ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Dipartimento di Ingegneria dell'Energia Elettrica e dell'Informazione "Guglielmo Marconi"

CORSO DI LAUREA IN INGEGNERIA ELETTRONICA PER L'ENERGIA E L'INFORMAZIONE

SVILUPPO DI UN SISTEMA EMBEDDED PER APPLICAZIONI DI DIAGNOSTICA AUTOMOTIVE

Elaborato in Elettronica dei Sistemi Digitali

Relatore:
Aldo Romani

Presentato da:
Lorenzo Porcheddu

Correlatore: Marcello Foschi

Anno Accademico 2020-2021

Indice

Introduzione

L'industria automobilistica (automotive) si occupa della progettazione, della costruzione, del marketing e della vendita di veicoli a motore. Nel 2019 la produzione mondiale di veicoli a motore, comprendendo automobili e veicoli commerciali, ha superato gli 80 milioni di unità. In Emilia Romagna, la regione definita anche “Motor Valley», con il 10% delle imprese è concentrata sulla produzione di veicoli di alta gamma e sul settore del Motorsport: l'industria automotive rappresenta il del PIL regionale ed annovera i brands più prestigiosi a livelo nazionale ed internazionale, quali Ferrari, Lamborghini, Ducati etc. Negli ultimi 30 anni l’elettronica è diventata sempre più presente nel mondo dell’automotive, iniziando con i primi sistemi di accensione ed iniezione elettronica del carburante, sistemi di sicurezza ABS fino ad arrivare al giorno d’oggi, dove l’auto ibrida inizia ad essere fortemente presente e si evidenzia una forte spinta per convertire il parco auto in sistemi completamente elettrici. La conseguenza di questo sviluppo è che sempre più sensori, attuatori e schede elettroniche sono integrati nelle nostre auto, rendendo più complicato il lavoro degli autoriparatori, i quali si trovano di fronte ad un mondo piuttosto complesso rispetto a quello di una volta. Sono quindi richieste nuove competenze elettroniche e nuovi modi di risolvere problemi su sistemi elettrici ed elettronici, sempre più diffusi.

Da quanto detto, si evidenzia la necessità di dotare il personale qualificato di strumenti adeguati alla crescita tecnologica dei veicoli, per essere più efficaci nel loro intervento.

In collaborazione con una ditta specializzata nel settore della diagnostica Automotive, ACTILAB di Foschi Marcello, è stato quindi sviluppato un progetto sulla diagnostica di officina, che permette di semplificare il lavoro di manutenzione, collaudo e riparazione elettronica, cioè è stato progettato e realizzato una sorta di “testbench” in grado di simulare e collaudare quanti più sensori e attuatori possibili presenti nei veicoli. Nello specifico, possono essere testati e pilotati sensori di giri, di fase, temperatura, pressione, ecc. Nel caso degli attuatori, si riescono a testare iniettori, riscaldatori, aria condizionata, elettroventole, ecc.

Il progetto si sviluppa fondamentalmente con componentistica elettronica su un PCB basato su piattaforma ESP32 di Espressif, che, tramite apposita circuiteria di contorno analogica e digitale, di segnale e di potenza, riesce a interfacciarsi con i sensori e gli attuatori presenti sui veicoli.

Capitolo 1: L’Hardware

I requisiti fondamentali alla base del progetto sono:

- Compattezza
- Funzionalità ed efficienza
- Economicità
- Facilità d’uso
- Robustezza elevata

Abbiamo quindi optato per i seguenti componenti elettronici:

- µC ESP32, Dual-Core, 240 MHz, 32-bit
- Display 2.8” Touch-Screen TFT SPI con controller ILI9341
- Power-MOSFET AUIRFS8403
- Signal-MOSFET AO3404
- Mosfet Driver MCP1416T
- Regolatore DC-DC TPS562211DRLR
- LDO NCP1117ST33T3G
- Diodo Shottky B540C
- Diodo TVS SMAJ14A

La scelta dell'unità logica di controllo è ricaduta su l‘ESP32, un particolare SoC prodotto dalla Espressif Systems (Shanghai), evoluzione del “vecchio” ESP8266. Bisogna tenere presente che questo mercato di micro sistemi è in così rapida crescita ed evoluzione che risulta essere problematica, per una azienda che si voglia cimentare nello sviluppo di questa tipologia di prodotti, la scelta di una piattaforma che sia abbastanza stabile nel tempo da garantire un ritorno degli investimenti.

Che cosa è un SoC: acronimo di System on a Chip, è un sistema su circuito integrato che in solo chip integra, oltre al processore centrale CPU, un chipset ed altri controllers come quello per la memoria RAM, la circuiteria input/output finanche il sotto sistema video. Questo singolo chip può contenere inoltre componenti digitali, analogici e circuiteria in radiofrequenza. I SoC, viste le dimensioni ridotte che si possono raggiungere con l'integrazione spinta di tutti i componenti, vengono utilizzati comunemente nelle applicazioni embedded, quali SmartPhones, IoT ed Industry 4.0

1.1 Il microcontrollore ESP32-WROVER-IE

Figura 1: Il Modulo ESP32-WROVER

Il microcontrollore (μC) scelto per lo sviluppo del progetto è il diffuso ESP32 di Espressif: è una soluzione SoC di rete Wi-Fi e Bluetooth di dimensioni 5x5 mm in grado di supportare le più svariate applicazioni software. Il dispositivo ha una memoria cache integrata che contribuisce a migliorare le prestazioni del sistema e ridurre i requisiti di memoria. Può assumere il compito di interfaccia Wi-Fi per l'accesso alle Local Area Network ed ad Internet in modalità wireless. E' possibile integrarlo a qualsiasi disegno microcontrollor-based perché il suo collegamento è semplice: basta un interfaccia SPI/SDIO o interfaccia bridge AHB. Potenza di elaborazione e capacità di archiviazione possono essere integrati per collegare sensori e altre attrezzature specifiche tramite le porte GPIO, per realizzare prototipi a basso costo ad alta velocità di sviluppo.

ESP32 è un sistema dual-core con due CPU Xtensa LX6 Harvard Architecture. Tutta la memoria incorporata, la memoria esterna e le periferiche si trovano sul bus dati e/o sul bus delle istruzioni di queste CPU. Con alcune piccole eccezioni, la mappatura degli indirizzi di due CPU è simmetrica, il che significa che usano gli stessi indirizzi per accedere alla stessa memoria. Più periferiche nel sistema possono accedere alla embedded memory tramite DMA. Le due CPU sono denominate "PRO_CPU" e "APP_CPU" (per "protocollo" e "applicazione"), tuttavia, per la maggior parte scopi le due CPU sono intercambiabili nel loro utilizzo. E' un sistema altamente integrato con antenna incorporata, balun RF, amplificatore di potenza, amplificatore di ricezione a basso rumore, filtri e moduli di gestione dell'alimentazione. Progettato per dispositivi mobili, elettronica indossabile ed applicazioni IoT, ESP32 raggiunge un consumo energetico estremamente basso con una

combinazione di diversi tipi di software proprietario. ESP32 include anche funzionalità all'avanguardia, come un raffinato clock gating e varie modalità di alimentazione e scalabilità dinamica. In ultimo, ESP32 è in grado di funzionare in modo affidabile in ambienti industriali, con una temperatura operativa compresa tra -40 ° C e + 125 ° C. Alimentato da circuiti di taratura avanzati, ESP32 può correggere dinamicamente le imperfezioni dei circuiti esterni e adattarsi ai cambiamenti delle condizioni esterne. Può funzionare come sistema standalone completo o come dispositivo slave per un MCU host, riducendo il sovraccarico di comunicazione sul processore dell'applicazione principale. ESP32 può interfacciarsi con altri sistemi per fornire funzionalità Wi-Fi e Bluetooth attraverso le sue interfacce SPI / SDIO o I2C / UART.

Per il nostro progetto, viene utilizzato nella sua forma di modulo; questa modalità ci permette di semplificare lo sviluppo della scheda e migliorarne la funzionalità, poiché il modulo contiene il microcontrollore e la circuiteria necessaria per il suo funzionamento. Sono infatti inclusi condensatori di disaccoppiamento e di alimentazione, un quarzo da 40 MHz, circuiteria per il matching RF ma soprattutto la memoria flash SPI, che altrimenti andrebbe aggiunta esternamente. Sempre collegata al bus SPI, troviamo la RAM pseudo-statica (PSRAM), un tipo di memoria che permette di unire la facilità d'uso delle RAM statiche con la densità di quelle dinamiche.

L'utilizzo di questo modulo fa risparmiare tempo e denaro, permettendoci di concentrarci sulle periferiche piuttosto che sulla circuiteria di supporto al microcontrollore.

Di seguito le principali caratteristiche tecniche del μC e funzionalità attraverso una tabella riassuntiva

- Microcontrollore Tensilica LX6, Dual core controllati indipendentemente con frequenza di clock regolabile, che vanno da 80 MHz a 240 MHz con 600 DMIPS e SRAM 520 KB integrata
- 4 MByte flash incluse nel modulo WROVER
- Un'uscita con guadagno di +19,5 dBm sull'antenna incorporata.
- Supporto Bluetooth classico per connessioni legacy, che supporta anche L2CAP, SDP, GAP, SMP, AVDTP, AVCTP, A2DP (SNK) e AVRCP (CT)
- Supporto per i profili Bluetooth Low Energy (BLE) inclusi i profili L2CAP, GAP, GATT, SMP e GATT come BluFi, SPP-like, ecc. per connettersi agli smart phones, trasmettendo beacon a bassa energia per un facile rilevamento
- Amplificatore analogico a bassissimo rumore, Oscillatore a cristallo 32 kHz,
- 26 x pin I / O digitali, 12 canali di ingresso ADC, 2 x audio I2S, 2 x DAC
- Ingresso / uscita PWM / timer disponibile su ogni pin GPIO

- Interfaccia di debug di OpenOCD con buffer TRAX da 32 kB, SDIO master / slave 50 MHz

- Supporto dell'interfaccia SD-card

L'assorbimento di corrente di deep sleep è inferiore a 5 μA, rendendolo adatto per applicazioni alimentate a batteria e elettronica indossabile. (Wareable Devices)

- Le periferiche disponibili includono sensori tattili capacitivi, sensore Hall, amplificatori di rilevamento a basso rumore, interfaccia scheda SD, Ethernet, SPI ad alta velocità, UART, I2S e I2C

Categories	Items	Specifications
Certification	RF certification	FCC/CE-RED/SRRC
Test	Reliablity	HTOL/HTSL/uHAST/TCT/ESD
Wi-Fi	Protocols	802.11 b/g/n (802.11n up to 150 Mbps)
		A-MPDU and A-MSDU aggregation and 0.4 μs guard interval support
	Frequency range	2412 ~ 2484 MHz
Bluetooth	Protocols	Bluetooth v4.2 BR/EDR and BLE specification
	Radio	NZIF receiver with –97 dBm sensitivity
		Class-1, class-2 and class-3 transmitter
		AFH
	Audio	CVSD and SBC
Hardware	Module interfaces	SD card, UART, SPI, SDIO, I^2C, LED PWM, Motor PWM, I^2S, IR, pulse counter, GPIO, capacitive touch sensor, ADC, DAC, Two-Wire Automotive Interface (TWAI®, compatible with ISO11898-1)
	On-chip sensor	Hall sensor
	Integrated crystal	40 MHz crystal
	Integrated SPI flash	4 MB
	Integrated PSRAM	8 MB
	Operating voltage/Power supply	3.0 V ~ 3.6 V
	Minimum current delivered by power supply	500 mA
	Recommended operating temperature range	–40 °C ~ 85 °C
	Package size	(18.00±0.15) mm × (31.40±0.15) mm × (3.30±0.15) mm
	Moisture sensitivity level (MSL)	Level 3

Tabella 1: Specifiche Tecniche del modulo ESP32 (Fonte [2])

….Insomma un piccolo mostro…..

Come si può rilevare dalle caratteristiche e dalla tabella, il microcontrollore è completo e ricco di funzioni Ad esempio troviamo funzionalità Wi-Fi e Bluetooth già integrati, un ADC a 12 bit, tutti i principali bus di comunicazione, DAC e tante altre funzionalità per l'IoT. Il microcontrollore contiene inoltre 448KB di ROM, 520KB di SRAM, e ben 34 GPIO. Da non trascurare il fatto che il software embedded può essere aggiornato tramite Wi-Fi in modalità FOTA, (Flash Over the Air), quindi senza la necessità di un collegamento fisico con la board.

Avendo già sperimentato questo microcontrollore, ho potuto apprezzare le sue grandi performance nella gestione di interfacce I/O time critical. Inoltre, il suo basso costo e la sua implementazione semplificata, hanno fatto di lui il candidato perfetto per il mio progetto.

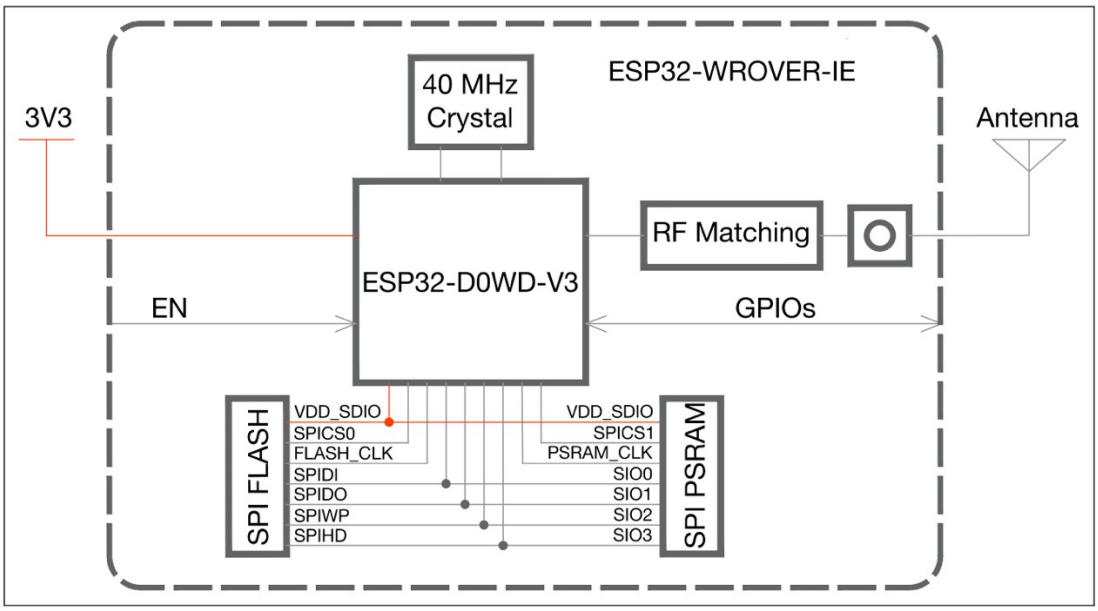

Figura 2: Schema a blocchi modulo ESP32 (Fonte [2])

Nello schema a blocchi si può constatare il µC, la memoria flash SPI e la PSRAM SPI che condividono lo stesso bus. Notiamo inoltre un oscillatore al quarzo da 40 MHz ed un connettore per l'eventuale antenna esterna per il Wi-Fi, completo di circuito di interfacciamento.

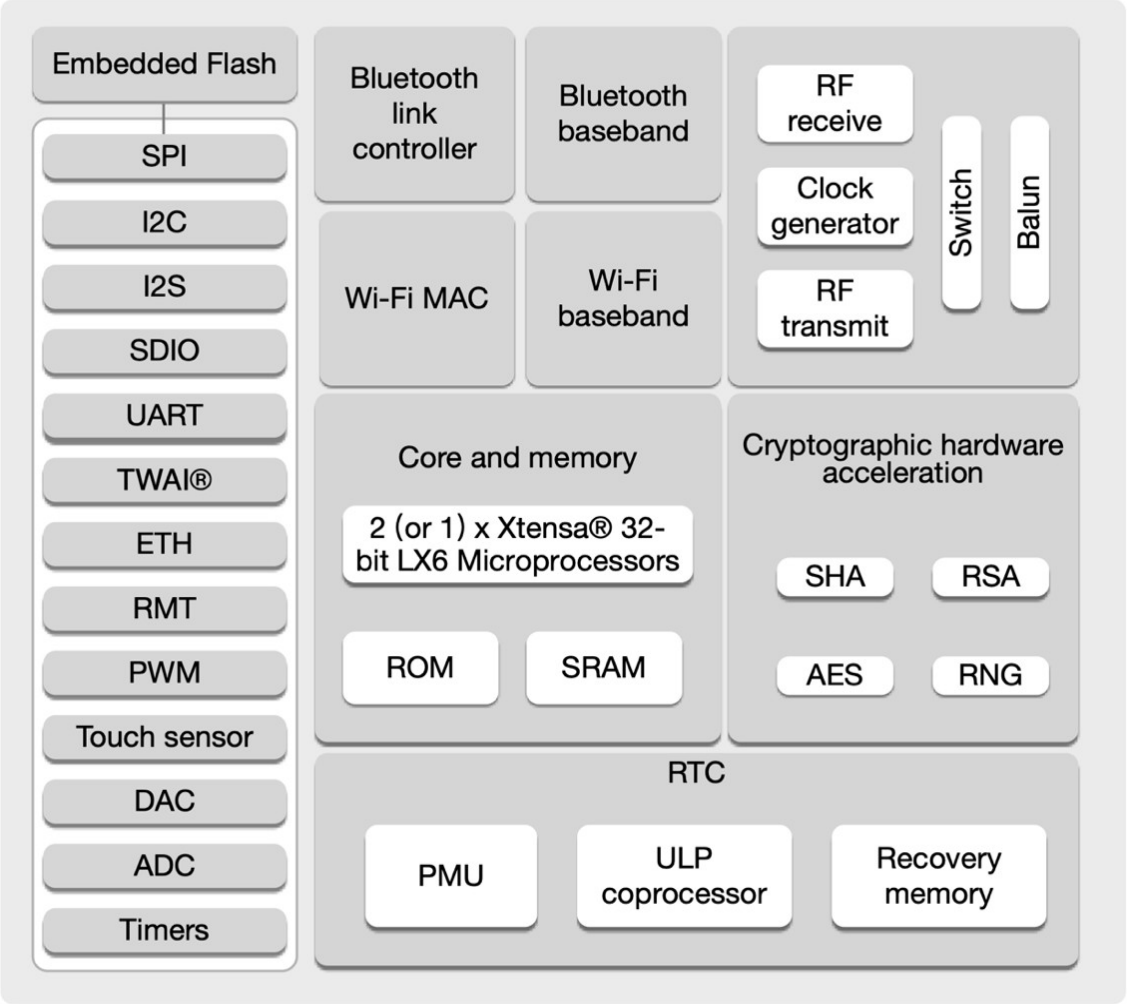

Figura 3: Schema MPU ESP32 (Fonte [1])

1.2 L’hardware di potenza

Abbiamo incluso nel nostro progetto due Power-MOSFET principalmente per pilotare gli iniettori, ma disponibili per altri dispositivi che richiedano corrente elevata. La nostra scelta è ricaduta sul componente AUIRFS8403, prodotto da International Rectifier: si tratta di un MOSFET Automotive Grade, adatto quindi all’utilizzo nell’ambito automotive. Il componente presenta una bassissima $R_{DS(on)}$, caratteristica che ben si adatta nel funzionamento come interruttore, poiché dissipando poca potenza riesce a contenere l'aumento di temperatura.

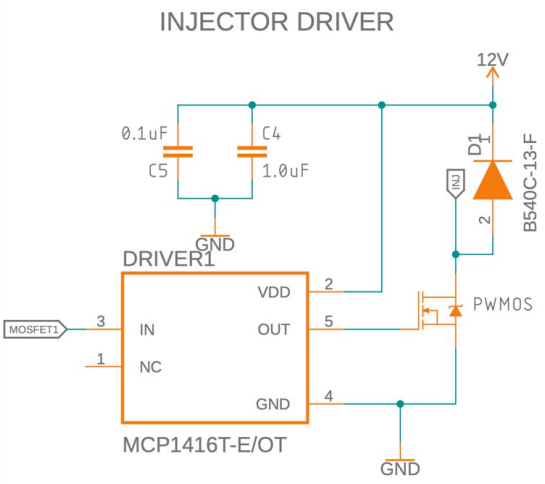

Figura 4: MOSFET con Driver e Schottky

Certamente ha una carica più elevata all’ingresso rispetto ad altri MOSFET ma, dato che la frequenza di switching degli iniettori è piuttosto bassa, cioè dell’ordine di qualche centinaio di Hertz, questo non rappresenta un problema dal punto di vista del consumo. Inoltre, abbiamo dotato ogni MOSFET AUIRFS8403 di un robusto MOSFET driver, in grado di sopperire ampiamente alle necessità di pilotaggio. La scelta di questa soluzione è scaturita dopo aver provato una semplice configurazione push-pull con due transistor; questa configurazione si rilevata troppo lenta nel caricare il MOSFET per le nostre necessità, per cui si è optato per l'utilizzo di un MOSFET driver a causa della elevata corrente che può erogare, per le migliori prestazioni dovute ad essere un circuito integrato dedicato per lo specifico compito ed infine per ridurre il footprint del componente.

Il MOSFET-driver scelto è un MCP1416T, prodotto dalla Microchip, in grado di accendere e spegnere il MOSFET in circa 65ns, un valore ottimo per le nostre esigenze, riducendo il calore prodotto nel transitorio. Questa scelta apre la strada per utilizzare l'uscita di potenza non solo per gli iniettori automotive, che hanno frequenze di switching dell'ordine di poche centinaia di Hertz, ma anche per apparati che necessitino di frequenze di switching più elevate.

Per la sezione di alimentazione abbiamo optato per un convertitore DC-DC di tipo switching: il TPS562211DRLR. La scelta è scaturita dall'utilizzo della utility Power Designer di TI WEBENCH, che permette di districarsi fra l’enorme quantità di regolatori switching oggi in commercio.

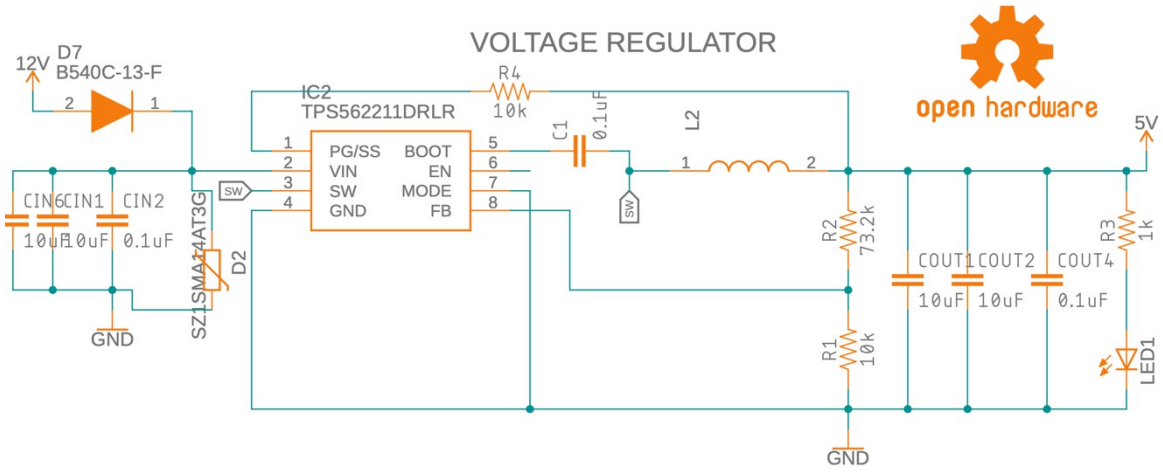

Figura 5: Il regolatore DC-DC

Fornendo alcuni dati in input, come la tensione in ingresso ed uscita e la corrente in uscita, il software restituisce i chip più adatti ed efficienti per soddisfare quanto inserito, comprendendo inoltre la circuiteria necessaria per gestire al meglio il singolo chip.

Abbiamo inserito inoltre il regolatore low-dropout (LDO) NCP1117ST33T3G, molto diffuso nell'ambito dei microcontrollori, che permette una corretta e stabile alimentazione del modulo, portando la tensione da 5V a 3.3V necessari. La scelta è ricaduta su un LDO perché il salto di tensione è relativamente contenuto ed inoltre, essendo un regolatore di tipo lineare, è meno rumoroso e di conseguenza porterà meno disturbi all'alimentazione del microcontrollore.

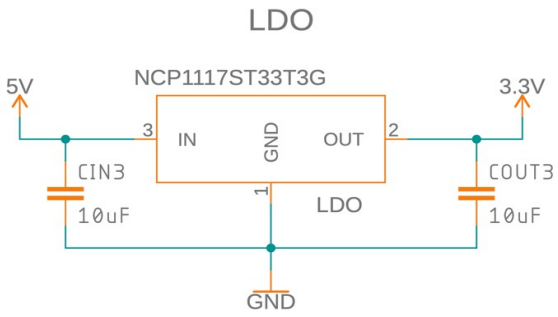

Figura 6: Regolatore LDO

Per quanto riguarda i componenti discreti, abbiamo inserito due diodi Schottky B540C collegati in antiparallelo al carico dei MOSFET, in modo tale da dissipare eventuali correnti generate principalmente da carichi induttivi come gli iniettori, che continuano a scorrere quando il MOSFET viene spento. In questo modo si evitano picchi di tensione con la conseguente dissipazione di potenza e calore sul MOSFET.

Abbiamo inserito un altro diodo Schottky all'ingresso del convertitore DC-DC per la protezione dall'inversione di polarità. Si è potuto utilizzare questo diodo in quanto la corrente che scorre è bassa, cioè nell'ordine di qualche centinaio di mA, ed ha una caduta di tensione di 0.2V, riducendo la dissipazione di potenza ad un valore irrisorio per la specifica applicazione, in cui i consumi specifici sono nell'ordine delle centinaia di mW. Abbiamo inserito anche un diodo SMAJ14A, di tipo TVS, per la protezione del circuito di alimentazione.

Infine abbiamo inserito due fusibili resettabili per la protezione del circuito di alimentazione a 12V ed un fusibile generale di alimentazione, oltre ad un interruttore principale e uno di alimentazione secondaria in uscita, per la simulazione del +12V sottochiave.

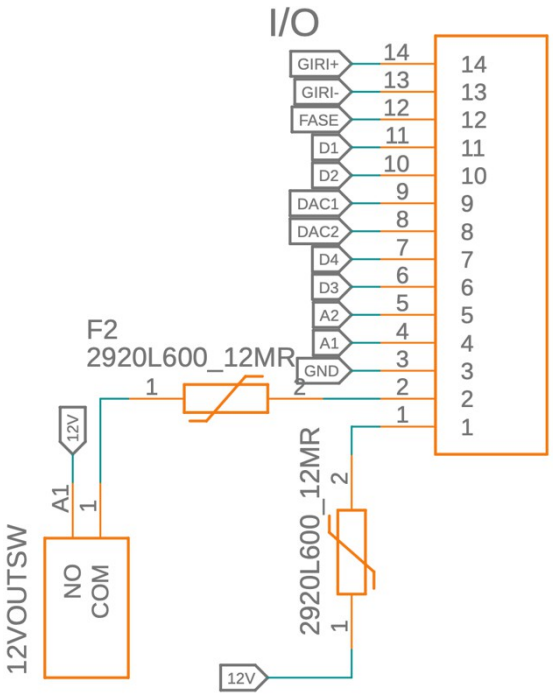

Figura 7: Connettore di uscita segnali con fusibili ed interruttore

1.3 L'hardware di segnale

Vista l'elevata integrazione di periferiche nel microcontrollore, si riduce la necessità di apparati elettronici esterni, quali DAC e ADC. E' necessario però modificare le tensioni di funzionamento per poter lavorare sul range 0-5V. Il microcontrollore opera correttamente a 3.3V, quindi abbiamo predisposto cinque I/O digitali dotati di un semplice circuito convertitore di livello, formato da un MOSFET di segnale e due resistori di pull-up. Inoltre, ai due DAC è stato collegato un amplificatore operazionale che, con un semplice circuito non invertente, aumenta la tensione di uscita in modo proporzionale.

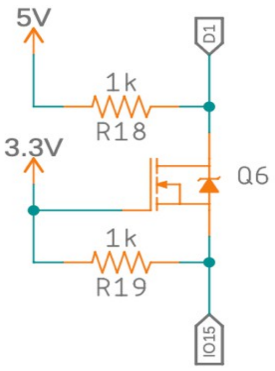

Figura 8: Convertitore di livello

Sono stati poi inseriti quattro ingressi analogici, che si avvalgono di un semplice partitore resistivo per diminuire la tensione. Sono presenti due potenziometri per gestire alcune variabili come numero di giri oppure il duty cycle delle forme d'onda in uscita. Completa la configurazione un pulsante di reset, un circuito di auto-reset e un led per segnalare la presenza di alimentazione.

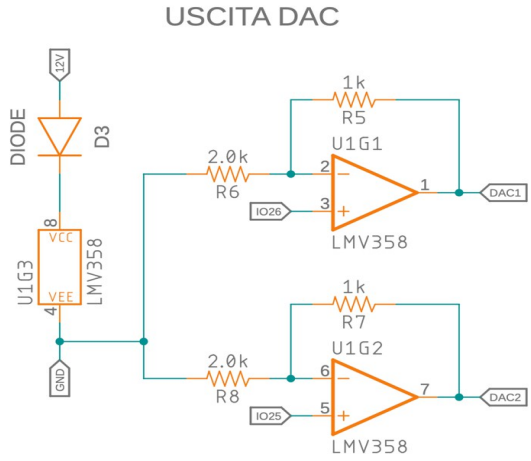

Figura 9: Uscita DAC con OPAMP

Il display di interfaccia è connesso tramite bus SPI e comprende due controller, uno per la grafica e l'altro per il touch screen. Vengono quindi predisposti e configurati due pin per il chip select del bus SPI.

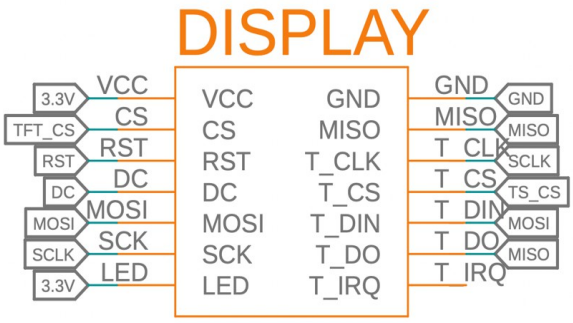

Figura 10: Connettore Display

1.4 Il sistema di I/O

Per realizzare le connessioni con le periferiche e l'alimentazione sono stati utilizzati ben cinque connettori. Il primo, a due pins, è quello di alimentazione in grado di sostenere correnti elevate; il secondo, ad otto pins, è quello di uscita per degli iniettori, quattro pins per ogni MOSFET, anche questo in grado di gestire elevate potenze in uscita.

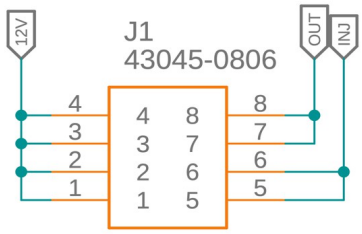

Figura 11: Connettore Iniettori

E' presente poi il connettore di programmazione a sei pins, quello di I/O dei segnali a 14 pins e quello dedicato al collegamento del display touch, a 14 pins anch'esso. Questi ultimi connettori sono costruiti per il trasporto di segnali e/o basse correnti, per questo possono essere più compatti.

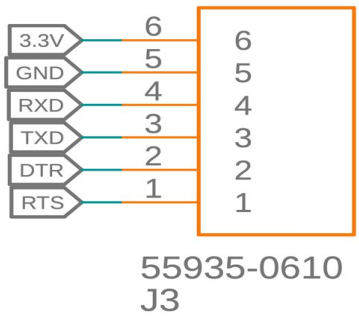

Figura 12: Connettore UART

In serie al connettore di alimentazione, è presente un fusibile da 20A, per la protezione del circuito, e l'interruttore di alimentazione principale. Il connettore UART include ulteriori due pin: Data Terminal Ready (DTR) e Ready to Send (RTS), che sono utilizzati per la programmazione e l'auto-reset del microcontrollore. Impostando infatti il pin Enable(EN) del microcontrollore a massa, si ottiene il reset del microcontrollore.

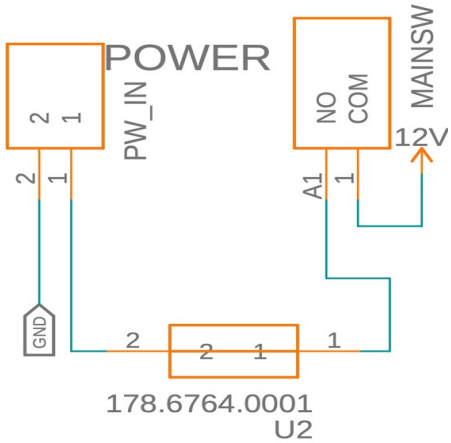

Figura 13: Connettore di alimentazione

Per programmare la flash memory interna tramite la porta seriale, occorre che in fase di avvio del microcontrollore il pin GPIO0 sia collegato a massa: in questo modo il sistema entra nella modalità di programmazione. Dopo la programmazione della Flash memory, il uC deve essere riavviato tramite un impulso di reset. Il tool di programmazione "esptool" si occupa autonomamente della procedura di programmazione e riavvio, semplificando il lavoro in fase di sviluppo, dove spesso è necessario ricaricare più volte il codice durante il collaudo. Una volta completata la fase di progettazione e collaudo, la versione definitiva per la produzione può eventualmente essere bloccata rimuovendo questa funzionalità: per l'eventuale riprogrammazione si è obbligati ad utilizzare due pulsanti per entrare in boot-mode manualmente.

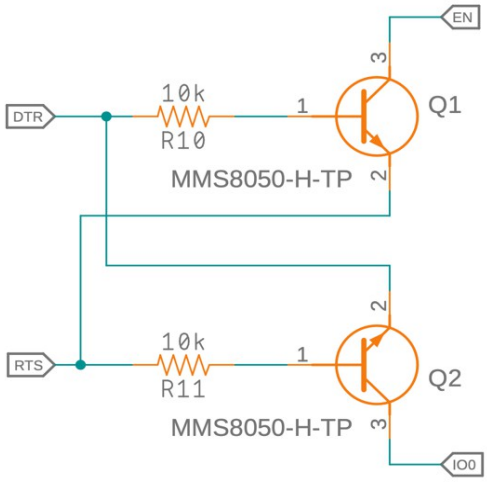

Figura 14: Circuito Auto-Reset

1.5 Lo schema circuitale ed il routing

Oltre ai componenti precedentemente citati, sono presenti alcuni condensatori vicino ai vari integrati, come previsto da datasheet, per effettuare il disaccoppiamento delle linee di alimentazione e ridurre le cadute di tensione ai terminali.

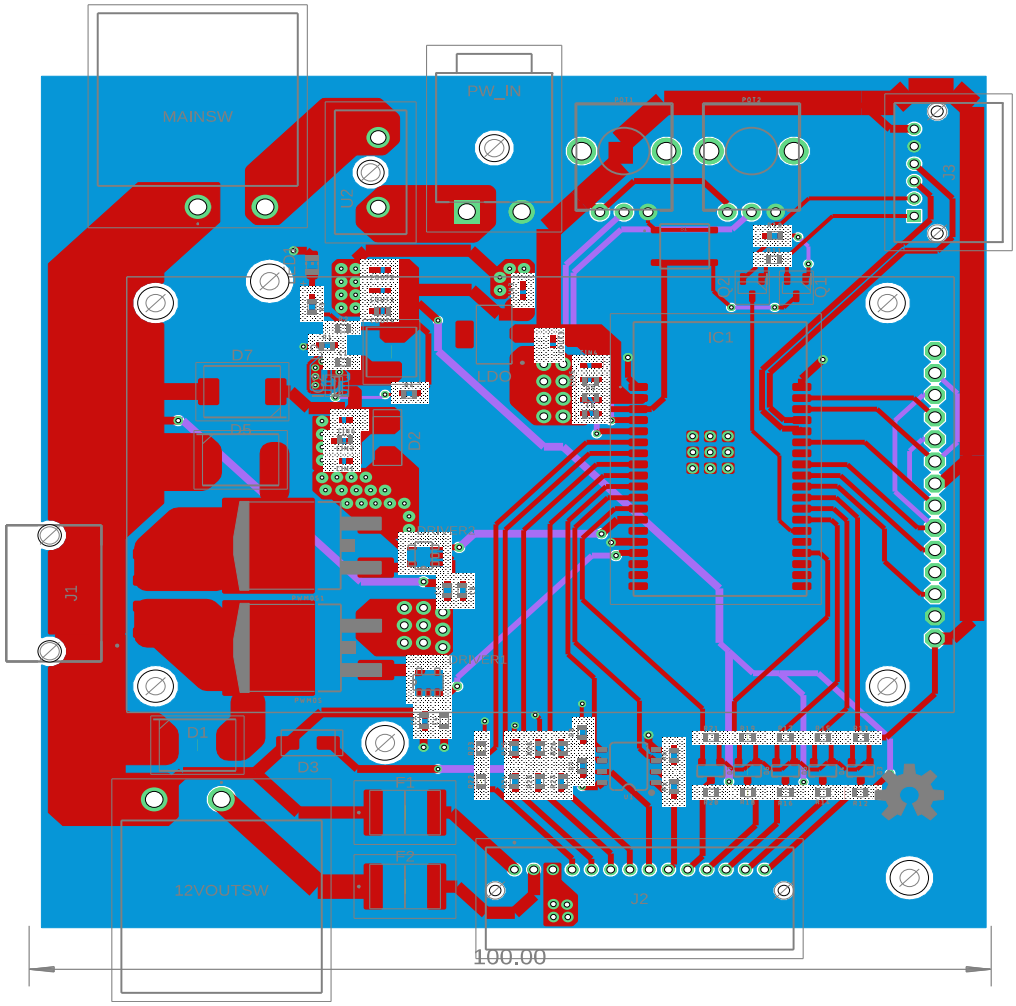

Figura 15: Layout del PCB realizzato

E' stato utilizzato un PCB stratificato a 4 layers, riservando il secondo e il quarto livello per due piani di massa che sono senza soluzione di continuità. Il primo livello è il layer principale, utilizzato sia per le piste di potenza che per quelle di segnale. Il terzo layer invece è utilizzato quasi esclusivamente per il trasporto di segnali o di piccole alimentazioni, che non potevano facilmente essere collegate attraverso il copper top.

La configurazione dei layers è piuttosto insolita, ma imposta dalla particolarità del circuito: si è scelto di non dedicare un layers esclusivamente all'alimentazione perché i componenti a 3.3V ed a 5V, escluso il modulo e il display touch, consumano una quantità di energia irrisoria. Inoltre il regolatore LDO, che è il consumatore maggiore di energia, è posizionato in prossimità del modulo. La parte alimentata a 12V invece richiede poligoni e componenti ravvicinati per evitare di aumentare l'impedenza. Si è quindi scelto di dedicare due layers

ininterrotti al collegamento di massa per ridurre le interferenze elettromagnetiche. L'uso di piani VDD e GND riduce l'interferenza elettromagnetica (EMI), a causa delle minori aree dei loop di corrente e fanno chiudere le linee del campo magnetico sui piani di massa.

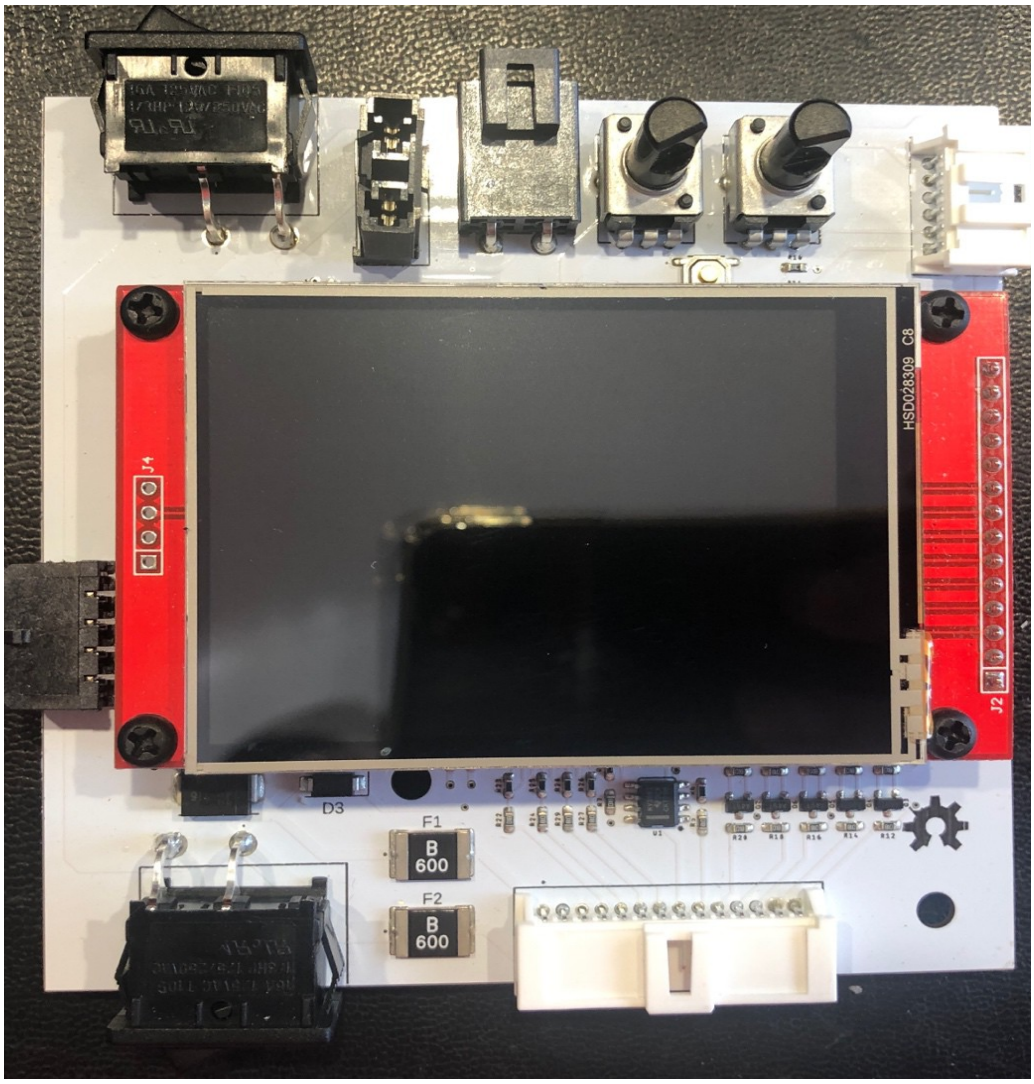

Figura 16: La scheda finita e montata con il display

Deve inoltre essere ricordato che i layers interni hanno uno spessore dimezzato rispetto a quelli esterni; quindi i circuiti di potenza devono essere necessariamente sbrogliati attraverso il top layer per usufruire di una maggiore conduttanza a parità di dimensioni delle piste. Lo spessore del layers dipende ovviamente dal produttore e dal budget a disposizione: con i layers interni dello stesso spessore di quelli esterni si esce dallo standard e si paga un sovrapprezzo di produzione.

Per ottimizzare le dimensioni della scheda si è cercato di compattare il più possibile i componenti, evitando in questo modo di avere piste di lunghezza eccessiva: per i componenti di potenza che si trovano a ridosso dell'alimentazione sono stati utilizzati dei poligoni per aumentare la larghezza delle piste al massimo possibile. Sempre per la riduzione delle dispersioni, anche i regolatori di tensione sono stati posizionati entrambi

molto vicino all'alimentazione principale. Completa l'opera di ottimizzazione la buona quantità di vias verso massa, che servono per ridurre al minimo la resistenza.

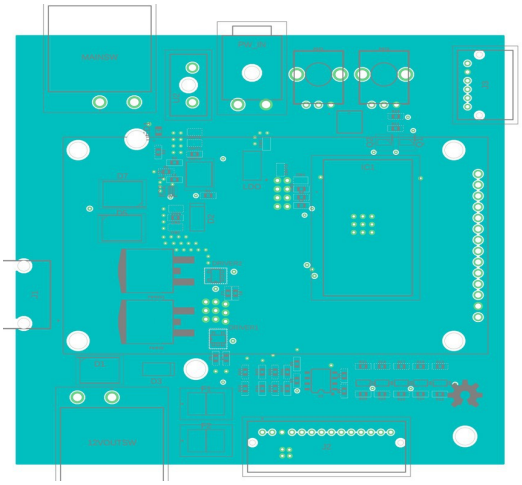

Figura 17: Secondo Livello, GND

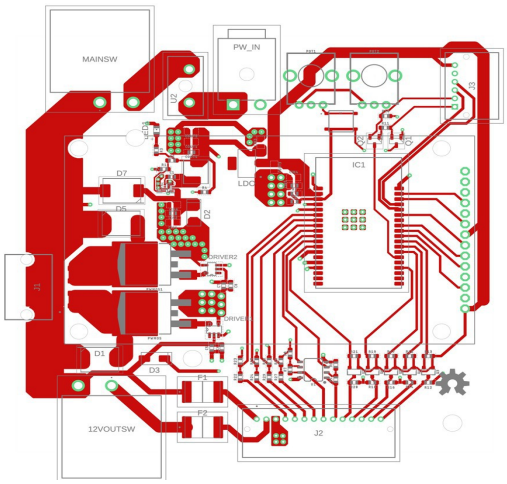

Figura 18: Top Layer

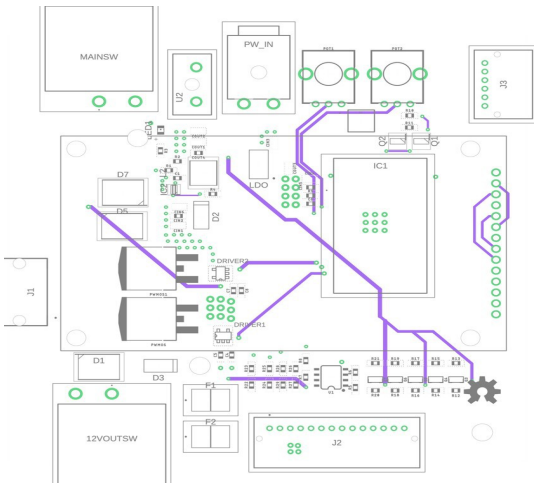

Figura 19: Terzo Livello, collegamenti

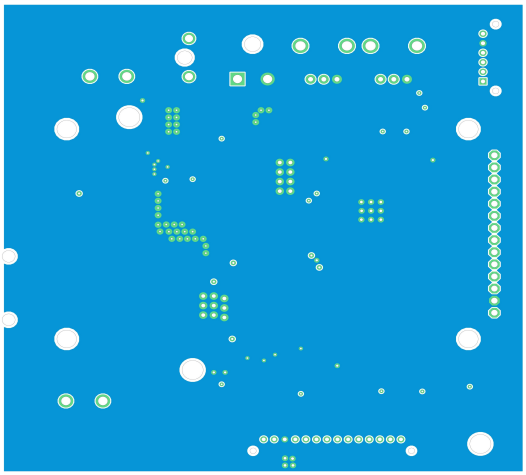

Figura 20: Bottom Layer GND

Capitolo 2: Il Software

2.1 L’ambiente di sviluppo

Consideriamo ora il software che gestisce l’intera scheda: il microcontrollore dispone di un framework proprietario, chiamato ESP-IDF, il quale non è stato utilizzato poiché si è preferito optatare per un altro framework di sviluppo basato sull'IDE (Integrated development environment) di Arduino, che utilizza un linguaggio di programmazione C++ semplificato. I vantaggi sono principalmente due: facilità di programmazione (precedente esperienza con il linguaggio per Arduino) e la presenza di un parco librerie ready-to-use molto vasto, che fornisce una gestione in modo agevole ed ottimizzato del display e di altri componenti aggiuntivi. Per questi motivi è stato utilizzato Visual Studio Code in combinazione con l'estensione PlatformIO, che consente di programmare tantissimi microcontrollori in vari linguaggi; questo strumento di sviluppo permette una gestione del codice più ordinata rispetto al classico IDE di Arduino, grazie alle funzioni a disposizione, tra cui non ultime la funzione di auto-completamento del codice ed il rilevamento automatico della porta seriale in fase di caricamento del codice su flash memory.

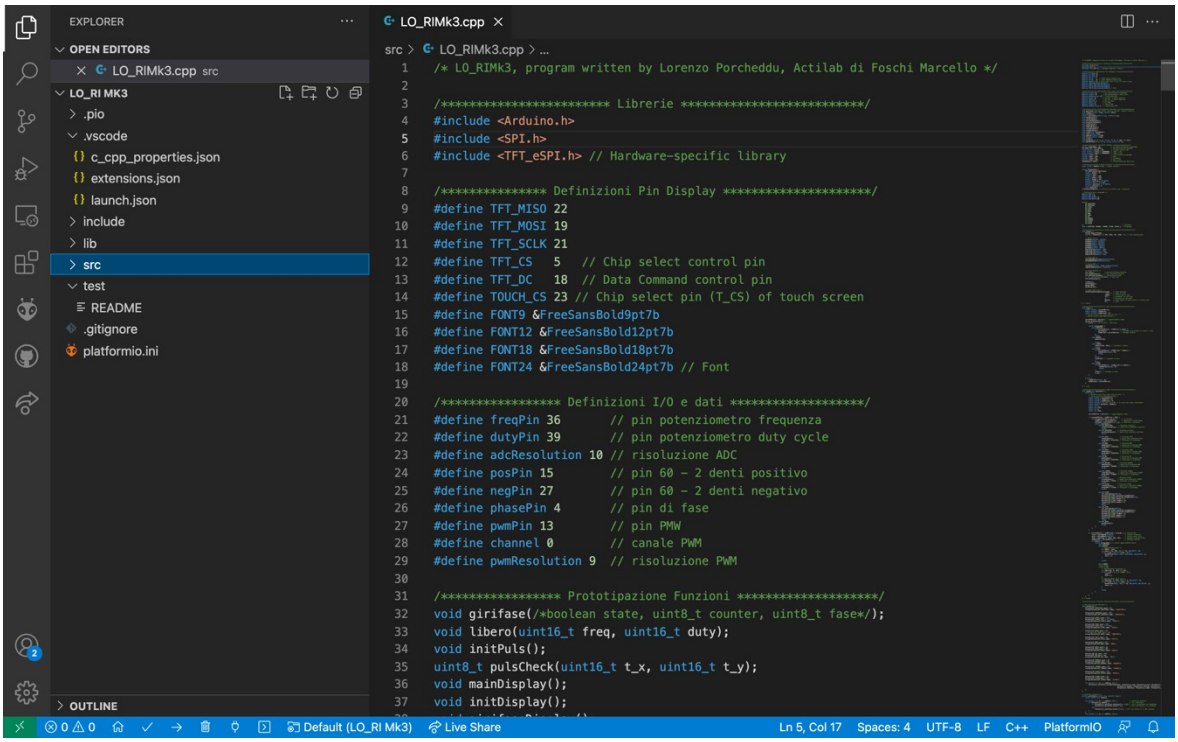

Figura 21: L'IDE Visual Studio Code

Ultimamente la fondazione Arduino ha rilasciato la versione 2.0 del suo IDE che integra queste migliorie, ma è ancora in fase beta e quindi si è preferito non utilizzarla per ora.

2.2 Sistema Operativo FreeRTOS

La scelta di un microcontrollore dual-core non è stata casuale; infatti, la comunicazione con display touch è molto onerosa dal punto di vista del tempo CPU, perché la sua scrittura richiede diverse centinaia di millisecondi. Per un microcontrollore che funziona a 240 MHz si tratta di un tempo molto elevato che non permetteva di svolgere in modo efficiente le altre funzioni.

La libreria tft-eSPI, che gestisce sia la parte grafica che il touchscreen, è ben ottimizzata ma comunque non permette lo svolgimento di altre operazioni real-time, visto i limiti imposti dalla comunicazione con il display stesso. Avendo invece a disposizione due core, è possibile riservare un core per la comunicazione del display e l'altro alle operazioni di lettura/scrittura delle GPIO, come la generazione di forme d'onda o la lettura degli ADC. Per sfruttare entrambi i core dobbiamo utilizzare le funzioni di FreeRTOS: si tratta di un sistema operativo real-time per i sistemi embedded, dove le varie funzioni vengono eseguite a seconda della priorità assegnata su vari livelli. Quindi, per esempio, possiamo assegnare una priorità elevata alle funzioni che sono fortemente dipendenti dal tempo e una più bassa alle funzioni che possono essere posticipate anche di qualche secondo. Ovviamente fare scheduling in questo modo può rivelarsi un'arma a doppio taglio, poiché si potrebbe perdere il controllo delle funzioni che vengono eseguite sul microcontrollore, visto che ci stiamo spostando verso una gestione più simile a quella dei microprocessori.

```
/* Setup Multicore */
xTaskCreatePinnedToCore(loop0,    // Task function
                        "Task_1", // name of task.
                        10000,    // Stack size of task
                        NULL,     // parameter of the task
                        1,        // priority of the task
                        &Task1,   // Task handle to keep t
                        0);       // Core
```

Figura 22: Impostazione del secondo core nel setup()

Questo sistema si rileva essere molto utile nell'ambito dell'IoT, dove vengono letti dei sensori e vengono trasmessi anche in remoto i valori di lettura in modalità asincrona: almeno per il momento, nel nostro caso non è necessario. Normalmente si utilizza del sistema solamente un core, per una programmazione di semplici funzionalità: per sfruttare anche l'altro è necessario istruire il sistema utilizzando una funzione che ho chiamato **loop0** che racchiude l'intero processo da mandare in esecuzione sul secondo core. Abbiamo quindi assegnato un task per ognuno dei due core.

2.3 Il codice

Il codice è formato principalmente da tre parti: il **setup**, che viene eseguito una volta sola all'avviamento del microcontrollore, il **loop**, che viene eseguito continuamente sul primo core ed il **loop0**, che viene eseguito perpetuamente sul secondo core. Nel **setup** si inseriscono le istruzioni di impostazione del sistema HW e SW, quali GPIO, ADC, DAC, PWM ed il task del secondo core. Nel **loop** sono presenti invece le funzioni di generazione di forme d'onda sia per le uscite di segnale che per quelle di potenza: in base al tasto premuto sul display, viene attivata, attraverso l'analisi tramite il costrutto switch(case), un'uscita piuttosto che un'altra. Sono inoltre presenti le temporizzazioni per la gestione di alcune forme d'onda, che sfruttano le funzioni millis() e micros() per la temporizzazione.

```
/************************ Setup ***********************
void setup() {
    Serial.begin(115200);
    uint16_t calData[5] = { 364, 3505, 313, 3438, 7 };

    /* Setup I/O */
    pinMode(phasePin, OUTPUT);
    pinMode(posPin, OUTPUT);
    pinMode(negPin, OUTPUT);
    pinMode(pwmPin, OUTPUT);
    pinMode(freqPin, INPUT);
    pinMode(dutyPin, INPUT);
    digitalWrite(posPin, LOW);
    digitalWrite(negPin, LOW);
    digitalWrite(phasePin, LOW);
    digitalWrite(pwmPin, LOW);

    /* Setup ADC */
    analogReadResolution(adcResolution);
    analogSetWidth(adcResolution);

    /* Setup PWM */
    ledcSetup(channel, freq, pwmResolution);
    ledcAttachPin(pwmPin, channel);

    /* Setup Display */
    tft.init();           // Inizializzazione Display
    tft.setRotation(1);   // Set rotazione display
    tft.setTouch(calData); // calibrazione touch
    // tft.setFreeFont(PULSFONT); // Set font
    tft.setTextSize(1);

    initPuls();
    initDisplay();
    delay(1000);
    mainDisplay();
```

Figura 23: Blocco Setup()

Nel **loop0** sono state inserite le funzioni per la gestione della grafica e dei pulsanti, oltre che la lettura dei potenziometri. Anche in questo caso, attraverso il costrutto switch(case), viene intercettato il pulsante premuto sul display ed attivato il submenù oppure la funzione corrispondente. Per far operare il tutto in multitasking, è stato necessario temporizzare tutti i processi attraverso le funzioni **millis**() e **micros**(), che ci restituiscono rispettivamente i millisecondi e i microsecondi trascorsi dall'accensione del microcontrollore; in questo modo è sufficiente controllare la scadenza di un determinato periodo per far commutare un'uscita oppure leggere i potenziometri.

```
void loop() {
    static uint32_t currentMicros;
    static uint32_t oldMicros;
    static uint32_t oldMicros2 = 0;

    currentMicros = micros(); // aggiornamento timer
    if (start) { // se lo start è abilitato

        switch (programma) {
            case GIRIFASE:
                if (currentMicros - oldMicros >= wait)
                    girifase();               // chiam
                    oldMicros = currentMicros; // stora
                }
                break;
```

Figura 24: Blocco loop()

2.4 Le funzioni implementate

Le funzioni del software sono state divise in due aree accessibili tramite submenu: iniettori e girifase. Nella prima sono presenti le funzioni relative ai MOSFET di potenza, mentre nella seconda le funzioni relative all'uscita dei segnali.

Figura 25: Schermata principale di selezione funzioni

Le funzioni per i MOSFET riguardano la generazione di forme d'onda rettangolari, con frequenza e duty-cycle variabile, pilotabili tramite i due potenziometri presenti sulla scheda, secondo la necessità dell'operatore.

Queste forme d'onda sono molto utili, ad esempio, per provare il funzionamento degli iniettori elettronici, ma possono essere usate anche per altre applicazioni, come per esempio il riscaldamento di una resistenza corazzata. Le funzioni per i segnali invece generano forme d'onda particolari che devono emulare dei veri e propri sensori di giri e di fase di un motore endotermico. Tendenzialmente il sensore di giri ha la stessa forma d'onda in quasi tutte le autovetture, vista la presenza di una ruota fonica; il sensore di fase, al contrario, può variare da produttore a produttore. Nel caso specifico abbiamo implementato il sensore di giri e di fase di una centralina Magneti Marelli 59F, utilizzata da Fiat su Punto e Panda a benzina.

Figura 26: Schermata iniettori, duty-cycle e frequenza

Figura 27: Schermata iniettori attiva: si può notare la frequenza in alto e il duty-cycle su barra

Figura 28: Schermata girifase con numero di giri x1000

Capitolo 3: Caratterizzazione Sperimentale

Una volta montata la scheda, ne abbiamo verificato il funzionamento effettivo tramite un oscilloscopio collegato opportunamente: si è stati in grado in questo modo di acquisire alcune forme d'onda in simulazione come illustrate di seguito.

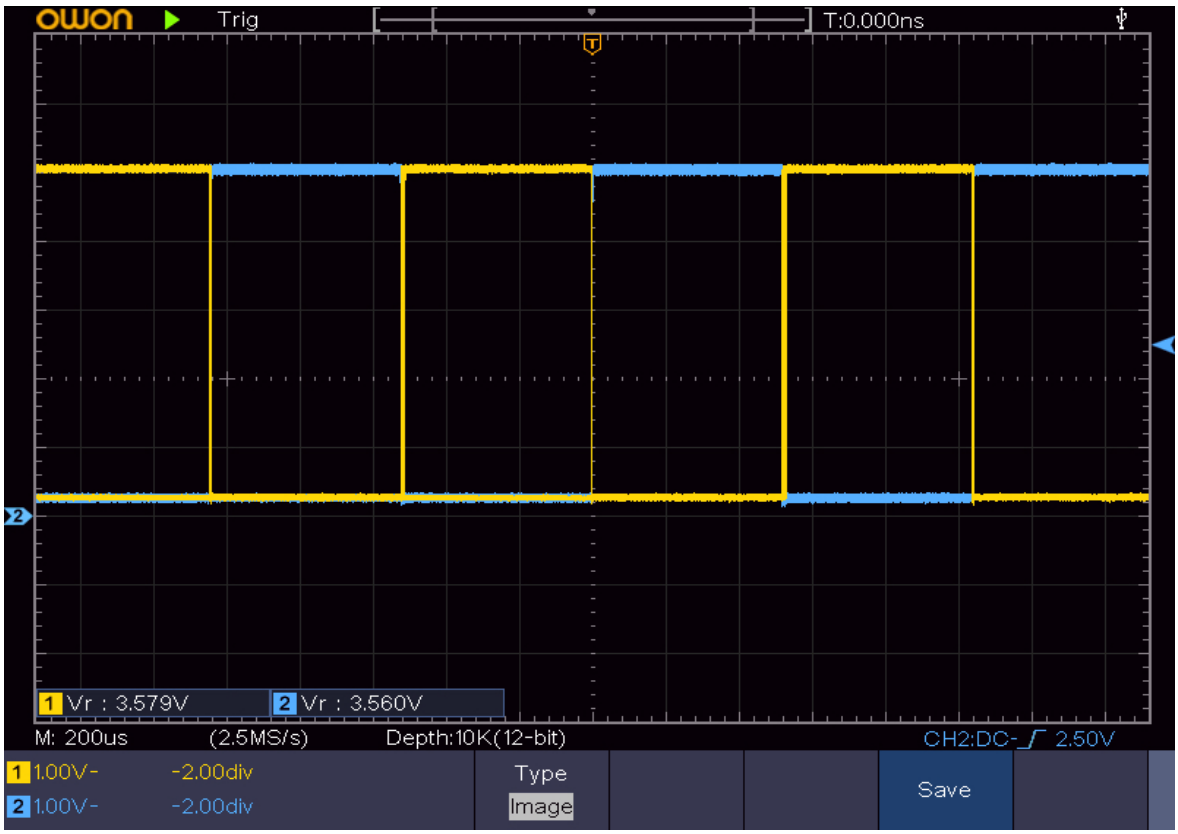

Figura 29: Forma d'onda Sensore giri

Nella precedente immagine si può notare la forma d'onda dell'uscita del sensore di giri emulata dalla scheda. La forma d'onda reale del sensore di giri è del tipo a due pin fuori massa, solitamente quadra o sinusoidale; la frequenza dell'onda determina il numero di giri a cui si trova il motore. In questo esempio sono state ricreate due onde quadre sfasate tra di loro di 180 gradi; così facendo possiamo simulare un'onda bipolare anche se in realtà bipolare non è, dato che l'uscita è fuori massa. Quest'onda ha anche un'altra particolarità: dopo 58 periodi è presente un "vuoto", ovvero corrispondente a due periodi in cui entrambe le onde sono a livello zero; questo indica che la ruota fonica è nella posizione del PMS (Punto Morto Superiore).

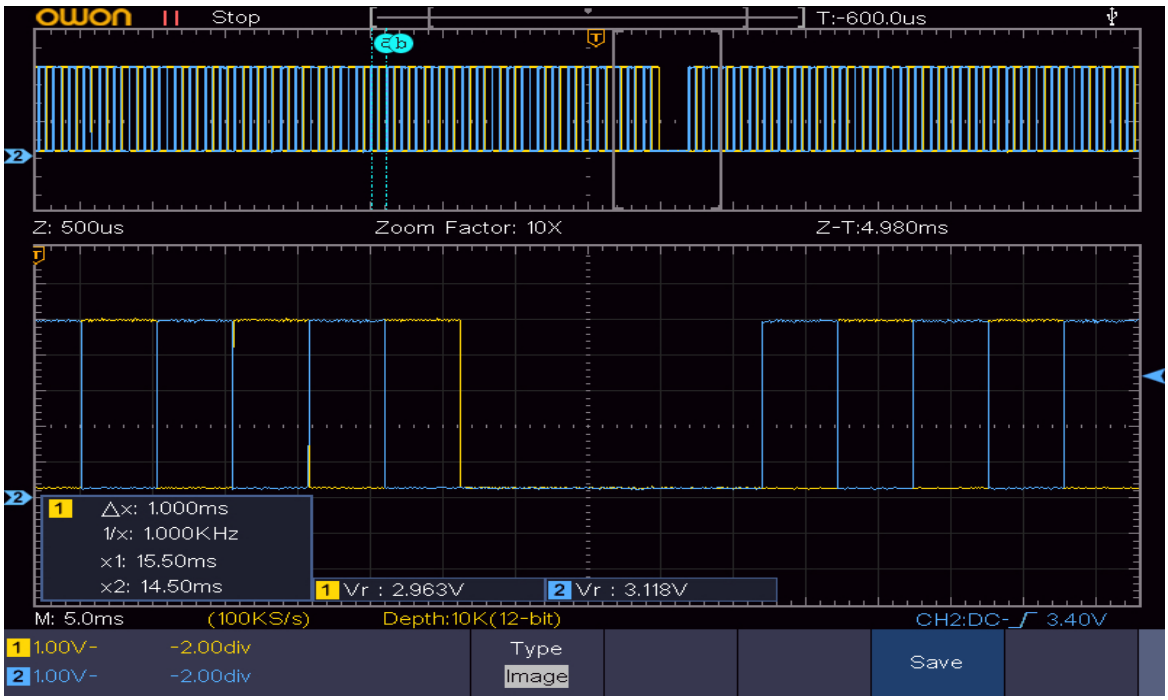

Figura 30: Zoom sul "vuoto", cioè il PMS

In questa immagine, tramite la funzione **zoom** dell'oscilloscopio, viene evidenziato il "vuoto" delle due forme d'onda, in coincidenza del P.M.S.

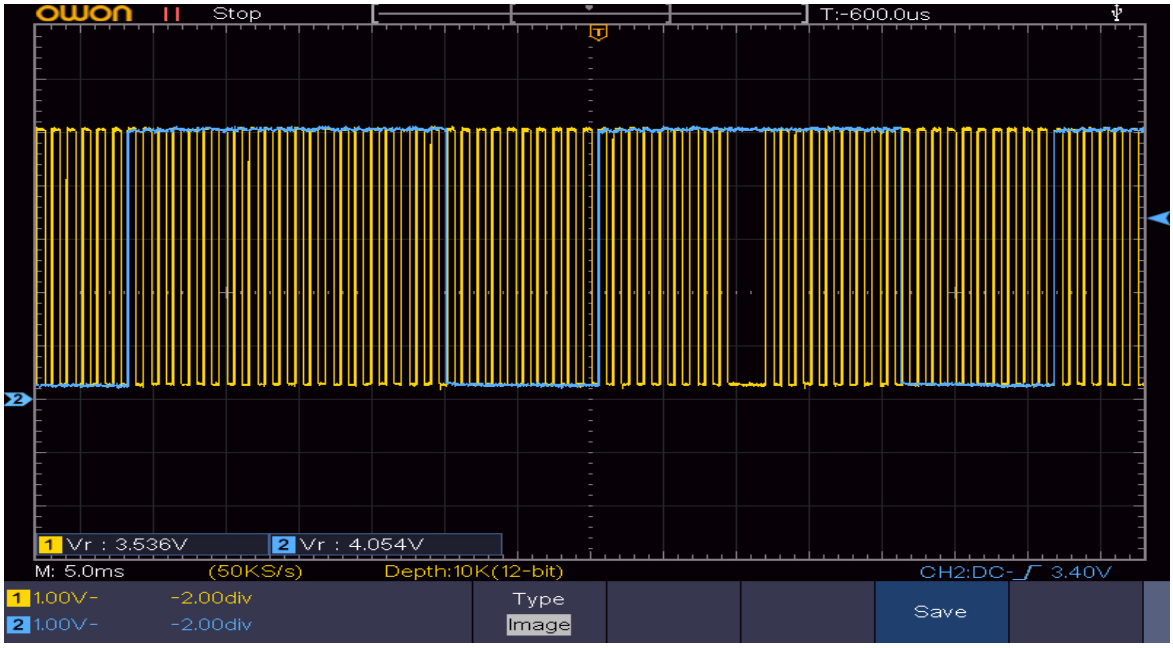

Figura 31: Forma d'onda di giri (giallo) e di fase (blu)

Nella rappresentazione in figura viene mostrato il segnale di giri (in giallo) sovrapposto a quello di fase (in azzurro). Come si può notare il segnale di fase cambia molto più lentamente.

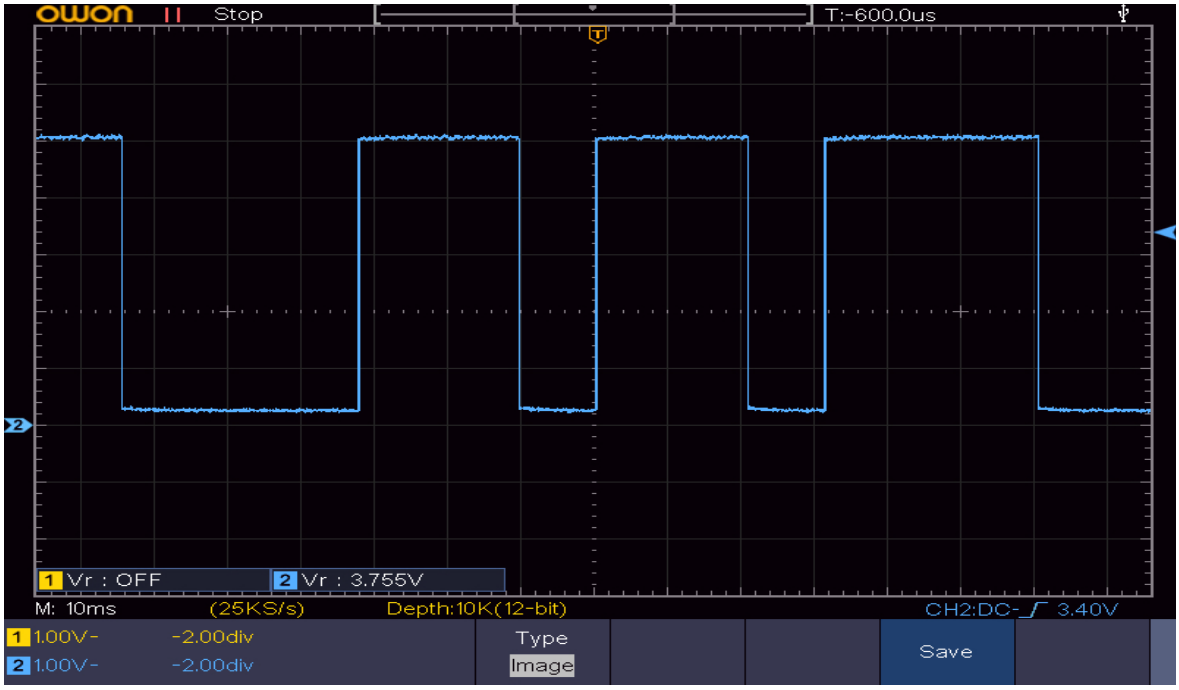

Figura 32: Segnale di fase

Nella rappresentazione in figura viene mostrato un segnale di fase puro isolato. Successivamente si è passati ad analizzare il comportamento del MOSFET.

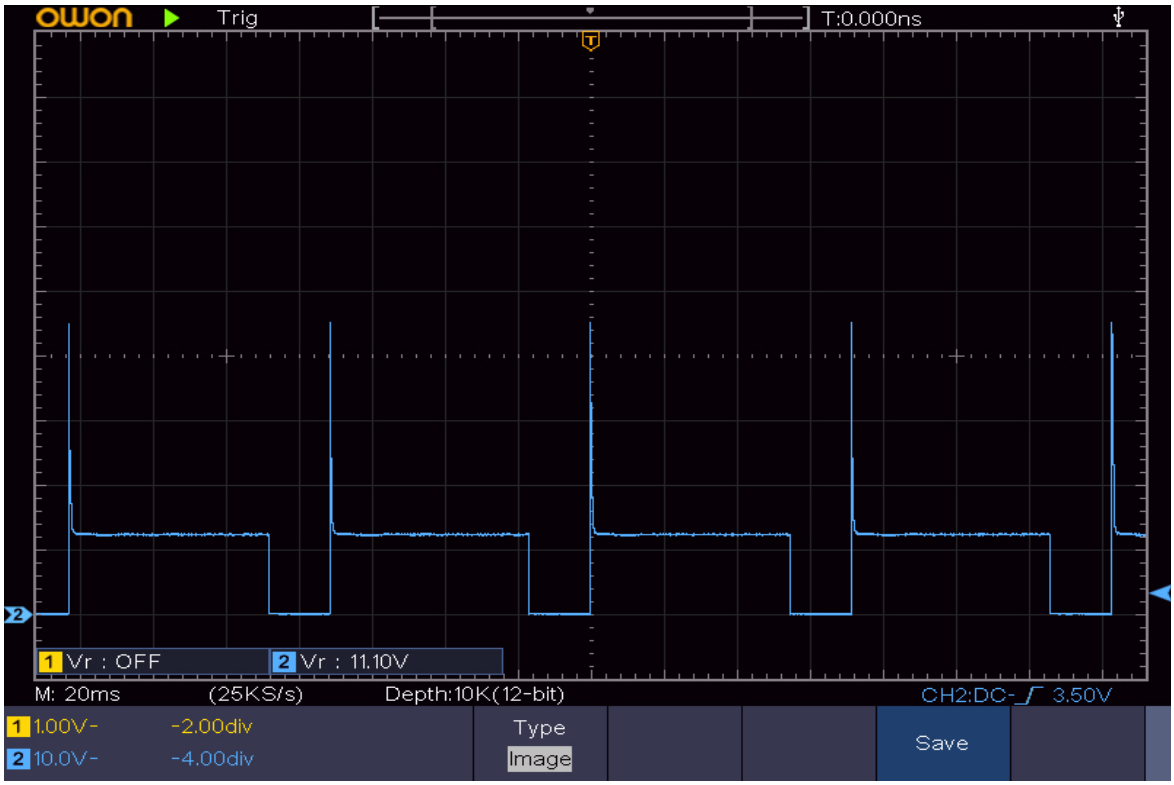

Figura 33: Uscita sul MOSFET

Nella rappresentazione in figura si può vedere la forma d'onda in uscita sul MOSFET quando è collegato un iniettore elettronico di tipo induttivo. Si è mantenuta la frequenza bassa per evitare che troppa energia venisse dissipata sul MOSFET. Infatti, durante il montaggio dei componenti sul PCB eseguito a macchina, c'è stato un errore in fase di

produzione della PCB, per cui è risultato invertito il verso dei diodi di ricircolo inseriti nello schema.

I diodi ovviamente sono stati smontati per evitare di danneggiare la scheda: senza la presenza di questo diodo, si può notare uno spike di tensione molto elevato in fase di apertura del MOSFET. Per evidenziare la particolarità di questo comportamento del MOFSET, ho catturato la forma d'onda in assenza del predetto diodo, constatando l'estrema utilità di questo diodo di ricircolo e confermando sul circuito reale le simulazioni fatte in fase di progettazione e design.

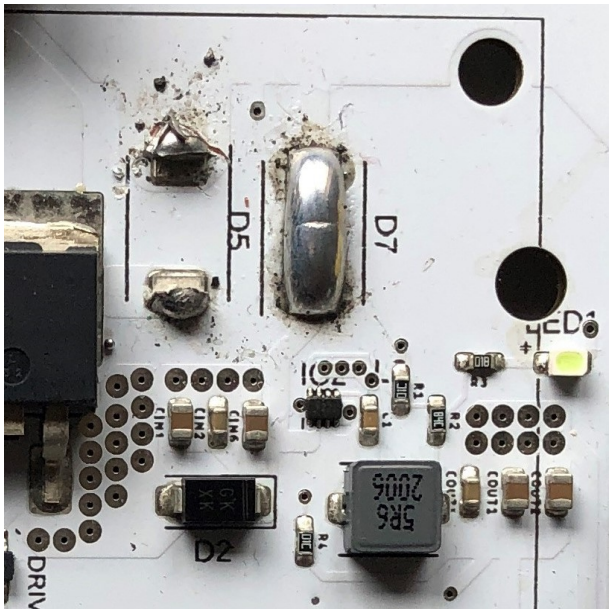

Figura 34: Diodi smontati per bypass

Dopo aver sostituito i due diodi montati al contrario con altri equivalenti, la forma d'onda ottenuta è stata quella per cui era stato progettato il circuito.

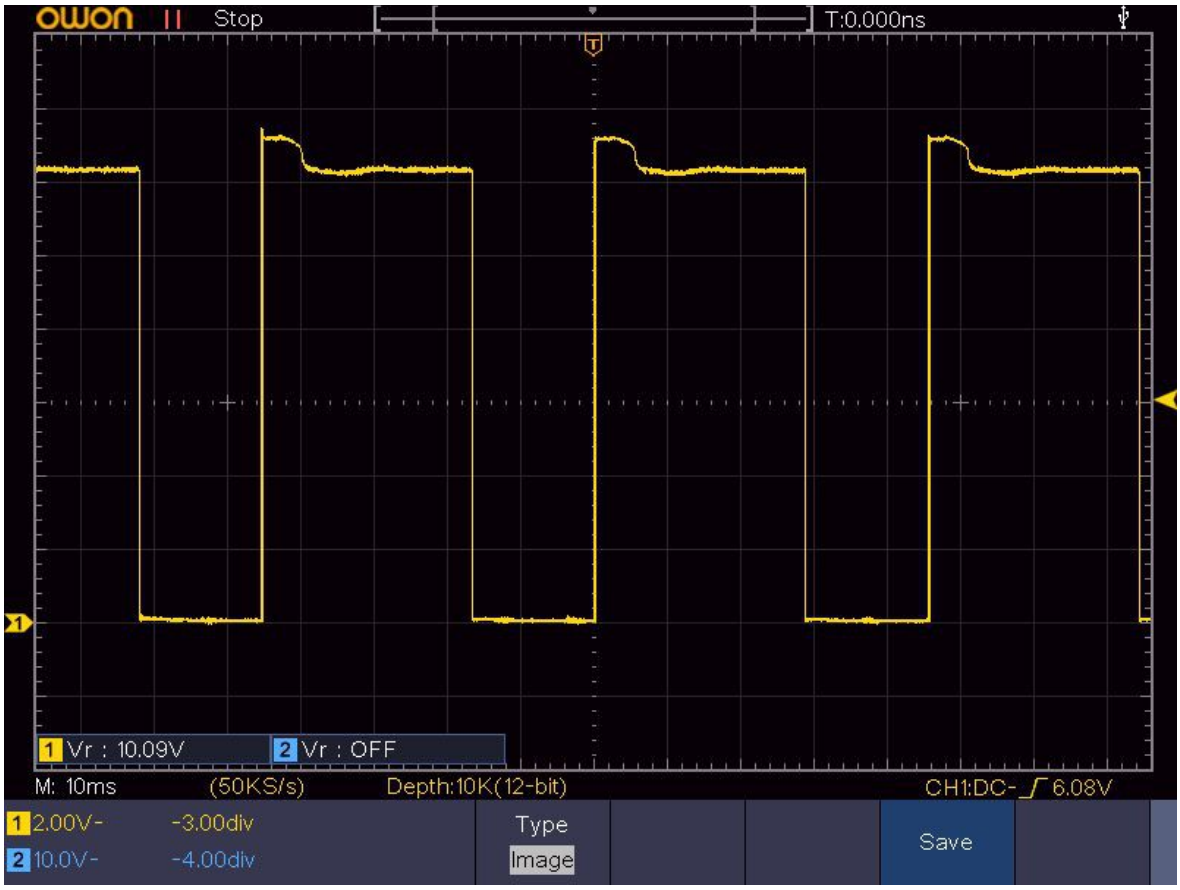

Figura 35: Uscita MOSET corretta

Conclusioni

L'esigenza di un apparato per il collaudo delle centraline elettroniche e della componentistica automotive si è evidenziata durante il mio percorso di alternanza scuola lavoro delle scuole superiori, effettuato presso la ditta Actilab di Cesena.
La ditta, specializzata nella diagnostica elettronica automotive, necessitava di un'apparato semplice ed economico per effettuare verifiche di funzionalità su centraline elettroniche e componentistica, prima di inviarle per la sostituzione alla casa madre. Tramite il loro known-how e le mie competenze elettroniche, in questo periodo di studi universitari abbiamo messo a punto un semplice strumento di collaudo basato su un microcontrollore montato su PCB, completo di componentistica elettronica di interfaccia che è in grado di effettuare simulazioni ed analisi di funzionalità per alcuni componenti elettronici del motore: la piattaforma ESP32 si è prestata egregiamente ad essere utilizzata grazie alle sue caratteristiche hardware e software.
Il circuito sviluppato risponde agli standard e ai requisiti di progettazione che ci eravamo prefissi; in futuro potrà essere ulteriormente migliorato aggiungendo nuove funzionalità a livello software ed hardware, per l'interfacciamento con altri tipi di sensori ed attuatori.
Non si esclude di poter realizzare un prodotto industriale per la diagnosi automotive da proporre agli specialisti del settore in alternativa ai ben più costosi strumenti di diagnosi proprietari.

Per lo sviluppo del PCB è stato utilizzato il software di progettazione Fusion 360 di Autodesk, che al momento integra il precedente EAGLE in un software unico, includendo anche un CAD meccanico.

Indice Figure

Bibliografia

1. Datasheet Microcontrollore ESP32
   https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf
2. Datasheet modulo WROVER basato su Microcontrollore ESP32
   https://www.espressif.com/sites/default/files/documentation/esp32-wrover-e_esp32-wrover-ie_datasheet_en.pdf
3. Datasheet diodo Schottky B540C
   https://www.diodes.com/assets/Datasheets/ds13012.pdf
4. Datasheet Signal MOSFET AO3404
   http://www.aosmd.com/pdfs/datasheet/AO3404.pdf
5. Datasheet DC-DC Buck Converter TPS562211
   https://www.ti.com/lit/ds/symlink/tps562211.pdf?ts=1636731799907&ref_url=https%253A%252F%252Fwww.google.com%252F
6. Datasheet OPAMP LMV324 https://www.ti.com/lit/ds/symlink/lmv324.pdf?ts=1636705475018&ref_url=https %253A%252F%252Fwww.google.com%252F
7. Datasheet MOSFET-driver MPC1416
   https://ww1.microchip.com/downloads/en/DeviceDoc/20002092G.pdf
8. Datasheet Small Signal Transistor MMS8050
   https://www.mccsemi.com/pdf/Products/MMS8050(SOT-23).pdf
9. Datasheet Low-Dropout Regulator LDO
   https://www.onsemi.com/pdf/datasheet/ncp1117-d.pdf
10. Datasheet TVS-Diode
    https://m.littelfuse.com/~/media/electronics/datasheets/tvs_diodes/littelfuse_tvs_diode_smaj_datasheet.pdf.pdf
11. Datasheet fusibile resettabile
    https://images.chipyun.com/pdf/C960036_FB5256E25139FFC7A5799EEF89905C7D.pdf
12. Libreria Controllo Display https://github.com/Bodmer/TFT_eSPI
13. Datasheet HEXFET® Power MOSFET AUIRFS8403

    https://www.infineon.com/dgdl/auirfs8403.pdf?fileId=5546d462533600a4015355b6fc7a14db