ALMA MATER STUDIORUM – UNIVERSITA' DI BOLOGNA
SEDE DI CESENA

SECONDA FACOLTA' DI INGEGNERIA CON SEDE A CESENA
CORSO DI LAUREA IN INGEGNERIA ELETTRONICA E TELECOMUNICAZIONI PER LO SVILUPPO SOSTENIBILE

PROGETTO DI SCHEDA A MICROCONTROLLORE PER LA TELEMETRIA DI VEICOLI ELETTRICI

Elaborato in:

Reti di Telecomunicazione L-M

Relatore:

Prof. Ing. Callegati Franco

Co-relatore

Dott. Ing. Ramilli Marco

Presentata da:

Zannoni Luca

Terza Sessione

Anno accademico 2010 / 2011

A tutti coloro che mi hanno
sopportato e, soprattutto, supportato

INDICE

INDICE.....5

1 INTRODUZIONE.....7

2 DESCRIZIONE DEL SISTEMA.....11

2.1 HARDWARE.....11

2.2 SOFTWARE.....25

3 REALIZZAZIONE DEL PROGETTO.....27

3.1 SPECIFICHE.....27

3.2 AMBIENTE DI SVILUPPO.....31

3.3 REALIZZAZIONE E LAYOUT.....34

4 CONCLUSIONI.....57

APPENDICE A – ARDUINO UNO.....59

APPENDICE B – ARDUINO ETHERNET.....61

APPENDICE C – CAN BUS SHIELD.....63

APPENDICE D – LIBELIUM GPS SHIELD.....65

BIBLIOGRAFIA.....67

1 INTRODUZIONE

In un'epoca in cui l'informatizzazione si diffonde a macchia d'olio in ogni aspetto della vita quotidiana e la possibilità di essere connessi ad internet risulta vitale per aggiornarsi o anche semplicemente per mantenere contatti è possibile e allo stesso tempo necessario cercare di sfruttare la rete nel migliore dei modi in ambito lavorativo, per migliorare i propri prodotti e cercando di offrire all'utente beni e servizi sempre migliori, al passo coi tempi e col pensiero moderno.
É in questo ambiente che la connettività si rende necessaria anche nel settore dell'automobile in modo da gestire in maniera efficiente l'enorme quantità di dati scambiati dalle varie sottoparti del sistema il cui compito è quello di supervisionare i componenti elettronici e meccanici. L'obiettivo è quello quindi di centralizzare ed elaborare le informazioni in modo da semplificare ed ottimizzare la gestione del veicoli per ottenere importanti vantaggi dalla fase di test fino a quella di utilizzo, passando per quella di manutenzione.
Per questo risulta fondamentale, nell'epoca in cui viviamo, concedere la possibilità al veicolo di interagire con la rete internet in modo da poter sfruttare tutti i vantaggi comunicativi, siano essi con l'ambiente circostante o con persone, che essa prevede.

Una volta quindi trovato il modo di interfacciarsi con la rete e sviluppato un software adeguato è fondamentale implementare fisicamente il dispositivo in modo da ottenere un dispositivo altamente integrabile nel sistema veicolo in modo da non alterare in maniera significativa la disposizione dei componenti di base (meccanici, elettrici ed elettronici) dell'automobile elettrica.
È in quest'ottica che s'inserisce il progetto di una scheda per una vera e propria telemetria del veicolo elettrico con l'obiettivo di ottenere un sistema ad hoc, ma che mantenga una molteplicità di interfacce che permettano al

dispositivo di rimanere aggiornato con l'evoluzione in atto relativa alle tecniche e ai protocolli (standard) di comunicazione permettendo quindi comunicazioni tramite rete ethernet, Wi-Fi o GPRS, cercando anche di sfruttando sistemi di posizionamento come il GPS.

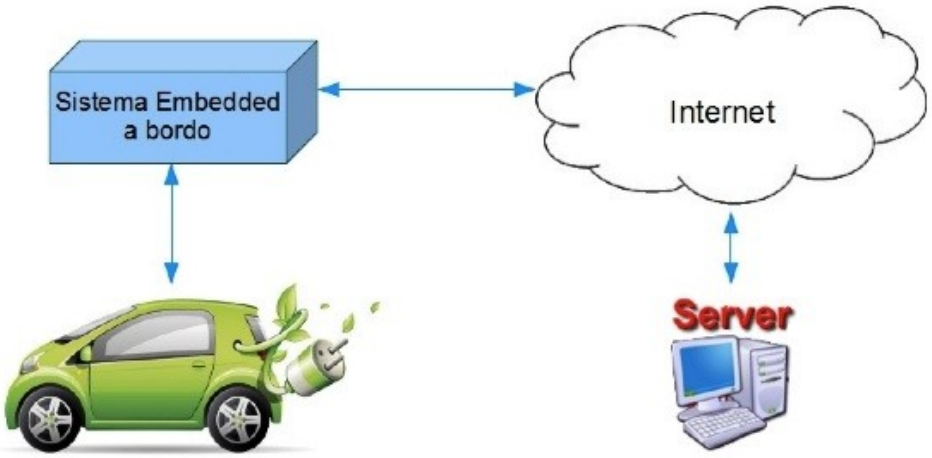


Fig. 1.1: Schema rappresentativo del progetto

Per questo motivo si è cercato di realizzare la scheda seguendo la filosofia dei sistemi embedded, architetture il cui compito è quello di eseguire operazioni molto specifiche spesso con vincoli sull'esecuzione in tempo reale. Questo permette di ridurre al minimo l'hardware in termini di spazio, consumo e costo di realizzazione.
Queste dispositivi sono evoluti recentemente virando sulla creazione di architetture modulari che permettono il riutilizzo delle risorse disponibili; in questo modo si ottengono comunque dispositivi ottimizzati ma in grado di mantenere un certo tipo di flessibilità nello sviluppo delle applicazioni e allargando quindi lo spettro dei possibili impieghi.

Secondo questi principi si cercherà quindi di realizzare la scheda in modo che implementi e realizzi il software dedicato alla comunicazione del veicolo con internet ma che, grazie all'hardware a disposizione, possa essere programmata da mani esperte anche per numerosi utilizzi alternativi e resa quindi disponibile all'utente finale in possibili forme.

In questo è risultato fondamentale l'utilizzo della piattaforma Arduino, basata sul microcontrollore ATmega328, che permette appunto una rapida espansione fisica del sistema come vedremo meglio nel capitolo 2.

2 DESCRIZIONE DEL SISTEMA

Come anticipato il sistema di partenza da cui si è deciso di sviluppare la scheda per la telemetria è basato sulla piattaforma di prototipazione Arduino. La scelta per lo sviluppo del software è ricaduta su questo prodotto per tre fondamentali motivi:

- **open-source:** la possibilità di modificare liberamente qualsiasi tipo di codice senza alcun tipo di ostacolo ha permesso la diffusione della piattaforma e la nascita di una vera propria community con conseguente aumento delle funzionalità del prodotto e diminuzione degli errori nella gestione delle periferiche

- **librerie software:** molte periferiche hardware collegabili con il le schede Arduino sono provviste di librerie per l'ambiente di programmazione che permettono di utilizzare i moduli di espansione in maniera più astratta e facilitano la comunicazione a basso livello con il microcontrollore

- **espandibilità hardware:** il mercato offre moduli di espansione collegabili ad incastro che permettono ad Arduino di avere interfacce diverse, spaziando dalla comunicazione (Ethernet, Bluetooth, Wi-Fi,...) alla sensoristica (attuatori, schermi LCD, telefonia mobile,...)

2.1 HARDWARE

Il progetto di partenza è suddiviso in due sezioni che si occupano rispettivamente di ricevere e rielaborare i dati utili del motore elettrico e di comunicare questi dati ad un database disponibile in rete. La prima è

composta da una scheda Arduino UNO con i moduli di espansione CAN e GPS, mentre la seconda è stata sviluppata con un Arduino UNO espanso con un modulo per la comunicazione via Ethernet. Vediamo di analizzare tutti i moduli coinvolti.

La scheda Arduino UNO, illustrata in Fig. 2.1.1, è il cuore del sistema nel suo complesso occupandosi di tutte le elaborazioni necessarie di entrambe le sezioni.

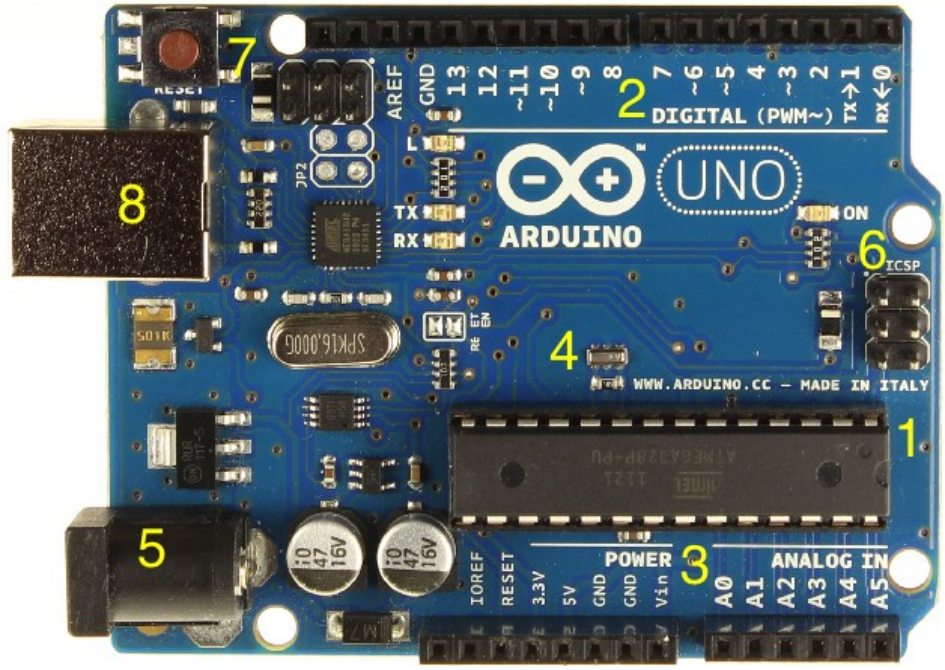


Fig. 2.1.1: vista frontale di Arduino UNO [1]

Il suo nucleo è composto dal microcontrollore ATmega328 (1) che mette a disposizione fino a 14 pin digitali (2) programmabili come ingressi/uscite (6 di essi prevedono la possibilità di essere utilizzati per modulazioni PWM), 6 ingressi analogici (3), un oscillatore a 16MHz (4), un jack per l'alimentazione esterna (5), un connettore ICSP per la programmazione diretta del microcontrollore (6), un pulsante di reset (7) ed un connettore USB (8) utilizzabile sia per alimentare la scheda che per programmare il microcontrollore. Si analizzano di seguito le principali caratteristiche del dispositivo

- **alimentazione:** può essere fornita tramite USB (5 V) o tramite il jack di alimentazione purchè la tensione sia tra i 7 e i 12 V (la scheda sopporta in realtà l'intervallo 6-20 V, ma fuori dal range 7-12 possono esserci problemi di stabilità nei 5 V o il danneggiamento della scheda a causa del surricaldamento dei regolatori). In alternativa l'alimentazione può arrivare da una sorgente esterna collegata al pin VIN.
  Nella scheda è possibile prelevare alimentazioni per dispositivi esterni da a due regolatori che forniscono 3.3 V e 5 V, dal pin VIN o dai pin GND per avere la massa di riferimento della scheda.

- **memoria:** l'ATmega328 ha a disposizione 32 KB di memoria che includono però anche il bootloader, cioè il programma che permette l'avvio del kernel del microcontrollore; dispone anche di 2 KB di SRAM (usate per le variabili e le costanti del software) e 1 KB di EEPROM (accessibile via software grazie alle relative librerie)

- **ingressi/uscite:** i 14 pin digitali possono essere impostati a scelta sia come ingressi che come uscite e supportano al massimo 40 mA; alcuni di essi però hanno anche funzioni specializzate; in particolare i pin 2/3 gestiscono interrupt, i pin 3/5/6/9/10/11 permettono una modulazione PWM, il pin 13 è collegato ad un led

- **comunicazione:** Arduino UNO prevede diversi protocolli di comunicazioni per poter comunicare con dispositivi diversi; i pin 0/1 chiamati anche RX/TX ricevono e trasmettono i dati seriali provenienti dall'integrato 16U2 che si occupa della conversione dei dati provenienti dall'USB in dati seriali (e viceversa) adatti per il microcontrollore; i pin 10/11/12/13 supportano la comunicazione SPI, una sorta di bus sincrono per la comunicazione tra microcontrollori; i pin A4/A5 implementano i protocolli I^2C o TWI

Il modulo di espansione CAN, illustrata in Fig. 2.1.2, è in realtà ben più complesso di quello che lascia pensare il nome

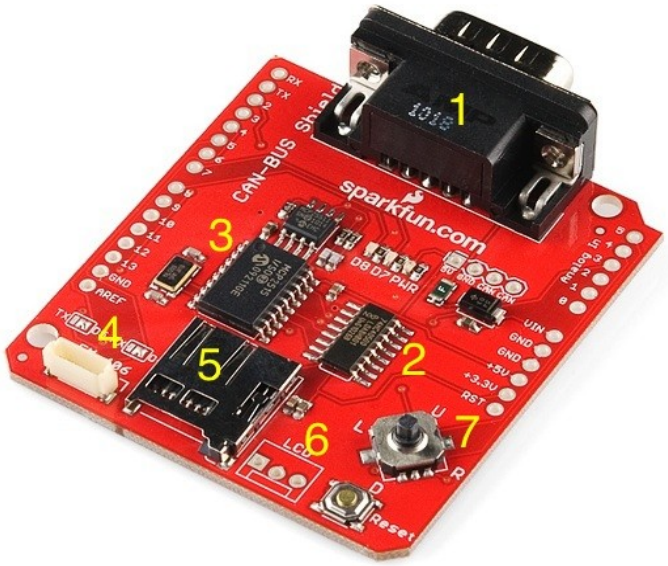


Fig. 2.1.2: shield CAN-Bus[2]

Oltre infatti al collegamento CAN effettuato tramite un connettore D-Sub a 9 poli (1), sono presenti i due microcontrollori MCP2551 (2) e MCP2515 (3) necessari alla comunicazione tra CAN-Bus e Arduino (come vedremo tra poco), un'interfaccia per il collegamento del modulo GPS EM406[3] (4), un alloggiamento per memoria flash di tipo micro-SD (5), un collegamento per uno schermo LCD di tipo seriale (6) e un piccolo joystick (7).
Ciò che interessa per il comportamento del sistema è la sezione relativa al CAN-Bus per cui vediamo di analizzarla in maniera un po' più approfondita.
Il Control Area Network è uno standard di comunicazione seriale di tipo multicast, cioè capace di connettere, in linea di principio, un numero infinito di dispositivi; il suo scopo è quello di definire le regole per implementare la struttura della rete (sfruttando poi invece protocolli di alto livello per gestire e garantire la comunicazione fra le varie parti del sistema).
In linea di principio il suo comportamento è illustrato in Fig. 2.1.3

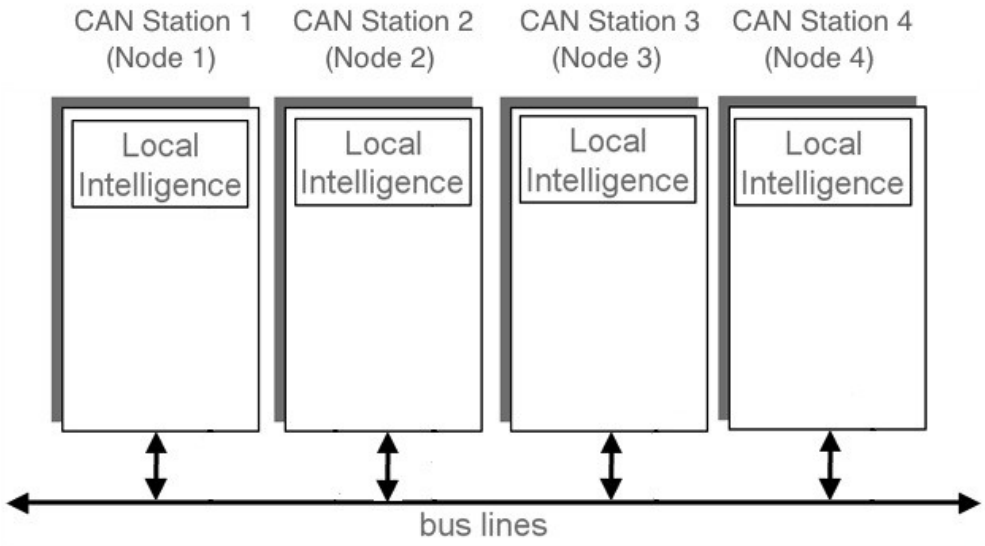


Fig. 2.1.3: schema di principio del CAN-Bus

Introdotto inizialmente come bus per autoveicoli (situazione che concerne proprio il sistema in realizzazione), se è evoluto entrando a far parte di molti processi di automatizzazione per via di numerosi vantaggi offerti in termini di

- **cablaggio:** il mezzo fisico è un semplice doppino telefonico ed essendo lo standard orientato a messaggi è semplice aggiungere o togliere nodi dalla rete

- **efficienza temporale:** la comunicazione dei dai è basata sul concetto di bit dominanti e recessivi; il nodo che trasmette bit dominanti ha priorità e questo riduce notevolmente i tempi di contesa evitando possibili ritardi

- **elevata immunità ai disturbi:** sfruttando come layer fisico una linea bilanciata a due fili il sistema è particolarmente resistente ai disturbi elettromagnetici

- **gestione errori:** se un nodo si ritrova ad avere errori hardware sistematici esso viene escluso

- **comunicazione:** ogni dispositivo può essere master del bus e quindi avere la possibilità di trasmettere in qualsiasi momento

Se poniamo ora l'attenzione al vero e proprio hardware coinvolto nel modulo utilizzato per implementare la comunicazione via CAN-Bus, allora la rete può essere schematizzata come in Fig. 2.1.4

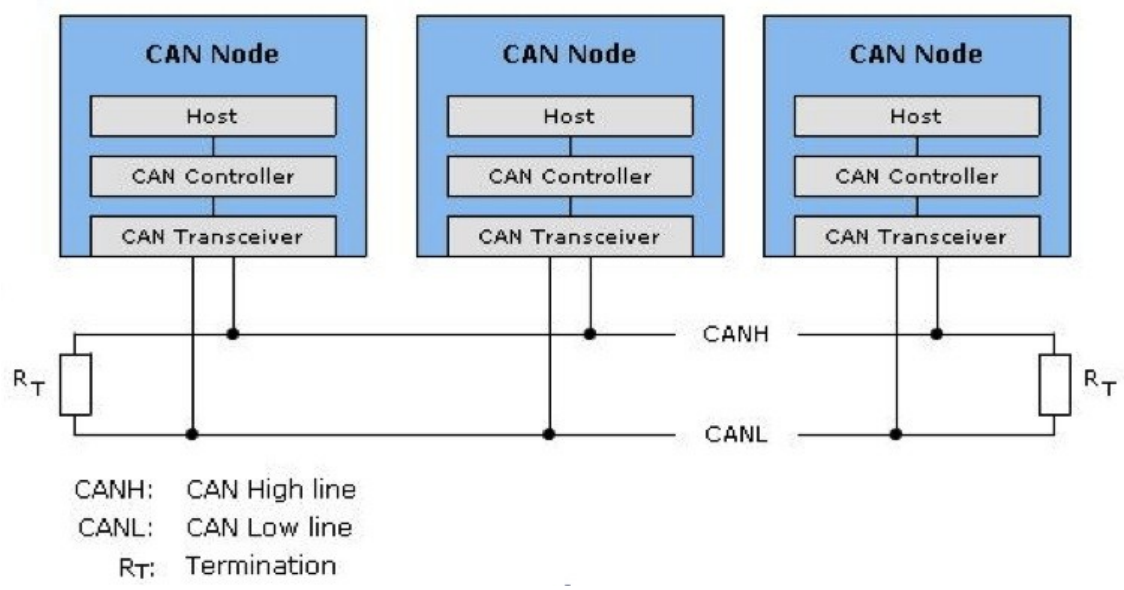


Fig. 2.1.4: stack CAN-Bus[4]

In questo caso le due linee CANH e CANL rappresentano il livello fisico bilanciato a due fili, mentre ogni nodo è composto da tre livelli logici

- **transceiver:** lo scopo di questo integrato è quello di rilevare lo stato del bus leggendo la differenza tra i livelli di tensione tra le due linee fisiche CANH e CANL e determinando quindi chi ha il diritto di trasmettere tra i nodi che tentano l'accesso
- **receiver:** questa sezione consente la comunicazioni dei dati ottenuti dal o per il CAN_Bus in modalità seriale, fungendo da collegamento tra il transceiver e l'host
- **host:** è colui che gestisce tutte le operazioni che deve effettuare il nodo; nel sistema che stiamo analizzando è ATmega328 che svolge questa funzione.

Il modulo di espansione scelto per implementare la sezione GPS in fase di prototipazione del sistema è quello fornito dalla Libellium[5], mostrato in Fig. 2.1.5, che permette un rapido interfacciamento fisico con Arduino UNO

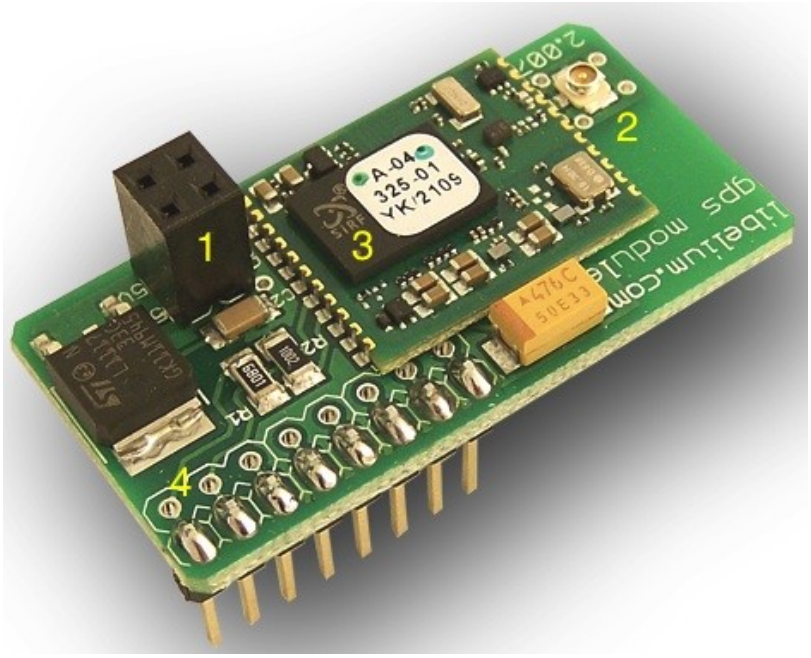


Fig. 2.1.5: shield GPS di Libellium

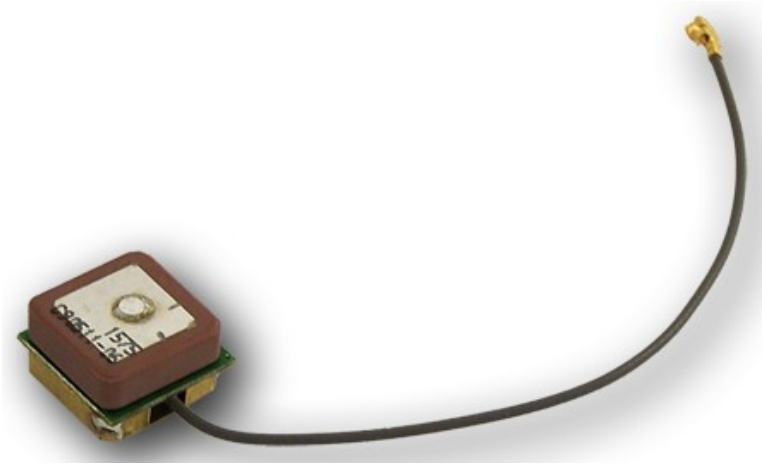


Fig. 2.1.6: antenna GPS con connettore UFL[6]

La scheda è molto semplice, ma permette un interfacciamento semplice e diretto; gli elementi principali sono un connettore da cui prelevare 5 V e GND (1), il connettore di tipo UFL per l'antenna esterna (2), il ricevitore GPS (3) composto dal modulo Vincotech A1080-B in grado di supportare lo

standard NMEA e basato sul microcontrollore SiRF III (un processore GPS fabbricato dalla SiRF Technology la cui peculiarità è quella di riuscire ad acquisire e mantenere il segnale nelle aree urbane e in quelle forestali densamente coperte) e due pin (4) per la comunicazione seriale con il resto del sistema.

Ovviamente l'elemento fondamentale di questa scheda è il Vincotech A1080-B[7], un dispositivo in grado di ricevere segnali da un massimo di 20 satelliti e trasferirli in informazioni di tempo e posizionamento attraverso una porta seriale.

Lo standard NMEA[8] (dall'ente "National Marine Electronics Association" che lo gestisce e lo sviluppa) che il modulo utilizza per la comunicazione GPS si basa sul principio per cui la fonte del segnale ("Talker") comunica con una o più riceventi ("Listeners") attraverso delle vere e proprie frasi di dati ("Sentences") con una comunicazione seriale in codifica ASCII; lo standard definisce anche i vari tipi di Sentences in modo che i riceventi possano analizzarli in maniera accurata.

Ogni frase, lunga fino ad un massimo di 80 caratteri, comincia con il simbolo "$" e termina con "*" seguito da due cifre esadecimali che rappresentano il checksum della sentence stessa.

Nel sistema sviluppato sono state ritenute necessarie per la rete CAN le informazioni relative al riferimento temporale e geo-posizionale del dispositivo in fase di lavoro.

Per quanto riguarda l'antenna invece occorre dire che è necessario utilizzarne una attiva di tipo GPS operante tra 3 e 5 V, che assorba un massimo di 50 mA, che abbia un guadagno tra i 20 e i 35 dB e una figura di rumore inferiore a 1.5 dB.

Il modulo di espansione Ethernet rappresentato in in Fig. 2.1.7 e il dispositivo che si occupa dell'interfacciamento principale con la rete Internet e che ha il compito di comunicare i dati elaborati al database hostato tramite il secondo Arduino UNO che si comporta in tutto e per tutto come un web server. Viene montato a castello sopra ad Arduino UNO e collegato in

maniera logica e fisica attraverso la porta SPI

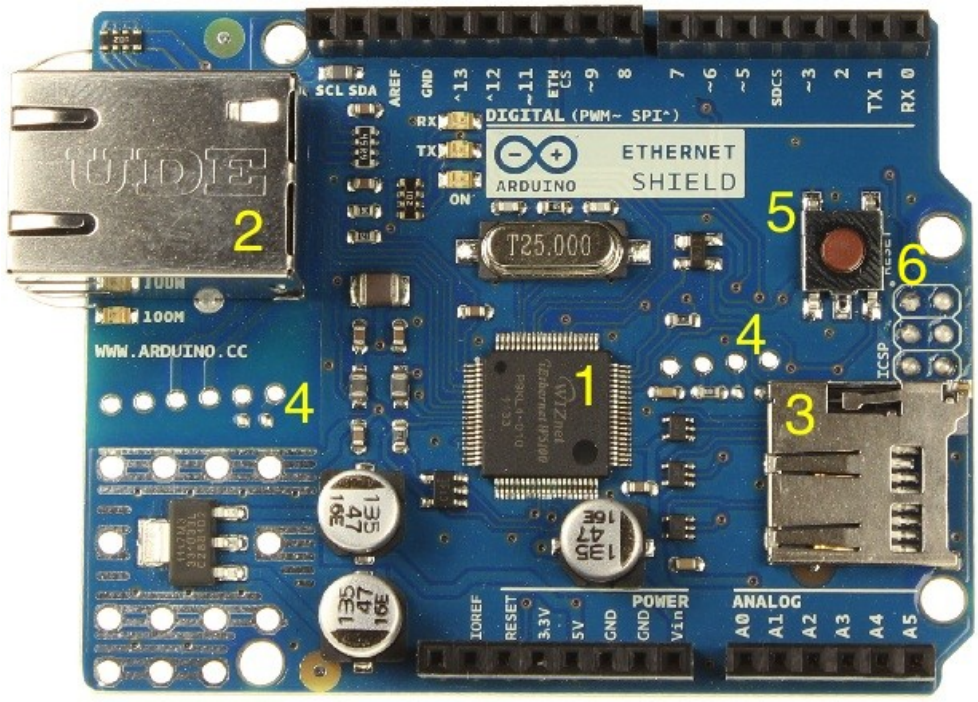


Fig. 2.1.7: shield ethernet

Il nucleo centrale dello shield ethernet[9] è costituito dall'ethernet chip (1) Wiznet W5100 che mette a disposizione un buffer interno di 16KB e fornisce entrambi gli stack di rete TCP/IP e UDP/IP.

Il collegamento alla rete, disponibile sia alla velocità di 10 Mb/sec che a quella di 100 Mb/sec, avviene attraverso un cavo di rete ethernet standard con connettore RJ_45 (2), visibile in Fig. 2.1.8.

La scheda presenta anche un lettore di schedina di tipo micro-SD (3) che può essere utilizzato come memoria dei dati da utilizzare per fare da server internet. Sia per la schedina che per la gestione della connessione ethernet sono disponibili delle librerie per l'IDE di Arduino che permettono di utilizzare l'hardware in tempi molto rapidi.

Fig. 2.1.8: cavo di rete ethernet con connettore RJ-45

Su questo shield, a differenza delle passate versioni, è presente anche la possibilità di montare un modulo Power over Ethernet (4), cioè un piccolo dispositivo in grado di ricavare l'alimentazione necessaria al sistema (tipicamente 9 V di output) attraverso proprio la connessione ethernet; viene usato tipicamente nelle situazioni in cui è difficile reperire fonti elettriche o per diminuire il numero di cavi del sistema. Per completezza, anche se non è stato utilizzato, vediamo in figura 2.1.9 un tipico modulo Poe compatibile con l'Arduino Shield Ethernet

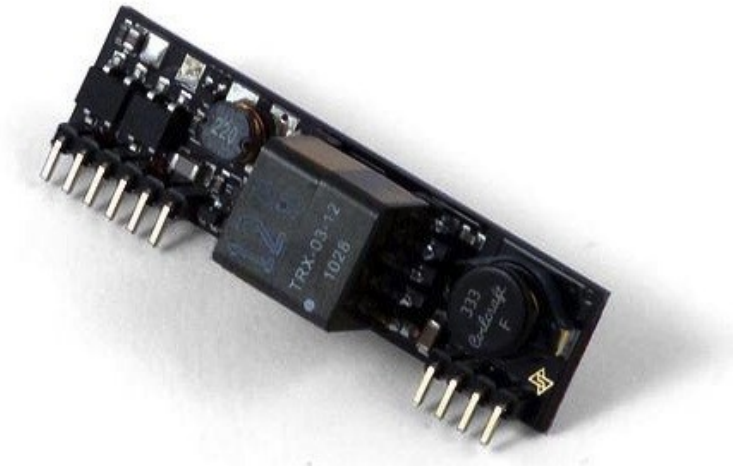


Fig. 2.1.9: modulo PoE Ag9120-S

Infine, come in tutti gli shield per Arduino, è presente un tasto di reset (5) che permette il riavvio sia del W5100 che dell'ATmega328 della scheda UNO.
Si fa notare qui che Arduino comunica sia con il W5100 che con la micro-SD usando l'SPI-Bus attraverso la connessione fisica ICSP che usa i pin digitali

- 4: per selezionare la micro-SD
- 10: per selezionare il chip W5100
- 11: pin MOSI (Master Output Slave Input)
- 12: pin MISO (Master Input Slave Output)
- 13: pin SCLK (Serial Clock)

Questi pin non possono più essere utilizzati come ingressi/uscite generali perchè sono necessarie affinchè la comunicazione ethernet avvenga in maniera corretta e con successo.
Si fa anche notare che le due sezioni principe di questo modulo condividono il bus seriale per cui solo uno di essi può essere attivo allo stesso tempo. La gestione di questo problema si risolve se non si usa una delle due periferiche o andando a variare alcuni valori all'interno di una delle due librerie a disposizione.

Veniamo infine all'ultimo shield di espansione, quello GPRS; anche in questo caso si è deciso di utilizzare quello fornito dalla Libellium[10] mostrato in Fig. 2.1.10

Fig. 2.1.10: schield GPRS di Libellium

Anche in questo caso il modulo è molto semplice; l'elemento principale è sicuramente il modulo GPRS Hilo prodotto dalla Sagem Communications (1) che viene accompagnato da un alloggiamento una SIM Card (2), da un regolatore da 3.3 V (3) e dal tasto di accensione/spegnimento (4).

Focalizziamo l'attenzione sull'Hilo che è sicuramente l'elemento cardine di questo modulo di espansione.
Progettato per applicazioni Machine to Machine (che favoriscono la comunicazione a distanza tra entità elettroniche attraverso la rete GSM/GPRS) con particolare attenzione all'industria automobilistica, ai sistemi di tracciamento e agli allarmi, il dispositivo contiene complete funzionalità di tipo GSM (chiamate, invio SMS, rubrica,...) e GPRS (fino a 85.6 Kbps in donw-wlink e 42.8 Kbps in up-link) in classe 10 (offre cioè 3 time slot per la ricezione e 2 per la trasmissione, adatto ai casi in cui lo scambio di dati è bilanciato). È un dispositivo quad-band cioè in grado di operare sia sulle frequenze 900/1800 MHz, tipicamente utilizzante in Europa, Africa ed Asia, che in quelle 850/1900 MHz in uso principalmente

nelle Americhe; risulta perciò essere molto versatile nell'utilizzo anche perchè, nonostante le sue dimensioni ridotte e il basso costo, ha un ampio range di tensione d'ingresso (3.2 – 4.5 V), bassissimi consumi in modalità IDLE (1.25 mA), ma soprattutto un range molto ampio di funzionamento in temperatura (tra i -40 °C e i +85 °C), elemento fondamentale se si considera il fatto che il sistema che stiamo studiando sarà montato su un veicolo in cui le variazioni di temperatura possono essere repentine a causa delle condizioni atmosferiche e dell'auto stessa.

Come la maggior parte degli shield anche quello GPRS può essere alimentato esternamente, ma anche dalla scheda Arduino; in questo caso c'è da prestare attenzione al consumo di corrente dello shield che in fase di comunicazione può raggiungere picchi di 2.2 A (range tipico 220 – 2200 mA). Arduino UNO infatti, se alimentato tramite USB, non è di fornire tutta la corrente necessaria allo shield GPRS per accendersi o svolgere in pieno tutte le sue funzionalità.
La soluzione è unica e semplice: utilizzare un'alimentazione esterna, sia essa collegata ad Arduino UNO o al solo shield; se anche in questo caso non si ha a disposizione un alimentatore con un output di almeno 2 A è consigliabile aggiungere un condensatore extra tra i 3.3 V e GND (tipicamente un condensatore elettrolitico da 220 μF).

Vista la panoramica sul sistema di partenza, in Fig. 2.1.11 riassumiamo come è stato realizzato il prototipo.
Si fa notare che in fase di progetto le due sezioni Master e Slave sono state sviluppate separatamente e poi messe in comunicazione attraverso il bus I^2C, facilmente implementabile tramite Arduino.

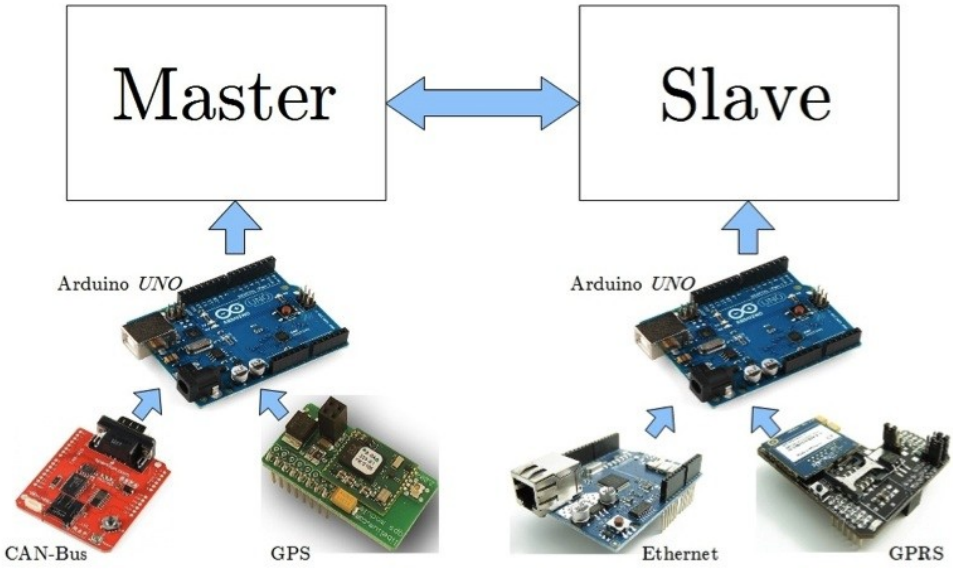


Fig. 2.1.11: schema progettuale e moduli utilizzati

Lo schema progettato sarà inserito all'interno di una centralina (rappresentata in Fig. 2.1.12) per un sistema veicolare completamente elettrico realizzato dal DIE, Dipartimento di Ingegneria Elettrica, dell'università di Bologna. Essa ha il compito di gestire la carica delle batterie e il controllo del motore del veicolo e comunica con tutte le sue parti (in particolare sensori) proprio attraverso il CAN-Bus su cui saranno disponibili i dati da analizzare con il sistema qui progettato; per questo motivo il bus è diviso in due sezioni: una interna per l'interfacciamento con le parti della centralina stessa e uno esterno a disposizione per la telemetria del veicolo (e in generale per il suo monitoraggio).

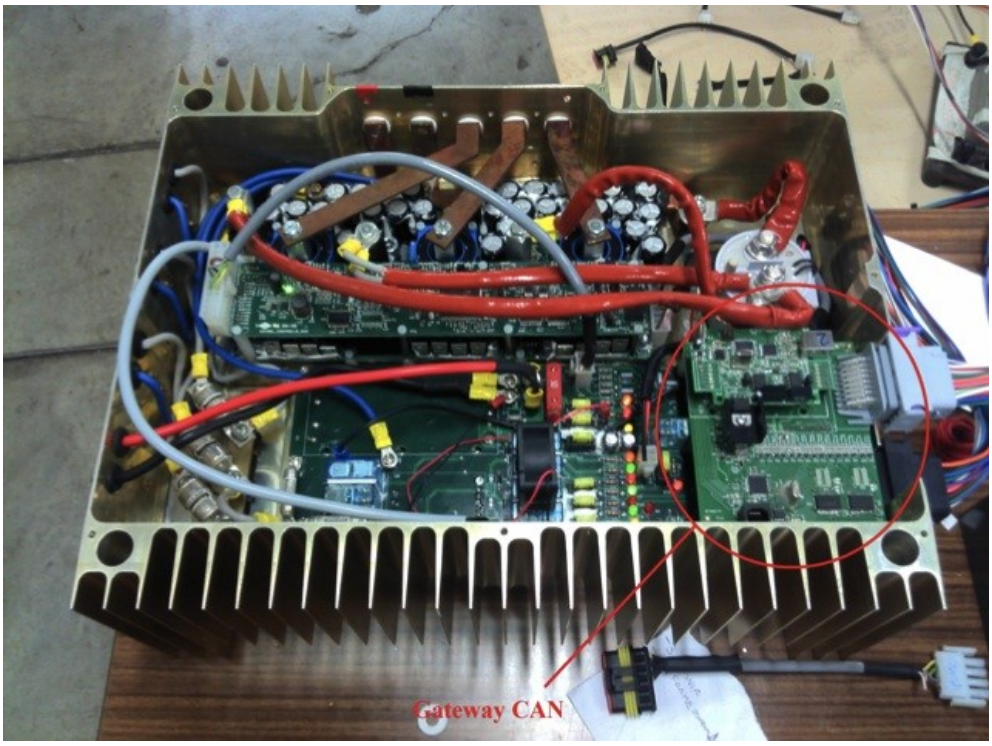


Fig. 2.1.12: centralina[4]

Nel prossimo paragrafo vediamo di illustrare in maniera chiara e concisa com'è sviluppato il software contenuto nel sistema appena presentato, spiegando cosa deve fare e quali sono state le scelte implementative

2.2 SOFTWARE

Diamo ora una breve descrizione del software implementato nel progetto iniziale sulla gestione dei dati acquisiti dalla centralina e su come essi siano resi disponibili su un database online.
Il processo può essere separato in due macro sezioni: la prima (Master) dedicata all'acquisizione dei dati e alla loro rielaborazione, la seconda (Slave) con l'obiettivo di creare un web server in cui andare ad inserire i dati sotto forma di database.

La sezione Master è stata articolata in tre aree.
La prima si occupa di dichiarare le librerie utilizzate per l'interazione con i

componenti hardware utilizzati e di definire tutte le variabili e gli array necessari ad accumulare i dati ricevuti dal CAN-Bus su ognuno dei tredici parametri del motore che si è deciso di monitorare in modo da riconoscere alcuni dei campi del protocollo che serviranno. La seconda parte è quella centrale e si occupa di mediare venti valori per ogni parametro (l'acquisizione è veloce e in questo modo il parametro sarà affidabile), leggere e ricavare tutte le informazioni in ricezione sull'antenna su posizionamento e tempo, preparare il messaggio (in byte) con tutte le informazioni per la sezione Slave ed inviarlo attraverso il protocollo I^2C sfruttando la libreria "Wire.h" per Arduino.

A questo punto è il software della sezione Slave che si occupa di rielaborare in maniera corretta queste medie per metterle a disposizione su un database.
Dopo avere inizializzato le librerie necessarie, avere definito alcuni parametri fondamentali per la connessione (ip, mac, subnet-mask, gateway) e avere impostato il dispositivo Slave come client, il software prevede la lettura dei dati proveniente dal Master ogni volta che c'è un byte a disposizione sul collegamento I^2C (in particolare sul pin A5). A questo punto i dati ottenuti, se ritenuti validi, vengono inviati al server web (è necessaria una codifica dei dati numerici ottenuti in decimale per non avere conflitti tra l'ASCII di Arduino e il web server) grazie all'invio di una richiesta "http" di tipo GET alla pagina "save.php" il cui compito è proprio quello di salvare i dati sulla tabella di un database.

Tutto il software è stato elaborato sull'IDE di sviluppo fornito dalla piattaforma Arduino stessa, basato su un linguaggio di programmazione "C-like" in un ambiente Object-Oriented rispettivamente dal Dott. Solari Samuele e Dott. Colella Simone

3 REALIZZAZIONE DEL PROGETTO

Una volta analizzato il sistema da cui si intende partire per realizzare effettivamente la scheda per telemetria andiamo a vedere come si è deciso di muoversi in fase di sviluppo del progetto finale, puntando l'attenzione su alcuni cambiamenti effettuati e sul perchè di tali.
L'idea che ha portato a queste modifiche è quella per cui si vuole cercare di sviluppare una scheda utilizzabile in maniera ottimale per effettuare la telemetria del veicolo elettrico, ma allo stesso tempo permetta di lasciarsi aperta la possibilità di avere infiniti modi di comunicare con il mondo esterno, permettendo così di utilizzare la scheda per sviluppare un gran quantitativo di progetti la cui priorità sia quella di avere comunicazione.

3.1 SPECIFICHE

Si è partiti tenendo conto dell'obiettivo principale, cioè la realizzazione della scheda per telemetria. Seguendo questo metodo e per quanto già ampiamente descritto nel Cap. 2 si è deciso di integrare nello schema due sezioni basilari

- **sezione CAN-GPS:** prevede d'inserire on-board un Arduino UNO il cui compito è quello di interfacciarsi con il connettore del CAN-Bus e controllare l'antenna esterna che permette l'acquisizione dati tramite GPS;
- **sezione Ethernet:** implementata tramite un Arduino UNO che gestisce completamente la sezione Ethernet collegandosi in maniera diretta al connettore RJ-45 che si interfaccerà con la rete esterna; anche in questo caso il tutto sarà implementato on-board, eliminando il castello e prendendo ispirazione dal sistema "Arduino Ethernet"[11] che presenta

alcune lievi differenze rispetto a quanto usato nel prototipo (Arduino UNO con shield Ethernet) delle quali parleremo approfonditamente più avanti;

A questi elementi fondamentali, come si può notare, è rimasta esclusa tutta la sezione relativa alla comunicazione via GPRS. Questo perchè si è voluto appunto dare vita non ad una scheda il cui unico scopo sia quello di realizzare la telemetria, ma permetta in un futuro di poter essere utilizzata per molte applicazioni.
Per questo motivo viene prevista una terza sezione all'interno dello schema che comprende un semplice Arduino UNO completo di castello in modo che su di esso possa essere montano un qualsiasi shield di comunicazione sia esso quello GPRS o Bluetooth o, ancora, Wi-Fi.
In questo modo non si vincolano attraverso l'hardware le modalità di comunicazione del sistema (a meno della sezione Ethernet che viene inserita come base per stabilità di connessione e soprattutto essendo oramai una tecnica affermata e in pieno sviluppo) in modo tale che esso possa adattarsi nel tempo alla tecnologia corrente semplicemente andando a modificare alcuni pezzi del software utilizzato.
A livello logico quindi il sistema può essere diviso come rappresentato in Fig. 3.1.1

Fig. 3.1.1: rappresentazione logica delle parti della scheda realizzata

Dopo avere realizzato la base del progetto si è poi cercato di approfondire decidendo il punto di arrivo del layout finale della scheda. Questo perchè una volta realizzata essa dovrà essere inserita all'interno della centralina del veicolo elettrico e quindi ha necessità di un certo tipo di interfacciamento. Per questo motivo, al di là della disposizione dei componenti per il funzionamento del dispositivo, si deciso di mantenere su un lato della scheda tutte le interfacce che devono comunicare internamente con la centralina, mentre dal lato opposto si vuole disporre tutti i connettori che hanno bisogno di collegarsi con l'ambiente esterno affinchè il funzionamento sia corretto e tutti quelli che permettono un'eventuale ampliamento del sistema, eventualmente anche a posteriori.

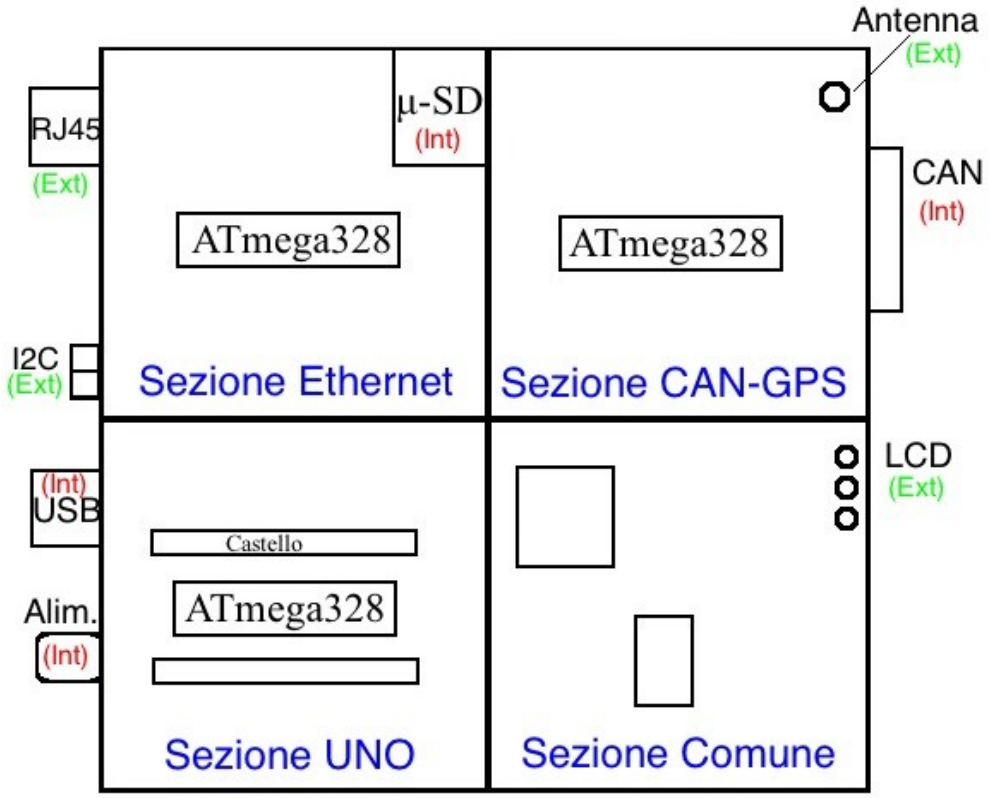


Fig. 3.1.2: indicazioni sul layout della scheda

In Fig. 3.1.2 vengono rappresentate le sezioni logiche del sistema e tutte le interfacce e le connessioni che il sistema avrà con l'ambiente esterno. In particolare vengono indicati con "(Int)" le interfacce che rimarranno all'interno della centralina perchè una volta che il sistema è reso operativo non sarà necessario che l'utente (o cliente) interagisca con esse a meno di dover risolvere guasti o effettuare aggiornamenti.

Allo stesso modo vengono indicate con "(Ext)" tutti quei collegamenti che permettono di collegare e controllare l'intero dispositivo con la rete o con altri moduli di espansione.

In maniera per il momento molto schematica viene rappresentano anche il cuore di ogni sezione; si può ben vedere che in tre di esse è presente un ATmega328 che sta ad indicare la presenza di una piattaforma Arduino, ma solo in una viene mantenuto il castello per l'interfaccia con moduli esterni perchè gli altri due sono già dedicati ad un compito preciso. Nella sezione comune sono presenti tutti quegli integrati per il controllo delle interfacce o

dei protocolli di comunicazione (USB, Ethernet, CAN-Bus,...).

3.2 AMBIENTE DI SVILUPPO

Per la realizzazione del circuito stampato della scheda studiata fino a questo momento si è deciso di utilizzare EAGLE[12], un software EDA (Electronic Design Automation) prodotto da Cadsoft sin dal 1988 che permette la progettazione a partire dallo schematico del progetto.
È forse tra i più popolari del genere e la scelta è ricaduta su esso perchè presenta alcuni vantaggi fondamentali rispetto a programmi simili, ma tipicamente più professionali

- **compatibilità:** tutti i prodotti Arduino e le maggior parte di quelli ad essi compatibili (in particolari tutti quelli necessari per il sistema che si sta sviluppando) condividono online in maniera gratuita tutti gli schematici e i PCB layout, realizzati con EAGLE; questo permette di accelerare i tempi di lavoro ereditando l'esperienza di chi ha già progettato sistemi analoghi e soprattutto permette di diminuire notevolmente il numero di errori anche grazie al continuo confronto che è possibile avere grazie ad una sviluppata community legata al sito ufficiale del software
- **gratuito:** EAGLE è liberamente scaricabile dal sito ufficiale di Cadsoft nella sua ultima versione Light (6.1); esistono due versioni (Standard e Professional) a pagamento che permettono di ottenere funzioni avanzate in fase di lavoro, in particolare concedono al progettista di sviluppare schede su un numero crescente di layer fino ad un massimo di sedici (la versione Light si limita a due)
- **disponibilità:** una vasta gamma di librerie contenenti rappresentazioni schematiche, associate ai package standard presenti sul mercato, di praticamente tutti i componenti pensabili sono rese disponibili dalla Cadsoft stessa[13]; allo stesso modo anche alcuni venditori di componenti elettronici offrono ad EAGLE le librerie dei loro prodotti che definiscono lo schematico, il pinout e le dimensioni che permettono una progettazione

precisa nella realizzazione del PCB layout (il più noto è SparkFun Electronics[14]).

Una volta avviato EAGLE abbiamo la possibilità di creare file di tipo schematico (.sch) che ci permettono di sviluppare i collegamenti logici tra tutti i componenti del sistema; una volta terminato il programma di permette in automatico di passare allo sbroglio del PCB layout (file .brd) e, avendo scelto i componenti giusti, ci troveremo le dimensioni corrette per ogni dispositivo del sistema in modo da poter lavorare a dimensione reale sul nostro progetto.

All'avvio di un nuovo progetto, la schermata sarà come quella in Fig. 3.2.1: una parte centrale occupata dalla griglia di disegno in cui disporre tutti i componenti necessari. In alto si può notare la barra con le possibili azioni che si possono svolgere; sono state evidenziate le due principali: quella che permette di aprire tutte le librerie disponibili in cui scegliere un componente per aggiungerlo allo schematico (1) e quella che permette di collegare i vari pin dei componenti facendo capire al programma che quello sarà un vero collegamento elettrico (2). In alto ci sono delle funzioni generali tra cui quella denominata “Board” (3) che ci permette in automatico di passare associare tutti i pattern reali dei componenti logici, facilitando così la realizzazione dei PCB layout che è il vero obiettivo quando si sta realizzando una scheda.

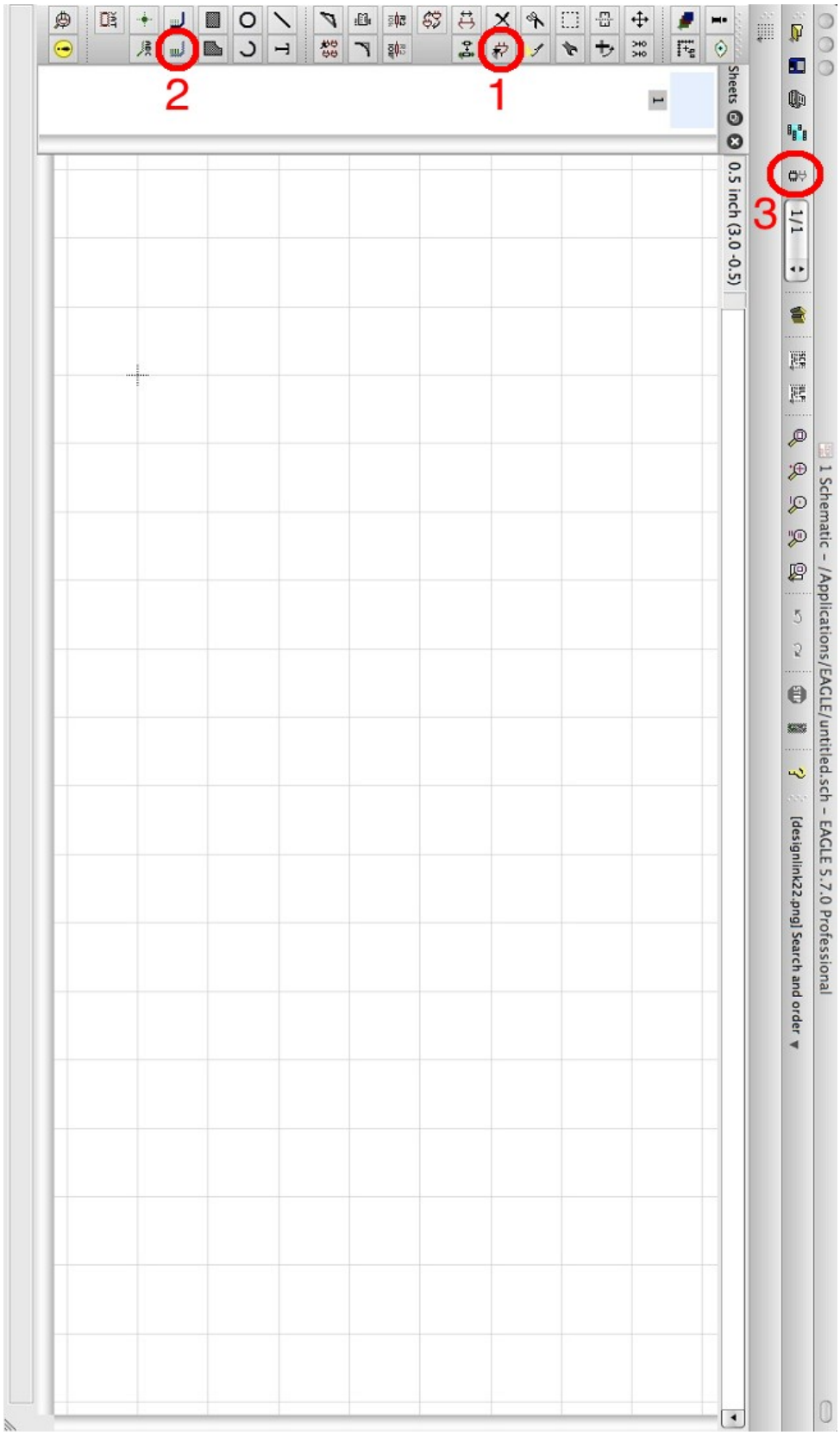


Fig. 3.2.1: schermata principale di EAGLE

3.3 REALIZZAZIONE E LAYOUT

Illustrato il programma e decisi i primi aspetti su come affrontare la realizzazione della scheda si è proceduto alla vera e propria realizzazione, partendo dalla parte principale che prevede la realizzazione dello schematico.
Il primo passo è stato prendere e analizzare tutti gli schematici disponibili gratuitamente online dei moduli illustrati nel Cap. 2, in modo da poter decidere quali elementi mantenere e quali eliminare al fine di ottimizzare il compito della scheda per rispettare gli obiettivi inizialmente postisi (vedi appendici A, B, C, D).
Dopo questa fase di analisi sono stati decisi alcuni cambiamenti volti appunti al miglioramento del prodotto finale:

- il modulo CAN-Bus di SparkFun utilizzato in fase di progetto prevedeva la possibilità di utilizzare un joystick e di controllare una scheda di tipo µ-SD; il primo necessita di cinque pin analogici per avere una completa funzionalità (quattro direzioni e click), ma l'ATmega328 ne mette a disposizione un massimo di sei di cui due già occupati per la comunicazione tra microcontrollori attraverso il protocollo I^2C per questo è stato eliminato (non essendo utile per il dispositivo telemetrico); la seconda invece è prevista anche nel modulo ethernet e quindi sarebbe stata un elemento doppio di scarsa utilità;
- si è optato per l'utilizzo di Arduino Ethernet (invece che Arduino UNO con shield ethernet) semplicemente perchè prevede tutte le funzionalità necessarie al sistema telemetrico già implementate on-board il che è già ottimo in ottica di ottimizzare lo spazio occupato dai vari componenti; non prevede il connettore USB, bensì un collegamento seriale a sei pin che però verrà eliminato per la sua non utilità e mantiene la possibilità di montare il modulo PoE;

- si è stati costretti a trovare un'alternativa al modulo GPS di Libelium utilizzato in fase di progetto perchè il ricevitore GPS Vincotech A1080-B, elemento cardine della scheda, è oramai obsoleto e fuori produzione e quindi non avrebbe avuto alcun senso mettere in produzione una scheda che necessitasse di esso per il funzionamento; l'alternativa è un modulo basato sul ricevitore GPS Copernicus® II[15] della Trimble che permette anch'esso l'utilizzo di un'antenna esterna, elemento fondamentale per il sistema telemetrico considerando il fatto che la scheda terminata sarà posta all'interno della centralina, ma dovrà permettere all'antenna di uscire "all'aperto" affinchè il segnale dei satelliti sia effettivamente ricevibile; allo stesso tempo è un dispositivo che permette l'utilizzo dello standard di comunicazione NMEA e perciò non sono richieste modifiche al software utilizzato;
- utilizzando più piattaforme Arduino, dagli schematici sarebbero previsti più connettori USB (e relativi chip di controllo) in modo da programmare indipendentemente ogni ATmega328. Mantenere questo modello costerebbe molto in termini di spazio occupato considerando che a parte per l'inserimento del software le stesse rimarrebbero praticamente inutilizzate. Si è deciso quindi di mantenere solo una connessione ed effettuare uno switch fisico dei piedini di programmazione attraverso tre coppie (sono solo due le linee necessarie) di jumper: basterà selezionare attraverso esso il microcontrollore corretto senza dover appesantire il numero di connettori esterni della scheda;
- come descritto nel Cap. 2 i tre microcontrollori comunicano logicamente tra loro attraverso il protocollo I^2C, fisicamente implementato sui pin analogici A4 e A5 di ogni ATmega328; quelli che si occupano del sistema telemetrico sono dedicati, ma la sezione denominata "UNO" può essere utilizzata anche per sviluppare sistemi indipendenti che possono avere bisogno di tutti i pin analogici, per questo motivo si è deciso di implementare la possibilità di scegliere se utilizzare A4 e A5 come I^2C o per altri scopi. Anche in questo caso lo switch sarà implementato da dei jumper che dovranno essere selezionati dal programmatore in base agli

obiettivi che egli si è prefisso;

- l'ultima scelta progettuale volta ad ottimizzare la scheda va a discapito dell'ottimizzazione dello spazio: i tre ATmega328 non saranno montati come componenti SMD (Surface Mounting Device) ma sugli zoccoli standard; questo perchè essendo essi i componenti critici del sistema, in caso di rottura nei casi in cui la scheda venga utilizzata per progettare nuovi sistemi si vuole avere l'opportunità di sostituirli senza essere costretti a cambiare l'intera scheda.

Fatte tutte queste premesse prima di passare definitivamente a vedere schematici e PCB layout riepiloghiamo in Fig. 3.3.1 i pin I/O utilizzati da ogni microcontrollore affinchè il sistema svolta correttamente le sue funzioni di telemetria (tra parentesi sono indicati i pin fisici dell'ATmega328)

ATmega328_ETHERNET	ATmega328_CAN_GPS	ATmega328_UNO
Pin A4,A5 (27,28) ---> I^2C Pin D4 (6) ---> SD Pin D10:D13 (16:19) ---> Eth Pin D9 (15) ---> LED Pin D0,D1 (2,3) ---> per comunicazione USB	Pin A4,A5 (27,28) ---> I^2C Pin D2, D10 (4,16) ---> CAN Pin D6 (12) ---> Serial LCD Pin D8,D9 (14,15) ---> GPS Pin D0,D1 (2,3) ---> per comunicazione USB	Pin A4,A5 (27,28) ---> I^2C (switchati) Pin D0,D1 (2,3) ---> per comunicazione USB

Fig. 3.3.1: pin I/O utilizzati nei tre ATmega328

È ora possibile andare ad analizzare gli schematici che realizzano tutte le connessioni logiche per il funzionamento della scheda per telemetria. Si fa presente che per comodità il lavoro è stato separato in tre fogli di lavoro afferenti allo stesso PCB layout: ogni foglio comprende un singolo ATmega328 con tutte le circuiterie necessarie al controllo delle interfacce ad esso collegate. Si spenderanno alcune parole sugli elementi relativi ad ogni schematico

In Fig. 3.3.2a-b-c è rappresentata la sezione che si occupa di controllare la ricezione via ethernet, la scheda μ-SD e il modulo PoE.

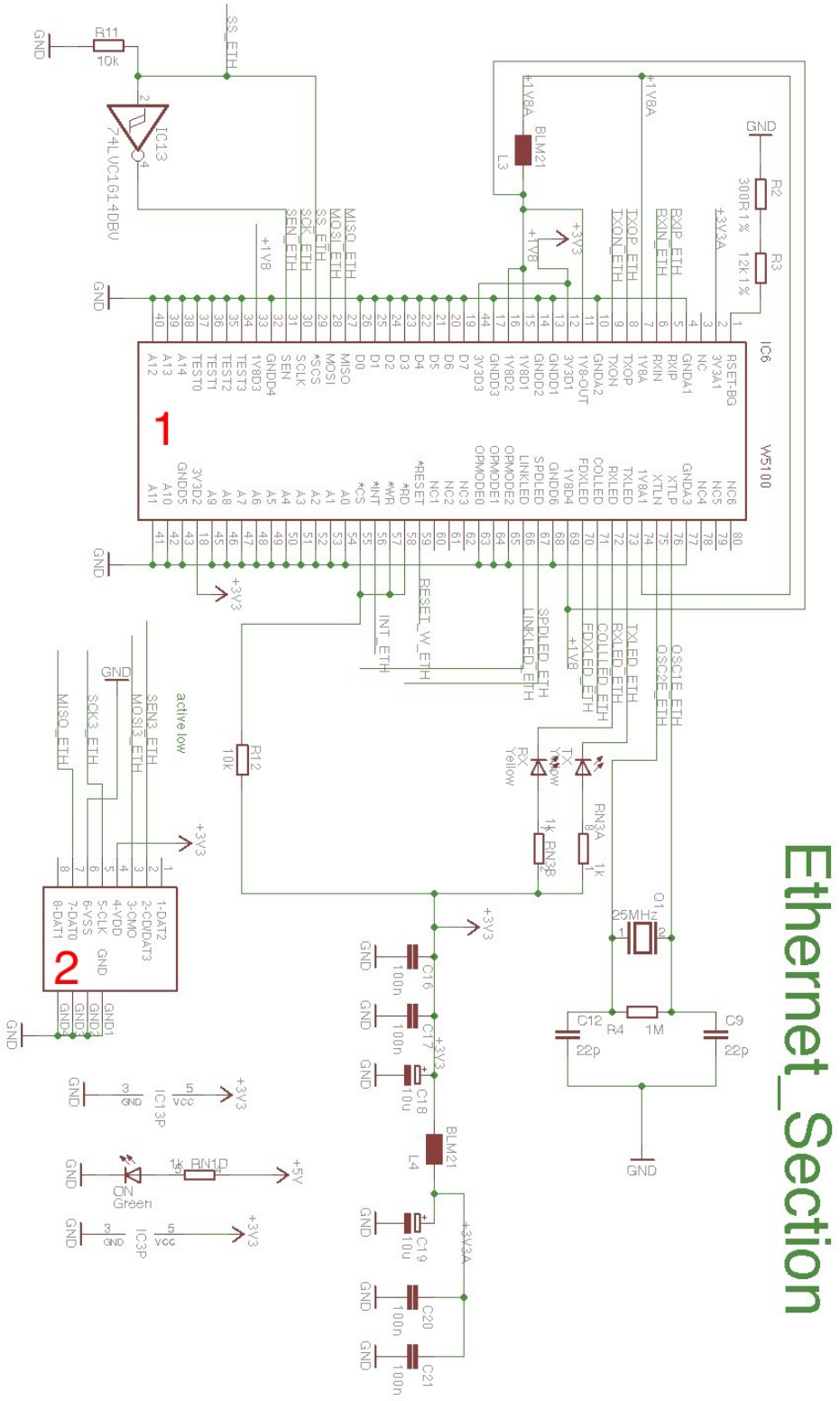
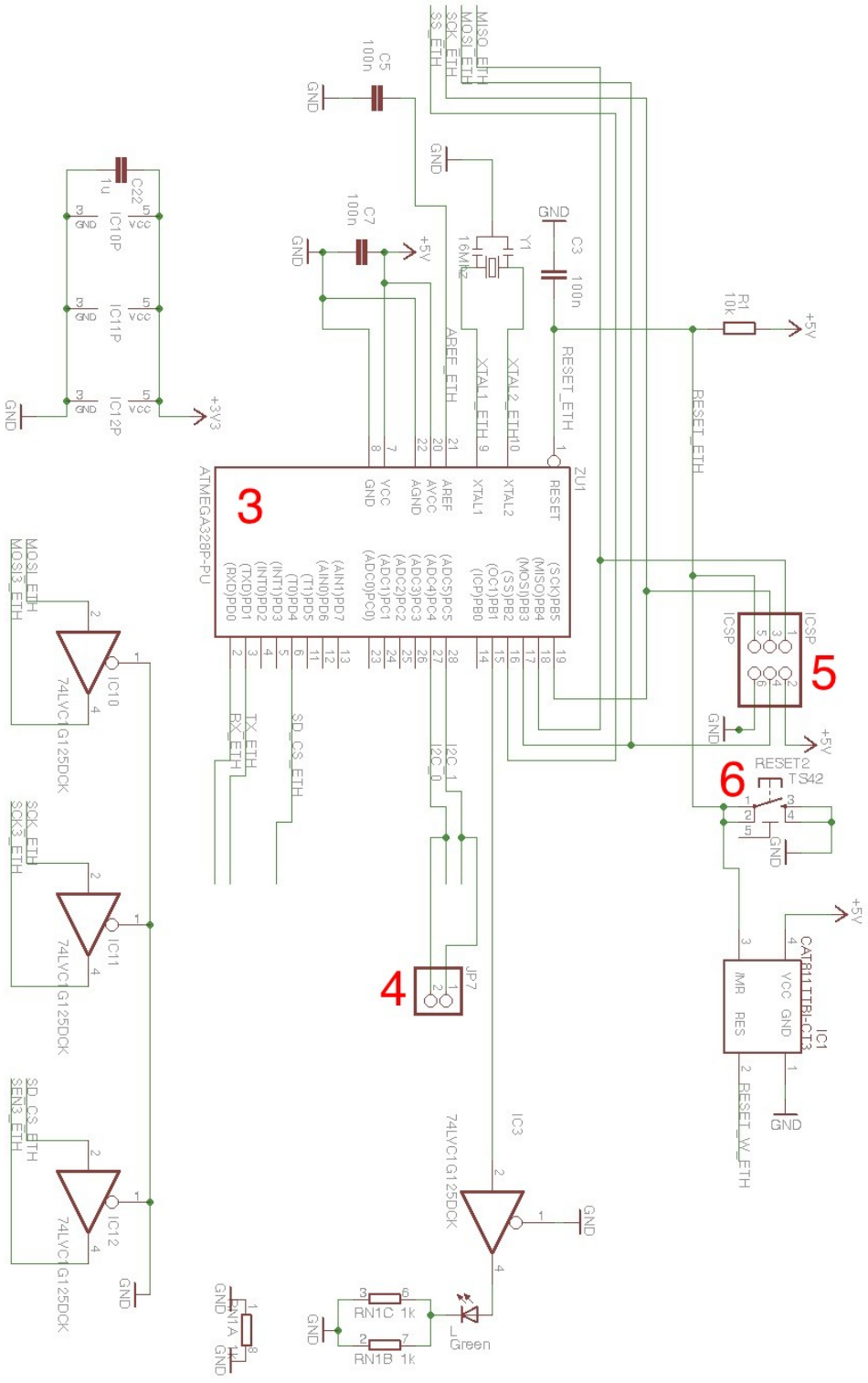
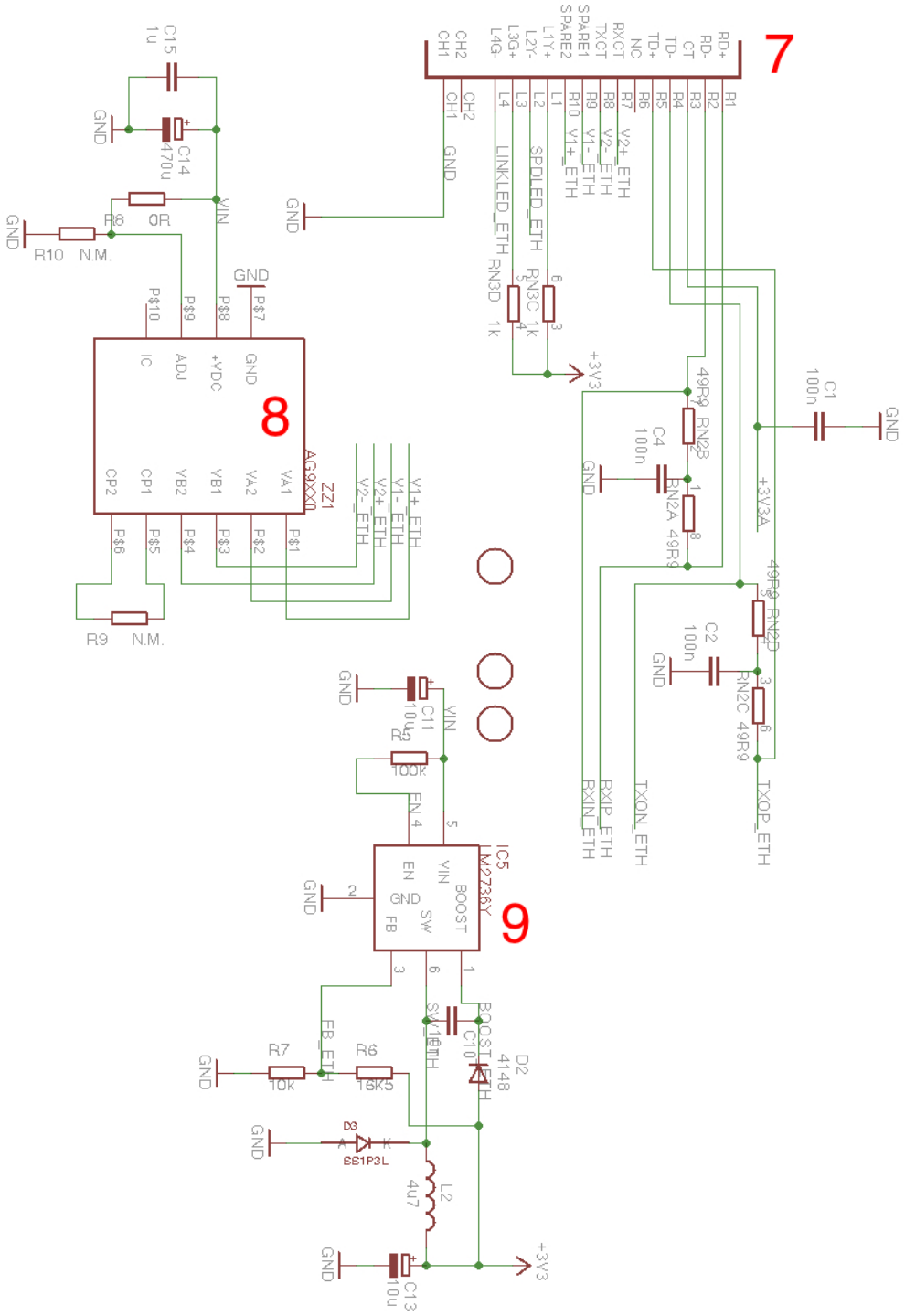


Fig. 3.3.2: ATmega328 per il controllo di ethernet, SD e PoE

(a)


3
ATMEGA328P-PU
ZU1
5
ICSP
6
RESET2
TS42
4
JP7
IC1
CAT811TTBI-GT3
IC3
74LVC1G125DCK
IC10
IC11
IC12
IC10P
IC11P
IC12P
Y1
16MHz
C5
C7
C22
C3
R1
RN1A
RN1B
RN1C
Green
RESET_ETH
RESET_W_ETH
MISO_ETH
MOSI_ETH
SCK_ETH
SS_ETH
ABEF_ETH
XTAL1_ETH 9
XTAL2_ETH10
TX_ETH
RX_ETH
SD_CS_ETH
I2C_1
I2C_0
MOSI3_ETH
SCK3_ETH
SEN3_ETH
+5V
+3V3
GND

(b)


7
8
9
C15
1u
C14
470u
R10 N.M.
R9 N.M.
C1
100n
C4
100n
C2
100n
C11
10u
100k
C10
D2
4148
R7
10k
R6
16K5
D3
SS1P3L
L2
4u7
C13
10u
IC5
LM2736Y
RN2A 49R9
RN2B 49R9
RN2C 49R9
RN2D 49R9
RN3C 1k
RN3D 1k
+3V3
+3V3A
GND
VIN
FB_ETH
RXIP_ETH
RXIN_ETH
TXON_ETH
TXOP_ETH
LINKLED_ETH
SPDLED_ETH
Y1+_ETH
Y1-_ETH
Y2+_ETH
Y2-_ETH

(c)

(1) Il chip W5100 di Wiznet è il dispositivo che permette la connettività internet via ethernet, appositamente pensato per sistemi embedded in cui integrazione, performance e stabilità sono necessari; esso prevede uno stack completo TCP/IP, integrando strato ethernet, MAC e fisico e supportando TCP, UDP, IPv4 e PPPoE. Internamente è presente un buffer di 16 KB adibito alla trasmissione dati.
A livello di schema esso si interfaccia con il connettore RJ_45 attraverso i collegamenti TXOP_ETH, TXON_ETH, RXIP_ETH e RXIN_ETH; allo stesso rimane sincronizzato con il relativo ATmega328 attraverso le linee di controllo MISO_ETH, MOSI_ETH, SS_ETH, SEN_ETH e SCK_ETH.

(2) Il socket per la scheda μ-SD si collega in comunicazione IN/OUT con l'ATmega328 rispettivamente tramite MISO e MOSI in un bus di tipo seriale; ovviamente è con esso sincronizzato e ha un chip select (sul pin 2) affinchè venga utilizzato solo al momento opportuno; la scheda può arrivare ad avere una velocità di lettura di 10.0 Mbytes/sec e una di scrittura di 5.02 Mbytes/sec.

(3) È il cuore dei questa sezione e abbiamo già parlato abbondantemente dell'ATmega328 nel Cap. 2; come detto il castello è stato eliminato perchè i pin input/output utilizzati sono esplicitamente dedicati al sistema telemetrico.

(4) Questa coppia di morsetti in uscita sono collegati alle due linee che creano il bus I^2C in modo che una volta terminata la scheda sia comunque possibile aggiungere dispositivi esterni che vogliano e possano comunicare attraverso questo protocollo.

(5) Il connettore ICSP (In-Circuit Serial Programming)[16] permette di programmare direttamente i microcontrollori già montati sulla scheda,

inserendo in essi il bootloader (programma che carica il kernel del sistema operativo permettendone l'avvio) e aumentando se possibili la flessibilità di questi prodotti. La programmazione avviene sempre attraverso le linee MISO e MOSI.

(6) Questo pulsante di reset consente il riavvio sia del microcontrollore che del W5100 contemporaneamente.

(7) Il connettore RJ-45, come detto al punto (1), utilizza quattro linee per il controllo della connessione ethernet; quattro linee vengono sfruttate per ricavare l'alimentazione PoE che si andranno ad interfacciare con il modulo esterno (V1+_ETH , V1-_ETH , V2+_ETH , V2-_ETH); altre due linee sono collegate a dei led che indicano la velocità di comunicazione e se essa sia attiva o meno.

(8) Questi otto fori permettono di montare il modulo Power over Ethernet da cui ricavare l'alimentazione necessaria per tutto il sistema direttamente dal connettore RJ-45.

(9) Questo regolatore di tensione alimentato da VIN (l'alimentazione dell'intera scheda) rende disponibile in uscita una tensione di 3.3 V utilizzata per alimentare gran parte dei chip presenti all'interno del progetto. Supporta fino a 750 mA più che sufficienti considerando i consumi nel suo complesso

In Fig. 3.3.3a-b è rappresentata la sezione che si occupa di gestire la comunicazione con il CAN-Bus e quella GPS attraverso l'antenna; rende inoltre disponibile l'interfacciamento con uno schermo LCD di tipo seriale

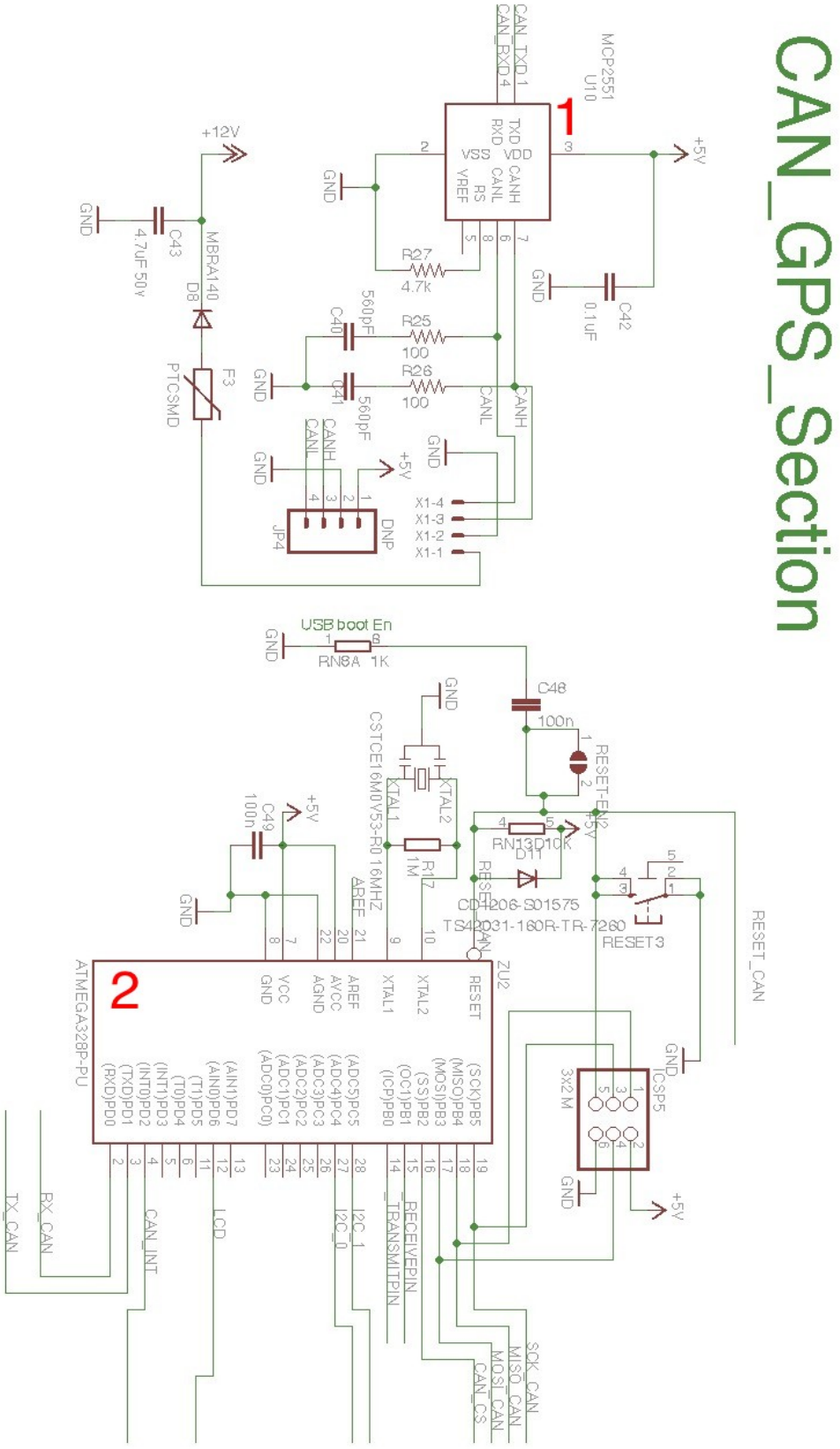
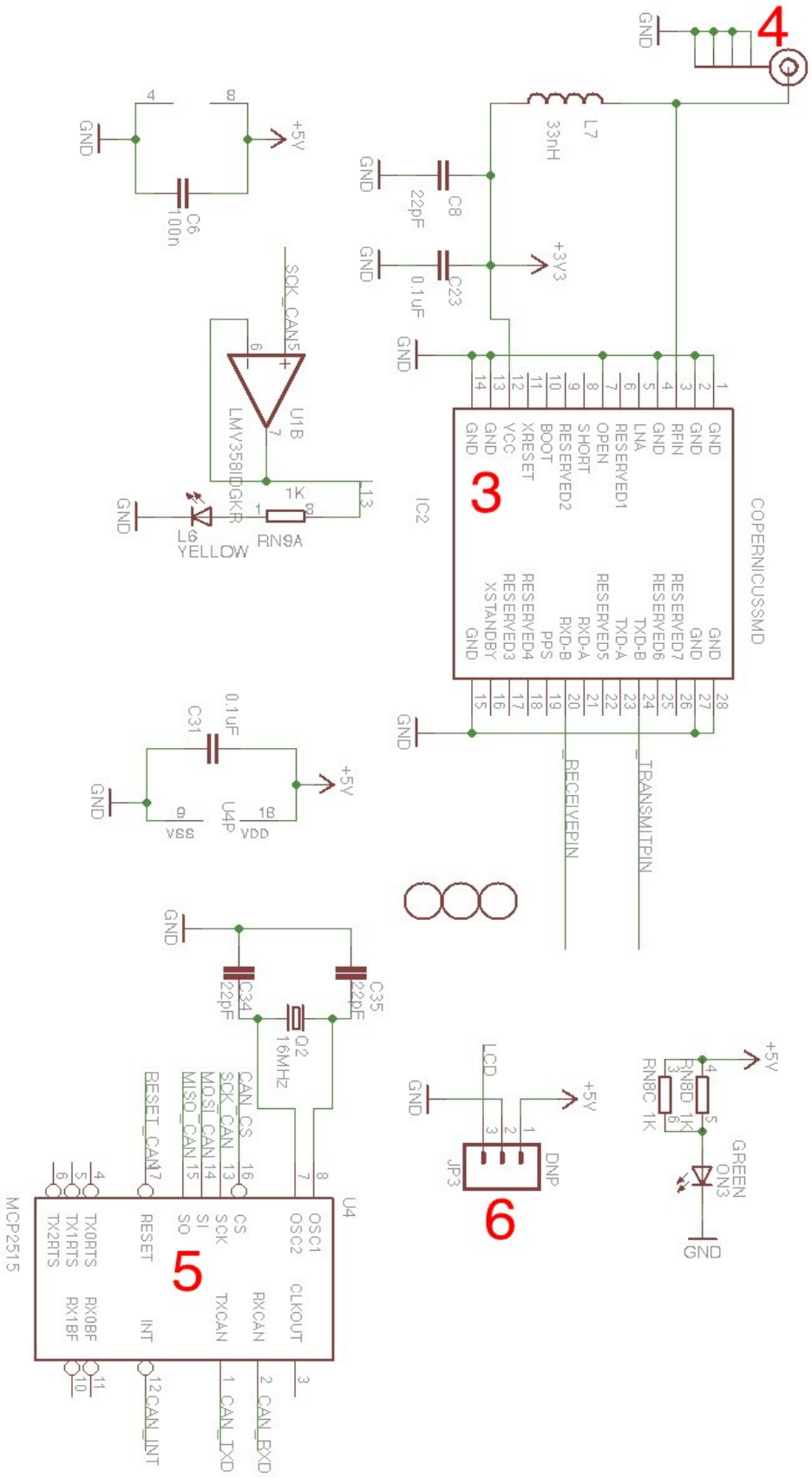


Fig. 3.3.3: ATmega328 per il controllo di CAN-Bus, GPS e LCD

(a)


4
GND
L7
33nH
C8
22pF
+3V3
C23
0.1uF
C6
100n
+5V
SCK_CAN5
U1B
LM358IDGKR
1K
RN9A
L6
YELLOW
IC2
3
COPERNICUSSMD
GND
RFIN
LNA
RESERVED1
OPEN
SHORT
RESERVED2
BOOT
XRESET
VCC
XSTANDBY
RESERVED3
RESERVED4
PPS
RXD-B
RXD-A
TXD-A
TXD-B
RESERVED6
RESERVED7
_RECEIVEPIN
_TRANSMITPIN
C31
0.1uF
U4P
VSS
VDD
C34
22pF
C35
22pF
Q2
16MHz
MCP2515
U4
5
RESET_CAN7
CAN_CS_16
SCK_CAN_13
MOSI_CAN14
MISO_CAN15
TX0RTS
TX1RTS
TX2RTS
RX0BF
RX1BF
RESET
INT
CS
SCK
SI
SO
TXCAN
RXCAN
OSC1
OSC2
CLKOUT
12 CAN_INT
1 CAN_TXD
2 CAN_RXD
LCD
JP3
DNP
6
RN8D 1K
RN8C 1K
GREEN
ON3

(b)

(1) L'MCP2551[17] è un dispositivo CAN ad alta velocità il cui compito è quello di fare da interfaccia tra il bus fisico e il protocollo che lo gestisce; svolge perfettamente le funzioni descritte nel Cap. 2.1 offrendo una trasmissione ed una ricezione differenziale operando fino ad una velocità di 1 Mbit/sec. Attraverso le due linee provenienti dal connettore CAN, CANH e CANL, fornisce i due segnali per la comunicazione CAN_RXD e CAN_TXD.

(2) È l'ATmega328 che si occupa di gestire questa sezione; anche in questo caso il castello è stato eliminato perchè il dispositivo è dedicato completamente alla gestione del sistema telemetrico; restano utilizzati un pin per la comunicazione col CAN-Bus (CAN_INT), due pin collegati al chip per il controllo della ricezione/trasmissione GPS (_TRANSMITPIT e _RECEIVEPIN), uno per lo schermo LCD (LCD) e i due pin analogici per inserire questa sezione sul bus I^2C (I2C_0 e I2C_1).

(3) Il modulo Copernicus II[18] è un ricevitore GPS in grado di gestire fino a 12 canali con un bassissimo consumo di potenza (132 mW) in grado di supportare sia antenne attive che passive; permette di sfruttare il protocollo NMEA di cui ha bisogno il nostro sistema e a livello di schema si collega direttamente con l'antenna sul pin di ricezione RFIN e comunica con l'ATmega328 attraverso le linee _TRANSMITPIT e _RECEIVEPIN

(4) Il connettore per l'antenna esterna; si è fatta questa scelta perchè montarla direttamente sulla scheda avrebbe significato chiuderla all'interno della centralina, causando probabilmente la mancata ricezione del segnale. In questo modo invece sarà possibile portarla all'esterno, in una zona decisamente più favorevole del veicolo.

(5) L'MCP2515[19] è un CAN controller che implementa tutte le specifiche

necessarie e ne permette l'interfacciamento con un microcontrollore attraverso il protocollo seriale. Il dispositivo è composto da tre blocchi concettuali:

- **modulo CAN:** si occupa della gestione di tutte le funzioni per trasmettere e/o ricevere messaggi attraverso il bus; controlla anche che tutti quelli individuati non contengano errori;
- **controllo logico:** si occupa del setup e delle operazioni necessarie al chip stesso per interfacciarsi, facendo uso di piedini per la gestione degli interrupt che permettono una maggiore flessibilità del sistema;
- **comunicazione SPI:** usando i comandi di read e write del protocollo seriale è possibile interagire con tutti i registri fisici del microcontrollore a cui il chip è collegato.

(6) Questa connessione permette il collegamento di un LCD seriale che può essere utilizzato per visualizzare dati importanti e utilizzabile anche come debugger in caso di assenza di un computer.

In Fig. 3.3.4a-b abbiamo l'ultima sezione, quella in cui è contenuta praticamente una piattaforma Arduino UNO con qualche lieve modifica. Il suo compito è quella di permettere alla scheda di essere usata anche in altri tipi di progetti e lasciare la possibilità di espanderla; nel caso specifico della scheda per la telemetria è previsto che sul castello venga collegato il modulo GPRS illustrato nel Cap. 2.1 o comunque un sistema di comunicazione alternativo a quello via ethernet.

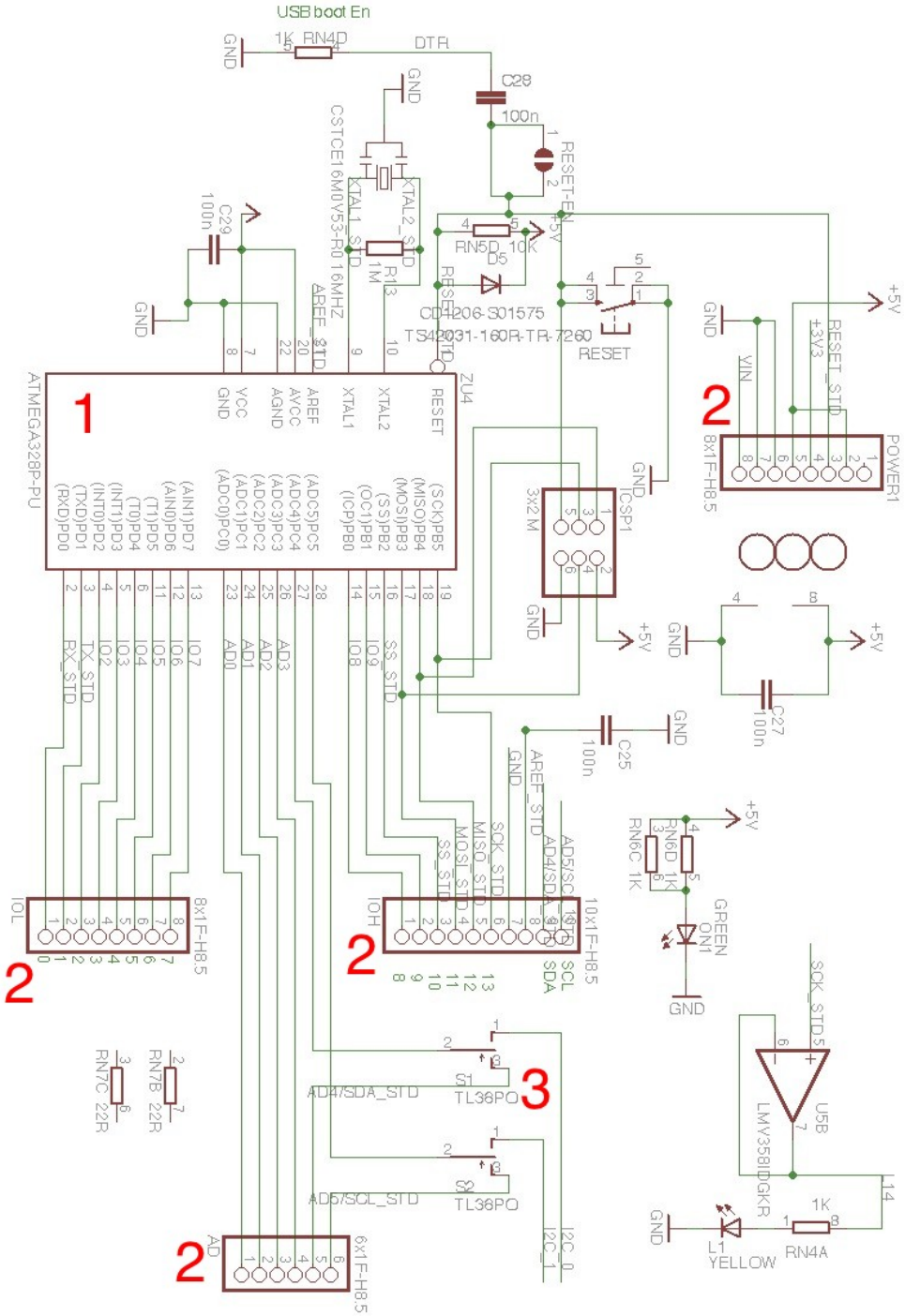
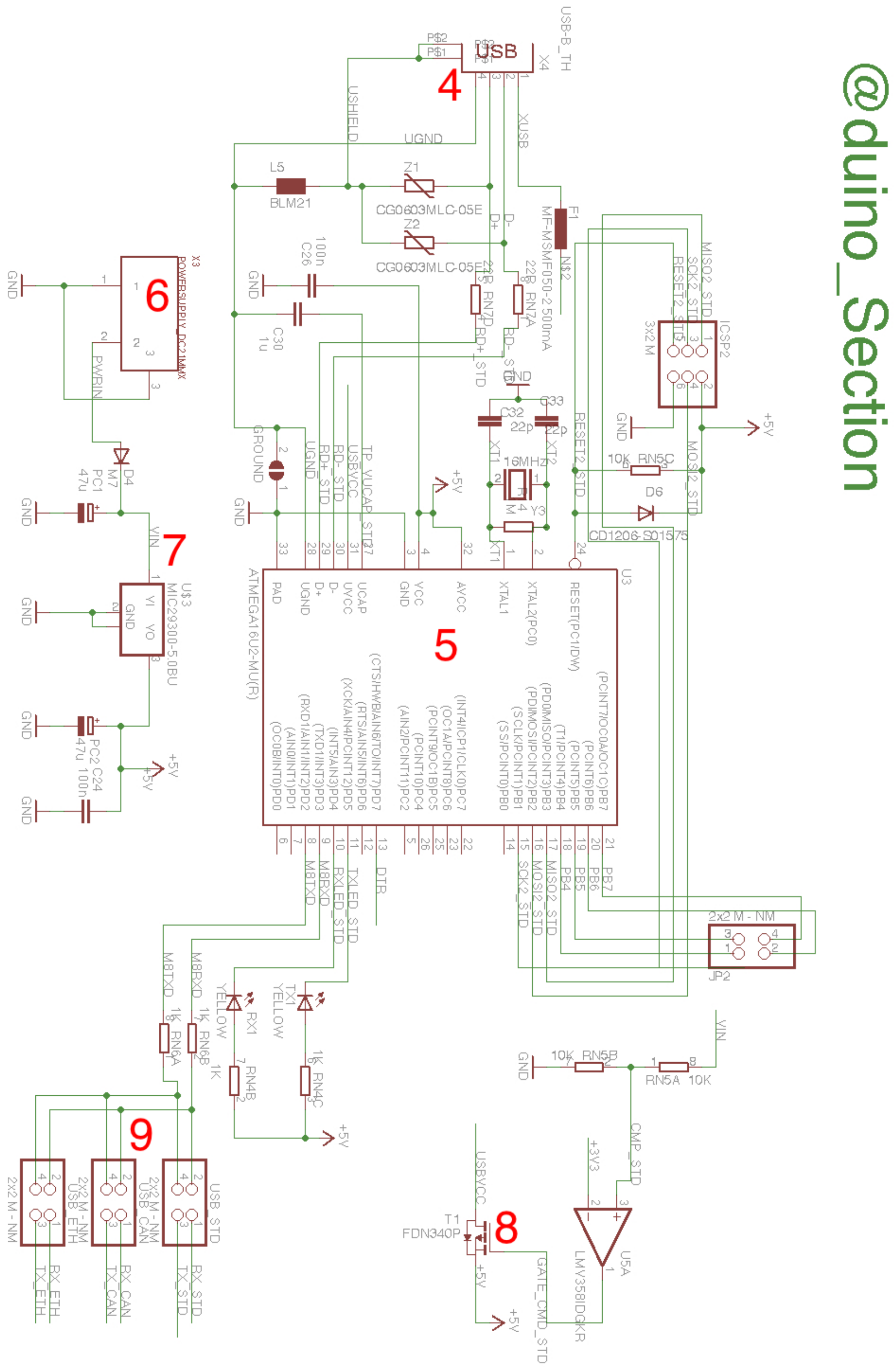


Fig. 3.3.4: ATmega328 per il controllo di moduli esterni

(a)


@duino_Section
USB
4
5
6
7
8
9
ATMEGA16U2-MU(R)
MIC29300-5.0BU
FDN340P
LMV358IDGKR
16MHz
BLM21

(b)

(1) L'ATmega328 di questa sezione non svolge compiti specifici e mantiene le sue connessioni standard previste dalla scheda Arduino UNO.

(2) Il castello, cioè tutti i connettori dei pin input/output, su cui è possibile collegare dispositivi esterni, moduli compresi, viene mantenuto nella sua totalità in modo che sia possibile connettere al microcontrollore interfacce di vario genere.

(3) L'unica lieve modifica sui pin viene fatta per A4 e A5: su di essi è stato inserito uno switch che permette di connettere le due uscite del microcontrollore al castello oppure dirottarli direttamente sul bus I^2C che permette di mettere in comunicazione questo ATmega328 con gli altri due presenti sulla scheda; la scelta su come disporre di questa linea è fatta dal progettista a seconda del progetto su cui si sta lavorando.

(4) Il connettore USB permette al sistema di collegarsi ad un computer in modo da poter essere alimentato, ma soprattutto da poter programmare il microcontrollore con il software da utilizzare; le due linee che permettono questo sono RD+_STD e RD-_STD che trasmettono un segnale differenziale codificato in tipo NRZI e arrivano all'ATmega16U2 che si occupa della gestione del segnale stesso.

(5) L'ATmega16U2[20] è un microcontrollore a 8 bit che nel nostro sistema si occupa principalmente di effettuare la conversione USB-to-Serial, cioè di convertire il segnale proveniente dalla presa USB in una comunicazione di tipo seriale in modo che la porta possa essere sincronizzata con l'ATmega328 per quanto riguarda il caricamento del software; mette a disposizione 16 Kbytes di memoria flash programmabile con capacità di lettura/scrittura, 512 bytes di EEPROM, 22 linee input/output e 32 registri generici, oltre a timer,

contatori, una porta seriale e un watch dog con oscillatore interno.

(6) Questo è il connettore per l'alimentazione esterna, di cui si è già parlato nel Cap. 2.1.

(7) Il MIC29300BU[21] è un regolatore di tensione per alte correnti e ad alta precisione il cui compito è quello ci prendere in ingresso VIN e convertirla in una tensione stabile di +5 V necessaria per buona parte delle alimentazioni dei circuiti della scheda. Il dispositivo non è lo stesso originalmente previsto nel sistema Arduino UNO, ma il cambiamento è stato necessario perchè nell'ipotesi in cui si monti il modulo GPRS questo arriverebbe a dei picchi di corrente assorbita di 2 A in fase di trasmissione di alcuni tipi di messaggio; il 29300 infatti è in grado di supportare fino a 3 A.

(8) Questo MOSFET ha il compito di fare da interruttore su come alimentare il sistema: attraverso il regolatore da +5 V oppure direttamente dalla connessione USB; la scelta viene effettuata comparando la VIN con una tensione di 3.3 V.

(9) Questi tre jumper sono la modifica più significativa di questa sezione. Il loro compito è quello di dirottare le linee TX ed RX in uscita dal microcontrollore Atmega16U2 ad uno solo dei tre ATmega328 presenti sulla scheda in modo da programmarlo in caso di necessità; questa modifica è necessaria perchè si è logicamente scelto di implementare una sola interfaccia USB per l'intero sistema. Come anticipato in precedenza la chiusura dei jumper sarà effettuata a mano dal programmatore nelle fasi precedenti all'inserimento della scheda nella centralina.

A questo punto il progetto è finito sulla carta; rimane da realizzare il PCB layout cioè disporre i package associati a tutti i componenti utilizzati negli

schematici rispettando le specifiche fatte in Fig. 3.1.2.
Si sono controllati i datasheet di ogni componente per controllare le sue reali dimensioni e una volta associato il package corretto si è sfruttato il comando di Eagle "Board" che crea anche i cosiddetti elastici, cioè delle linee gialle che mantengono i collegamento logico tra componenti in modo da essere facilitati quando si andranno a disegnare le linee elettriche reali.
A tal proposito si fa notare che la scheda è stata realizzata sfruttando due layer di piste

- **TOP:** è il livello superiore, quello in cui vengono montati i componenti e su EAGLE è indicato con il colore rosso;
- **BOTTOM:** come intuibile, è il livello inferiore dove rimangono le saldature dei componenti non SMD e utilizzato per fare passare tutte le linee che il TOP non riesce a connettere.

In Fig. 3.3.5 è rappresentato la bozza di disposizione componenti che rispetti le specifiche relative ai componenti che dovranno interfacciarsi con l'interno o l'esterno della centralina, in particolare quelle Int rimangono nella parte superiore dell'immagine, le Ext in quella inferiore.

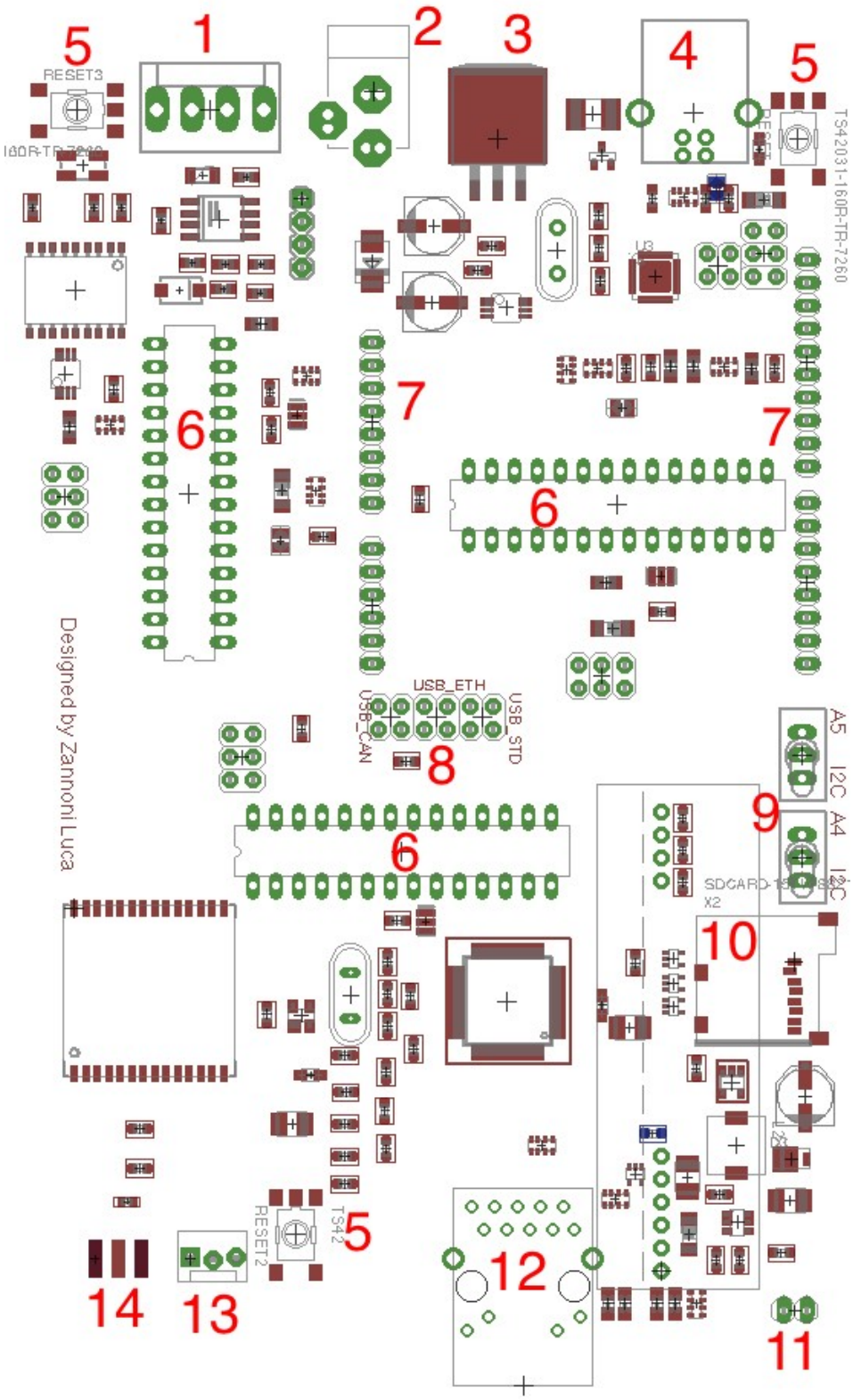


Fig. 3.3.5: disposizione componenti

Si illustra brevemente qual'è la corrispondenza con lo schematico degli elementi principali.

(1) Connettore CAN.
(2) Connettore per alimentazione esterna.
(3) Regolatore con uscita da +5 V, disposto sull'esterno perchè scalda molto e si vuole evitare il rischio di danneggiare le altre parti del sistema.
(4) Connettore USB.
(5) Pulsanti di reset dei tre ATmega328.
(6) ATmega328 delle tre sezioni logiche di lavoro del sistema.
(7) Castello di pin connessi all'ATmega328 della sezione UNO.
(8) Jumper di deviazione della connessione USB (da considerare a coppie).
(9) Switch di deviazione dei pin A4 e A5 dell'ATmega328 della sezione UNO tra castello e bus I^2C.
(10) Scheda μ-SD.
(11) Connettore del bus I^2C.
(12) Connettore ethernet RJ-45.
(13) Connettore di uscita per il collegamento dello schermo LCD seriale.
(14) Connettore per antenna.

In Fig. 3.3.6 infine abbiamo, per completezza, l'elenco di tutti i componenti, messo a disposizione del DIE con il quale si sta lavorando all'ottimizzazione del layout definitivo per mettere in produzione la scheda e passare poi alla fase di test

Partlist

Exported from Layout_1.sch

EAGLE Version 5.7.0 Copyright (c) 1988-2010 CadSoft

Part	Value	Package	Library	Sheet
AD	6x1F-H8.5	1X06	SmartPrj	3
C1	100n	C0603-ROUND	SmartPrj	1
C2	100n	C0603-ROUND	SmartPrj	1
C3	100n	C0603-ROUND	SmartPrj	1
C4	100n	C0603-ROUND	SmartPrj	1
C5	100n	C0603-ROUND	SmartPrj	1
C6	100n	C0603-ROUND	SmartPrj	2
C7	100n	C0603-ROUND	SmartPrj	1
C8	22pF	C0603-ROUND	SmartPrj	2
C9	22p	C0603-ROUND	SmartPrj	1
C10	10n	C0603-ROUND	SmartPrj	1
C11	10u	SMC_B	rcl	1
C12	22p	C0603-ROUND	SmartPrj	1
C13	10u	SMC_B	rcl	1
C14	470u	PANASONIC_D	rcl	1
C15	1u	C0603-ROUND	SmartPrj	1
C16	100n	C0603-ROUND	SmartPrj	1
C17	100n	C0603-ROUND	SmartPrj	1
C18	10u	SMC_B	rcl	1
C19	10u	SMC_B	rcl	1
C20	100n	C0603-ROUND	SmartPrj	1
C21	100n	C0603-ROUND	SmartPrj	1
C22	1u	C0603-ROUND	SmartPrj	1
C23	0.1uF	C0603-ROUND	SmartPrj	2
C24	100n	C0603-ROUND	SmartPrj	3
C25	100n	C0603-ROUND	SmartPrj	3
C26	100n	C0603-ROUND	SmartPrj	3
C27	100n	C0603-ROUND	SmartPrj	3
C28	100n	C0603-ROUND	SmartPrj	3
C29	100n	C0603-ROUND	SmartPrj	3
C30	1u	C0603-ROUND	SmartPrj	3
C31	0.1uF	0603-CAP	SparkFun	2
C32	22p	C0603-ROUND	SmartPrj	3
C33	22p	C0603-ROUND	SmartPrj	3
C34	22pF	0603-CAP	SparkFun	2
C35	22pF	0603-CAP	SparkFun	2
C40	560pF	0603-CAP	SparkFun	2
C41	560pF	0603-CAP	SparkFun	2
C42	0.1uF	0603-CAP	SparkFun	2
C43	4.7uF 50v	0603-CAP	SparkFun	2
C48	100n	C0603-ROUND	SmartPrj	2
C49	100n	C0603-ROUND	SmartPrj	2
D2	4148	MINIMELF	diode	1
D3	SS1P3L	DO220AAL	SmartPrj	1
D4	M7	SMB	diode	3
D5	CD1206-S01575	MINIMELF	diode	3
D6	CD1206-S01575	MINIMELF	diode	3
D8	MBRA140	SMA-DIODE	SparkFun	2

D11	CD1206-S01575	MINIMELF	diode	2
F1	MF-MSMF050-2 500mA	L1812	rcl	3
F3	PTCSMD	PTC-1206	SparkFun	2
GROUND		SJ	jumper	3
IC1	CAT811TTBI-CT3	SOT143	SmartPrj	1
IC2	COPERNICUSSMD	COPERNICUS	SparkFun	2
IC3	74LVC1G125DCK	SC70-5	74xx-little-us	1
IC5	LM2736Y	SOT23-6	SmartPrj	1
IC6	W5100	SQFP-S-10X10-80	SmartPrj	1
IC10	74LVC1G125DCK	SC70-5	74xx-little-us	1
IC11	74LVC1G125DCK	SC70-5	74xx-little-us	1
IC12	74LVC1G125DCK	SC70-5	74xx-little-us	1
IC13	74LVC1G14DBV	SOT23-5	74xx-little-de	1
ICSP	ICSP	2X03	pinhead	1
ICSP1	3x2 M	2X03	SmartPrj	3
ICSP2	3x2 M	2X03	SmartPrj	3
ICSP5	3x2 M	2X03	SmartPrj	2
IOH	10x1F-H8.5	1X10	SmartPrj	3
IOL	8x1F-H8.5	1X08	SmartPrj	3
J$1	SMA_EDGE	SMA-EDGE	SparkFun	2
JP2	2x2 M - NM	2X02	SmartPrj	3
JP3	DNP	MOLEX-1X3_LOCK	SparkFun	2
JP4	DNP	1X04	SparkFun	2
JP7		1X02	pinhead	1
L	Green	CHIPLED_0805	led	1
L1	YELLOW	CHIP-LED0805	led	3
L2	4u7	SRR0604	SmartPrj	1
L3	BLM21	0805	wuerth-elektronik	1
L4	BLM21	0805	wuerth-elektronik	1
L5	BLM21	0805	wuerth-elektronik	3
L6	YELLOW	CHIP-LED0805	led	2
L7	33nH	C0402	SparkFun	2
ON	Green	CHIPLED_0805	led	1
ON1	GREEN	CHIP-LED0805	led	3
ON3	GREEN	CHIP-LED0805	led	2
PC1	47u	PANASONIC_D	rcl	3
PC2	47u	PANASONIC_D	rcl	3
POWER1	8x1F-H8.5	1X08	SmartPrj	3
Q1	25MHz	QS	special	1
Q2	16MHz	CRYSTAL-SMD-5X3	SparkFun	2
R1	10k	R0603-ROUND	SmartPrj	1
R2	300R 1%	R0603-ROUND	SmartPrj	1
R3	12k 1%	R0603-ROUND	SmartPrj	1
R4	1M	R0603-ROUND	SmartPrj	1
R5	100k	R0603-ROUND	SmartPrj	1
R6	16K5	R0603-ROUND	SmartPrj	1
R7	10k	R0603-ROUND	SmartPrj	1
R8	0R	R0603-ROUND	SmartPrj	1
R9	N.M.	R0603-ROUND	SmartPrj	1
R10	N.M.	R0603-ROUND	SmartPrj	1
R11	10k	R0603-ROUND	SmartPrj	1
R12	10k	R0603-ROUND	SmartPrj	1
R13	1M	R0603-ROUND	SmartPrj	3
R14	1M	R0603-ROUND	SmartPrj	3
R17	1M	R0603-ROUND	SmartPrj	2
R25	100	0603-RES	SparkFun	2
R26	100	0603-RES	SparkFun	2

R27	4.7k	0603-RES	SparkFun	2
RESET	TS42031-160R-TR-7260	TS42	SmartPrj	3
RESET-EN		SJ	jumper	3
RESET-EN2		SJ	jumper	2
RESET2	TS42	TS42	st-mod v1.4	1
RESET3	TS42031-160R-TR-7260	TS42	SmartPrj	2
RN1	1k	CAY16	resistor-dil	1
RN2	49R9	CAY16	resistor-dil	1
RN3	1k	CAY16	resistor-dil	1
RN4	1K	CAY16	resistor-dil	3
RN5	10K	CAY16	resistor-dil	3
RN6	1K	CAY16	resistor-dil	3
RN7	22R	CAY16	resistor-dil	3
RN8	1K	CAY16	resistor-dil	2
RN9	1K	CAY16	resistor-dil	2
RN13	10K	CAY16	resistor-dil	2
RX	Yellow	CHIPLED_0805	led	1
RX1	YELLOW	CHIP-LED0805	led	3
S1	TL38PO	TL3XPO	switch	3
S2	TL38PO	TL3XPO	switch	3
T1	FDN340P	SOT-23	zetex	3
TX	Yellow	CHIPLED_0805	led	1
TX1	YELLOW	CHIP-LED0805	led	3
U$1	POE-RJ45	POE-RJ45	SmartPrj	1
U$3	MIC29300-5.0BU	TO263-3	linear	3
U1	LMV358IDGKR	MSOP08	linear	2
U3	ATMEGA16U2-MU(R)	MLF32	SmartPrj	3
U4	MCP2515	SO-18W	microchip_can	2
U5	LMV358IDGKR	MSOP08	linear	3
U10	MCP2551	SO-08M	skpang_lib	2
USB_CAN	2x2 M - NM	2X02	SmartPrj	3
USB_ETH	2x2 M - NM	2X02	SmartPrj	3
USB_STD	2x2 M - NM	2X02	SmartPrj	3
X1		KK-156-4	con-molex	2
X2	SDCARD-15TW-8821	SDCARD-15TW-8821	st-mod v1.3	1
X3	POWERSUPPLY	POWERSUPPLY_DC-2	SmartPrj	3
X4	USB-B_TH	PN61729	SmartPrj	3
Y1	16Mhz	RESONATOR	Nano_MySMD2	1
Y2	16MHZ	RESONATOR	SmartPrj	3
Y3	16MHz	QS	SmartPrj	3
Y6	16MHZ	RESONATOR	SmartPrj	2
Z1	CG0603MLC-05E	CT/CN0603	varistor	3
Z2	CG0603MLC-05E	CT/CN0603	varistor	3
ZU1	ATMEGA328P-PU	DIL28-3	SmartPrj	1
ZU2	ATMEGA328P-PU	DIL28-3	SmartPrj	2
ZU4	ATMEGA328P-PU	DIL28-3	SmartPrj	3
ZZ1	AG9XX0	AG9000	SmartPrj	1

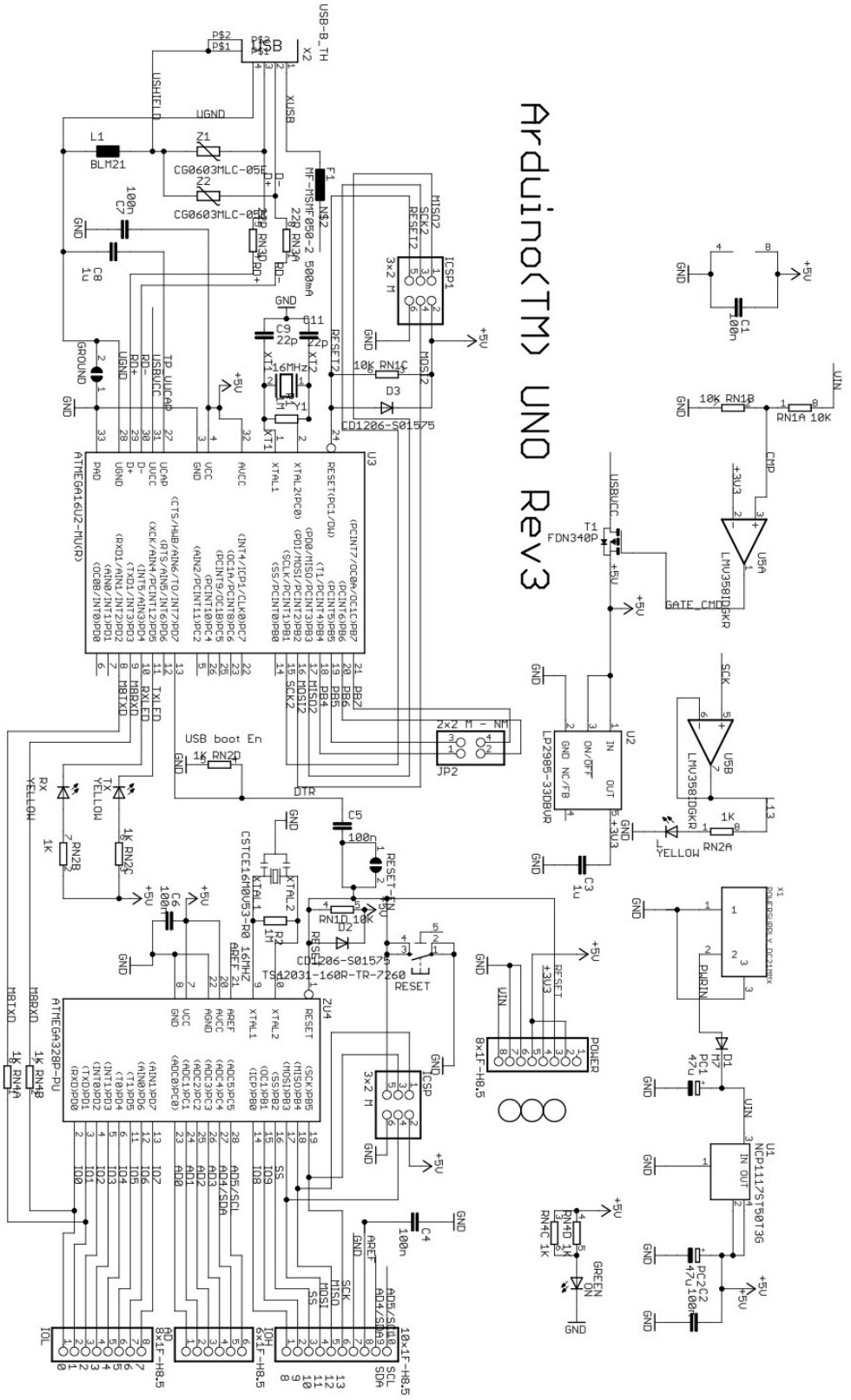
Fig. 3.3.6: elenco componenti

4 CONCLUSIONI

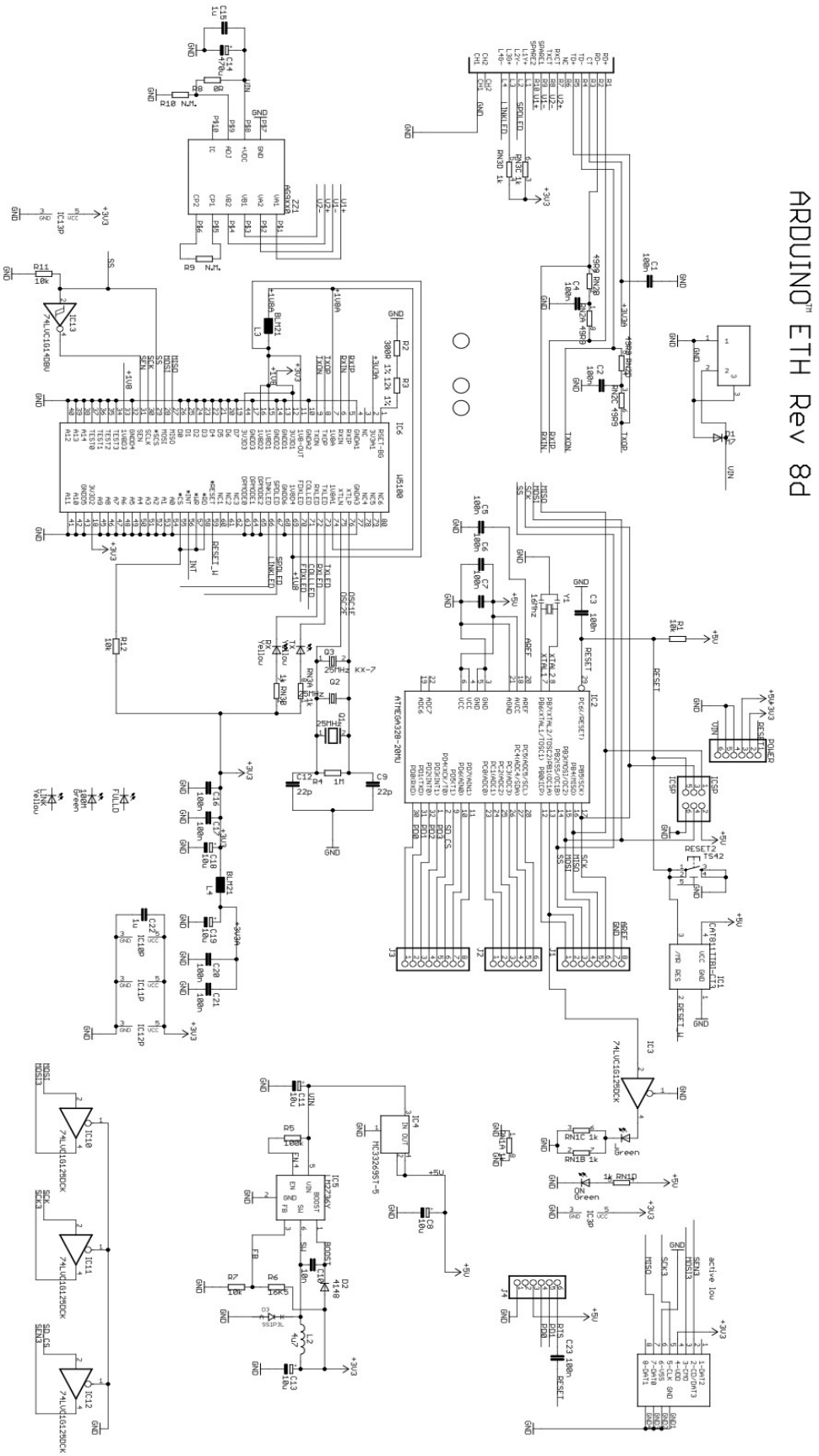
Dopo avere introdotto il sistema basato su Arduino nella sua interezza di hardware e software ci siamo focalizzati sul fulcro del lavoro di tesi.

L'obiettivo è stato quello di dare vita ad una scheda che basandosi sugli schematici di partenza e tenendo conto delle necessità del software, riuscisse ad implementare in un unico dispositivo il progetto iniziale basato sulla piattaforma Arduino, dimostrando ulteriormente che l'utilizzo di dispositivi open source paga tantissimo in termini di prestazione e permette anche una riduzione delle spese in fase di progetto e realizzazione.
Si sottolinea il fatto che oltre al progetto iniziale la scheda progettata lascia ampio spazio di manovra al lavoro di un progettista perchè grazie alla capacità di poter essere espansa e comprendere un ampio numero di interfacce permette lo sviluppo di sistemi e idee diverse tra loro, mettendo a disposizione una piattaforma completa e soprattutto in grado di rimanere aggiornata e compatibile in termini di nuovi prodotti disponibili nel settore, come ad esempio il modulo di espansione Wi-Fi che potrà permettere un'alternativa alla connessione via ethernet.
Questi obiettivi sono stati raggiunti a livello di studio e realizzazione e dovranno essere confermati in fase di test che dovrebbero avvenire in tempi brevi (la scheda è in produzione al momento della stesura) sempre grazie alla collaborazione del DIE che si è allo stesso tempo occupato dell'ottimizzazione del PCB layout, in modo da realizzare un prodotto allettante anche per una possibile interazione col settore di mercato relativo.

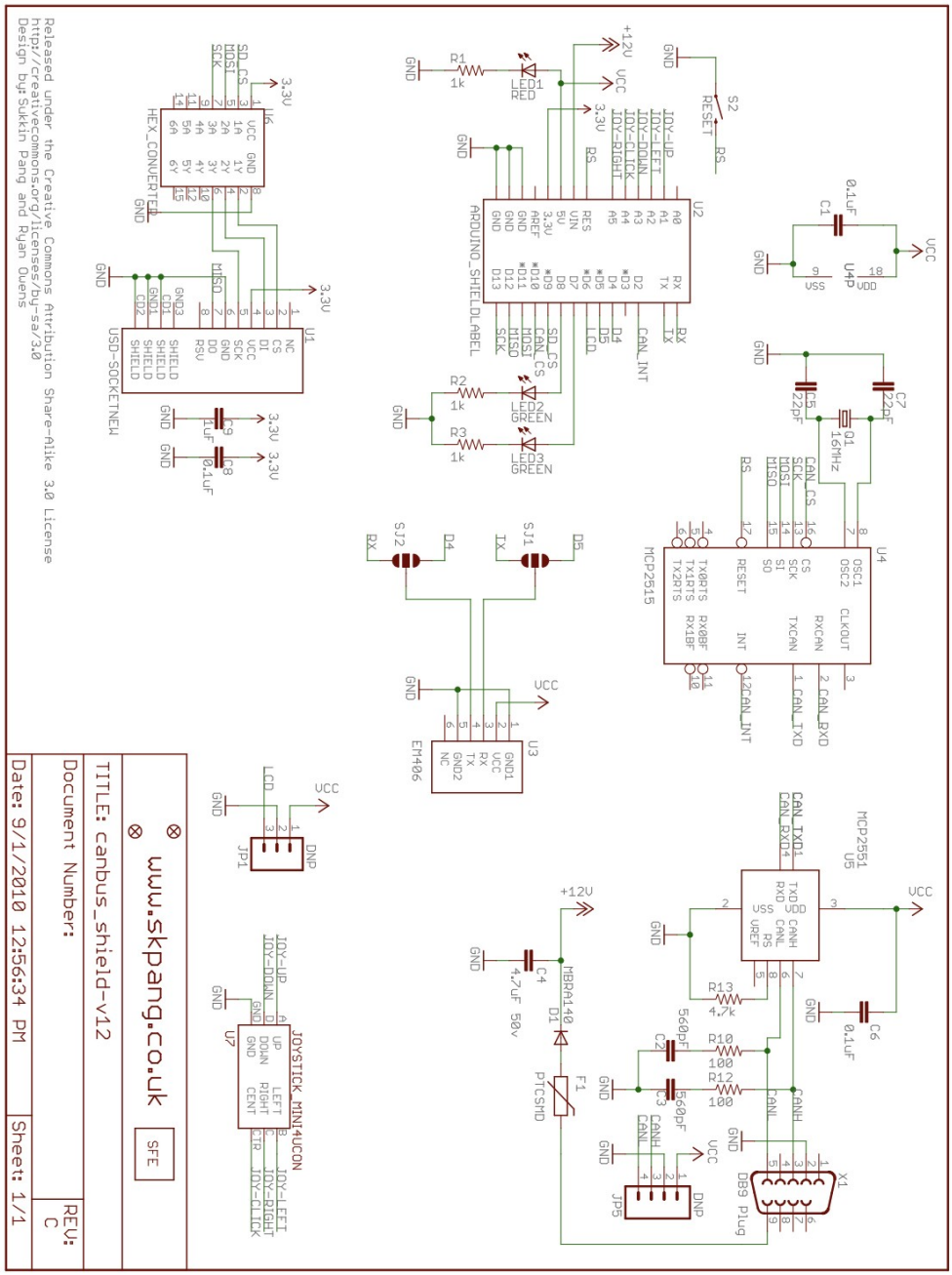
APPENDICE A – ARDUINO UNO



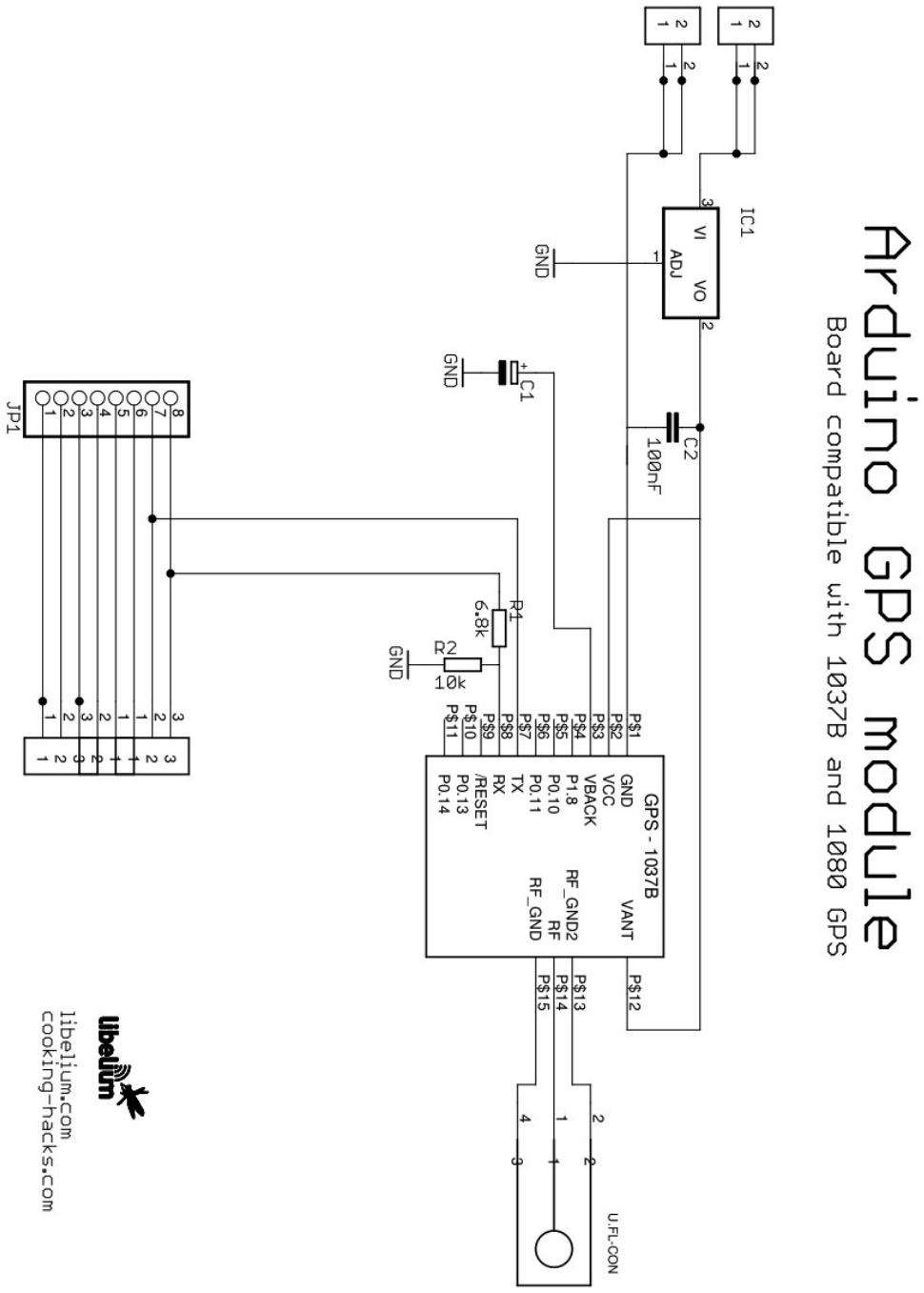
APPENDICE B – ARDUINO ETHERNET



APPENDICE C – CAN BUS SHIELD



APPENDICE D – LIBELIUM GPS SHIELD



BIBLIOGRAFIA

[1]. http://arduino.cc/en/Main/ArduinoBoardUno

[2]. http://www.sparkfun.com/products/10039

[3]. http://edge.rit.edu/content/P07122/public/EM-406%20SIRF3

[4]. Solari S., *Interfacciamento di reti di bordo per autoveicoli con internet*, Tesi di laurea magistrale, Cesena, Università di Bologna, II Facoltà di Ingegneria, Dicembre 2011

[5]. http://www.cooking-hacks.com/index.php/gps-module-for-arduino.html

[6]. http://www.cooking-hacks.com/index.php/internal-gps-antenna.html

[7]. Vincotech GPS Receiver Module A1080-A/-B, User's Manual (http://www.datasheetpro.com/node/60621?page=3)

[8]. http://en.wikipedia.org/wiki/NMEA_0183

[9]. http://arduino.cc/en/Main/ArduinoEthernetShield

[10]. http://www.cooking-hacks.com/index.php/shop/arduino/arduino-gprs-module.html

[11]. http://arduino.cc/en/Main/ArduinoBoardEthernet

[12]. http://www.cadsoftusa.com/?language=en

[13]. http://www.cadsoftusa.com/downloads/libraries/?language=en

[14]. http://www.sparkfun.com/

[15]. http://www.trimble.com/embeddedsystems/copernicus2.aspx?dtID=overview&

[16]. http://ww1.microchip.com/downloads/en/devicedoc/30277d.pdf

[17]. http://ww1.microchip.com/downloads/en/devicedoc/21667d.pdf

[18]. http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/GPS/63530-10_Rev-B_Manual_Copernicus-II.pdf

[19]. http://ww1.microchip.com/downloads/en/devicedoc/21801e.pdf

[20]. http://www.atmel.com/Images/doc7799.pdf

[21]. http://docs-europe.electrocomponents.com/webdocs/0adb/0900766b80adbbbe.pdf