Panda 4x4 OBD

Filippo Fondi - www.panda4x4.info

ATTENZIONE!
Il progetto e' stato testato solo su una Panda 4x4 Trekking 1108 SPI del 1999, non si garantisce il corretto funzionamento per altri modelli. Nel caso in cui testate il progetto su un modello diverso e funziona fatemelo sapere.

Storia del progetto

Tutto nasce dalla mia ossessione nel tenere sotto controllo la temperatura del motore, ogni 2 minuti l'occhio mi cadeva sul quadro strumenti e un giorno mi domandai tra me e me se era possible collegarsi in un qualche modo alla centralina, poi scopro la presenza della porta apposita nel cofano dedicata alla diagnostica. Compro un cavo che converte la porta 3pin al classico connettore obd2 e un cavo vag com kkl, mi collego con successo mediante l'utilizzo di un computer portatile. Allora iniziai a pensare a come creare una specie di computer di bordo, il portatile era troppo scomodo in quanto avrei dovuto accenderlo ogni volta ed aspettare che caricasse il Sistema operativo con il relativo software. Mi venne in mente di comprare un raspberry pi ma anche quest'ultimo avrebbe impiegato troppo tempo ad accendersi senza contare il fatto che il software utilizzato su windows non era disponibile per linux e i driver non erano compatibili. Ed infine la soluzione: Arduino, accensione istantanea e versatilita' di utilizzo con l'elettronica.

Funzionalita'

-Modalita' standalone e debug (OK)

-Lettura temperatura acqua e temperatura aria float (OK)

-Lettura tensione batteria float (OK)

-Lettura codice ISO all'avvio (OK)

-Lettura giri motore (OK), Velocita' in KM/H (Non Possibile)

-Allarme temperatura (aria o acqua) > 90 gradi (OK)

-Allarme voltaggio batteria <10v (OK)

-Verifica connessione avvenuta (OK)

-Lettura tempo iniezione (OK)

-Lettura anticipo accesione (OK)

-Lettura pressione aspirata (OK)

-Lettura posizione farfalla (OK)

-Lettura Posizione Stepper (OK)

-Lettura correzione sonda lambda (OK)

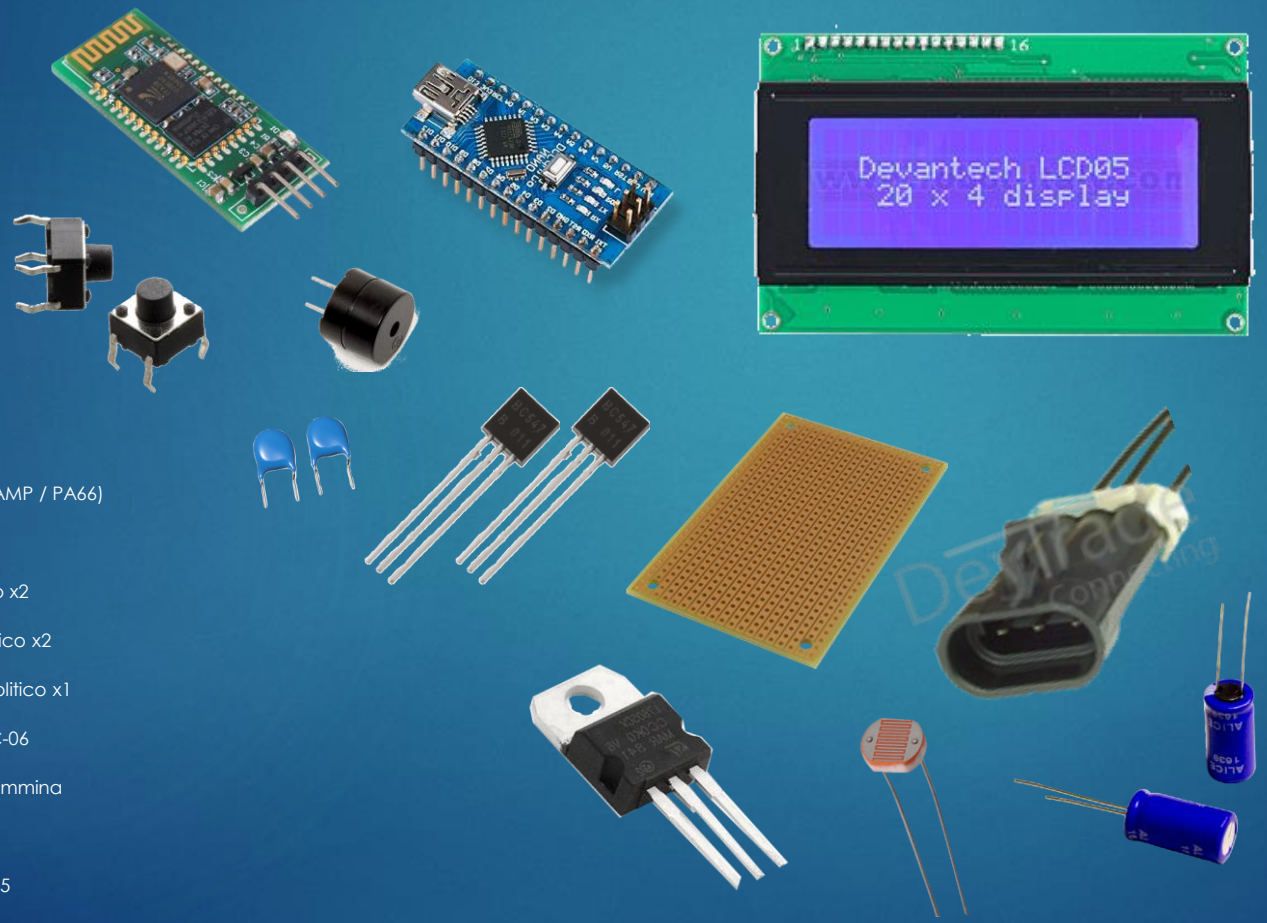
-Lettura marcia corrente + led/pixel blink (...)

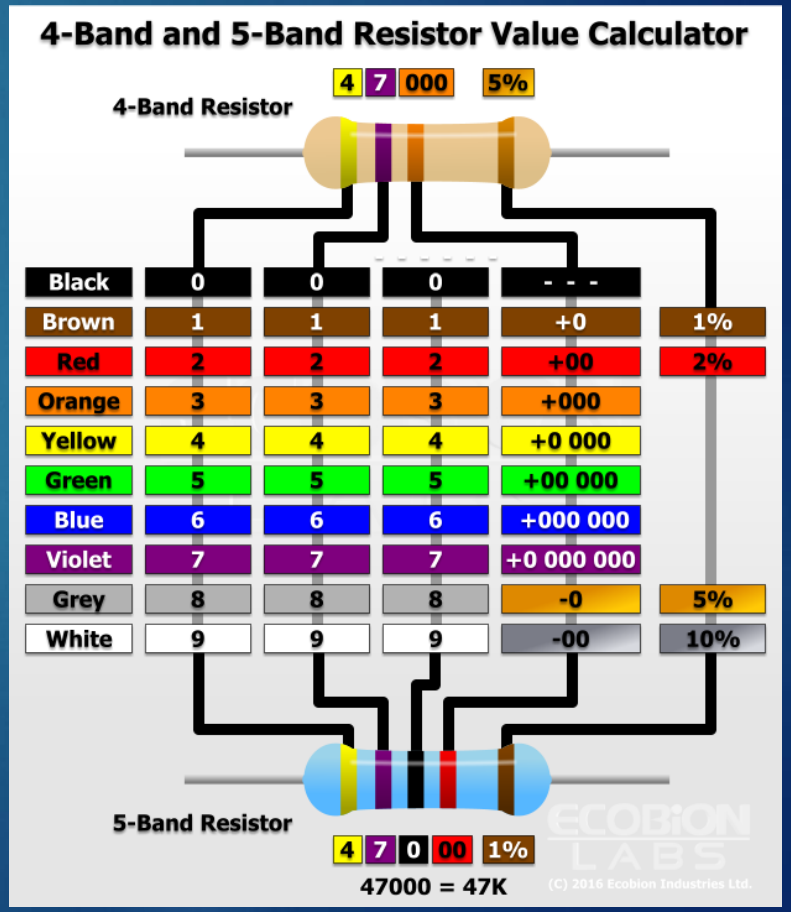
-Funzionalita' bluetooth + app Android (OK)
-Lettura Correzione Integrale Minimo (OK)
-Lettura Correzione Proporzionale Minimo (OK)
-Lettura Trimmer Titolo (OK)
-Lettura Obiettivo Giri Minimo (OK)
-Lettura Offset Giri al Minimo (OK)
-Lettura Delta Regolatore Minimo (OK)
-Lettura Correzione Passi da FLT (OK)

Sezione Hardware

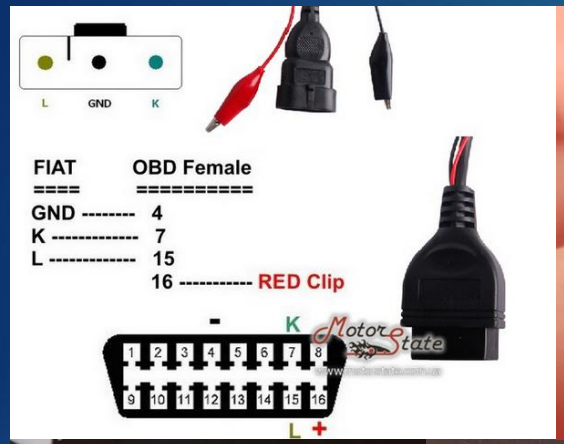
Prima di spiegare il funzionamento software della centralina, illustro l'hardware e la componentistica necessaria per interfacciarsi correttamente.
Ecco la lista dei componenti necessari per la costruzione del modello completo:

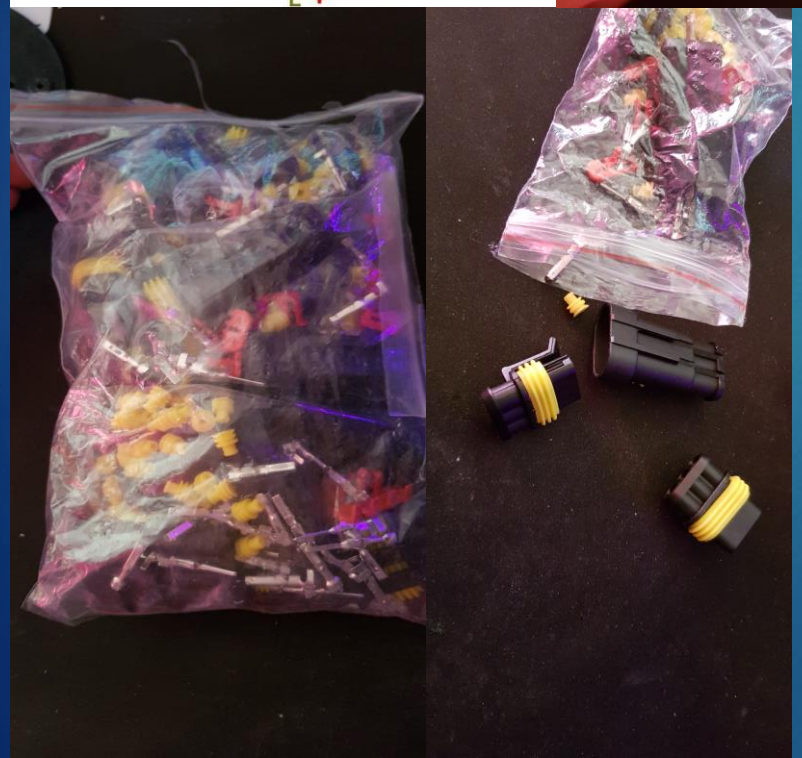
- -Arduino Nano x1
- -Millefori (5x7cm) x1
- -Transistor BC547 x2
- -Buzzer passivo x1
- -Resistenza 33 kohm x1
- -Resistenza 47 kohm x1
- -Resistenza 10 kohm x2
- -Resistenza 4.7 kohm x1
- -Resistenza 560 ohm x2
- -Display LCD 20X4 i2c
- -Connettore maschio 3 pin (AMP / PA66)
- -Fotoresistenza
- -Condensatore 1nF ceramico x2
- -Condensatore 10uF elettrolitico x2
- -Condensatore 3300uF elettrolitico x1
- -Modulo bluetooth HC-05/HC-06
- -Connettore USB maschio&femmina
- -Pulsante
- -Stabilizzatore di tensione 7805





Sezione Hardware





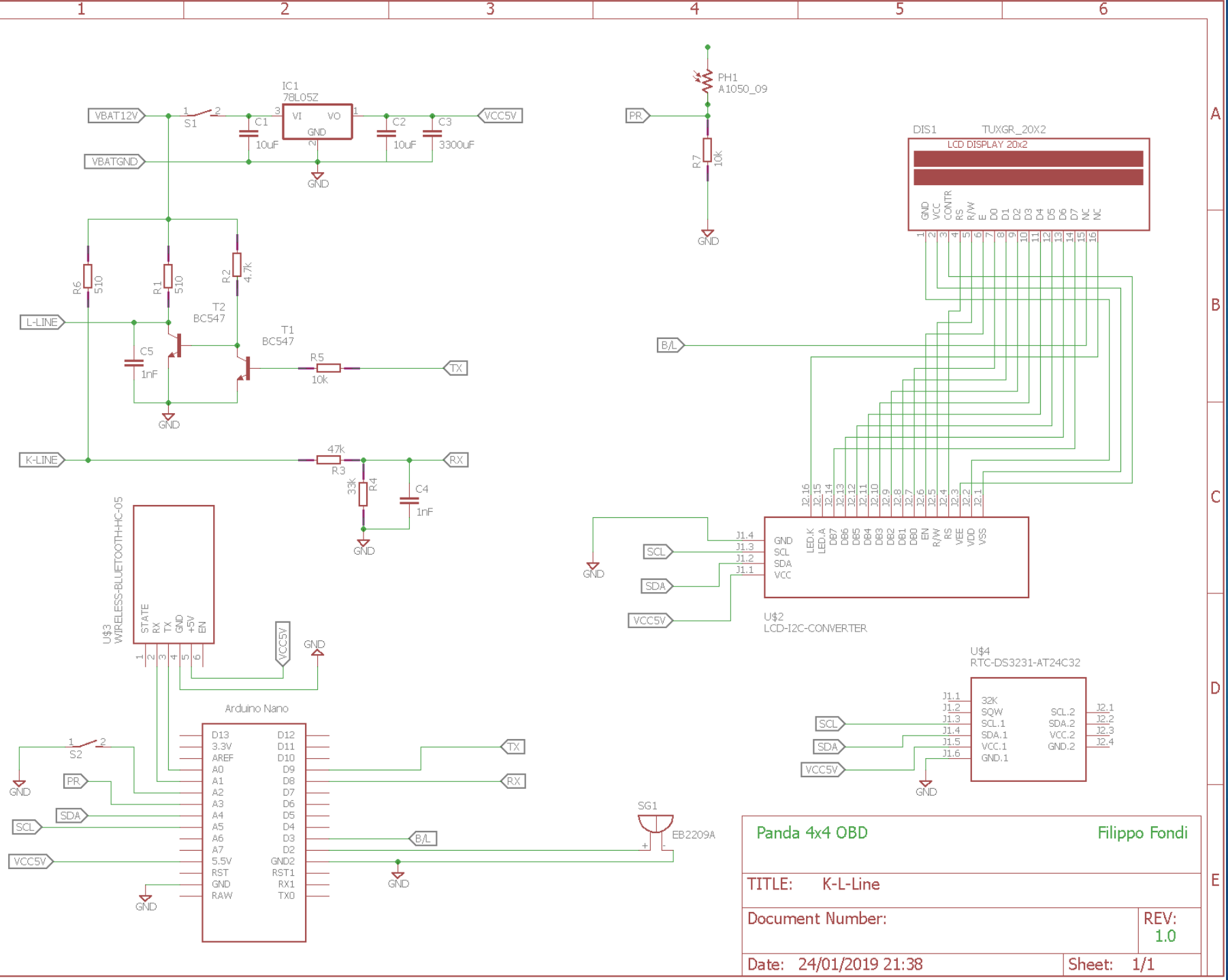
Partiamo illustrando il connettore alla quale ci collegheremo con Arduino. Il connettore e' un tipico Fiat 3 PIN composto da 3 pin:

-L: L-Line, linea utilizzata dalla centralina per la ricezione di richieste riguardanti i parametri del motore. Quindi su questa linea si inviano le domande.

-K: K-Line, linea utilizzata dalla centralina per l'invio delle informazioni richieste. Quindi su questa linea si ricevono le risposte.

-GND: massa del segnale.

Sulle altre automobili viene utilizzata la K-Line in modo bidirezionale (Per invio e ricezione) la panda invece le utilizza entrambe in modo unidirezionale divise una per ricezione ed una per trasmissione.



Sezione Hardware

Lo schema elettrico rappresentato e' quello piu' aggiornato.

In alto a sinistra abbiamo il 7805 che stabilizza la tensione da 12v della batteria a 5v per il nostro Arduino, I 2 condensatori da 10uf servono per filtrare la tensione in entrata e in uscita. Il regolatore interno ad Arduino supporta la 12v ma quando la macchina e' in funzione la batteria puo' arrivare anche a 15v, meglio non rischiare, potrebbe cuocersi.

Sotto al 7805 abbiamo la linea TX che useremo per inviare da Arduino le richieste sulla L-Line. I due transistor formano un level shifter, il compito di T1 e' di convertire I 5v di Arduino in 12v (perche' la centralina lavora con 0-12v, Arduino invece lavora con 0-5v), ma facendo questo T1 inverte anche il segnale, il compito di T2 e' di invertire nuovamente il segnale. Il resistore da 510ohm e' di pull-up.

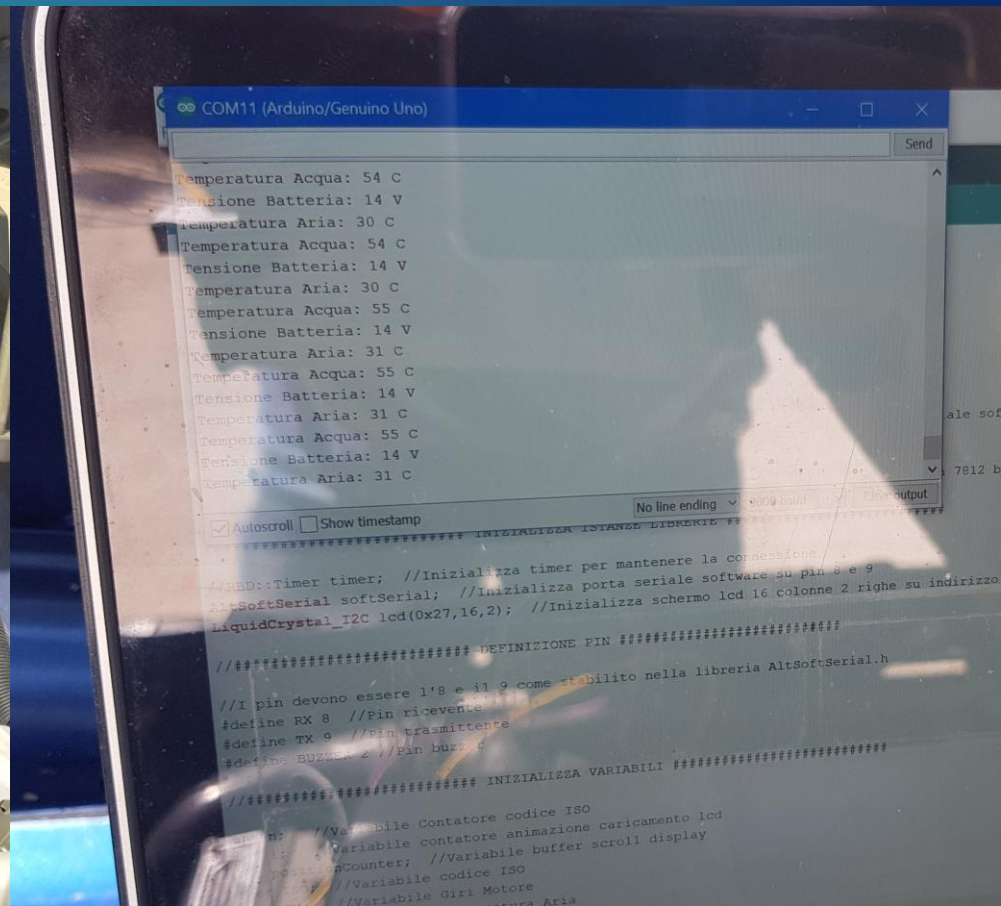
Piu' sotto ancora abbiamo la linea RX che servira' a ricevere le risposte inviate dalla centralina tramite la K-Line. Essa e' formata da un partitore di tensione che abbassa I segnali di 12v a 5v per essere letti da Arduino.

A destra c'e' il circuito relativo al display lcd con il relativo modulo i2c, che ci permette di utilizzare il display tramite due soli pin invece di 6.

I due condensatori ceramici da 1nF servono a filtrare i disturbi. La sezione alto-centrale e' necessaria per acquisire il livello di luce e regolare di conseguenza tramite la tecnica PWM la retroilluminazione del pannello LCD.

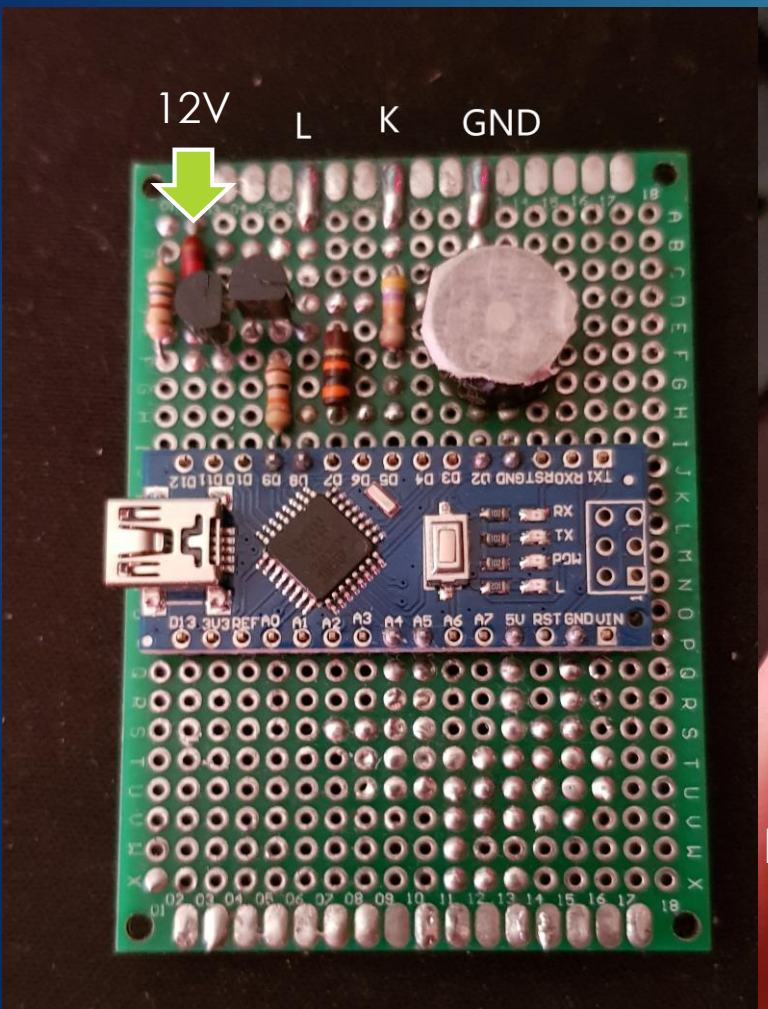
Sezione Hardware

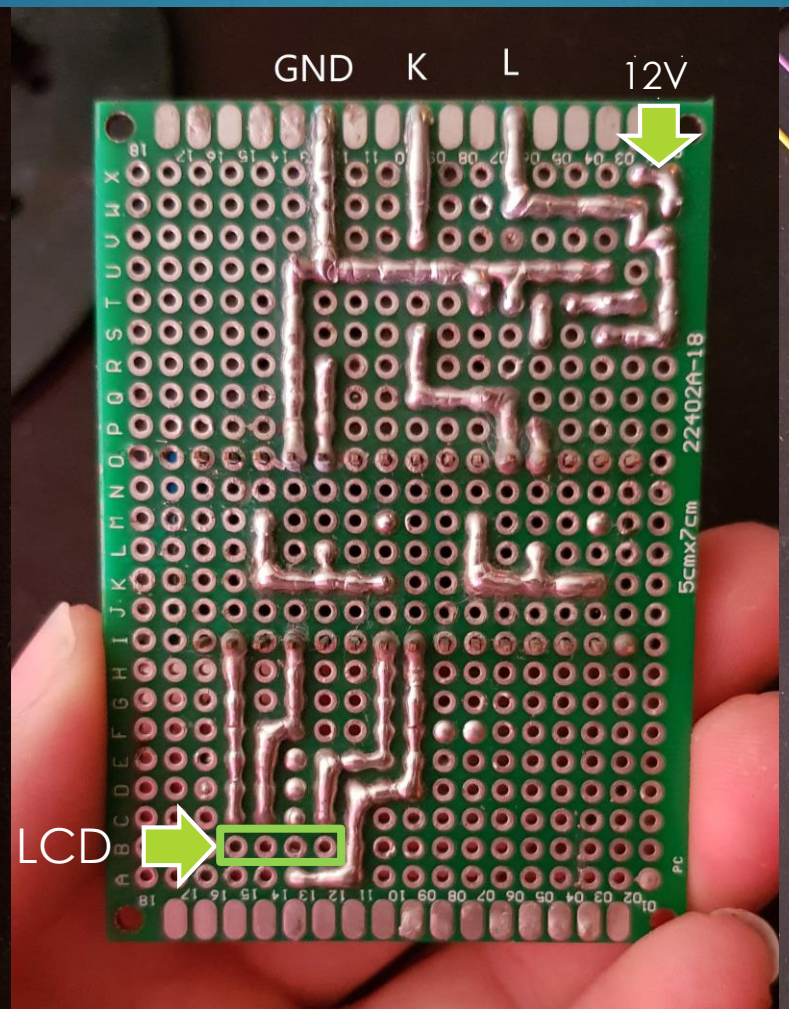
Prima di saldare il circuito che ho progettato l'ho prima testato su una breadboard per assicurarmi che il tutto funzionasse. Mi sono collegato con successo alla centralina e ricevuto correttamente i valori delle temperature acqua-motore e la tensione della batteria in accordo al programma caricato su arduino che illustrero' successivamente.

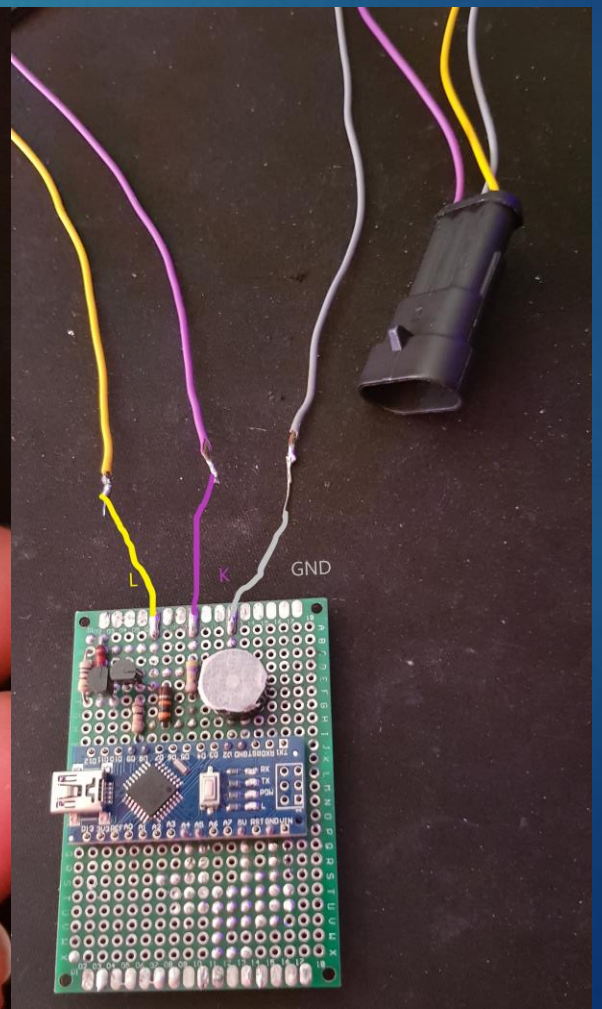


Sezione Hardware

Il primo prototipo che ho realizzato e' questo:







Come potete notare la parte del circuito relativa alla stabilizzazione della tensione dalla batteria e' assente perche' questo e' un prototipo della scheda. Infatti mi collegavo con il computer ad arduino tramite la porta micro usb per leggere i valori con il monitor seriale e in contemporanea lo alimentavo quindi la stabilizzazione da 12v a 5v non era necessaria. Ma la versione finale ne ha bisogno. Infatti in questa versione il display lcd non veniva collegato ma avevo predisposto le piste per il connettore.

Perfettamente funzionante in modalita' debug (monitor seriale con computer).

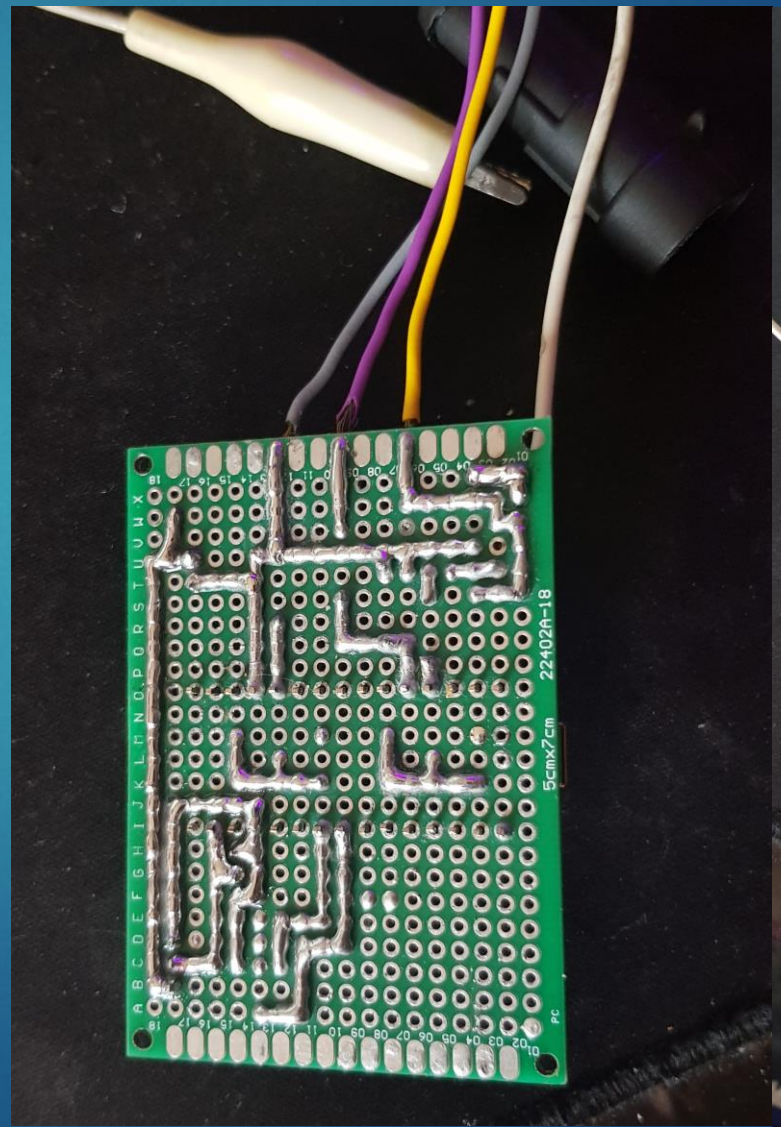
Sezione Hardware

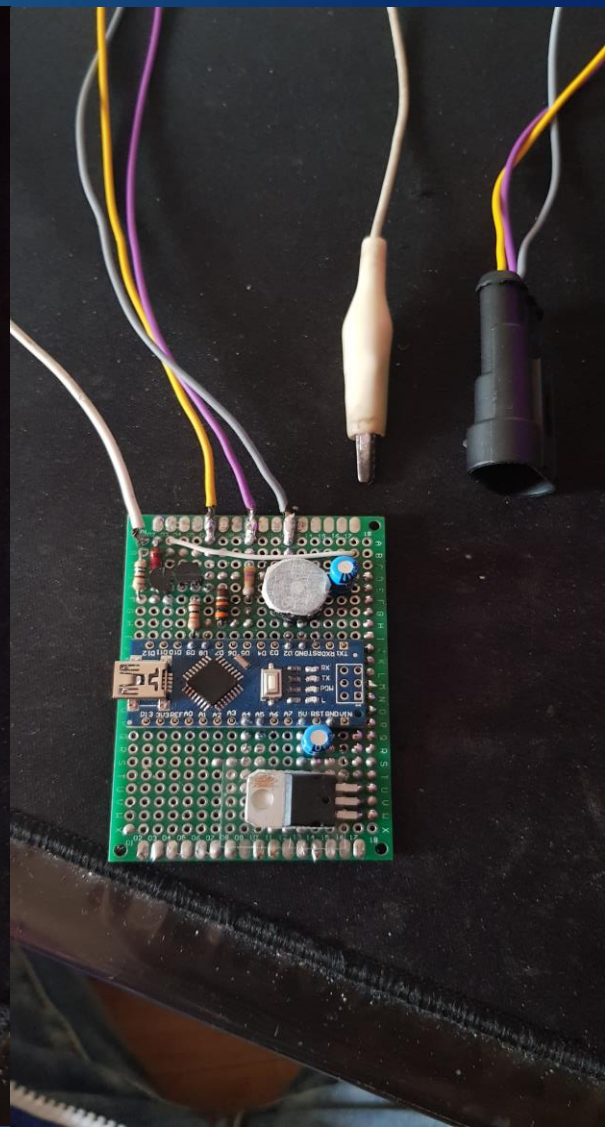
Ecco il secondo prototipo:

In questa versione ho inserito il circuito riguardante la stabilizzazione a 5v per Arduino mediante il 7805 e i relativi condensatori per filtrare la tensione e compensare la lunghezza dei fili.

Una volta testata questa versione con alimentazione stabilizzata + cavo usb mi sono reso conto che leggevo i valori solo a motore spento, (chiave on), se lo accendevo i valori diventavano irreali.

Il problema l'ho risolto con il terzo prototipo che segue.



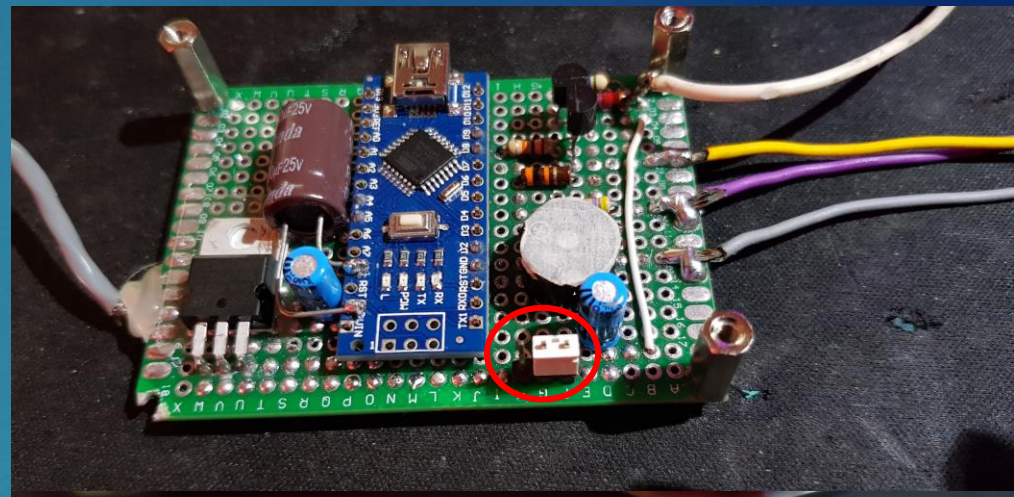


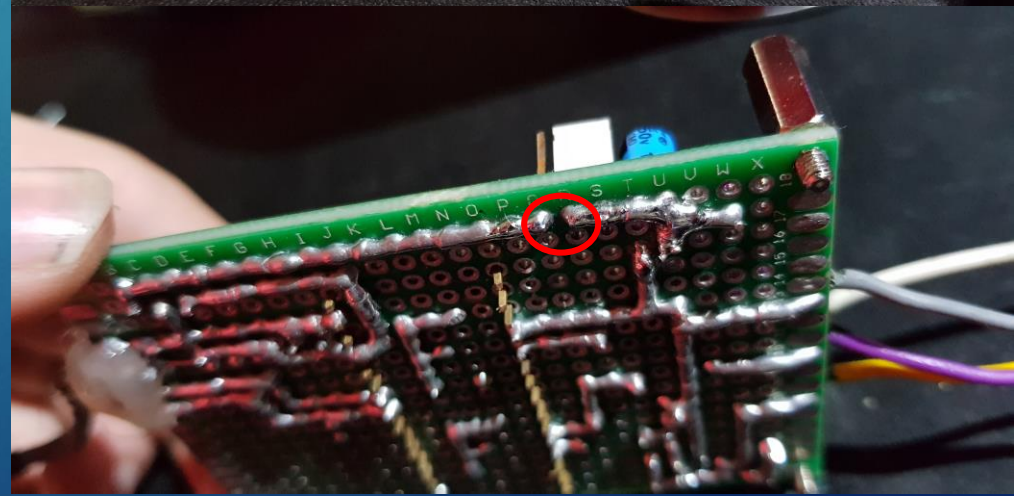
Sezione Hardware

Terzo prototipo:

In questa versione ho aggiunto un condensatore da 1000uf, un jumper per interrompere la pista dei 12v che va sul 7805 e il connettore per il display lcd. Con la versione precedente a motore acceso leggevo valori irreali, questo succedeva perche' andando ad accendere il motore, il motorino di avviamento consumava energia e causava un calo di tensione tale da non fornire il voltaggio necessario al 7805 per alimentare arduino a 5v e percio' arduino perdeva la connessione. Per risolvere il problema ho aggiunto un condensatore da 1000uf che funziona da "magazzino" durante quell'intervallo di tempo ed evitare che arduino si resetti.

Per evitare di alimentare arduino con 2 sorgenti ho inserito un jumper che mi permette di scegliere se alimentarlo in modo stabilizzato tramite la batteria della macchina (versione finale con lcd), oppure alimentarlo via usb (modalita' debug senza lcd, solo per modifiche allo sketch).

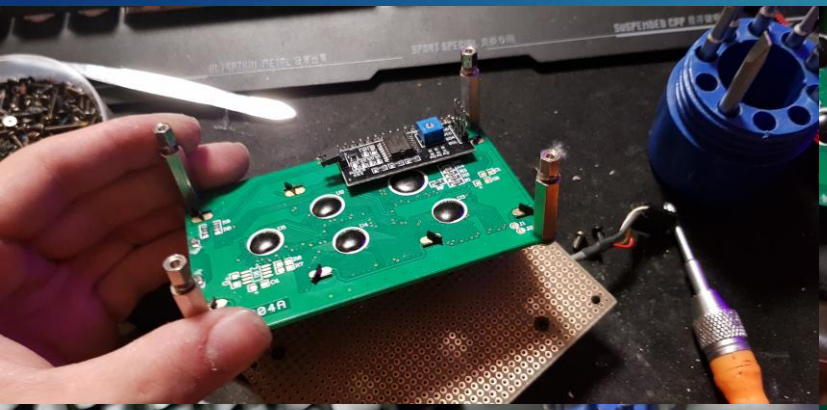


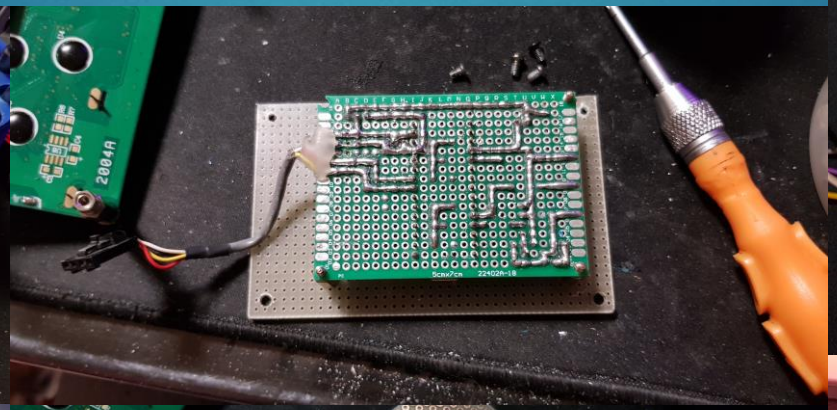


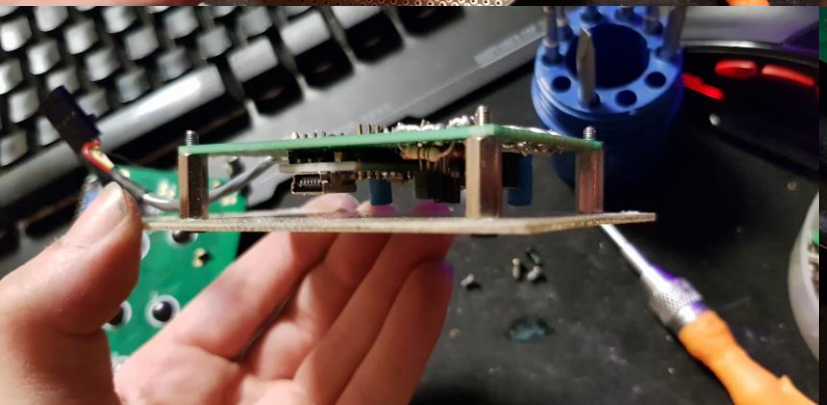
Sezione Hardware

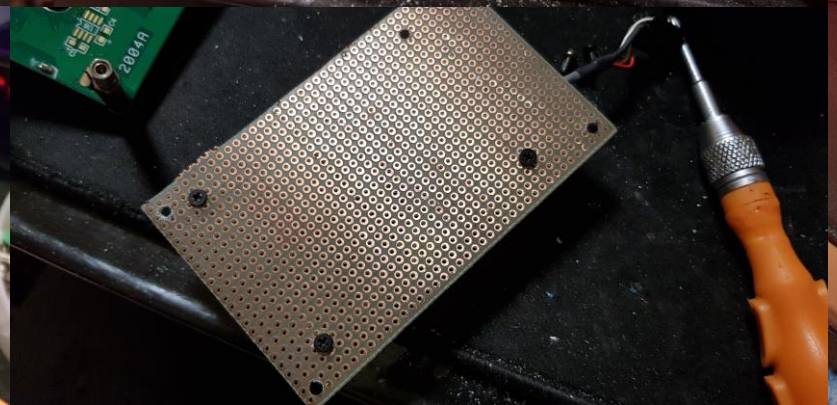
Terzo prototipo:

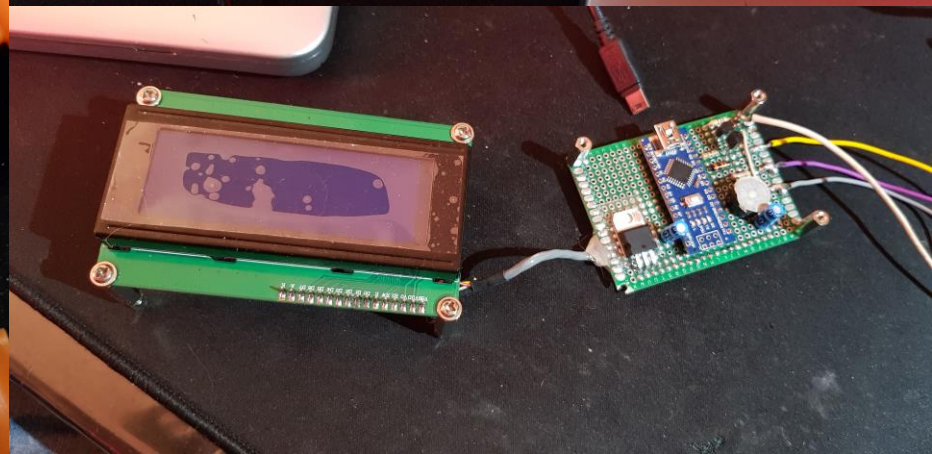
Inoltre ho costruito uno chassis per rendere lo strumento compatto e bello da vedere.





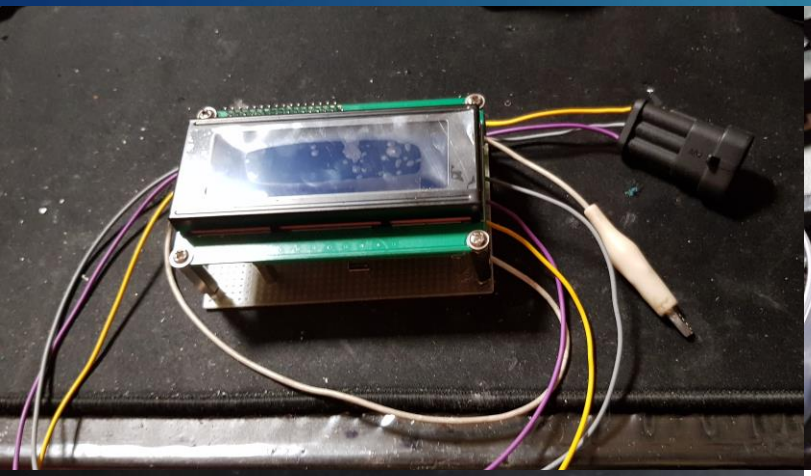


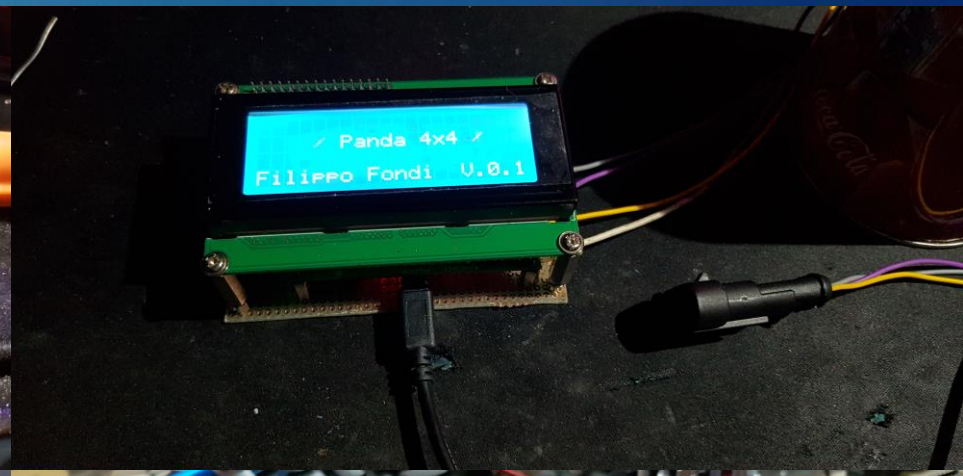


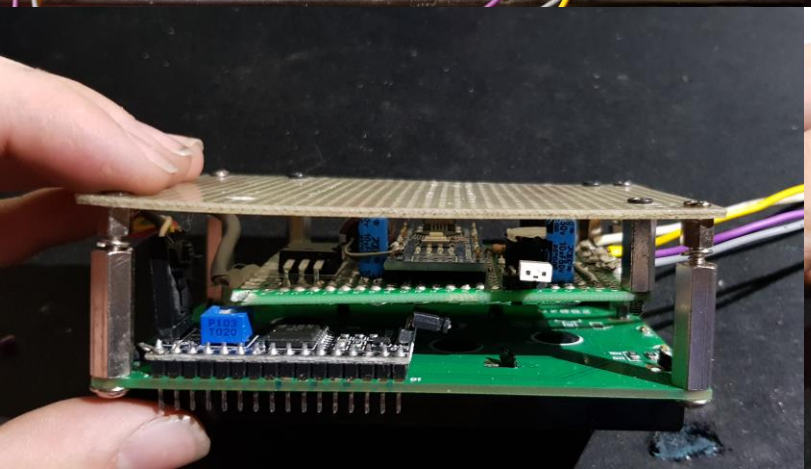


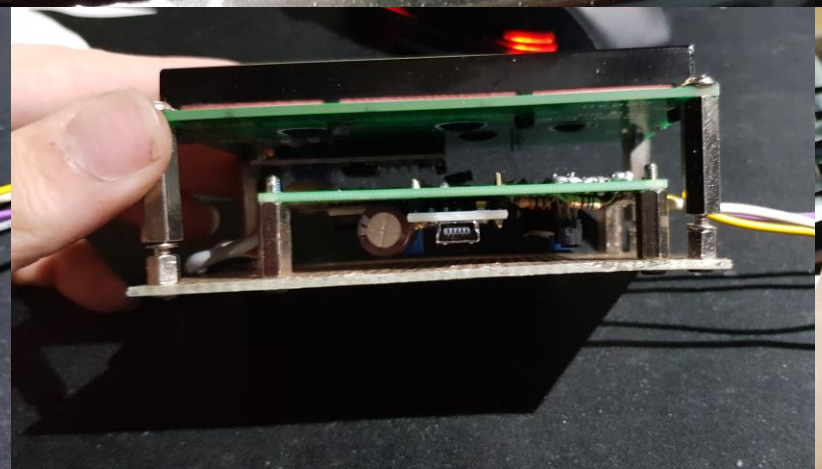
Sezione Hardware

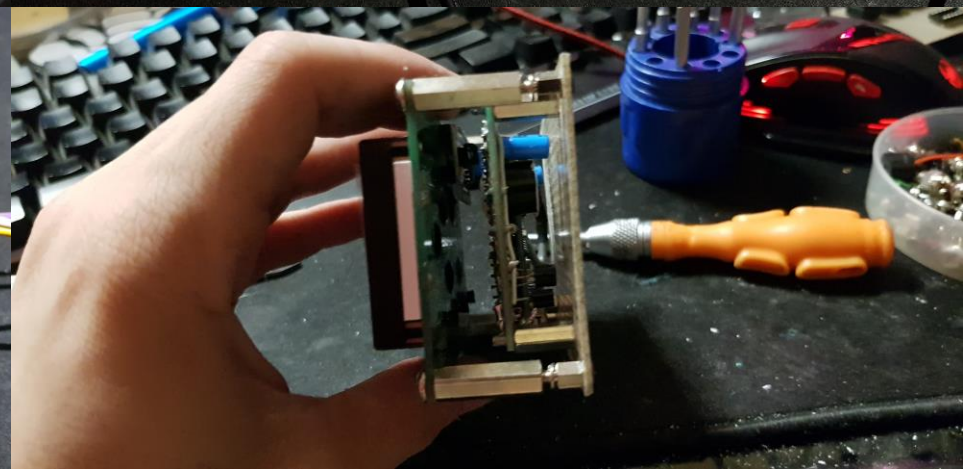
Terzo prototipo:











Sezione Hardware

Modalita' Terzo prototipo:

Sicuramente vi sarete chiesti cosa sono la modalita' standalone e la modalita' debug. Ecco la spiegazione:

E' possibile cambiare modalita' tramite l'apposito jumper che interrompe la 12v al 7805.

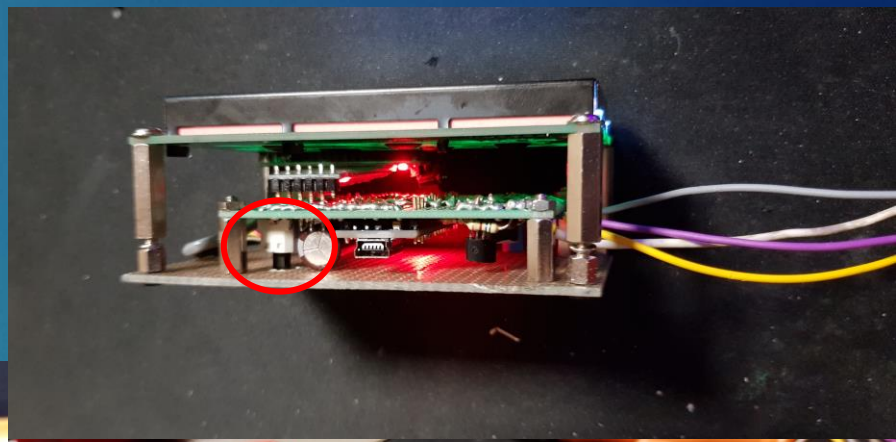
Modalita' standalone: strumento completo e funzionante da solo senza l'utilizzo del computer, in questa modalita' l'alimentazione viene fornita dalla batteria e stabilizzata per alimentare arduino. Tutto cio' che dovrete fare e' collegare il connettore di diagnostica e la 12v alla scheda.

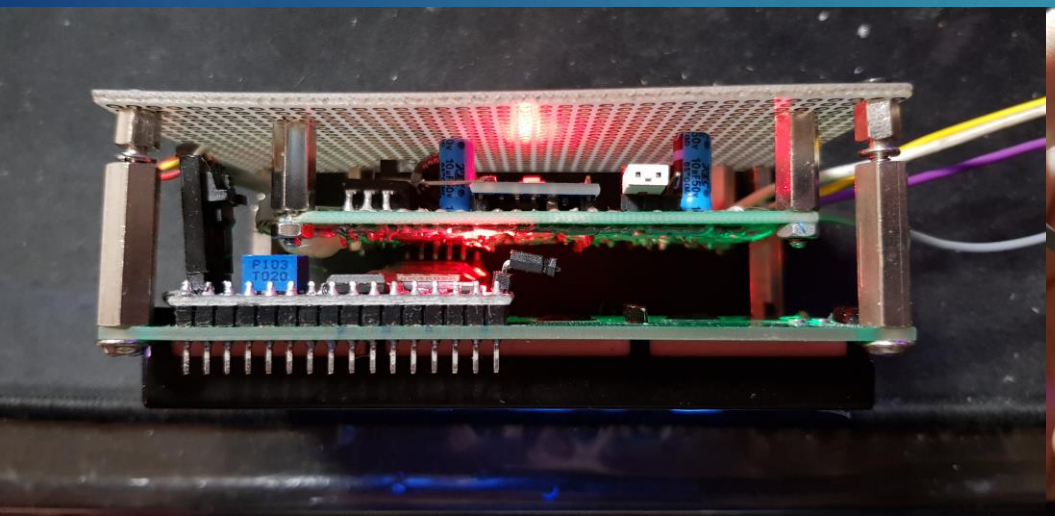
Modalita' debug: da usare solo se sapete cosa state facendo. In questa modalita' l'alimentazione viene fornita dall'usb, avrete quindi bisogno di utilizzare il computer con l'ide di arduino per avere il monitor seriale e leggere i valori direttamente sul computer. Questa modalita' e' da usare quando dovete modificare o aggiornare il programma di arduino e verificare i dati.

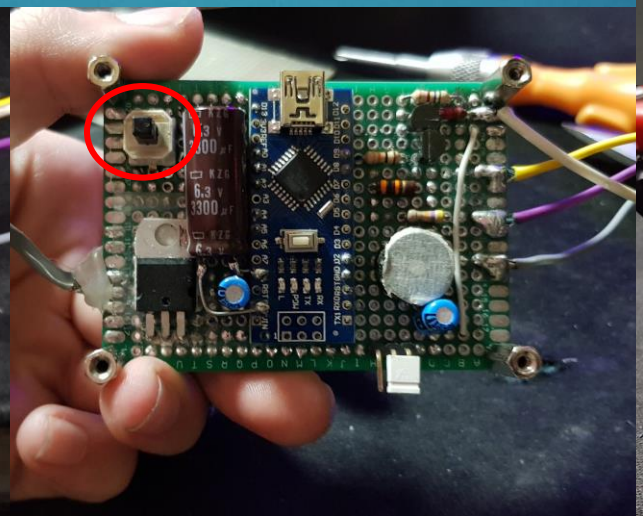
Sezione Hardware

Quarto prototipo:

In questa versione ho rimpiazzato il condensatore da 1000uf con uno da 3300uf per motivi di alimentazione. Ho aggiunto un pulsante per cambiare pagina del menu in modo da visualizzare piu' elementi possibili ed ho aggiunto un modulo bluetooth che permette allo strumento di collegarsi con uno smartphone ed inviare tutti i parametri direttamente su un applicazione dedicata.



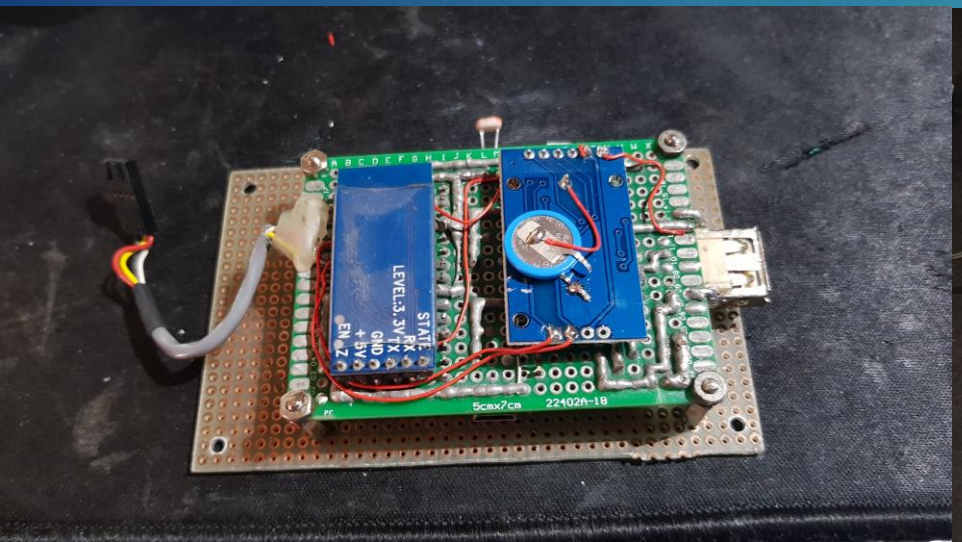


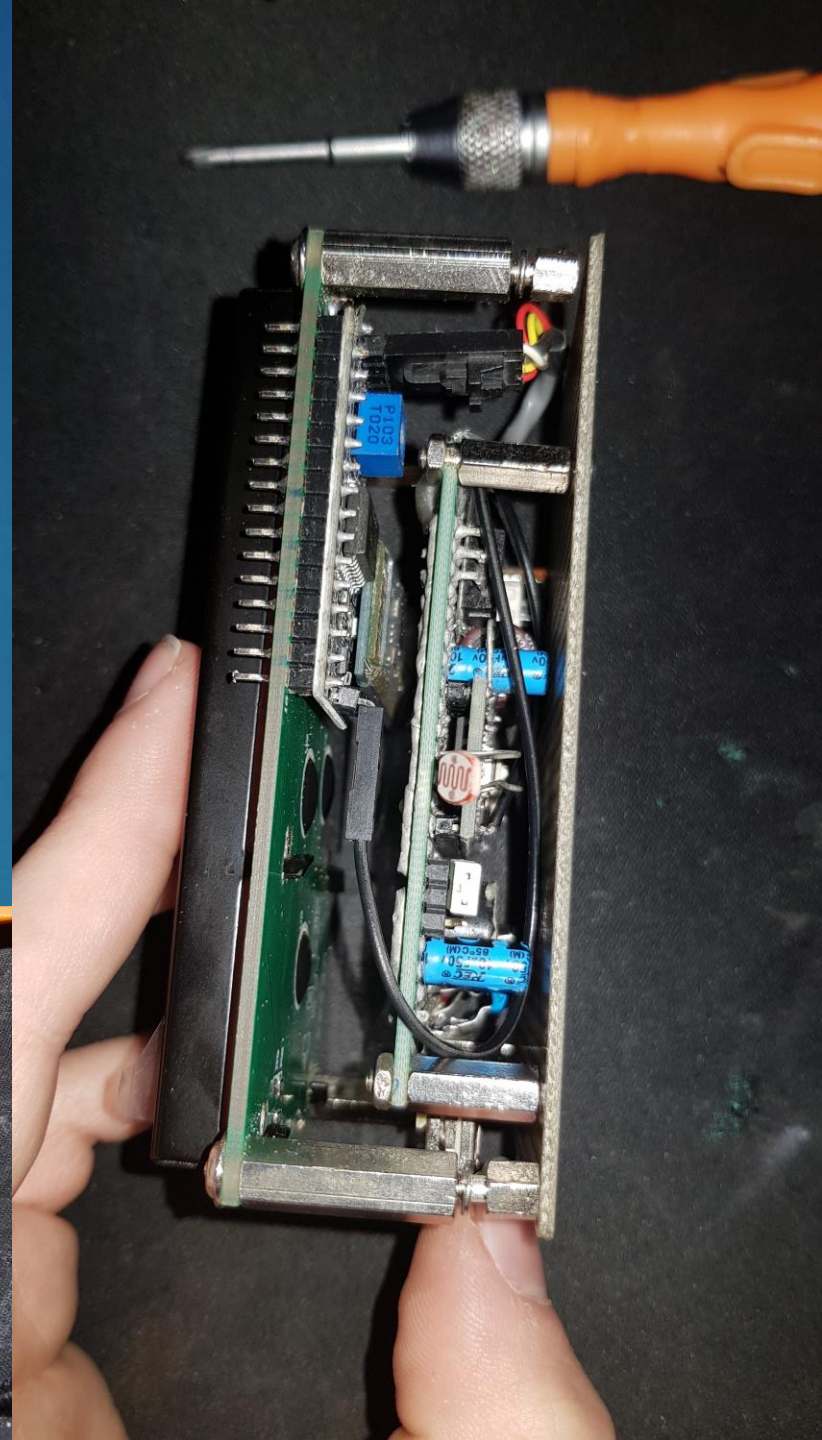


Sezione Hardware

Quinto prototipo:

In questa versione ho aggiunto un connettore USB e ricostruito il cavo della diagnostica, in questo modo posso staccarlo in qualunque momento e renderlo piu' portatile. Inoltre ho aggiunto un circuito per il controllo della retroilluminazione del pannello LCD mediante una fotoresistenza. Ho aggiunto un resistore di pull-up da 560ohm da 12v alla k-line per risolvere il problema del blocco del dispositivo. Ed infine ho aggiunto due condensatori ceramici da K-line a GND e da L-line a GND da 1nF per filtrare i disturbi.





Sezione Hardware

IAW16F:

Il cuore della centralina e' un microcontrollore ad 8 bit di Motorola, il 68HC11F1.

Features

- M68HC11 Central Processing Unit (CPU)
- Power Saving STOP and WAIT Modes
- 512 Bytes (EEPROM)
- 1024 Bytes RAM, Data Retained During Standby
- Nonmultiplexed Address and Data Buses
- Enhanced 16-Bit Timer
- Three Input Capture (IC) Channels
- Four Output Compare (OC) Channels
- One Additional Channel, Selectable as Fourth IC or Fifth OC
- 8-Bit Pulse Accumulator
- Real-Time Interrupt Circuit
- Computer Operating Properly (COP) Watchdog
- Enhanced Asynchronous Nonreturn to Zero (NRZ) Serial Communications Interface (SCI)
- Enhanced Synchronous Serial Peripheral Interface (SPI)
- Eight-Channel 8-Bit Analog-to-Digital (A/D) Converter
- Four Chip-Select Signal Outputs with Programmable Clock Stretching

— Two I/O Chip Selects

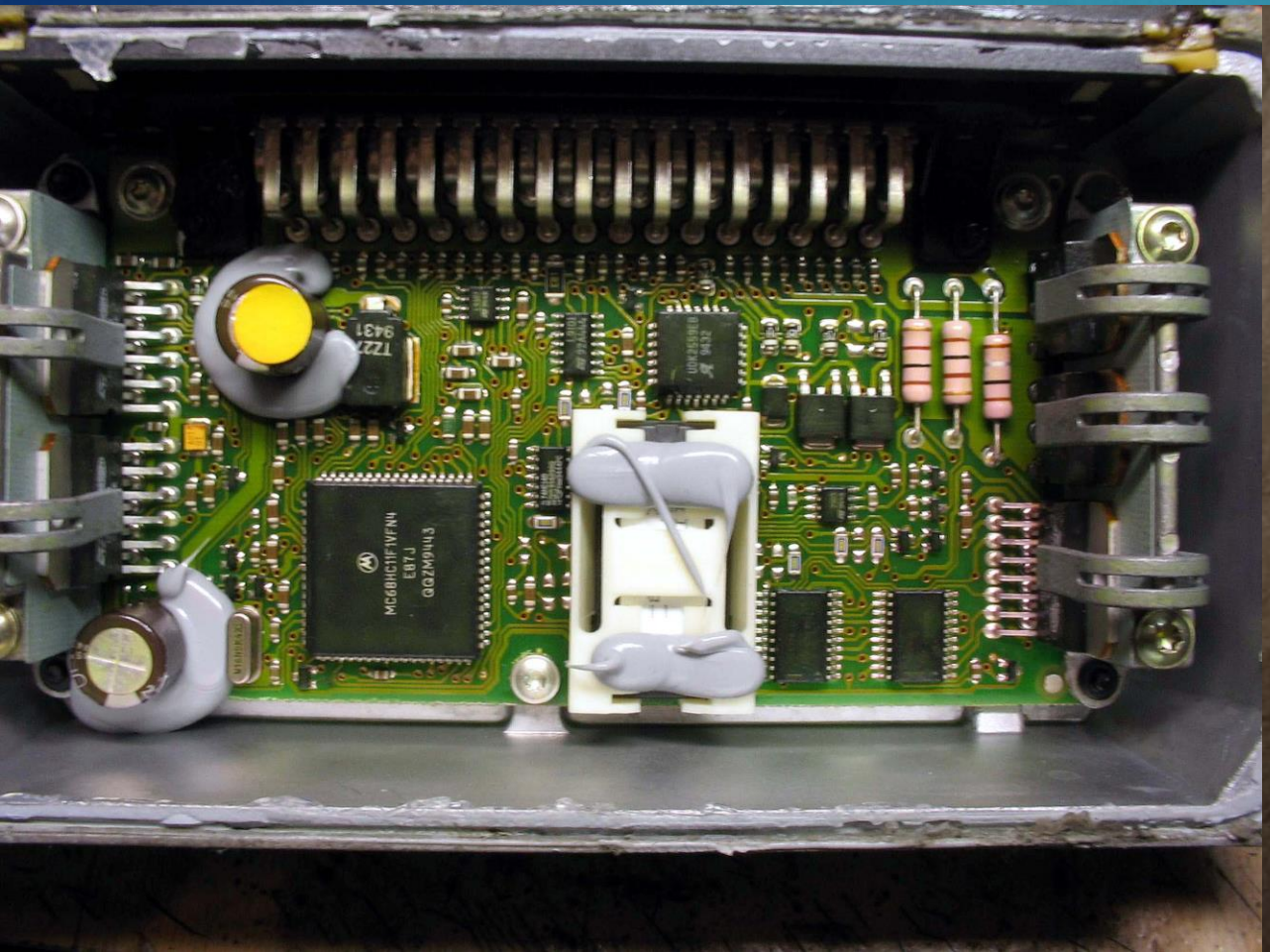
— One Program Chip Select

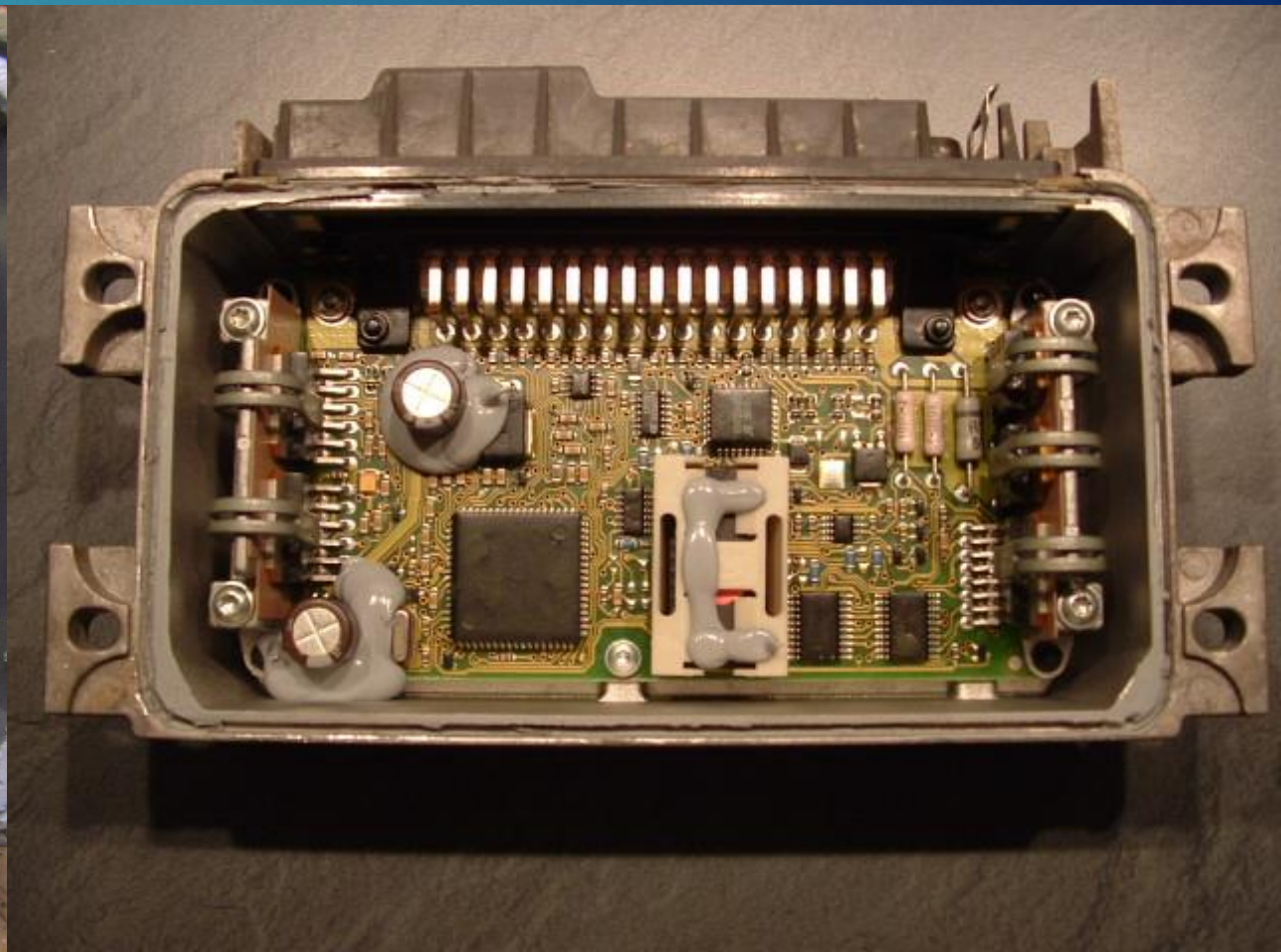
— One General-Purpose Chip Select

- Available in 68-Pin Plastic Leaded Chip Carrier (PLCC) and 80-Pin Plastic Quad Flat Pack (QFP)

Sezione Hardware

Foto interne della centralina:





Sezione Software

Iniziamo ora a parlare della parte software. Non mi mettero' a spiegare come programmare arduino da zero, quindi salto la parte relativa all'inizio del programma (variabili, definizioni, librerie, ecc.), se siete curiosi di leggere quella parte basta che aprite lo sketch e troverete i commenti di descrizione. La centralina in questione e' una IAW16F della Magneti Marelli.

Essendo la centralina di tipo FREE-RUNNING, ogni volta che giriamo la chiave ed accendiamo il quadro la centralina invia il codice ISO senza bisogno di inizializzarla (codice identificativo dell'automobile).

Perche' il codice ISO venga inviato il power latch deve essere terminato. Il power latch è il tempo che intercorre tra lo spegnimento del quadro vettura e la reale disalimentazione della centralina motore. In questo periodo di tempo la centralina salva tutto quello che ha in memoria ram sulla eeprom. Ci vogliono un paio di minuti per far terminare il power latch e far si che al prossimo avvio il codice ISO venga inviato di nuovo.

Il codice ISO viene inviato sulla K-Line a 1200 Baud ed e' composto da 6 bytes:

-Il Byte 55 Hex di autosincronismo

-4 Bytes di campo informativo (Keyword)

-1 Byte di controllo Checksum

Il Codice ISO dovrebbe essere 55-D0-85-8A-94-C8 (Panda 4x4 SPI 1100). In caso contrario la connessione non e' andata a buon fine. LASCIARE IL QUADRO ACCESO!

Sezione Software

Dopo aver ricevuto il codice ISO bisogna attendere almeno 500ms per poter inviare sulla L-Line 3 Bytes di valore 0F-AA-CC Hex con Baud Rate 1200 per richiedere la connessione a 7812 baud. Dopo aver ricevuto questa richiesta, la centralina entra in modalita' comunicazione a 7812,5 Baud. Una volta instaurata la comunicazione a 7812 Baud sara' possibile inviare sull L-Line le richieste per i parametri del motore le cui risposte verranno inviate dalla centralina sulla K-Line. IL TUTTO DEVE AVVENIRE CON IL QUADRO ACCESO!

Non e' obbligatorio leggere il codice ISO, e' solo uno step per essere sicuri che il tutto sta funzionando correttamente ma puo' essere ignorato. L'importante e' inviare i 3 bytes 0F-AA-CC. Quindi sul programma di Arduino:

```
//############## INIZIALIZZAZIONE COMUNICAZIONE 7812 BAUD ################

  delay(500); //Aspettare almeno 500 millisecondi dalla ricezione del codice ISO prima di inviare la richiesta di comunicazione a 7812 Baud
  softSerial.write(0x0F);  //Primo byte di inizializzazione
  delay(110);              //Delay 110ms +- 10ms
  softSerial.write(0xAA);  //Secondo byte di inizializzazione
  delay(110);              //Delay 110ms +- 10ms
  softSerial.write(0xCC);  //Terzo byte di inizializzazione
  delay(150);              //Delay 110ms +- 10ms

  softSerial.begin(7812);  //Inizializza seriale software a 7812 baud per lettura parametri dalla ECU
```

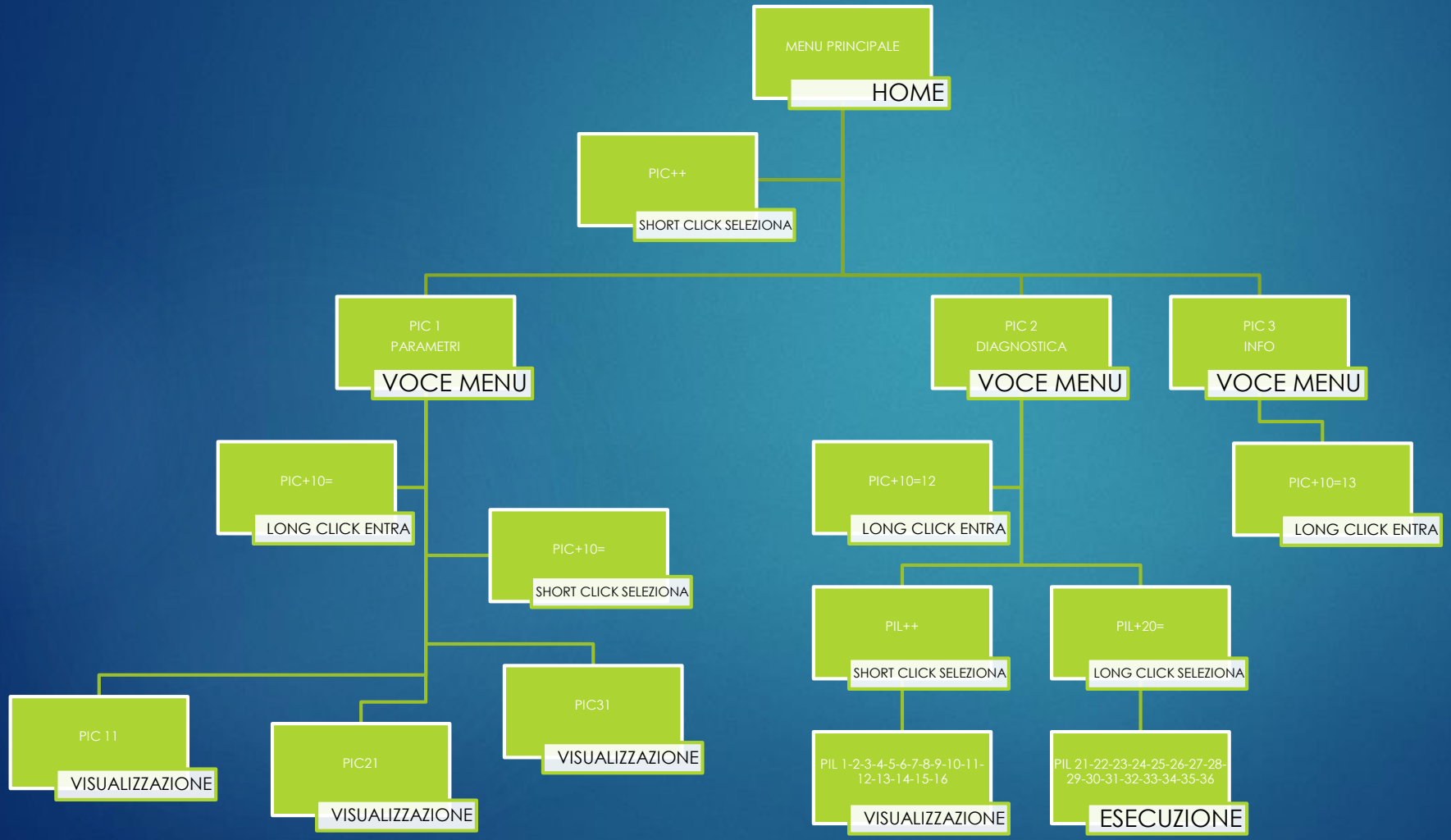
Sezione Software

Una volta che la centralina entra in modalita' di comunicazione a 7812 Baud possiamo iniziare ad inviare a piacere dei codici (Domande) alla quale la centralina rispondera' con dei valori grezzi. Per esempio, vogliamo leggere la temperatura dell'acqua, per conoscere il codice della richiesta bisogna far riferimento al manuale della centralina. Una volta che conosciamo il codice di richiesta del parametro non ci resta che inviarlo con Arduino e prepararci a riceverlo e visualizzarlo sul display:

```
//#########################  TEMPERATURA ACQUA  ##################################

  softSerial.write(0x08);  //Richiesta parametro temperatura acqua
  delay(110);  //Ritardo di 110ms
  Serial.print("Temperatura Acqua: ");  //Scrivi sul Display
  TW=softSerial.read()-40;     //Il valore della temperatura (non grezzo) secondo il manuale e' uguale al valore letto - 40.
  Serial.print(TW, DEC);  //Visualizza sul monitor seriale il dato acquisito
  Serial.println(" C");
  delay(1000);
```

Sezione Software



Descrizione Menu:

Dal menu principale abbiamo 3 opzioni evidenziabili con uno short click.

Con un long click possiamo entrare all'interno dell'opzione evidenziata.

Nel menu parametri si possono sfogliare con uno short click fino a 3 pagine di dati da visualizzare in modo ciclico.

Nel menu diagnostica si possono sfogliare 2 pagine di test diagnostici con uno short click e selezionarli con un long click per poi essere eseguiti.

Nel menu info una volta entrari con un long click si possono visualizzare le informazioni del progetto.

1 CARATTERISTICHE DIAGNOSTICHE DEL SISTEMA
1.1 TABELLA DIAGNOSTICA

TABELLA RIASSUNTIVA

SOTTOGRUPPO COMPONENTE IL SISTEMA	PRESTAZIONI DI ASSISTIBILITA' OTTENIBILI COLLEGANDO IL FIAT LANCIA TESTER ALLA PRESA DI DIAGNOSI, A CENTRALINA ALIMENTATA E FUNZIONANTE		PO	CK	ER	VR	NOTE
CENTRALINA	**Identificazione del sistema** :						**Protocollo di comunicazione con strumento di diagnosi** :
	– Lettura CODICE ISO	[6 Bytes / Hex]	*	*	*	*	I dati vengono codificati e trasmessi in **NRZ** con logica positiva alla velocità di **7812,5** Baud. La struttura dei Bytes è la seguente: 1 Start Bit ("0"), 8 Bit di Dato, 1 Stop Bit ("1"), no parity.
	– Lettura CODICE RICAMBIO	[11 Bytes/ACII]	*	*	*	*	Al Power On, la centralina trasmette i 6 byte contenenti il CODICE ISO a 1200 baud. Dopo almeno 500 msec dall'arrivo del codice ISO, lo strumento di diagnosi deve trasmettere verso la centralina un proprio codice di riconoscimento composto da tre Bytes esadecimali **0Fh, AAh, CCh**, trasmessi alla velocità di **1200** Baud con un tempo di interbytes (tempo tra un Byte ed il successivo) di 110 msec +/–10 msec.
	Visualizzazione di :	n° Bytes/valore	*	*	*	*	
	– Giri motore	[2 Bytes/ Rpm]	*	*	*	*	
	– Tempo iniezione	[2 Bytes/ msec.]	*	*	*	*	
	– Anticipo accensione	[1 Byte /°Ang.]	*	*	*	*	Se questa procedura ha esito positivo, la centralina si predispone per la comunicazione seriale con strumento di diagnosi alla velocità di 7812,5 Baud e modalità secondo specifica (vedi NORMA di PRODUZIONE).
	– Pressione aspirata	[1 Byte/mmHg]	*	*	*	*	
	– Temperatura aria	[1 Byte / ° C]	*	*	*	*	
	– Temperatura acqua	[1 Byte / ° C]	*	*	*	*	La comunicazione è configurata come ISO_4 : linea "**L**" utilizzata per inizializzare e trasferire Bytes da strumento di diagnosi verso centralina, e linea "**K**" utilizzata per trasferire Bytes da centralina verso strumento di diagnosi.
	– Posizione farfalla	[1 Byte/ °Ang.]	*	*	*	*	
	– Tensione Batteria	[1 Byte / Volt]	*	*	*	*	
	– Correzione Sonda Lambda	[1 Byte / %]	*	*	*	*	**Riconoscimento degli errori** :
	– Posizione stepper	[1 Byte / Passi]	*	*	*	*	L'errore viene rilevato tramite una macchina di validazione che funziona nel seguente modo:
	– Correzione integrale minimo	[1 Byte / Passi]	*	*	*	*	Quando un errore viene rilevato, questo viene controllato per un certo tempo per evitare i possibili
	– Correzione proporzionale minimo	[1 Byte / Passi]	*	*	*	*	rumori sulla linea diagnosticata; se passa questo stadio viene considerato presente e viene memo-
	– Trimmer Titolo	[1 Byte]	*	*	*	*	rizzato in ERR–CO–xx (**errore filtrato**), dopo di che si passa alla fase successiva in cui si controlla
	– ERR–CO–XX (errori filtrati)	[3 Byte / Hex]	*	*	*	*	che l'errore sia sempre presente per un altro tempo (detto di validazione). Se viene superata questa
	– FGSTAT (byte di stato)	[1 Byte / Hex]	*	*	*	*	fase l'errore viene memorizzato in ERR–VA–xx (**errore validato**) e, se è previsto, viene accesa la
	– ERR––VAXX (errori validati)	[3 Bytes / Hex]	*	*	*	*	lampada di avaria. La fase di scrittura dell'errore in EEPROM avviene, a differenza degli altri sistemi,
	– Codice Ricambio	[11Bytes/ASCII]	*	*	*	*	immediatamente (sul task di 4 msec.).
	– Correzione Autoadattatività stepper	[2 Bytes / Passi]	*	*	*	*	**Memorizzazione degli errori** :
	– Correzione Autoad. stepper con Condizionatore	[2 Bytes / Passi]	*	*	*	*	A partire dallo stato di OK, in caso di rilevamento errore su di una linea, viene attivata una procedura
	– Obiettivo giri minimo	[1 Byte / Rpm]	*	*	*	*	di **filtro** che consiste nel campionare la linea per un tempo T1. Se al termine di T1 il numero di cam-
	– Offset giri al minimo	[1 Byte / Rpm]	*	*	*	*	pioni errati supera un valore prefissato Q1, viene superato lo stadio di filtro e attivata la scrittura
	– Delta regolatore minimo	[1 Byte / Passi]	*	*	*	*	dell'errore in memoria volatile RAM (**ERR–CO–xx**) ; se invece lo stadio di filtro non viene superato,
	– Correzione Passi stepper al minimo da FLT	[1 Byte / Passi]	*	*	*	*	si torna nello stato OK. In questa condizione la spia di avaria non viene ancora accesa, perchè l'erro-
	– FGSTAT2 (Byte di stato)	[1 Byte / Hex]	*	*	*	*	re non è ancora validato.
	– SE–CO–XX, SE–VA–XX (segno degli errori)	[6 Bytes / Hex]	*	*	*	*	Dopo il superamento del filtro, viene attivata una procedura di **validazione**, che consiste nel campio-
	– Offset Autoad. Titolo fuori minimo/Canister Off	[2 Bytes / msec]	*	*	*	*	nare la linea per un tempo T2. Se il numero di campioni errati supera un valore prefissato Q2, viene
	– Offset Autoad. Titolo fuori minimo/Canister On	[2 Bytes / msec]	*	*	*	*	superato lo stadio di validazione dell'errore e viene scritto in memoria E^2PROM (**ERR–VA–xx**) con
	– Offset Autoadattività Titolo al minimo	[2 Bytes / msec]	*	*	*	*	accensione della spia di avaria, se è previsto.
	– Guadagno Autoadattatività Titolo a medio carico	[2 Bytes / %]	*	*	*	*	Se la procedura di validazione dell'errore non viene superata, si cancella il contenuto della RAM e si
	– Errori / Stato Immobilizer UNIVAS, EEVAS	[2 Bytes / Hex]	*	*	*	*	torna in condizione OK (validazione di bontà). Ad ogni variabile rappresentativa di errori confermati o
	– Contatore età errore Immobilizer CRDVAS	[1 Byte / Hex]	*	*	*	*	validati corrisponderà una variabile con il senso degli errori : cicuito aperto, corto circuito a Gnd o +
							Vbatt. Se è presente un errore confermato di un certo senso e la centralina rileva errore del senso
	Segnalazione di :						opposto, allora il processo di validazione dell'errore va avanti ma il senso dell'errore viene invertito.
	– Errore RAM (non superamento test) – WL=On		*				Se è invece presente un errore validato e la centralina rileva un errore di senso opposto, il senso
	– Errore EPROM (checksum errato della memoria ROM) – WL=On		*	*	*	*	dell'errore validato non viene invertito. La memorizzazione avviene solo se l'errore non era presente
	– Errore EEPROM (checksum errato memoria EEPROM) – WL=On		*				in precedenza. Quindi in caso di errori di tipo diverso sulla stessa linea, la memoria di segno errori
	– Errore Microprocessore (funzionalità del Micro) – WL=On		*	*	*	*	conterrà il segno del primo errore validato.
	– **N.B. : WL = Lampada di Avaria posta sul cruscotto**						I valori di T1, Q1, T2, Q2 sono valori in calibrazione specifici per ogni linea.
							Alcuni errori particolarmente "gravi" vengono immediatamente validati alla prima rilevazione e me-
	Recovery :						morizzati in memoria permanente; per questo tipo di errori anche l'accensione della spia di avaria è
	– non è prevista la limitazione delle prestazioni						immediata.

LEGENDA: PO= CHIAVE ON – CK= IN AVVIAMENTO – ER= MOTORE IN MOTO – VR= VEICOLO IN MOVIMENTO

SEGUE **TABELLA DIAGNOSTICA**

TABELLA RIASSUNTIVA

SOTTOGRUPPO COMPONENTE IL SISTEMA	PRESTAZIONI DI ASSISTIBILITA' OTTENIBILI COLLEGANDO IL FIAT LANCIA TESTER ALLA PRESA DI DIAGNOSI, A CENTRALINA ALIMENTATA E FUNZIONANTE		PO	CK	ER	VR
CENTRALINA (continua)	Parametri :	Formule di conversione :				
	– Giri motore	(15 x 10^6) / $01$02	*	*	*	*
	– Tempo iniezione	(2 x $03$04) / 10^3	*	*	*	*
	– Anticipo accensione	$05 / 2	*	*	*	*
	– Pressione aspirata	$06 x 3	*	*	*	*
	Temperatura aria	$07 – 40	*	*	*	*
	– Temperatura acqua	$08 – 40	*	*	*	*
	– Posizione farfalla	$09 x 4,234 – 2,9638	*	*	*	*
	– Tensione Batteria	$0A x 0,0625	*	*	*	*
	– Correzione Lambda	$0B x 0,002656 + 0,66	*	*	*	*
	– Posizione stepper	$0C	*	*	*	*
	– Correzione integr. minimo	$0D / 2	*	*	*	*
	– Correzione prop. minimo	$0E / 2	*	*	*	*
	– Trimmer Titolo	80H= 00, FFH= +127, 00H= –127	*	*	*	*
	– ERR–CO–XX	$10 (INP), $11 (OUT), $12 (FUNZ)	*	*	*	*
	– FGSTAT (byte di stato)	$13	*	*	*	*
	– ERR–VA–XX (errori)	$14 (INP), $15 (OUT), $16 (FUNZ)	*	*	*	*
	– Codice Ricambio	$17..$21	*	*	*	*
	– Correzione AA stepper	$22$23	*	*	*	*
	– Correz. AA stepper Cond.	$24$25	*	*	*	*
	– Obiettivo giri minimo	$26 x 8	*	*	*	*
	– Offset giri al minimo	$27 x 8	*	*	*	*
	– Delta regolatore minimo	$28 – 128	*	*	*	*
	– Correzione Passi da FLT	$29 – 128	*	*	*	*
	– FGSTAT2 (Byte di stato)	$2A	*	*	*	*
	– SE–CO–XX, SE–VA–XX	$2B..$30	*	*	*	*
	– Offset AA Tit Canister Off	$32$33	*	*	*	*
	– Offset AA Tit Canister On	$34$35	*	*	*	*
	– Offset AA Tit al minimo	$36$37	*	*	*	*
	– Guad. AA Tit/medio carico	$38$39	*	*	*	*
	– Errori / Stato Immobilizer	$71(UNIVAS), $72(EEVAS)	*	*	*	*
	– Cont. età errore CRDVAS	$73 (valore max = FF Hex)	*	*	*	*
	Memoria Errori relativi ai componenti del sistema :					*
	– Sensore Potenziometro Farfalla		*	*	*	*
	– Sensore di Pressione		*	*	*	*
	– Sonda Lambda				*	*
	– Sensore Temperatura Acqua		*	*	*	*
	– Sensore Temperatura Aria		*	*	*	*
	– Tensione Batteria		*	*	*	*
	– Sensore Giri Motore			*		*
	– Comando Mono–Iniettore			*	*	*
	– Comando Bobina 1			*	*	*
	– Comando Bobina 2			*	*	*
	– Comando Canister				*	
	– Comando Condizionatore	[solo con condiz. presente]			*	
	– Comando Pompa Carburante		*	*	*	

LEGENDA: PO= CHIAVE ON – CK= IN AVVIAMENTO – ER= MOTORE IN MOTO – VR= VEICOLO IN MOVIMENTO

NOTE

i Bytes che contengono i dati della memoria errori sono denominati :

ERR–CO–xx = ERRORI FILTRATI (copia in RAM)
ERR–VA–xx = ERRORI VALIDATI (copia da Bytes ERR–CO–xx)
SE–CO–xx = Segno degli ERRORI FILTRATI (copia in RAM)
SE–VA–xx = Segno degli ERRORI VALIDATI (copia da Bytes SE–CO–xx)
FGSTAT/FGSTAT2 = STATO SISTEMA

Oltre ai Bytes sopra descritti, esistono 2 Bytes specifici per la Chiave Elettronica (UNIVAS, EEVAS) che identificano lo Stato ed Errori della stessa secondo lo schema di seguito descritto :

UNIVAS = Stato della Chiave / Errori filtrati in RAM
EEVAS = Errori validati in EEPROM

Cancellazione errori :
La cancellazione errori può avvenire tramite 2 modalità :
– Tramite stumento di diagnosi (FLT) con comando specifico da diagnosi attiva ($84 Hex)
– Tramite incremento contatore Trip in calibrazione fino al valore N–AVV (in calibrazione), valore per il quale al Power–Latch successivo vengono cancellati dalla RAM e dall'EEPROM tutti gli errori validati.
Quanto descritto sopra differisce unicamente per gli errori della Chiave Elettronica il cui contatore CRDVAS viene settato ad FF Hex con errore validato e decrementato di 1 ad ogni Key–On in assenza di errore fino al valore 0 per il quale si resettano gli errori in EEVAS.

Gestione della Warning Lamp (spia di avaria) :
Per verificarne la funzionalità, la spia viene accesa ad ogni Power–On, e resta illuminata per 4 secondi. La spia poi viene accesa ogni qual volta è presente un errore validato, cioè fino a quando permane la situazione di guasto, e viene spenta alla scomparsa dello stesso.
Di eventuali guasti avvenuti in precedenza viene mantenuta una "storia" nella memoria EEPROM

Accensione della Warning Lamp (spia di avaria) :
È possibile definire per ogni linea se attivare o meno la lampada di segnalazione a cruscotto e/o limitare le prestazioni del motore (limitazione regime)
La lampada spia si **attiverà**, per gli errori per cui è prevista l'attivazione, ogni qualvolta l'errore è presente in ERR–CO–xx e ERR–VA–xx (errore filtrato e validato) e si disattiverà nel momento in cui l'errore non è più presente (ERR–CO–xx) secondo lo schema di seguito descritto :

WL = On ⇒ ERR–CO–xx = **On** ERR–VA–xx = **On**
WL = Off ⇒ tutte le altre condizioni

Informazioni relative allo Stato del Sistema e del Motore :
Esistono 2 Bytes che contengono informazioni relative allo stato del sistema e del motore (**FGSTAT, FGSTAT2**)

FGSTAT :
Bit_0 = 1 Ok Diagnosi attiva stepper in Run e TH2O Ok
Bit_1 = 1 Motore in moto
Bit_2 = 1 Quadro Segnali Ok
Bit_3 = 1 Farfalla in minima o piena apertura
Bit_4 = 1 Titolo in Closed Loop
Bit_5 = 1 Richiesta attivazione Condizionatore
Bit_6 = 1 Autoadattatività Titolo abilitata
Bit_7 = 1 Test Stepper attivo in esecuzione

FGSTAT2 :
Bit_0 = Free (libero)
Bit_1 = Free (libero)
Bit_2 = 1 Autoadattatività Stepper abilitata
Bit_3 = Free (libero)
Bit_4 = Free (libero)
Bit_5 = Free (libero)
Bit_6 = 1 Guasto ca su linea Power–Latch
Bit_7 = Free (libero)

F.A.Q.

-Dove trovo tutti i componenti? Su Amazon dovrebbe esserci tutto il necessario per realizzare il progetto.

-Non sono esperto di elettronica? Ho provato a semplificare il piu' possibile il circuito per essere alla portata di tutti, lo stesso vale per il programma di Arduino, se non avete mai usato Arduino, e' il momento giusto per farlo, scoprirete un nuovo mondo e vi accorgerete di quanto sia utile e semplice da usare. Se proprio non riuscite a capire il tutto, fatevi aiutare da un amico piu' esperto.

-La centralina si puo' danneggiare? Ni, se non sapete cosa state facendo rischiate di danneggiarla, l'importante e' evitare che i cavi si tocchino tra di loro evitando un eventuale corto circuito, non c'e' il pericolo di inviare richieste sbagliate o scrivere la centralina poiche' le linee utilizzate (L e K) sono fatte apposta per la diagnostica, nel caso in cui viene inviata una richiesta sbagliata la centralina ignorera' la domanda.

-Con quale auto e' compatibile il circuito? Sono illustrate nella pagina seguente.

-L'applicazione android si vede male/non entra nello schermo: L'applicazione e' stata creata e testata su un Galaxy s8, non l'ho provata su altri dispositivi e quindi gli elementi sono stati disposti in base alla risoluzione del display dell's8. Se proprio e' inutilizzabile scrivetemi il modello del vostro smartphone e provvedero' a sistemarla.

-E' possibile acquistarlo gia' costruito? Purtroppo sono uno sviluppatore indipendente, cio' significa che dovrei realizzarli tutti a mano uno per uno. Quindi, per motivi di tempistiche e materiali non vendero' il progetto gia' realizzato.

Compatibilita'

IAW16F:

{"55D085859443", "Cinquecento 899 SPI ECE F2"},
{"55D0858694C4", "Cinquecento 1108 SPI ECE F2"},
{"55D0850194BF", "Punto 1.1 SPI Em.04 Est Europa"},
{"55D085029440", "Punto 55 1.1 SPI 5M/6M ECE F2"},
{"55D0850494C2", "Punto 60 1.2 SPI CM ECE F2 T.i.T."},
{"55D085079445", "Punto Selecta 1.2 SPI ECE F2"},
{"554C851092C8", "Panda 1000 SPI ECOL"},
{"55D085089446", "Panda 1000 SPI ECE F2"},
{"55D0850B9449", "Panda 1108 SPI CA ECE F2"},
{"55D0850E15CD", "Tipo/Tempra 1.6 SPI USA'83 (TOFAS)"},
{"55D08510154F", "Lancia Y 1.2 SPI CA ECE F2"},
{"55D0858994C7", "Panda 899 SPI ECE F2"},
{"55D0858A94C8", "Panda 1108 4x4/4x4 ECE F2"},
{"55D0858C15CB", "Tipo/Tempra 1.6 SPI Em.04 (TOFAS)"},
{"55D0858F15CE", "Lancia Y 1.2 SPI CM ECE F2"},
{"55D085911651", "Tipo 1372 SPI ECE 04 (TOFAS)"},
{"55D085921652", "131 Bn/Sw 1.6 SPI USA'83 (TOFAS)"},
{"55D085139754", "Seicento 0.9 CM SPI F2"}

IAW04:

{"55BC83019429", "Alfa 155 2.0 16V 4x4"},
{"55CE850894C4", "Dedra 2.0 16V"},
{"55CE85079443", "Dedra 2.0 16V 4x4"},
{"553883029426", "Tempra 2.0 8V 4x4"},
{"553883019425", "Tempra 2.0 8V M/T"},
{"5538830194A8", "Tempra 2.0 8V A/T"},
{"55CB8202943B", "Dedra 2.0 8V A/T"},
{"55CE8504943B", "Nuova Delta 2.0 T/C"},
{"55CE85049440", "Nuova Delta 2.0"},
{"55CB850194BA", "Coupe S 2.0 T/C"},
{"55CE858394BF", "Coupe S 2.0"},
{"55CD852613E0", "Delta Evoluzione 2000 16V 4x4"},
{"55CB859B13D3", "Coupe ESSE 2.0 16V T/C"},
{"55CB8592134A", "Nuova Delta 2.0 16V T/C 4x2"},
{"55CE8501943D", "Nuova Delta 2.0 16V"},
{"55D00292914A", "TIPO 2000 16V"},
{"55CD850813C2", "Delta Evoluzione 2.0 16V ECO"}

IAW18F:

{"55318002941C", "Punto 75 1.2 Fire 8V ECE F2"},
{"5531808A1323", "Punto 75 1.2 Fire 8V ECE ECOL"},
{"5531800E97AB", "Palio 1.2 Fire 8V ECE F2"},
{"5531800794A1", "Delta/Dedra Bn/Sw 1.8 ECE F2"},
{"55318083949D", "Alfa 145/146 1.3 Boxer ECE F2"},
{"55318085941F", "Delta 1.8 90 CV ECE F2"},
{"553180869420", "Tipo/Tempra Bn/Sw 1.8 ECE F2"}

IAW8F:

{"55B683079126", "ALFA 33 1360 MPI CM"}};

IAW18FD:

{"5531800D1629", "Punto 1242 FIRE 16V CEE F2"},
{"5531808C16A8", "Siena 1.4 8V (IAW-1G7SP)"},
{"553180081523", "Palio 1.0 8V (IAW-1G7SD)"}

Lo strumento funziona sicuramente con le auto che montano la IAW16F. Per quanto riguarda le altre centraline, vanno ancora testate ma probabilmente funziona in quanto la modalita' di inizializzazione e le linee dati utilizzate dovrebbero essere le stesse.

Problemi Noti

Come tutti (o quasi) i progetti fai da te, ci sono sempre dei problemi che devono ancora essere risolti.
Qui di seguito illustro una lista di problemi noti che ho riscontrato durante l'utilizzo:

-Problema casuale durante la marcia, lo strumento potrebbe andare in tilt e visualizzare valori irreali.
Riavviate lo strumento. (Risolto)

Se riscontrate altri tipi di problemi non esitate a contattarmi e provvedero' se possibile a risolverli.