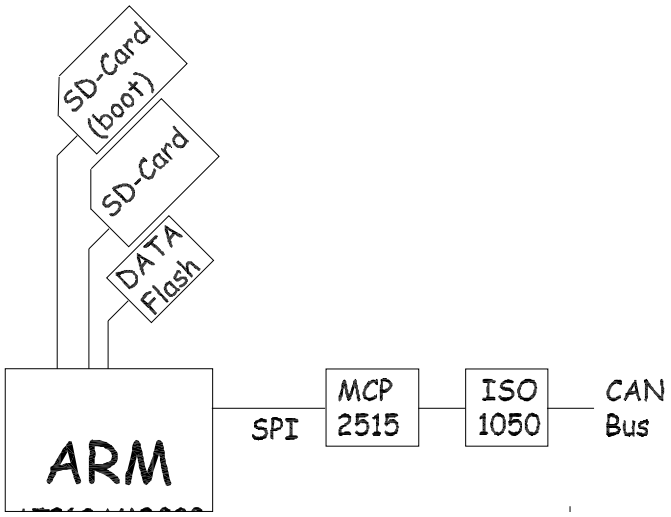


Figura 3.6: Schema di collegamento delle memorie con il processore.

Dal confronto fra Microcontrollore e processori ARM si nota come quest'ultimo sia quasi del tutto sprovvisto di memoria, quindi si rende necessario per il funzionamento, l'utilizzo di memorie esterne. I requisiti del sistema ci impongono di dotare la nostra OBU di almeno un supporto removibile; ma dato che il processore poteva essere collegato con differenti supporti removibili e anche una memoria flash fissa, abbiamo deciso di sfruttare appieno queste potenzialità. Per quello che riguarda la memoria flash il percorso è stato alquanto semplice, infatti questa dialoga con l'ormai noto bus SPI. La scelta della memoria removibile ha comportato un'analisi più approfondita: infatti il processore può interfacciarsi sia con memorie di tipo SD-Card che utilizzano un bus di tipo seriale a 5 terminali (o all'occorenza utilizzando sempre il bus SPI ma con una velocità di trasmissione minore rispetto allo standard SDIO); oppure con memoria di tipo Compact Flash che invece utilizzano un bus parallelo a 44 pin. L'utilizzo di quest'ultima soluzione, anche se permetterebbe una più alta capacità di storage (le memorie CF hanno una capacita massima di 64GB contro i 16GB delle SD), andrebbe a ridurre in maniera drastica le possibilità di future espansioni (per via del maggiore numero di pin da connettere con la memoria e quindi non più disponibili). Quanto detto ha fatto propendere la nostra scelta verso le SD-card. Inoltre sfruttando appieno le capacità del processore (che dispone di due bus SD indipendenti) ne sono state installate due unità: la prima prevista nel BIOS per il boot, del sitema operativo ed in generale per tutte quelle routine e programmi atti

al funzionamento del sistema, la seconda per lo storage dei dati.

3.4 Piattaforma ARM

Oltre alle interfacce appena citate, in fase di sviluppo ne saranno necessarie anche altre, questo poichè non avendo un display, si deve disporre di un terminale dove poter leggere i dump dei processi. Per riuscire a semplificare la fase di debug abbiamo dotato la nostra board di numerose intrefaccie: dalle porte seriali, dove, tramite un semplice terminale (Hyper Terminal in Windows e Minicom in Unix) possiamo andare a leggere tutte le operazioni effettuate dalla board, dalla porta Ethernet con la possibilità con quest'ultima anche di trasferire dati nelle memorie. Quest'ultima interfaccia inoltre ci dà la possibilità, ancora più importante, di creare un bridge tra le reti veicolare e quella fissa. Vediamo ora in Figura 3.4 una schema a blocchi che rappresenta l'architettura definitiva della nostra board:

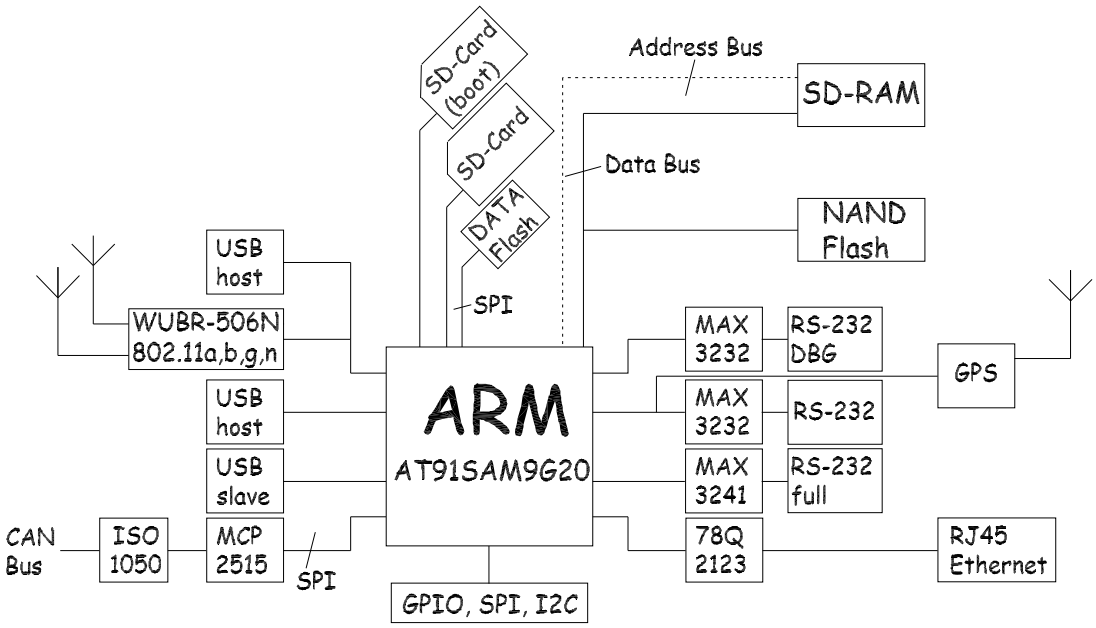


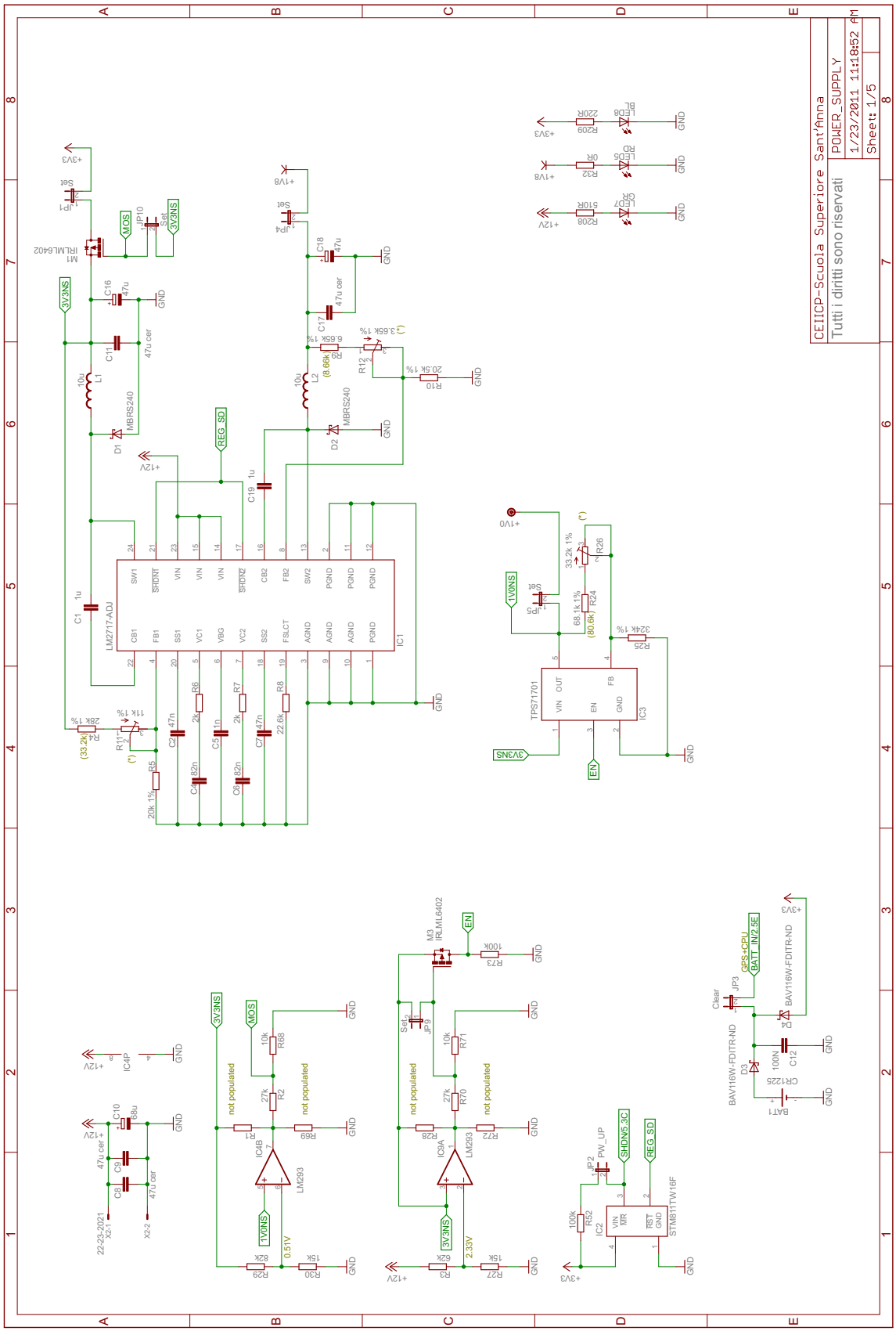
Figura 3.7: Schema a blocchi della On Board Unit.

3.5 Schema elettronico

Il disegno dello schema elettronico, assieme al successivo sbroglio, possono essere considerate le parte fondamentali del nostro progetto.

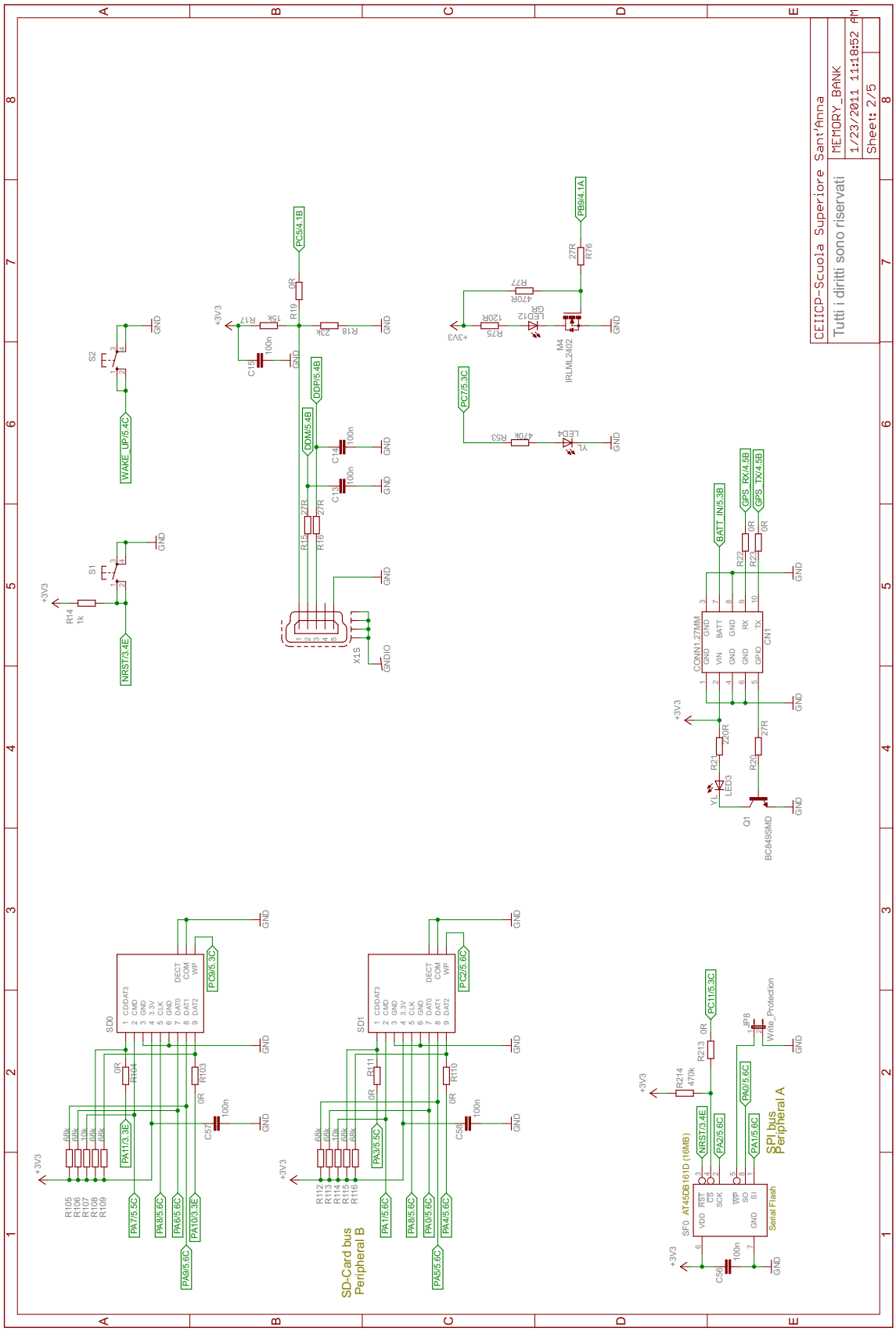
3.5.1 Stadio Alimentazione

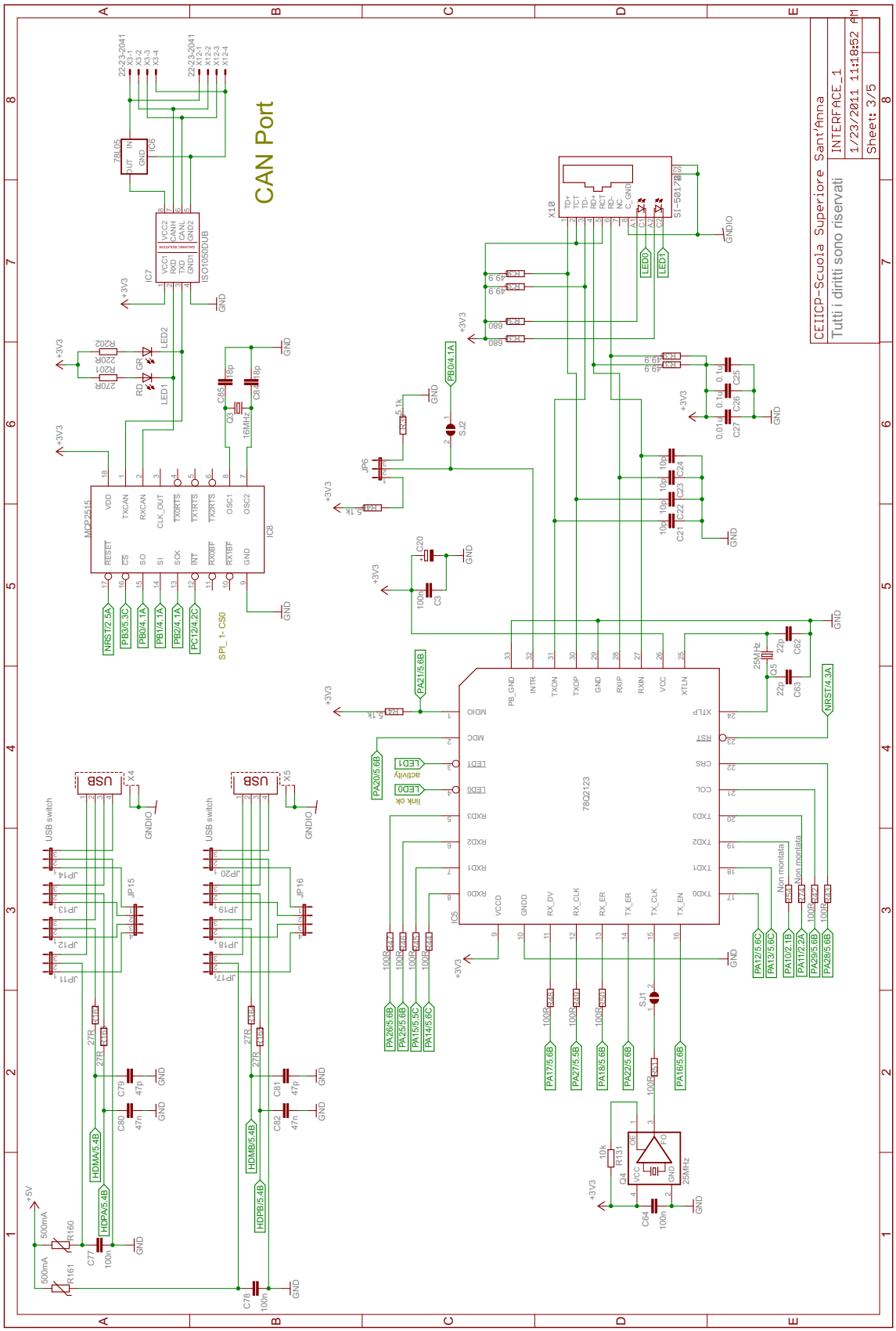
La parte di alimentazione ha riservato i maggiori problemi: infatti sono state necessarie ben 4 alimentazioni differenti, due per l'ARM (1V e 3.3V) una per le memorie RAM (1.8V) ed infine una 5V per alimentare i dipositivi USB. queste sono state ottenue da 3 alimentatori switching e una da regolatori lineari. Un ulteriore problema risiedeva nel fatto che l'ARM deve ricevere le alimentazioni: la 3.3 V per le periferiche e la 1 V per il core, in contemporanea. Per questo motivo si è reso necessario l'utilizzo di due amplificatori operazionali (visibile in B-C 1,2,3) funzionanti come comparatori di tensione: se la 3.3 V non è presente, neanche la tensione da 1 V verrà erogata al processore e viceversa. Infatti le due tensioni vengono erogate da alimentatori differenti nell'unico modo per aveve un controllo di massima sulle tensioni. L'ultima parte non convenzionale dello schema è rappresentatato dall'integrato STM811 il quale ci permette di effettuare lo shutdown del sistema (e il successivo riavvio) via software.

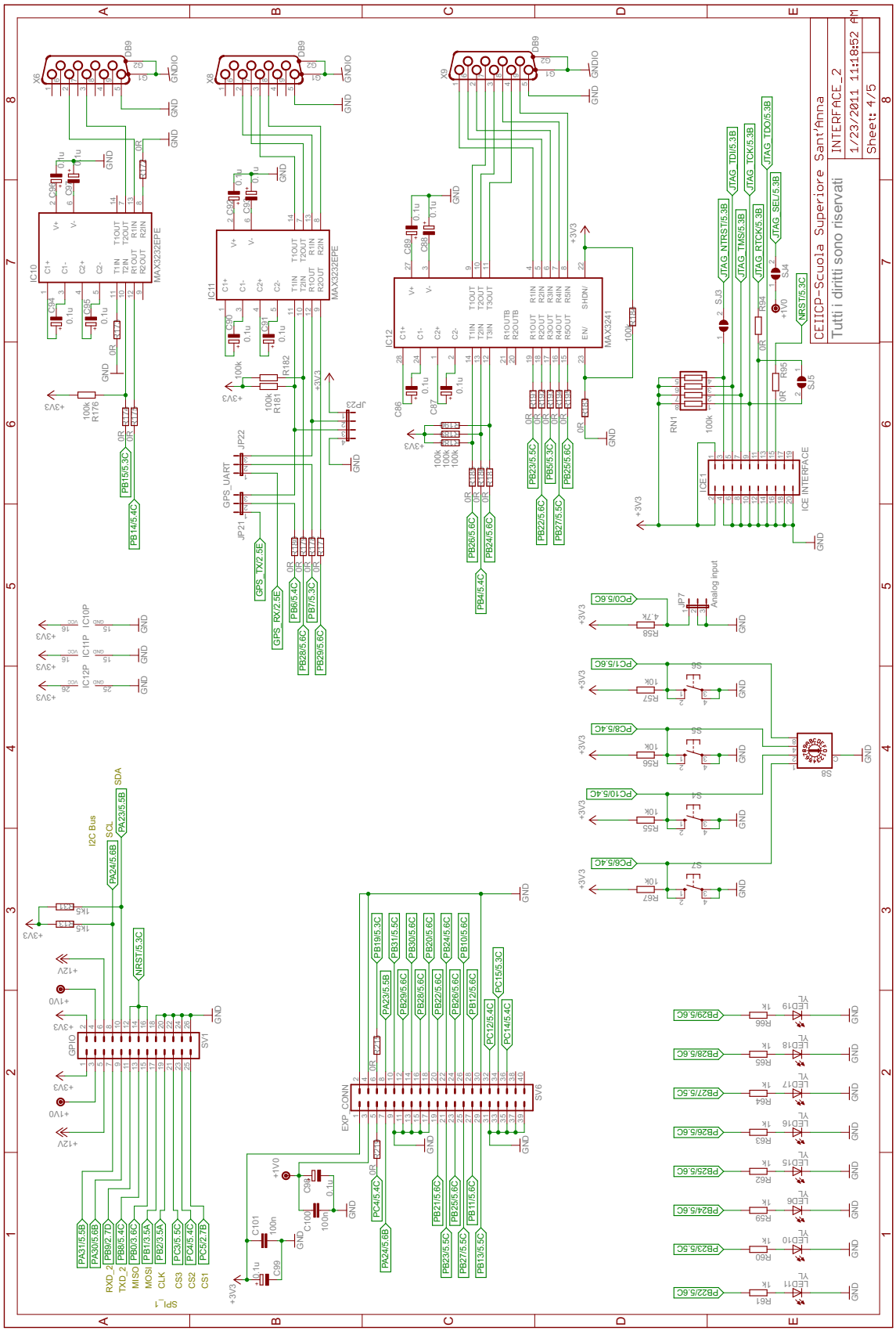

CEIICP-Scuola Superiore Sant'Anna
Tutti i diritti sono riservati
POWER_SUPPLY
1/23/2011 11:18:52
Sheet 1/5
LM2717-ADJ
IC1
TPS71701
IC3
LM293
IC4B
IC3A
IC4P
IC2
STM8111TW16F
IRLML6402
M1
M3
MBRS240
D1
D2
BAV116W-FDITR-ND
D3
D4
BAT1
CR1225
22-23-2021
X2-1
X2-2
GND
+12V
+3V3
+1V8
+1V0
3V3NS
1V0NS
MOS
REG_SD
EN
SHDN5.3C
PW_UP
GPS+CPU
BATT_IN/2.5E
Clear
not populated

3.5.2 Interfacce

Le interfaccie, già ampiamente descritte al capito precedente sono state implementate avvelendoci dei datasheet e dalle applications notes dei vari integrati e del processore. Si noti che nelle connessioni dei due host USB e di una seriale, sono presenti appositi jumper per poter connettere il dispositivo anche su di un connettore strip line (o a pettine), per poter utilizzare anche schede previste per l'integrazione diretta su circuito stampato.


CEIICP-Scuola Superiore Sant'Anna
Tutti i diritti sono riservati
MEMORY_BANK
1/23/2011 11:18:52 AM
Sheet: 2/5
SD-Card bus
Peripheral B
SPI bus
Peripheral A
Serial Flash
AT45DB161D (16MB)
CONN1 27MM
BC849SMD
IRLML2402
Write_Protection

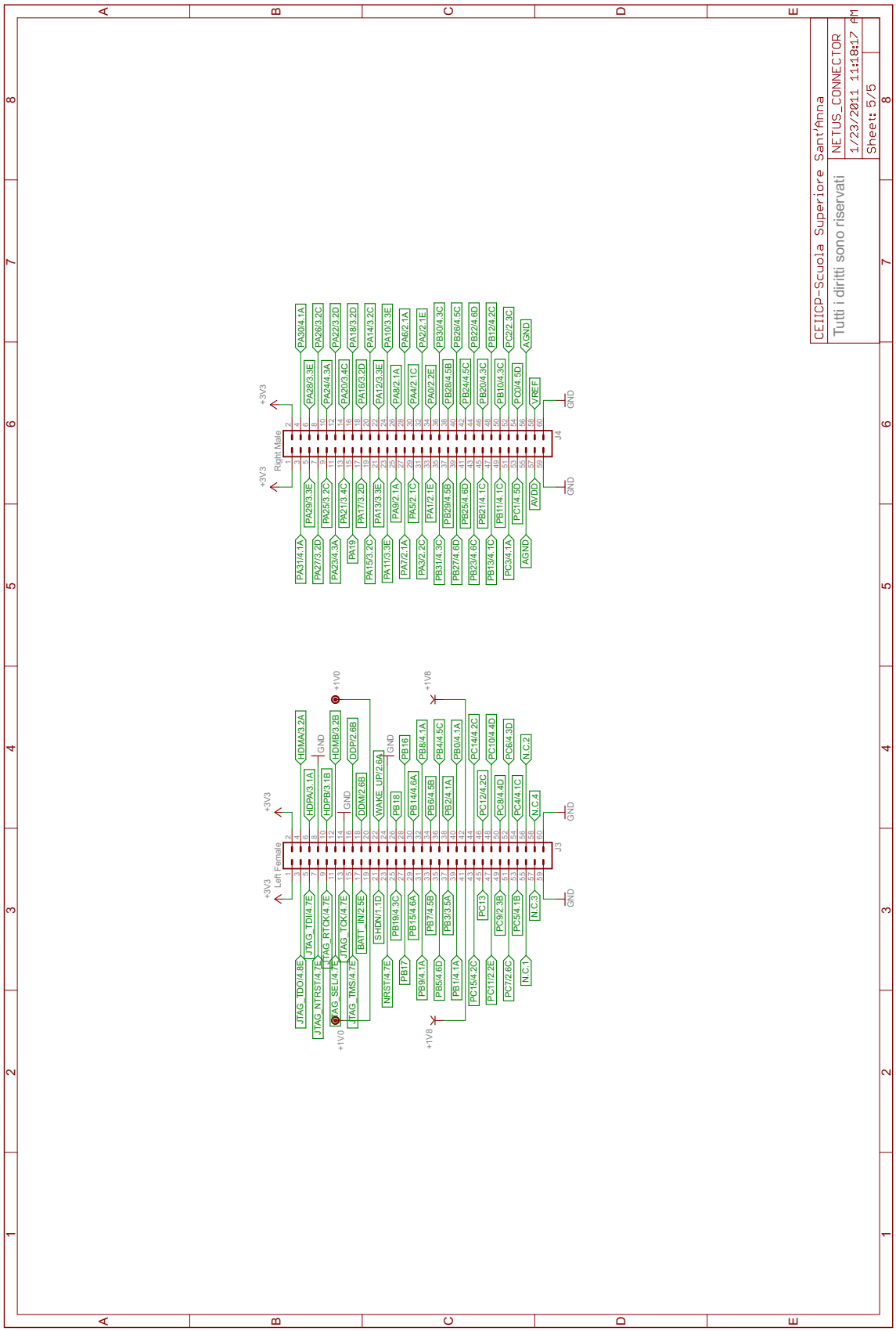

CAN Port
CEIICP-Scuola Superiore Sant'Anna
Tutti i diritti sono riservati
INTERFACE_1
1/23/2011 11:18:52
Sheet: 3/5
MCP2515
ISO1050DUB
78Q2123
USB
USB switch
SPI_1- CS0
LED0
LED1
activity
link ok
Non montata
GND
+3V3
+5V
GNDIO


CEIICP-Scuola Superiore Sant'Anna
Tutti i diritti sono riservati
INTERFACE_2
1/23/2011 11:18:52
Sheet: 4/5
MAX3232EPE
MAX3241
ICE INTERFACE
GPIO
EXP CONN
I2C Bus
SPI_1
GPS_UART
JTAG_TDI/5.3B
JTAG_TCK/5.3B
JTAG_TDO/5.3B
JTAG_NTRST/5.3B
JTAG_TMS/5.3B
JTAG_RTCK/5.3B
JTAG_SEL/5.3B
NRST/5.3C
Analog input

3.5.3 Processore

Il processore insieme alle memorie necessarie per il suo funzionamento (SDRAM e flash), sono collocate su di una scheda esterna. Infatti per via dei brevissi tempi di prototipizzazione, non sarebbe stato possibile collocare il processore direttamente sulla nostra board. I problemi sono molteplici: il più importante risiede nel fatto che il processore dialoga con le memorie ad una velocità di 133 MHz; in questo frangente, le piste non possono essere più viste come semplici 'fili' ma bensì, sono vere e proprie linee di trasmissione, che necessitano adattamenti, spessori e terminazioni adeguate. La scheda CPU da noi utilizzata, è la Netus Board G20 che equipaggia la nota FoxBoard.[1]

[1] inserire url


CEIICP-Scuola Superiore Sant'Anna
Tutti i diritti sono riservati
NETUS_CONNECTOR
1/23/2011 11:18:17 AM
Sheet: 5/5
Left Female
Right Male
J3
J4
+3V3
GND
+1V0
+1V8
JTAG_TDO/4.8E
JTAG_NTRST/4.7E
JTAG_SEL/4.7E
JTAG_TMS/4.7E
JTAG_TDI/4.7E
JTAG_RTCK/4.7E
JTAG_TCK/4.7E
BATT_IN/2.5E
NRST/4.7E
PB17
PB9/4.1A
PB5/4.6D
PB1/4.1A
PC15/4.2C
PC11/2.2E
PC7/2.6C
N.C.1
SHDN/1.1D
PB19/4.3C
PB15/4.6A
PB7/4.5B
PB3/3.5A
PC13
PC9/2.3B
PC5/4.1B
N.C.3
HDMA/3.2A
HDPA/3.1A
HDPB/3.1B
ODM/2.6B
WAKE_UP/2.6A
PB18
PB14/4.6A
PB6/4.5B
PB2/4.1A
PC12/4.2C
PC8/4.4D
PC4/4.1C
N.C.4
HDMB/3.2B
ODP/2.6B
PB16
PB8/4.1A
PB4/4.5C
PB0/4.1A
PC14/4.2C
PC10/4.4D
PC6/4.3D
N.C.2
PA31/4.1A
PA27/3.2D
PA23/4.3A
PA19
PA15/3.2C
PA11/3.3E
PA7/2.1A
PA3/2.2C
PB31/4.3C
PB27/4.6D
PB23/4.6C
PB13/4.1C
PC3/4.1A
AGND
PA29/3.3E
PA25/3.2C
PA21/3.4C
PA17/3.2D
PA13/3.3E
PA9/2.1A
PA5/2.1C
PA1/2.1E
PB29/4.5B
PB25/4.6D
PB21/4.1C
PB11/4.1C
PC1/4.5D
AVDD
PA30/4.1A
PA26/3.2C
PA22/3.2D
PA18/3.2D
PA14/3.2C
PA10/3.3E
PA6/2.1A
PA2/2.1E
PB30/4.3C
PB26/4.5C
PB22/4.6D
PB12/4.2C
PC2/2.3C
AGND
PA28/3.3E
PA24/4.3A
PA20/3.4C
PA16/3.2D
PA12/3.3E
PA8/2.1A
PA4/2.1C
PA0/2.2E
PB28/4.5B
PB24/4.5C
PB20/4.3C
PB10/4.3C
PC0/4.5D
VREF

Capitolo 4

Piattaforma Software

Come piattaforma software intendiamo l'insieme di sistema operativo, librerie e applicazioni che andremo ad installare sulla piattaforma ARM, oltre alla loro configurazione per realizzare l'obiettivo desiderato.

4.1 Sistema Operativo

Per riuscire a sfruttare apieno le pontenzialità della nostra archittetura, come predentemente introdotto, necessitiamo di un sistema operativo completo e non di un firmware, per quanto questo possa essere complesso.
Sempre più spesso, i processori di tipo ARM equipaggiano gli oggetti che giornalmente andiamo ad uilizzare: dagli smartphone, ai decoder televisivi passando per i router wireless. Tutti questi dispositivi fanno parte, insieme naturalmente a quelli studiati per applicazioni di tipo industriale, di quel ramo di tecnologia che viene definita elettronica embedded. I dispositivi sopra citati, devono disporre al loro interno di un sistema operativo; che a seconda delle interfaccie presenti sara più o meno complicato. Da questa problematica nasce lo sviluppo di una miriade di S.O. definiti appunto embedded studiati per tutti quei sistemi dove le risorse sono limitate.
In questa panoramica si distinguono quei S.O. che basano il loro kernel su Linux (e dunque open source). Fra questi si distingue, per via dei numerosi driver che

man mano vengono implementati e grazie ad una vastissima comunità di sviluppo, OpenWRT.

4.2 OpenWRT

Il sistema operativo che si è deciso di adottare è OpenWRT[1], una distribuzione Linux per sistemi embedded, nella versione trunk [2].
OpenWrt utilizza i più comuni strumenti di Linux embedded, come uClibc, busybox, interprete shell e fornisce un livello di astrazione hardware e gestore di pacchetti. Vediamo in Tabella4.1 il software stack che utilizza OpenWrt:

UCI	IPGK	User Program
Busybox		
uClibc		
Linux kernel		

Tabella 4.1: Stack software di OpenWRT

Ogni architettura utilizza un kernel Linux diverso; ma permette la condivisione tra più dispositivi dell'ambiente user-space. Pertanto basterà ricompilare solo le uClibc e le applicazioni per ottenere gli stessi programmi in esecuzione su un dispositivo embedded completamente diverso.

4.2.1 Configurazione del sistema e dei pacchetti

UCI che sta per Unified Configuration Interface è una libreria C che permette la configurazione dello spazio utente e del sistema di gestione. Queste sono adottate in OpenWRT poichè possono essere utilizzate in tutti quei sistemi che non disponendo di memorie NVRAM (Non Volatile Random Access Memory), devono memorizzare i file

[1] http://openwrt.org/
[2] https://svn.openwrt.org/openwrt/trunk/

di configurazione su memorie flash; inoltre risultano facilmente integrabili all'interno di sistemi esistenti o per sviluppare configurazioni di storage.
OpenWRT ha il vantaggio di essere facilmente modificabile e configurabile, sia in fase di compilazione che dopo l'installazione; in particolare supporta l'installazione di nuovo software, compatibilmente con le poche risorse disponibili, sia attraverso un sistema autonomo, ipkg, basato sul sistema di gestione pacchetti di Debian, sia attraverso la copia di nuovi files sulla flash, gestita attraverso un filesystem JFFS (Journalling Flash File System). Permette anche l'accesso tramite web attraverso ed SSH.

4.2.2 Compilazione

I sorgenti possono essere agevolmente compilati anche su una piattaforma Linux, ma richiedono diversi prerequisiti.[3] Il configure di OpenWRT controlla automaticamente la presenza di queste dipendenze. Una volta scaricata la cartella di OpenWRT e lanciato il comando `make menuconfig`" si entra nel menù di configurazione basato su ncurses; dove potremmo poi navigare nei vari sottomenù per poter selezionare tutti i pachetti desiderati.
Dovremo inoltre attivare nella configurazione le librerie necessarie per i programmi che intendiamo compilare; una parte fondamentale riguarda i driver USB per l'interfaccia wireless.
Inoltre dovremmo andare a selezionare l'abilitazione dell'interfaccia SPI:
Il menuconfing ci permette di abilitare moltissime interfaccie (presenti come moduli aggiuntivi al kernel): I^2C, UART, 1-wire, ecc; nella nostra implementazione saranno find'ora abilitate prevedendo upgrade futuri.
Per finire si installa l'interprete Python e uno tra gli editor testuali più famosi in ambiente Unix: vim.
La configurazione prodotta è memorizzata nel file ".config", ed è possibile copiarla per successive installazioni. Una semplice invocazione di make senza argomenti a

[3] I pacchetti richiesti come dipendenza sono g++, ncurses, zlib, bison,flex, autoconf, wget, GNU awk (gawk), GNU tar (gtar), GNU patch(gpatch) e le GNU findutils.

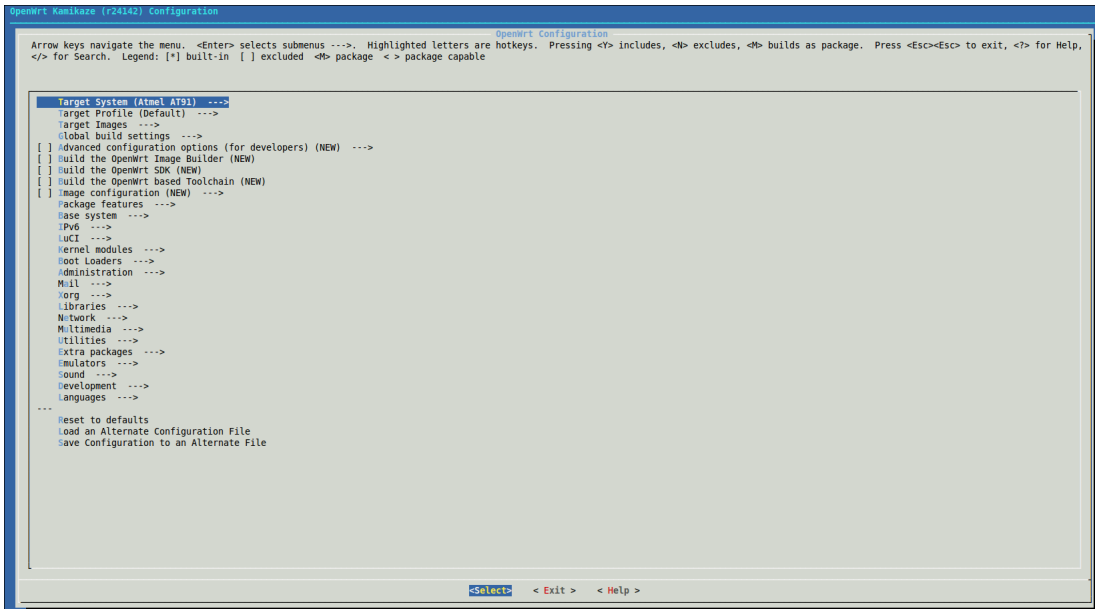


Figura 4.1: Schermata principale del menù di configurazione

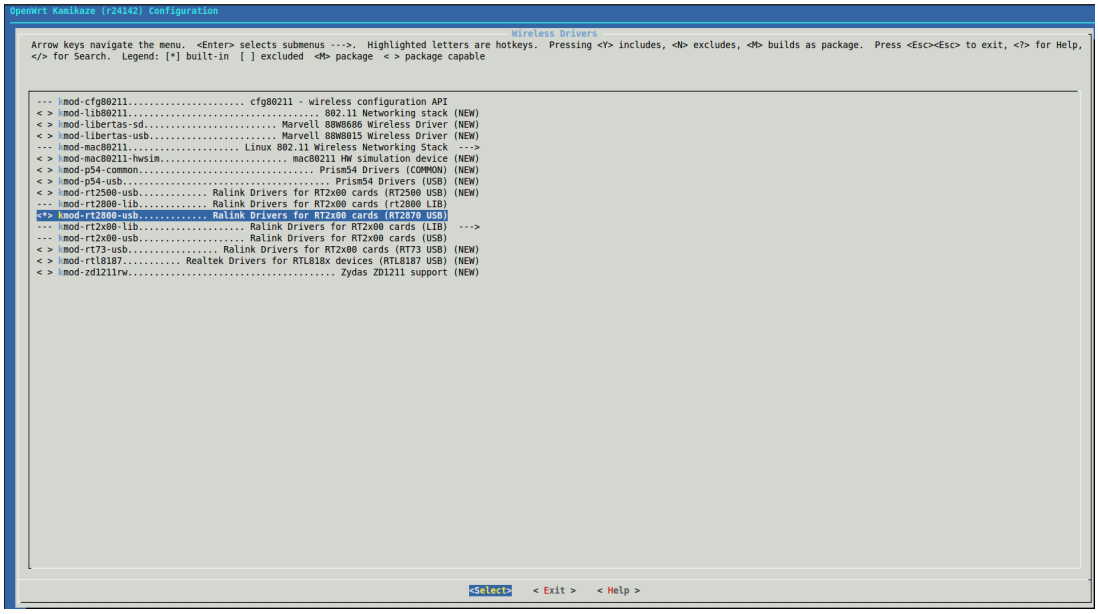


Figura 4.2: Inserimento dei moduli per l'interfaccia wireless

questo punto sarà sufficiente a compilare tutta la catena per mips-linux, il kernel e le librerie. Qualora make sia invocato senza argomenti ma il file ".config" risulti assente, il Makefile istruisce il programma ad eseguire preventivamente la fase di configurazione precedentemente vista. Una volta completata l'installazione, la directory bin/

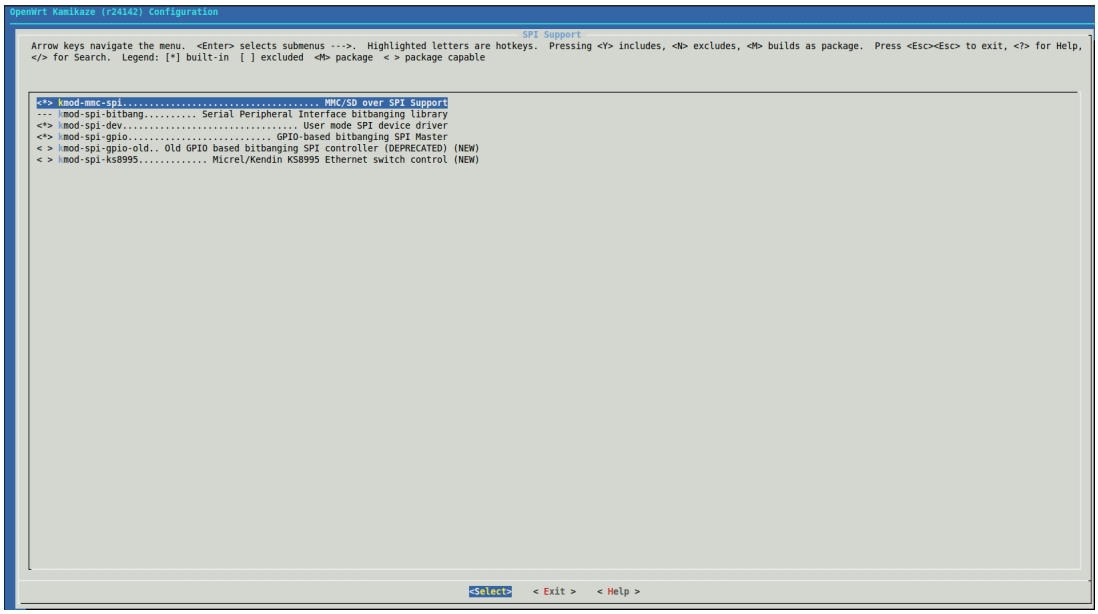


Figura 4.3: Abilitazione dell'interfacia SPI

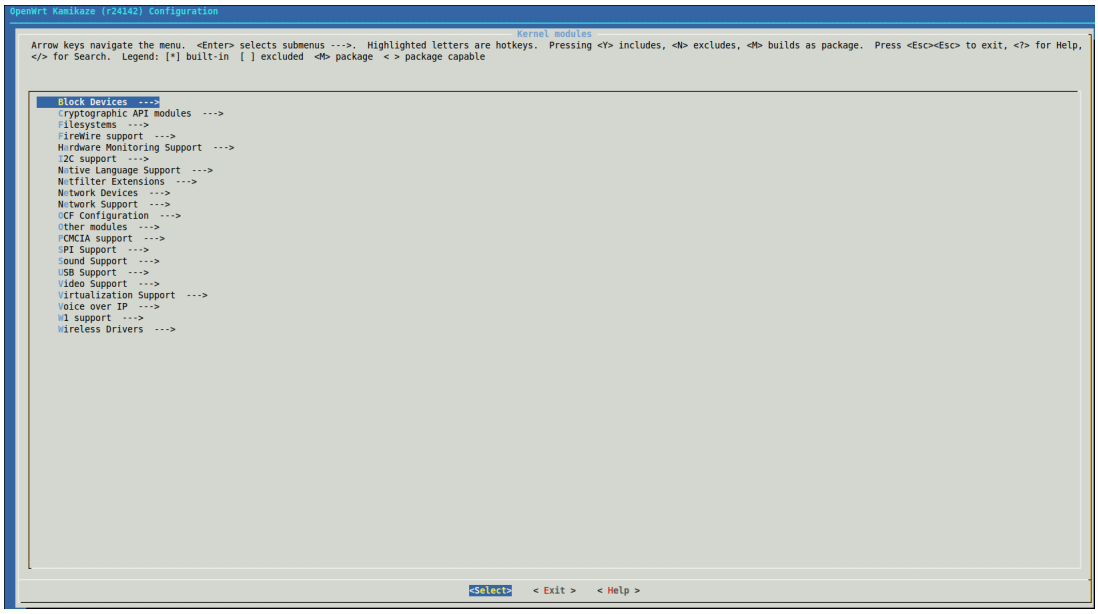


Figura 4.4: Sottomenu dei moduli del kernel

conterrà, oltre all'immagine del sistema operativo, anche il file uImage necessario in fase di boot del sistema.

Dopo questo prima compilazione, si dovrà eseguire il comando make kernel_menuconfig per andare ad abilitare ulteriori moduli del kernel (che non possono essere abilitata

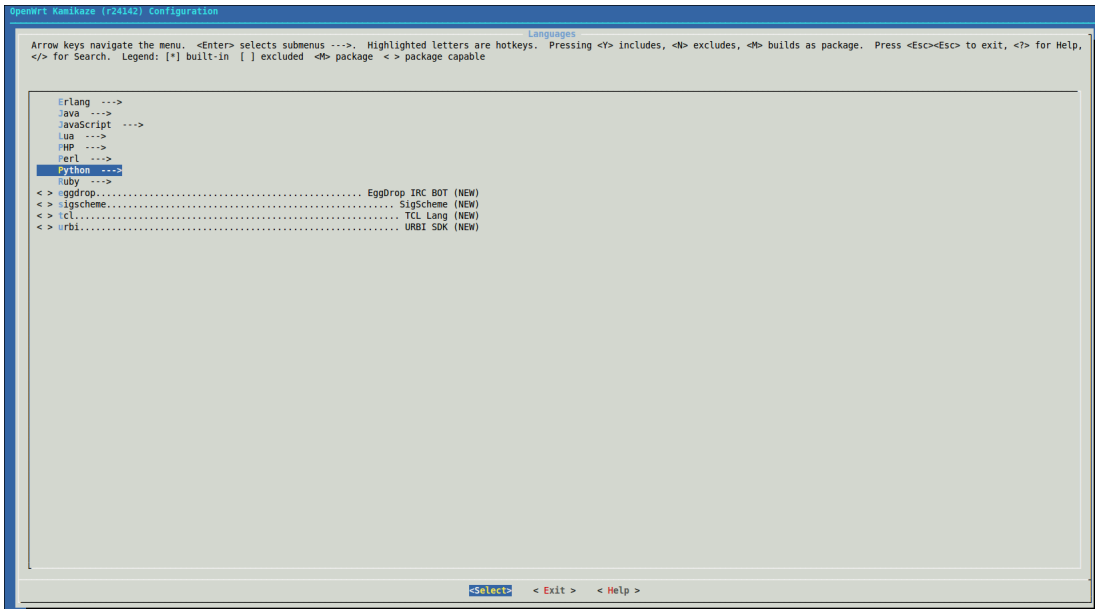


Figura 4.5: Interprete Python

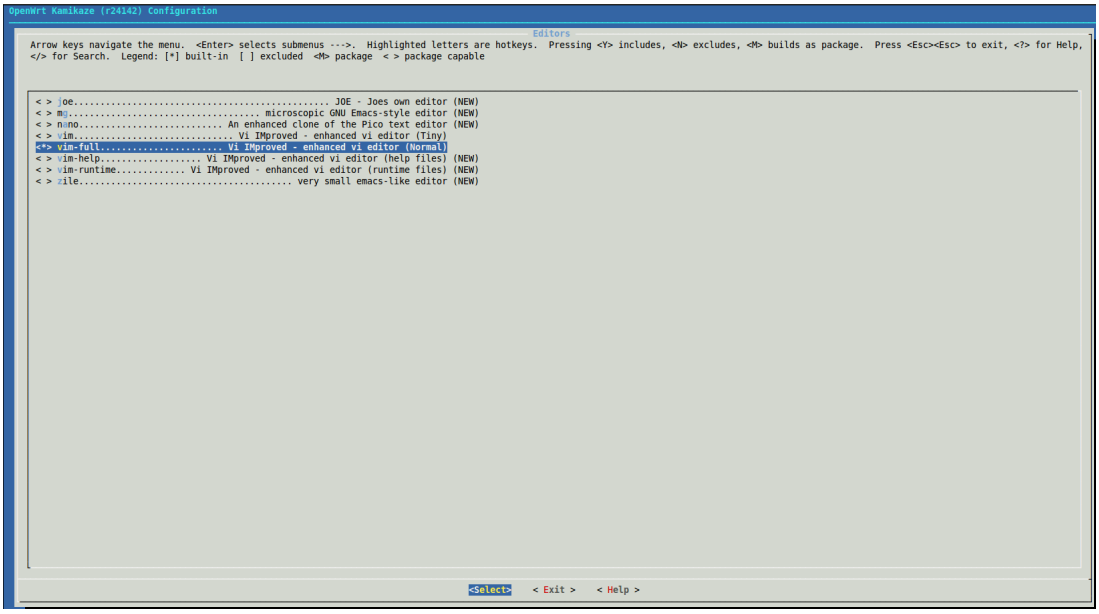


Figura 4.6: Installazione di VIM

prima che il sistema sia stato compilato almeno una volta dopo il download). In questo secondo menù andremo come visibile nelle figure, ad abilitare i driver etherne, i moduli del CAN bus e i dettagli del sistema.

4.3 Cross-compilazione dei programmi

La nostra piattaforma hardware è dotata degli strumenti per la compilazione (gcc, g++, ecc), ma date le limitate capacità computazionali, rispetto ad un pc desktop non è adatta alla compilazione del software: questa fase deve essere svolta dunque su una piattaforma esterna in crosscompilazione. Questa è una tecnica con la quale si compila un codice sorgente con un cross-compilatore, ottenendo così un file binario eseguibile su di un elaboratore con architettura diversa da quella della macchina su cui si è lanciato il cross-compilatore stesso. La cross-compilazione viene usata anche quando è necessario compilare un programma per un sistema operativo differente dalla macchina su cui si trova il compilatore e linker. La macchina che esegue il cross-compilatore viene detta host mentre la macchina sulla quale si eseguirà il binario ottenuto dal processo di cross-compilazione è detta target. Per una cross-compilazione di base con i tool del progetto GNU è necessario avere una toolchain completa di cross-compilatore, delle binutils più la glibc (questi ultimi compilati anch'essi per generare codice per il target). Nel caso in cui invece, si vogliano compilare applicazioni più complesse, come le applicazioni grafiche o che necessitano comunque di particolari librerie extra, è necessario aggiungere alla propria toolchain anche queste librerie preventivamente cross-compilate. Anche in questo caso ci viene in aiuto il menuconfig di OpenWRT che permette oltre che a generare i file del S.O. anche quelli per la toolchain.

4.4 Esempi di applicazioni

La OBU prototipata permetterà dunque lo sviluppo di un set di applicativi che presuppongono la comunicazione wireless fra un sistema intelligente di trasporto ed un veicolo. In Figura 4.8 sono riportati in maniera pittorica alcuni esempi di applicativi per le reti veicolari:

- Sistemi di navigazione OEM possono fruire dell'informazione digitale acquisita in tempo reale dall'ITS attraverso l'interfaccia IEEE802.11a/p module; thin

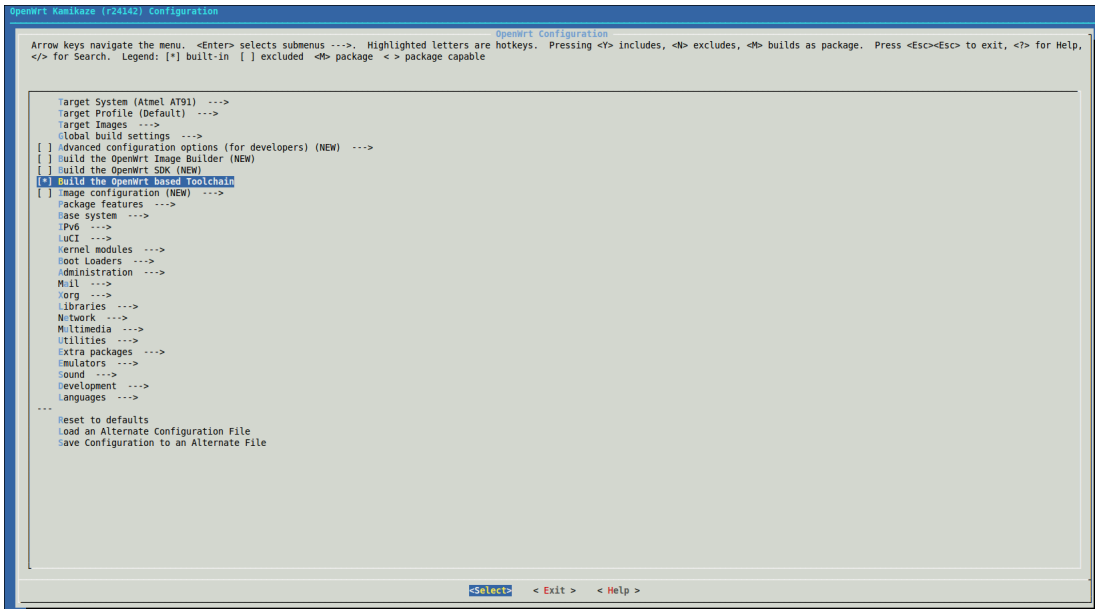


Figura 4.7: Compilazione della Toolchain

applications del tipo “park finder” (Figura 4.8/a) e “path discovery” (Figura 4.8/b) risulterebbero più accurate dal momento che farebbero uso di dati freschi piuttosto che di serie storiche;

- In caso di incidente o di altro pericolo il sistema avverte con un segnale acustico e visuale agendo sulla dashboard e sugli altoparlanti; nel caso in cui sistemi di sicurezza attiva siano in funzione, la OBU agisce sulla centralina e frena il veicolo;

- Il valore attuale di consumo medio del veicolo viene utilizzato per dare le informazioni di autonomia residua sulla dashboard; nel caso in cui, dall’ITS provengano informazioni riguardanti il traffico ed il profilo altimetrico del percorso rimanente, queste possono essere essere utilizzate per estrarre previsioni di autonomia più precise.

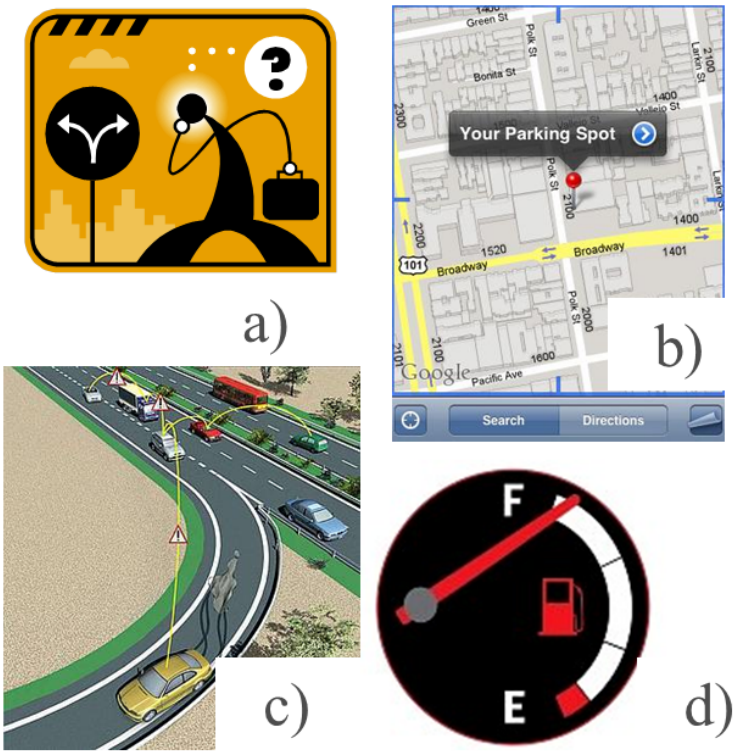


Figura 4.8: Compilazione della Toolchain

Capitolo 5

Conclusioni

In questa tesi è stata presentata la progettazione e la messa in funzione di una On-Board Unit che equipaggia un core computazionale di tipo ARM, un'interfaccia di comunicazione con la rete veicolare conforme allo standard IEEE802.11a/p ed un'interfaccia per la comunicazione intraveicolare attraverso il bus CAN.
Dopo aver presentato lo scenario applicativo (Capitolo 1), i requisiti di progetto (Capitolo 2), si sono valutate le scelte di architettura mostrando gli schematici ed il layout della OBU (Capitolo 3). In ultimo (Capitolo 4) è stata presentata la tool chain software basata su Linux Embedded e si sono proposti alcuni esempi applicativi.
Per lo svolgimento di questa tesi, si è acceduto al Know-How della Scuola Superiore Sant'Anna e del Consorzio Nazionale Interuniversitario per le Telecomunicazioni che mantengono le proprietà di sfruttamento e di trasferimento tecnologico del prototipo qui presentato.
Il progetto di ricerca IPERMOB della Scuola Superiore Sant'Anna di Pisa prevede di applicare la tecnologia sviluppata in questa tesi su opportuni casi di studio legati alla infomobilità e alla gestione del traffico urbano.