**UNIVERSITÀ POLITECNICA DELLE MARCHE**
**FACOLTÀ DI INGEGNERIA**
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea in Ingegneria Elettronica


**TESI DI LAUREA**

# Sistema di riconoscimento di cavi/accessori connessi tramite connettori USB-C a sensori portatili basati su chip nRF52840

# Recognition system for cable/accessories connected via USB-C connectors to portable sensors based on the nRF52840 chip

Relatore

Prof. Giorgio Biagetti

Candidato

Tommaso Giacani

**ANNO ACCADEMICO 2021-2022**

*La tecnologia non tiene lontano l'uomo dai grandi problemi della natura,*
*ma lo costringe a studiarli più approfonditamente.*

Antoine De Saint-Exupèry

**Sommario**

Lo sviluppo del sistema embedded Grasense in grado di identificare i virus tramite un sensore al grafene necessita di una programmazione per il riconoscimento dei cavi e gli accessori visti in ingresso. Di seguito verrà posta l'attenzione sui metodi utilizzati per la distinzione di tali accessori e la programmazione effettuata.

**Keyword**: Sistemi Embedded, Microcontrollore, IDE, Resistenze di Pull-up e Pull-down, Circuito Equivalente, Macchina a stati

# Indice

# Elenco delle figure

# Elenco delle tabelle

# Introduzione

Dagli anni Sessanta ad oggi i sistemi embedded si sono sempre più diffusi nella vita di tutti i giorni. Senza che ce ne accorgessimo sono diventati parti integrante della quotidianità andando a far parte di tutti quei strumenti che utilizziamo ogni giorno. Dalla lavatrice al condizionatore, dalle apparecchiature sugli aerei agli optional nelle automobili sono incorporati per lavorare nascosti. Infatti, embedded in inglese significa "incorporato, incapsulato" e di fatto sono dei microcontrollori all'interno di questi strumenti che, tramite una specifica programmazione, riescono a controllarli. Di fatto il sistema embedded è un insieme di hardware e software che ha come miglior caratteristica la versatilità. Altri aspetti fondamentali sono la semplicità nel programmarli e il loro basso costo, che li ha resi più preferibili rispetto ai circuiti integrati che, sebbene siano più preformanti, sono più complessi nella programmazione e sopratutto molto più costosi.

Questa tecnologia è stata fondamentale anche durante la più recente pandemia, andando a far parte di tutti quei macchinari necessari al monitoraggio e alla cura dei pazienti. Ma il loro utilizzo negli ospedali non è la loro unica applicazione in campo medico. Di recente è stato pubblicato un progetto da Romagnoli *et al.* [2023] che parla dello sviluppo di un sistema embedded in grado di individuare la presenza di un virus all'interno di una soluzione tramite un sensore al grafene. Questo dispositivo è stato progettato per supportare diversi sensori, dunque la sfida proposta in questa tesi è quella di riuscire a fare la distinzione delle diverse periferiche collegate.

La tesi è composta da sei capitoli strutturati come di seguito specificato:

- Nel Capitolo 1 ci sarà un'introduzione sui dispositivi utilizzati nello studio, ai sistemi embedded e ai software utilizzati
- Nel Capitolo 2 si analizzeranno i circuiti equivalenti delle resistenze in ingresso al dispositivo nRF52840
- Nel Capitolo 3 verrà analizzato il programma scritto in C fatto sull'IDE Eclipse per la distinzione dei dispositivi visti in ingresso.
- Nel Capitolo 4 verranno analizzate le resistenze interne al dispositivo, andando a descrivere il codice utilizzato.

- Nel Capitolo 5 verranno riportati i dati raccolti dalle misurazioni effettuate.
- Nel Capitolo 6 verranno tratte le conclusioni e verranno discussi alcuni possibili sviluppi futuri del sistema embedded realizzato.

# CAPITOLO 1

# Il dispositivo nRF52840

In seguito verranno analizzati tutti i dispositivi e software utilizzati per realizzare il sistema embedded capace di riconoscere i cavi e gli accessori. L'analisi verrà fatta sulle proprietà tecniche e applicative del microcontrollore nRF52840 e dei cavi USB-c. Il sistema embedded è stato realizzato tramite l'utilizzo dell'IDE[1] Eclipse®e i circuiti analizzati con l'ausilio dei software LTspice®XVII e MATLAB®

## 1.1 Microcontrollore nRF52840

Il microcontrollore nRF52840 (Figura1.1) fa parte della famiglia degli nRF52 ed è un sistema wireless su chip (SoC[2]) che lavora a bassissima potenza da 2,4 GHz. Al suo interno è presente una CPU Cortex-M4F, un ricetrasmettitore multi-protocollo da 2,4 GHz e una memoria flash.

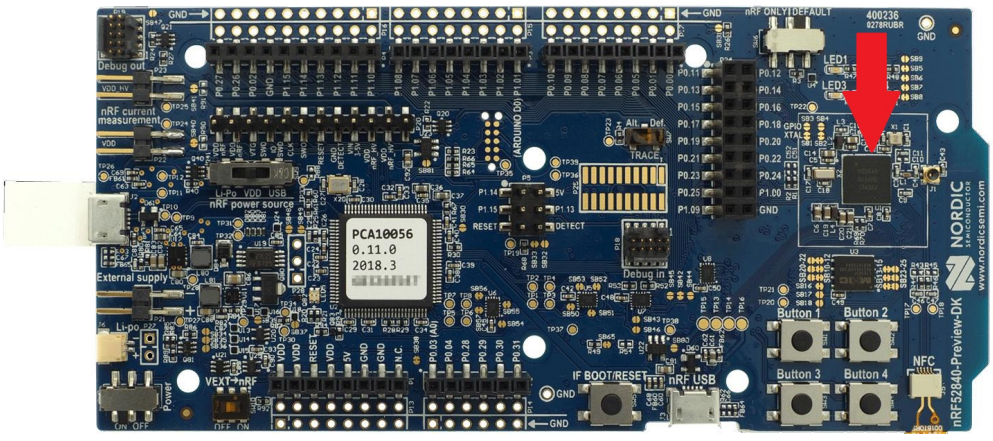


**Figura 1.1:** Microcontrollore nRF52840, foto presa dalla Nordic Semiconductor®

[1]integrated development environment, ovvero ambiente di sviluppo integrato
[2]System on chip

### 1.1.1 Caratteristiche del nRF52840

Il dispositivo ha le seguenti caratteristiche tecniche:

- Ricetrasmettitore bluetooth 5, IEEE 802.15.4-2006, 2.4 GHz con:
  - -95 dBm di sensibilità in 1 Mbps e -103 dBm a 125 kbps a low energy
  - Compatibilità con i dispositivi della serie nRF52, nRF51, nRF24L e nRF24AP
  - Velocità di trasmissione dati supportate di 2 Mbps, 1 Mbps, 500 kbps e 125 kbps
  - Antenna di output a singola uscita
  - corrente massima di trasmissione a 4,8 mA e di ricezione a 4,6 mA
- 1 MB di memoria flash e 256 kB di memoria RAM
- processore ARM Cortex-M4®32-bit con FPU a 64 MHz
- Gestione flessibile della potenza di alimentazione
- Interfacce avanzate sul chip come:
  - USB 2.0 full speed a 12 MBps
  - interfaccia QSPI a 32 MHz
  - SPI ad alta velocità a 32 MHz
  - 48 pin di I/O a general purpose
  - interconnessione periferica programmabile (PPI[3])

### 1.1.2 Possibilità di utilizzo

Il microcontrollore nRF52840 può essere utilizzato per molte applicazioni differenti che fanno soprattutto uso della connessione bluetooth. Alcuni esempi possono essere le periferiche I/O di computer oppure i sensori di movimento o per la domotica. Gli utilizzi variano molto passando dal settore automobilistico, per la realizzazione di radar che comunicano con l'autovettura, alla realtà virtuale dove viene utilizzato come mezzo di comunicazione tra i vari dispositivi collegati.

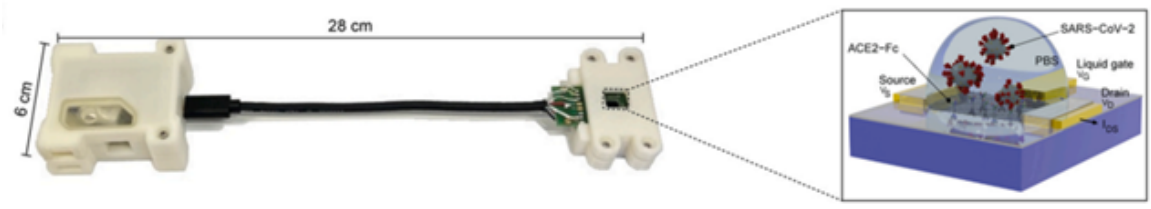


**Figura 1.2:** il dispositivo Grasense collegato con rilevatore SARS-COV-2, immagine presa da Romagnoli *et al.* [2023]

[3]Programmable peripheral interconnect

Nei prossimi capitoli lo studio che verrà condotto per il riconoscimento di cavi e accessori in ingresso al microcontrollore nRF52840 verrà applicato a un prototipo, in figura 1.2. Tale prototipo permette di riconoscere, tramite opportuni sensori intercambiabili collegati al microcontrollore, se in una certa quantità di soluzione sono presenti o meno dei virus. Il transistor ad effetto di campo in grafene o gFET[4] del sensore invia dei dati al dispositivo che a sua volta li trasmette tramite bluetooth low energy al computer. Il dispositivo è alimentato a batterie e può essere trasportato facilmente date le dimensioni ridotte, questo facilita il lavoro degli operatori che dovranno utilizzarlo.

## 1.2 USB-C

L'USB-C è (Figura 1.3) il più recente tipo di connettore per la trasmissione di dati e l'alimentazione elettrica che offre una serie di vantaggi fra cui la reversibilità, ovvero la possibilità di collegare il connettore indipendentemente dal verso.

**Figura 1.3:** USB-C


**Figura 1.4:** Confronto tra i due tipi di USB, immagini prese da Wikipedia

La nomenclatura tipo C indica la tipologia di connettore, ne esistono diverse versioni che vengono suddivise per la velocità con cui trasmettono i dati come mostrato nella tabella 1.1.

| Tipologia | velocità di trasmissione |
|---|---|
| 2.0 | 480 Mbit/s |
| 3.0 | 5 Gbit/s |
| 3.1 | 10 Gbit/s |
| thunderbolt | 40 Gbit/s |

**Tabella 1.1:** Le differenti tipologie di USB-C con relativa velocità di trasmissione

Queste elevate velocità di trasmissione insieme alla flessibilità di applicazione e le dimensioni ridotte permettono alle USB-C di essere utilizzate praticamente in ogni dispositivo, sia mobile che fisso.

[4] graphene field-effect transistor

### 1.2.1 Pin di porta

La USB-C ha ventiquattro pin che sono suddivisi in due serie A e B da dodici ciascuno chiamati $A_1, \ldots, A_{12}$ e $B_1, \ldots, B_{12}$. Gli schemi con cui sono disposti i pin nella parte A e nella parte B sono opposti tra loro rispetto all'asse di simmetria per mantenere la reversibilità. Nella presa (figura 1.5) sono presenti due coppie di pin D+ e D-, mentre nella spina, figura 1.6, è presente solo una coppia. Questa differenza nella disposizione dei pin è fatta per riuscire a distinguere il verso del collegamento in quanto il sistema riesce a riconoscere dove è presente una differenza di potenziale.

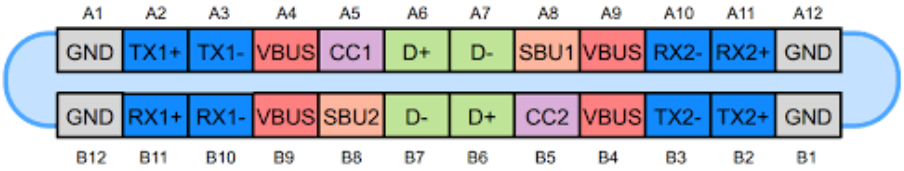


**Figura 1.5:** I pin della presa del connettore USB-C

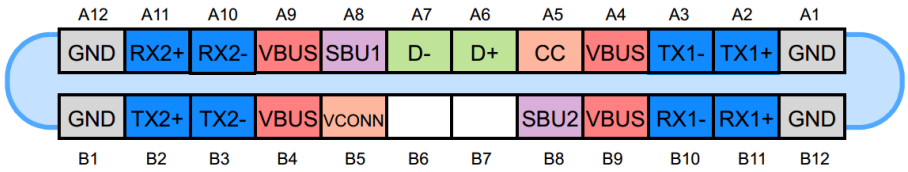


**Figura 1.6:** I pin della spina del connettore USB-C, immagini prese da Microchip develop helper

Ognuno dei 12 pin ha una funzione che viene descritta come segue:

- Pin GND è la massa.
- Pin TX1+ e TX1- (o TX2+ e TX2-) sono una coppia di differenziali che servono per la trasmissione.
- Pin VBUS serve per il trasporto di potenza e la tensione predefinita è 5 V. VBUS può avere una tensione fino a 20 V grazie al Power Delivery. Si può aumentare la corrente massima fino a 5 A per avere una potenza in uscita di 100 W[5].
- Pin CC1 (o CC2) serve per la configurazione del canale. Oltre a rilevare l'orientamento del connettore, serve anche come funzione di rilevamento del collegamento o della rimozione. Come si può vedere nella figura 1.7, il collegamento che si viene a formare tra CC1 della spina e CC1 della presa oppure tra CC2 della spina e CC1 della presa, dà luogo a un solo percorso di corrente alla volta che serve a distinguerne il verso.
- Pin D+ e D- vengono utilizzati per la connettività e sono le coppie differenziali.

[5]Può essere utile quando si vuole utilizzare il collettore come caricatore di un dispositivo di grandi dimensioni

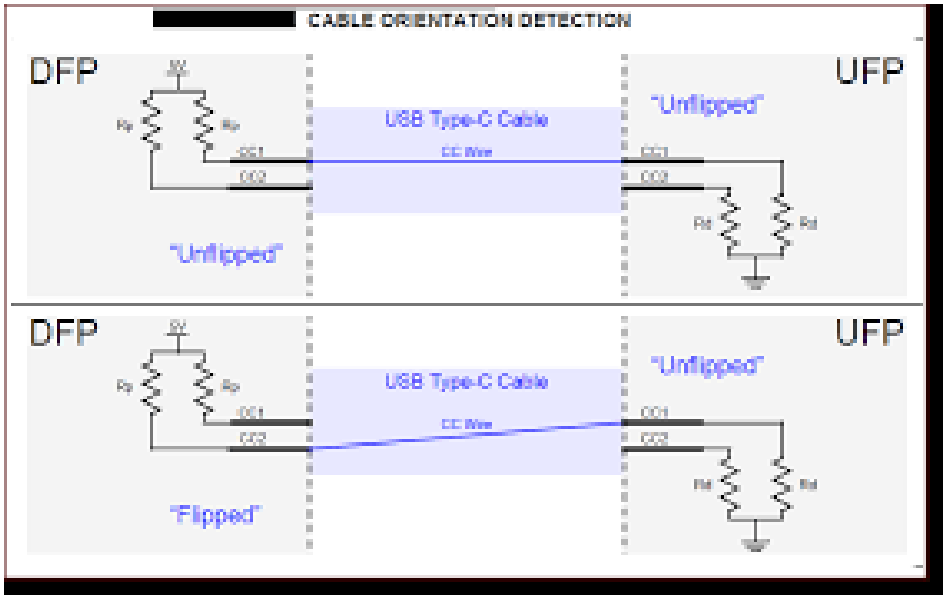


**Figura 1.7:** Collegamento tra DFP e UFP, immagine presa da Microchip develop helper

- Pin SBU1 (o SBU2) viene utilizzato solo in modalità alternativa come percorso per segnali a bassa velocità.
- Pin RX1+ e RX1- (o RX2+ e RX2-) sono una coppia di differenziali che servono per la ricezione.

Per lo studio preso in considerazione, il connettore USB-C non è stato utilizzato in modo standard, bensì per fare la connessione con degli oggetti personalizzati di diverso tipo, tra cui dei sensori al grafene. Questi dispositivi connessi possono essere distinti in base al valore di un resistore collegato tra il pin CC1 (o CC2) e il pin VBUS (o GND).

## 1.3 Sistemi embedded

I sistemi embedded sono sistemi elettronici di elaborazione basati su microprocessore che servono alla realizzazione di uno scopo preciso, sono quindi detti *special purpose*. Questa è la caratteristica principale che li distingue da quelli progettati per realizzare una elaborazione più generica ovvero a *general purpose*, cioè l'impossibilità di essere riprogrammati dagli utenti per svolgere lavori diversi da quelli per cui sono stati progettati. Questi sistemi non si limitano a fornire un output bensì si approcciano al mondo fisico andando ad interagire con esso.

Le possibilità di applicazione sono innumerevoli e molto differenti tra loro, questo a fatto sì che il loro utilizzo si è ampiamente diffuso sin dalla nascita nel 1971. Oggigiorno la maggior parte dei dispositivi elettronici ha un sistema embedded integrato dato il loro basso costo e la facilità di implementazione rispetto ai circuiti integrati personalizzati. Un aspetto molto importante dei sistemi embedded è la flessibilità in quanto diversi microcontrollori possono essere utilizzati per diverse applicazioni in base alla programmazione effettuata.


**Figura 1.8:** Immagine presa da Informatica e ingegneria online

I sistemi embedded possiedono delle caratteristiche prestazionali e architetturali,

che li differenziano in base all'ambito in cui vengono utilizzati. La maggior parte delle caratteristiche prestazionali vengono regolate dal mercato e dall'utilizzo, mentre quelle architetturali sono regolate dalle specifiche tecniche e dalle dimensioni. A livello esemplificativo sono queste le caratteristiche che devono essere realizzate;

- Prestazioni: i parametri variano in base all'applicazione e generalmente sono il tempo di gestione e di reazione ad un determinato tipo di evento che si verifica
- Volume: i volumi di produzione previsti in fase di progettazione
- Peso e dimensioni: quanto spazio occupa il sistema; rappresenta un fattore determinante per i dispositivi mobili
- Dimensione del codice: dato che il software risiede in un supporto di memoria permanente all'interno del chip del microprocessore, le dimensioni tendono ad essere le più ridotte possibile
- Consumo energetico: particolarmente importante nelle applicazioni che utilizzano le batterie
- Costo: intrinsecamente connesso a tutti gli altri parametri

L'importanza dei fattori varia in base all'utilizzo, i monitor da gaming premieranno sistemi con le prestazioni e il consumo energetico migliore, mentre gli accessori all'interno di un'automobile ricercherà sistemi embedded con un costo e un volume minore.

Solitamente le caratteristiche architetturali sono:

- Affidabilità: determinata dopo una serie di controlli per analizzare i potenziali rischi
- Manutenibilità: determina con quale probabilità il circuito deve essere riparato o sostituito dopo un certo periodo
- Disponibilità: legata ad affidabilità e manutenibilità, determina la probabilità che il sistema funzioni
- Sicurezza: capacità di resistere alle violazioni di utilizzo
- Time-to-market: tempo necessario affinché il prodotto venga venduto una volta introdotto nel mercato
- Interfacce di comunicazione: la complessità dipende dal costo del sistema
- Tempo reale: la progettazione del sistema affinché possa operare in base a determinati parametri temporali
- Interfacce utente: variano in base al compito che deve svolgere
- Safety: esistono vari livelli e sono legati alla possibilità di provocare danni in caso di malfunzionamento

Queste proprietà servono a definire un sistema embedded, ma è necessario fare una considerazione anche dal punto di vista della programmazione software. Si diversifica dalla normale programmazione per computer in quanto il sistema deve interagire con il mondo fisico e di conseguenza c'è un legame più forte tra parte software e hardware. Inoltre il microprocessore è meno performante rispetto a quello di un PC e quindi bisogna avere una conoscenza più approfondita sulle sue capacità in modo da sfruttarle al meglio. Il codice del software dovrà essere efficiente e robusto così da lavorare indipendentemente dall'ausilio di un programmatore, ma anche riutilizzabile così da poter essere applicato su diversi hardware.

## 1.4 Software utilizzati

### 1.4.1 Eclipse

Eclipse è un ambiente di sviluppo integrato multi-linguaggio e multipiattaforma *open source* e può essere utilizzato per la programmazione di vari software diversi. L'IDE è progettato per la Java, ma può essere utilizzato per altri linguaggi come il C/C++, utilizzato per la programmazione del sistema embedded preso di questa tesi.

La piattaforma è composta da un'insieme di plug-in che descrivono le funzioni di Eclipse ed è possibile aggiungerli da un apposito store, in quanto plug-in, per implementarne delle altre. Quella utilizzata per la programmazione in C è Eclipse Kepler for C/C++ developers, mentre quello per la comunicazione con il microcontrollore nRF52840 è GNU Tools for ARM Embedded Processors.

### 1.4.2 LTspice XVII

LTspice è un software per la simulazione di circuiti elettronici analogici basato su SPICE[6] e il XVII è l'ultima versione rilasciata. Questo software è stato utilizzato per la simulazione dei circuiti che si sono formati nel collegamento del connettore USB-C con l'ingresso del microcontrollore nRF52840.

### 1.4.3 MATLAB

MATALB è un ambiente di calcolo numerico e analisi statistica realizzato in C. Oltre ad essere un ottimo ausilio per il calcolo e la risoluzione delle equazioni, è stato indispensabile per la possibilità di realizzare grafici di funzioni per uno studio più accurato sui parametri dei circuiti.

[6]Simulation Program with Integrated Circuit Emphasis

# CAPITOLO 2

# Studio del dispositivo nRF52840

*Lo scopo principale dello studio è la distinzione dei dispositivi collegati al microcontrollore nRF52840. Tale distinzione viene fatta tramite il convertitore analogico digitale, chiamato ADC, presente all'interno del microcontrollore che permette di stimare il valore delle resistenze con una precisione tale da poter distinguere i diversi dispositivi. Il microcontrollore della Nordic Semiconductor®ha questo tipo di convertitore all'interno del dispositivo stesso e ciò ha molto semplificato lo studio. Non è stato necessario l'implementazione di un dispositivo esterno per le misurazioni, permettendo così di evitare tutti i problemi relativi all'incompatibilità dei diversi dispositivi o la difficile lettura dei dati per via di diverse forme di linguaggio.*

## 2.1 Circuiti equivalenti delle resistenze in ingresso

I dispositivi in ingresso, come mostrato in figura2.1 al microcontrollore nRF52840 possono essere visti come delle resistenze in ingresso alle porte del dispositivo. In base al collegamento che hanno i differenti cavi con i pin presenti nella porta USB si vengono a creare diverse configurazioni circuitali. Diventa necessario separare e distinguere queste tipologie di collegamento in varie casistiche. La divisione è fatta in base a come la resistenza equivalente del dispositivo collega tra loro i pin CC1 o CC2 con il pin VBUS. Se è presente il collegamento tra il pin CC1 oppure CC2 e la massa allora si può parlare di resistenza di pull-down; altrimenti se il collegamento avviene con VBUS al posto della massa allora si parla di resistenza pull-up.

In base al voltaggio dei pin di porta del microcontrollore, che viene registrato dall'ADC, è possibile distinguere i diversi dispositivi che vengono collegati identificando sia il tipo di collegamento che il valore della resistenza. Dal momento che la precisione dell'ADC presente all'interno del microcontrollore è del millivolt non è possibile stabilire un valore estremamente preciso di queste resistenze. Infatti, la tensione di alimentazione (indicata successivamente con Vdd) è circa 1,8 V, mentre le correnti che circolano nel dispositivo sono nell'ordine del microampere. Considerato che le resistenze che interfacciano il dispositivo sono dell'ordine dei kiloohm è difficile avere un valore preciso all'ohm delle resistenze per via della elevata differenza degli ordini di grandezza. Fortunatamente l'obiettivo dello studio è quello del riconoscimento dei dispositivi visti in ingresso, e la precisione riscontrata dall'applicazione

**Figura 2.1:** Il dispositivo Grasense collegato con il cavo USB-C

delle equazioni, che verranno mostrate successivamente, è tale da permettere questa distinzione.

Per trovare i valori delle resistenze viste in ingresso sono state risolte le equazioni dei relativi circuiti equivalenti applicando le KVL[1] e le KCL[2]. Tramite il software MATLAB sono state risolte le equazioni applicati ai circuiti equivalenti, mentre la verifica di quest'ultime è stata fatta simulando i circuiti tramite LTspice XVII.

I casi simulati sono stati in totale nove.

1. Resistenza pull-down CC1 (Figura2.2)
   - $Idc1 = (Vdd - V\_cc1\_pu)/Rpuc1$
   - $I2c1 = V\_cc1\_pu/R2c1$
   - $I1c1 = Idc1 - I2c1$
   - $R\_cc1\_pd = (V\_cc1\_pu - (R1c1 \cdot I1c1))/I1c1$
2. Resistenza pull-down CC2 (Figura2.3)
   - $Idc2 = (Vdd - V\_cc2\_pu)/Rpuc2$
   - $I2c2 = V\_cc2\_pu/R2c2$
   - $I1c2 = Idc2 - I2c2$
   - $R\_cc2\_pd = (V\_cc2\_pu - (R1c2 * I1c2))/I1c2$
3. Resistenza di pull-up CC1 (Figura2.3)

[1] Legge di Kirchhoff delle tensioni
[2] Legge di Kirchhoff delle correnti

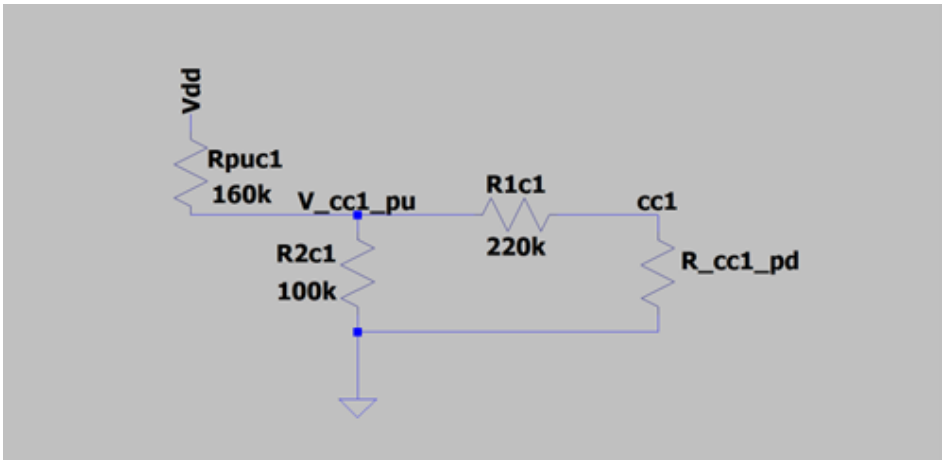


**Figura 2.2:** Pull-down CC1

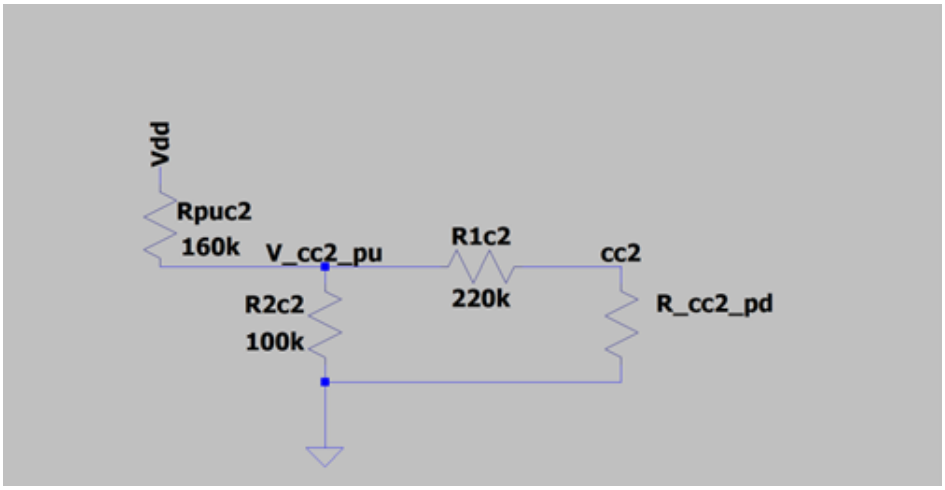


**Figura 2.3:** Pull-down CC2

- $R\_cc1\_pu = ((V\_vbus / V\_cc1\_np) \cdot R2c1) - R1b - R1c1 - R2c1$

4. Resistenza di pull-up CC2 (Figura2.3)
   - $R\_cc2\_pu = ((V\_vbus / V\_cc2\_np) * R2c2) - R1b - R1c2 - R2c2$
5. Doppia resistenza di pull-up (Figura2.6)
   - $Ic1 = V\_cc1\_np / R2c1$
   - $Ic2 = V\_cc2\_np / R2c2$
   - $Ir1b = Ic1 + Ic2$
   - $V\_vbus = Vdd'$[3] $-(Ir1b \cdot R1b)$
   - $R\_cc1\_pu = (V\_vbus \cdot R2c1) / V\_cc1\_np - R2c1 - R1c1$
   - $R\_cc2\_pu = (V\_vbus \cdot R2c2) / V\_cc2\_np - R2c2 - R1c2$

[3] In realtà non è propriamente il valore in tensione dell'alimentazione ma siccome la resistenza *Ron* è molto piccolo in confronto alle altre, la differenza di tensione ai capi di quest'ultima è trascurabile

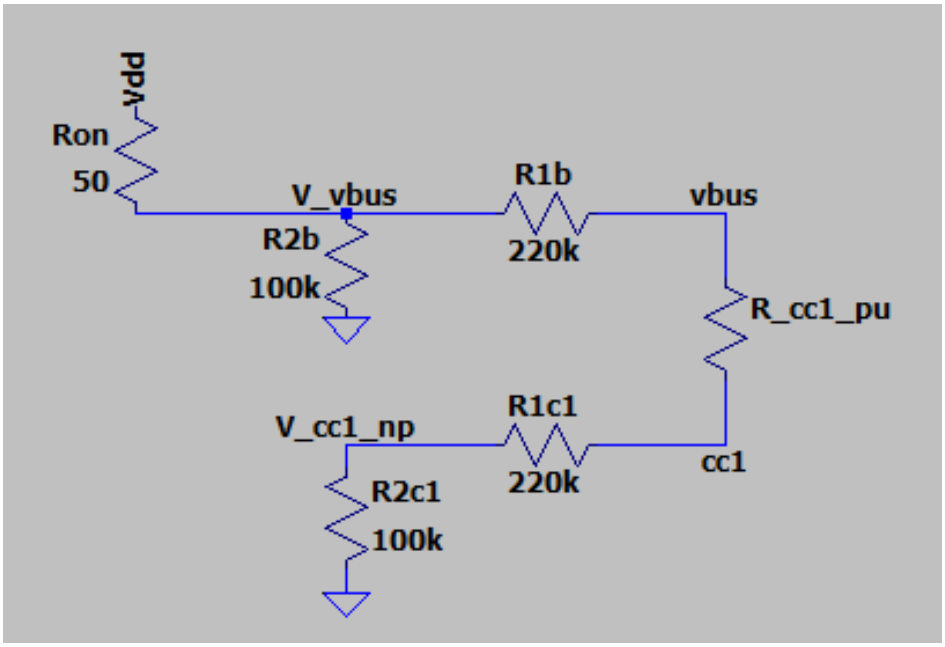


**Figura 2.4:** Pull-up CC1

6. Resistenza di pull-down CC1 con generatore esterno (Figura2.7)
   - $Idc1 = (Vdd - V\_cc1\_pu)/Rpuc1$
   - $I2c1 = V\_cc1\_pu/R2c1$
   - $I1c1 = Idc1 - I2c1$
   - $R\_cc1\_pd = (V\_cc1\_pu - (R1c1 \cdot I1c1))/I1c1$
7. Resistenza di pull-down CC2 con generatore esterno (Figura2.8)
   - $Idc2 = (Vdd - V\_cc2\_pu)/Rpuc2$
   - $I2c2 = V\_cc2\_pu/R2c2$
   - $I1c2 = Idc2 - I2c2$
   - $R_cc2_pd = (V\_cc2\_pu - (R1c2 * I1c2))/I1c2$
8. Resistenza di pull-up CC1 con generatore esterno (Figura2.9)
   - $Iry1 = V\_cc1\_np/R2c1$
   - $Vcc1$[4]$= R1c1 \cdot Iry1 + V\_cc1\_np$
   - $R\_cc1\_pu = (V\_ext - Vcc1)/Iry1$
9. Resistenza di pull-up CC2 con generatore esterno (Figura2.10)
   - $Iry2 = V\_cc2\_np/R2c2$
   - $Vcc2$[5]$= R1c2 \cdot Iry2 + V\_cc2\_np$
   - $R\_cc2\_pu = (V\_ext - Vcc2)/Iry2$

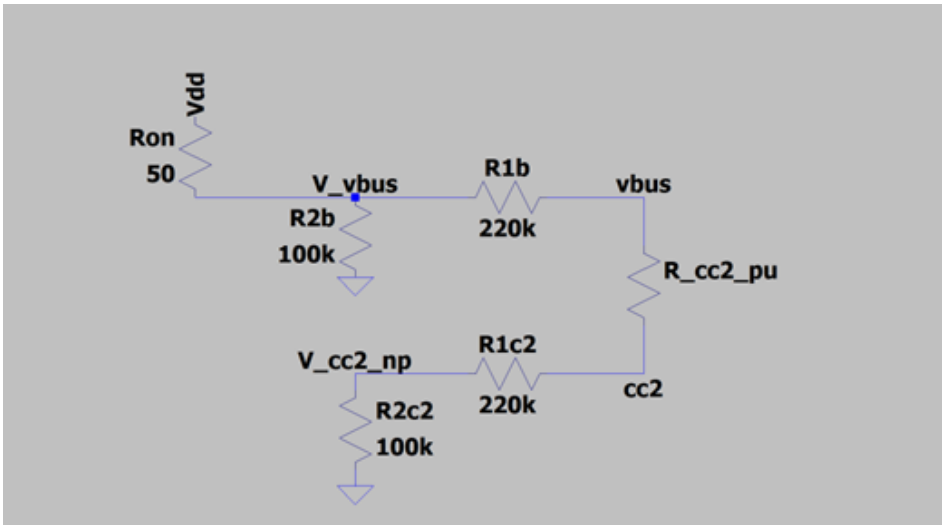


**Figura 2.5:** Pull-up CC2

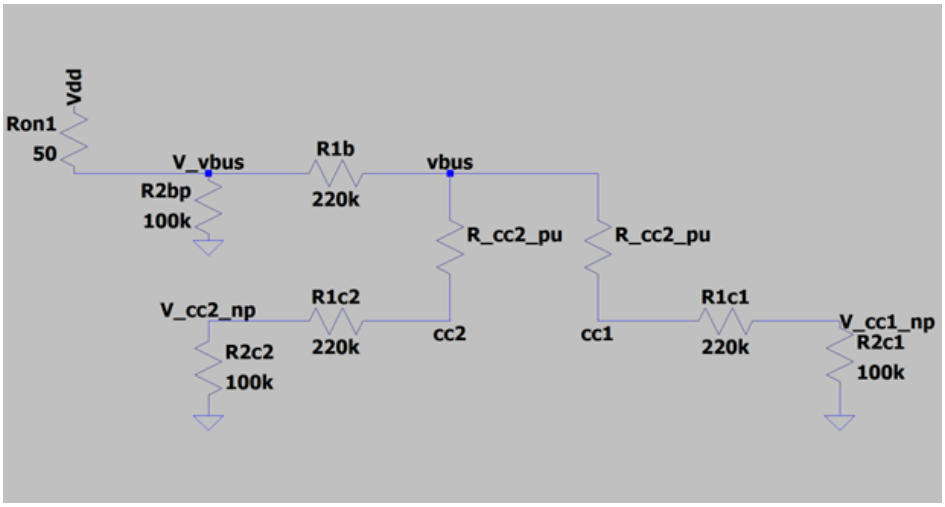


**Figura 2.6:** Doppio pull-up

Tutte le figure sono state realizzate tramite il software LTspice®XVII

### 2.1.1 Considerazioni sui casi trattati

Possiamo osservare che i casi 2.2 e 2.7 e i casi 2.3 e 2.8 sono uguali tra loro. Ciò si verifica perché il generatore esterno che si va ad interfacciare con il pin VBUS non è collegato, tramite la resistenza esterna, al pin CC1 (o CC2) e di conseguenza non va ad influire sulle tensioni riportate dall'ADC.

In tutti e 4 i casi sopra citati sono state svolte le seguenti equazioni per trovare il valore della resistenza esterna di pull-down. Per primo è stata trovata la corrente che scorre su $Rpuc1$ e poi quella su $R2c1$, poi tramite la KCL al nodo $V\_cc1\_pu$ è stata trovata la corrente che scorre sulla resistenza esterna. Infine, tramite la KVL alla

[4]Tensione su cc1
[5]Tensione su $cc2$

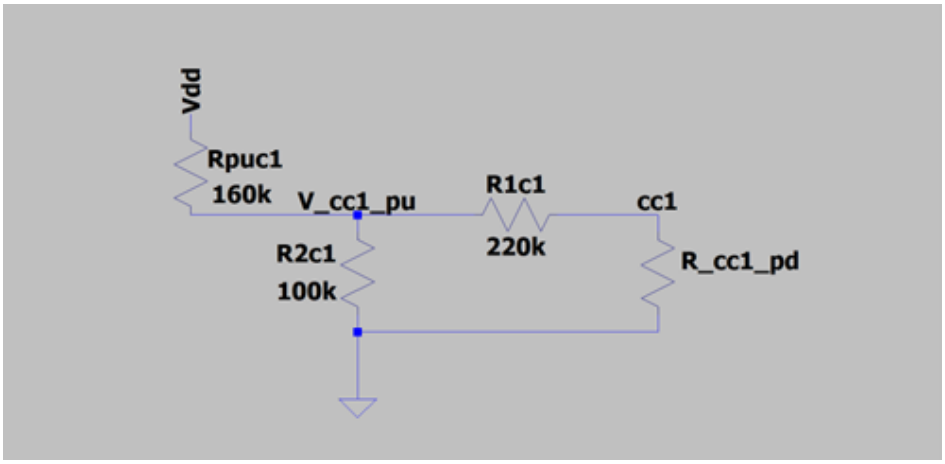


**Figura 2.7:** Pull-down CC1 con generatore esterno

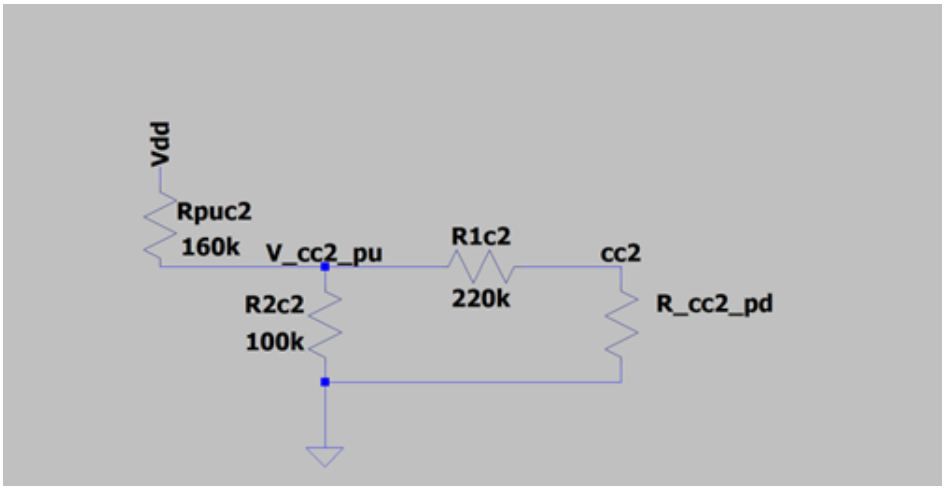


**Figura 2.8:** Pull-down CC2 con generatore esterno

maglia, è stato trovato il valore della resistenza. Gli stessi calcoli sono stati svolti per il pin CC2, con i componenti appropriati.

Il caso del pull-up per CC1 2.4 (o CC2 2.4) è stato il più semplice da analizzare, in quanto è bastato considerare il partitore di tensione a più resistenze venutosi a formare tra il nodo $V\_vbus$ e $V\_cc1\_np$ (o $V\_cc2\_np$) .

Non si può dire altrettanto per il caso del doppio pull-up 2.6, dove l'analisi ha richiesto più passaggi e come si vedrà successivamente più condizioni da rispettare. Per prima cosa sono state trovate le correnti che scorrono sulla resistenza $R2c1$ e $R2c2$. Tramite la KCL al nodo $vbus$ è stata trovata la corrente che scorre su $R1b$ così da ottenere la tensione presente sul nodo stesso applicando l'equazione per la differenza di tensione ai capi di una resistenza. Infine, tramite un partitore di tensione a più resistenze sono state trovare le resistenze di pull-up di Cc1 e di CC2.

Nell'ultimo caso considerato, la del pull-up con generatore esterno per CC1 2.9 (o CC2 2.10 , è stato trovato il valore della corrente che scorre su $R2c1$ per trovare la caduta di tensione su $cc1$. Tramite quest'ultima è stata trovata la resistenza di pull-up con l'equazione inversa della differenza di tensione ai capi della resistenza. Anche questa volta gli stessi calcoli sono stati svolti per il pin CC2, con i componenti

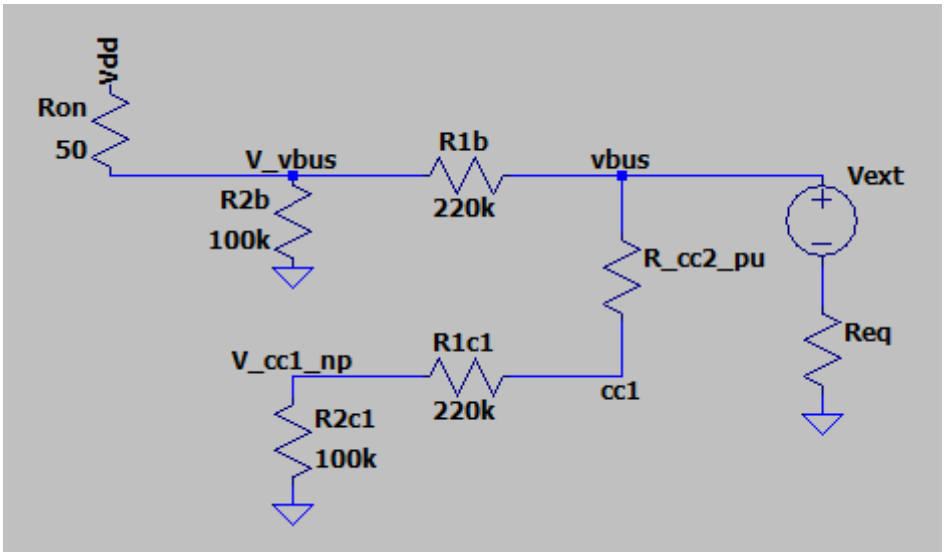


**Figura 2.9:** Pull-up CC1 con generatore esterno

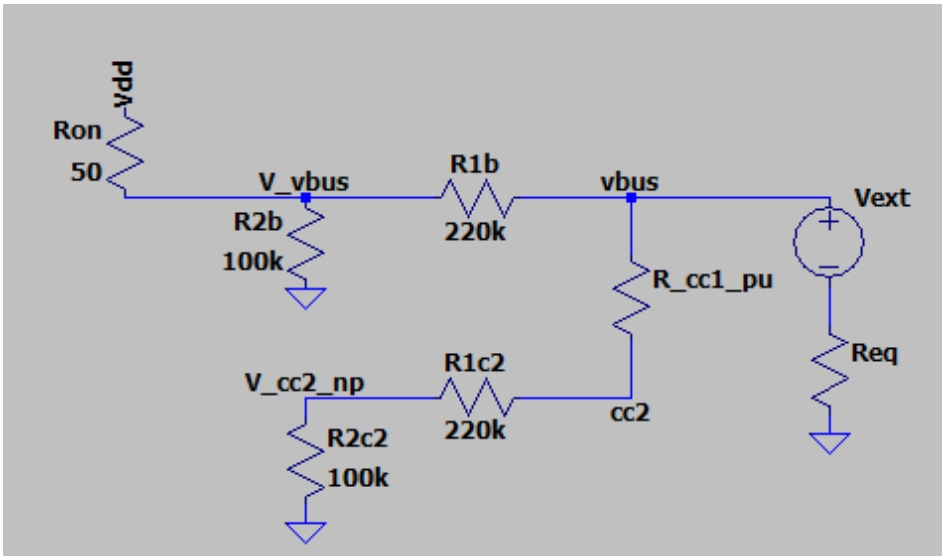


**Figura 2.10:** Pull-up CC2 con generatore esterno

appropriati.

### 2.1.2 Casistiche non realizzabili

Non viene preso in considerazione il caso del doppio pull-down ovvero quando i pin CC1 e CC2 presentano una resistenza di pull-down simultaneamente. Le resistenze sono tra loro indipendenti perché sono collegate alla resistenza di pull-up interna all'alimentazione propria del pin e di conseguenza non ha senso considerarle come un caso a parte perché studiabili separatamente come pull-down di CC1(2.2) e pull-down di CC2(2.3). Ciò non accade per il doppio pull-up 2.6 dove il collegamento delle resistenze avviene tra CC1 e VBUS e tra CC2 e VBUS. Siccome VBUS è un pin comune, è necessario studiare i due casi nello stesso momento perché il collegamen-

to simultaneo al pin comporta un abbassamento dei livelli massimi delle tensioni registrate da ADC.

Infine, non viene considerato nemmeno il caso del doppio pull-down con generatore esterno per i motivi sopra citati e nemmeno il caso del doppio pull-up con generatore esterno. In quest'ultimo è il generatore esterno che, con ipotesi che abbia un voltaggio impattante, permette di considerare i due casi indipendenti. Inoltre, non è più necessario considerare la tensione portata dall'alimentazione perché nel nodo che si viene a formare con il collegamento del pin VBUS è presente la tensione del generatore. Questo fa si che restante parte del circuito può essere considerata come un semplice partitore di tensione che parte da *vbus* e passa per *cc*1 (o *cc*2), poi per $V\_cc1\_np$ (o $V\_cc2\_np$) ed infine a massa.

CAPITOLO 3

# Programmazione della funzione che permette di distinguere le resistenze in ingresso

*In questo capitolo verrà analizzata la funzione che servirà al microcontrollore nRF53840 per riconoscere cavi e accessori visti in ingresso. Per farlo sarà necessario riconoscere il tipo di connessione e il valore della resistenza. L'attenzione verrà posta sulle variabili utilizzare e soprattutto sulla logica applicata per la distinzione dei casi.*

## 3.1 Programmazione

La programmazione della funzione che permette di distinguere le resistenze in ingresso è stata fatta sull'IDE Eclipse tramite linguaggio C. Il programma completo si chiama "grasense-fw" e comprende al suo interno i file in C: ble_bas; ble_measure; ble_system; bluetooth; converters; build_date; main; measure; power; usb. Sono presenti anche i file in H: ble_measure; ble_system; bluetooth; BQ25155; converters; custom_board; i2c; measure; power; usb.

La funzione realizzata, chiamata "find_resistance", è stata inserita all'interno di power.c e inizializzata all'interno di power.h, dove sono presenti le altre funzioni che lavorano con l'ADC. La chiamata della funzione viene fatta all'interno di "compute_power_status" dove è presente la lettura dei valori delle tensioni ai pin CC1, CC2 e VBUS ed infine l'invio dei dati tramite bluetooth.

## 3.2 Logica applicata per la distinzione dei casi

Le varie tipologie di connessioni si differenziano, a livello logico, in base al valore registrato dall'ADC. Durante lo studio dei circuiti, sono stati registrati i valori limite per cui si viene a creare un circuito aperto o un corto circuito al posto della resistenza di pull-up o pull-down. Tramite questi valori si identifica un range di tensioni entro il quale è possibile riconoscere le tipologie di resistenze e, tramite la risoluzione delle opportune equazioni, il loro valore.

La prima suddivisione per l'identificazione consiste nel verificare o meno la presenza del generatore esterno, poi seguono in *parallelo*[1] le condizioni per CC1 e CC2. Le condizioni servono per identificare la presenza delle resistenze di pull-up o pull-down nei rispettivi pin se questo non avviene significa che c'è un'errore nella misurazione.


**Figura 3.1:** Flow chart realizzato con https://app.diagrams.net/

Quando viene identificata una certa resistenza si applicano le equazioni relative al caso e si trova il risultato e poiché non è possibile avere contemporaneamente sia pull-up che pull-down nello stesso pin, viene assegnato il valore *INFINITY*[2] alla resistenza non identificata per rappresentarne l'assenza. Nel caso non sia collegato nulla al microcontrollore, viene registrato come una resistenza di pull-down per entrambi i pin e questa avrà un valore che tende all'infinito. Nel caso di errore di misurazione il programma non si blocca ma viene comunque assegnato il valore *NAN*[3] alla variabile contenete la resistenza.

[1] Le condizioni vengono verificate nello stesso momento

[2] Non rappresenta l'infinito ma il valore massimo raggiungibile dal tipo di variabile ed ha un carattere speciale apposito

[3] Not a number e viene rappresentato tramite un carattere speciale per rappresentare l'assenza di un numero e l'utilizzo della variabile

## 3.3 Descrizione della funzione

### 3.3.1 Variabili utilizzate

Nella prima parte della funzione sono presenti le inizializzazioni di tutte le variabili necessarie per la risoluzione dei circuiti equivalenti delle connessioni viste in ingresso.

```
1046        // cc1
1047        //float R1c1 = 220000;
1048        //float R2c1 = 100000;
1049        //float Rpuc1 = 160000;
1050        //float Rpdc1 = 160000;
1051        // cc2
1052        //float R1c2 = 220000;
1053        //float R2c2 = 100000;
1054        //float Rpuc2 = 160000;
1055        //float Rpdc2 = 160000;
1056        // v_bus
1057        //float R1b = 220000;
1058        //float R2b = 100000;
1059        //float Rpub = 160000;
1060        //float Rpdb = 160000;
1061        // tensions and currents
1062        float Vdd = 1.8;
1063        float Vext_pu;
1064        float Ir1b;
1065        float Iry1;
1066        float Iry2;
1067        float Vcc1;
1068        float Vcc2;
1069        float Vext;
1070        float Vbus;
1071        float Idc1;
1072        float Idc2;
1073        float I2c1;
1074        float I1c1;
1075        float I2c2;
1076        float I1c2;
1077        float Ic1;
1078        float Ic2;
1079        //for the calculation
1080        float R_cc1_pu;
1081        float R_cc1_pd;
1082        float R_cc2_pu;
1083        float R_cc2_pd;
1084        float V_ext;
1085        // full scale
1086        float V_fs = Vdd;
1087        float N_fs = 1 << 16;
1088        // higher value possible for v_cc1_pu and v_cc2_pu
1089        float v_max_cc1_pu;
1090        float v_max_cc2_pu;
1091        /*
```

**Figura 3.2:** Le variabili utilizzate

Le variabili utilizzate sono le seguenti:

- le variabili da $R1c1$ a $Rpdb$ sono i valori ideali delle resistenze interne del microcontrollore usati inizialmente per fare i calcoli
- le variabili da $Vdd$ a $Ic2$ contengono i valori delle tensioni e delle correnti risultanti dalle applicazioni delle equazioni per la risoluzione dei circuiti
- le variabili da $R\_cc1\_pu$ a $V\_ext$ servono per conservare temporaneamente i valori delle resistenze da trovare
- le variabili $V\_fs$ e $N\_fs$ sono rispettivamente il fondo scala analogico e il fondo scala digitale e servono per la normalizzazione dei valori registrati dall'ADC

- le variabili *v_max_cc*1*_pu* e *v_max_cc*2*_pu* contengono i valori che fanno da limite per la condizione del caso del pull-up con generatore esterno

Le prime variabili contenenti i valori ideali delle resistenze interne sono state *commentate* perché, come si vedrà più nel dettaglio nel capitolo 4, verranno sostituite dai valori reali delle resistenze interne calcolate da un'apposita funzione chiamata "find_resistor_divider". Tale funzione elabora i dati forniti dalle misurazioni fatte su measure.c e li immagazzina in una variabile esterna struct chiamata *internal_resistance*.

```
struct sys_cable_report
{
    uint16_t v_cc1_pu;
    uint16_t v_cc2_pu;
    uint16_t v_bus_pu;
    uint16_t v_bus_np;
    uint16_t v_cc1_np; // measured with v_bus_hi
    uint16_t v_cc2_np; // measured with v_bus_hi
} static cable_report;

//struct for the value of the external resistances
struct sys_cable_resistance
{
  uint16_t r_cc1_pu;
  uint16_t r_cc1_pd;
  uint16_t r_cc2_pu;
  uint16_t r_cc2_pd;
  uint16_t v_ext; // voltage value
} static cable_resistance;
```

**Figura 3.3:** Le variabili esterne cable_report e cable_resistance

Infine vengono utilizzate le variabili esterne *cable_report* e *cable_resistance*, entrambe delle struct. La prima contiene tutti i valori in tensione da normalizzare registrati dall'ADC nei pin CC1, CC2 e VBUS nel caso di collegamento in pull-up con l'alimentazione e no-pull. La seconda servirà a contenere tutti i valori delle resistenze in ingresso elaborati durante l'applicazione della funzione . Si può notare che, sebbene il nome "cable_resistance", al suo interno è presente *cable_resistance.v_ext* che contiene un valore di una tensione, quello del generatore esterno.

Il motivo per cui si è optato di salvare i dati su una variabile esterna è la possibilità di riutilizzarli in altre parti del codice al di fuori della funzione "find_resistance".

### 3.3.2 Distinzione dei casi

In questa parte avviene l'individuazione delle resistenze ammissibili in ingresso con i relativi tipi di collegamento tramite l'applicazione dell'albero decisionale mostrato precedentemente nella sezione 3.2. Le decisioni vengono realizzare tramite una successione di *if* ed *else*.

```
/*
external generator case
*/
// the external generator must have at least 1 volt,
// so the v_bus_np will be a least at 0.3 volt
  if (cable_report.v_bus_np > 10923) /*109023=0.3 volt*/ {

  Vext = cable_report.v_bus_np * (i_r.R2_VBUS + i_r.R1_VBUS) / i_r.R2_VBUS;
  V_ext = Vext * V_fs / N_fs;

  v_max_cc1_pu = i_r.R2_CC1 / (i_r.R1_CC1 + i_r.R2_CC1) * Vext;
  v_max_cc2_pu = i_r.R2_CC2 / (i_r.R1_CC2 + i_r.R2_CC2) * Vext;
  Vext_pu = V_ext * (1 / ((1 / i_r.R2_VBUS) + (1 / i_r.RPU_VBUS))) /
            (1 / ((1 / i_r.R2_VBUS) + (1 / i_r.RPU_VBUS)) + i_r.R1_CC2);
```

**Figura 3.4:** La condizione per il rilevamento del generatore esterno

La prima condizione (figura 3.4) serve a rilevare la presenza di un generatore di tensione esterno collegato tramite usb al microcontrollore. Se presente significa che viene registrato dall'ADC un valore di tensione superiore a 0.3V[4] nel pin VBUS in no-pull[5]. Questa condizione è necessaria ad escludere i generatori di tensione inferiori a 1.000V e il limite viene posto a 0.300V perché sono presenti delle resistenze tra connessione e ADC che formano una caduta di tensione. L'ultima utilità di questa condizione è quella di dividere lo studio in due macro-aree dove verranno poi identificate le resistenze di pull-up o pull-down per CC1 e CC2.

```
// CC1 pull-up, ext. gen.
if(cable_report.v_cc1_np>100) {
                  //100=2.747mV;
    if (cable_report.v_cc1_np < v_max_cc1_pu)
    {
    //Ir1b = (Vext - cable_report.v_bus_pu) / R1b;
      Iry1 = cable_report.v_cc1_np / i_r.R2_CC1;
      Vcc1 = i_r.R1_CC1 * Iry1 + cable_report.v_cc1_np;
      R_cc1_pu = (Vext - Vcc1) / Iry1;
      R_cc1_pd = INFINITY;

    } else
    {
      //ERROR
      R_cc1_pu = NAN;
      R_cc1_pd = NAN;
    }
} else {
//CC1 pull-down, ext. gen.
// 25486=0.7 V  and 19297= 0.530 V
   if (25486 >= cable_report.v_cc1_pu && cable_report.v_cc1_pu >= 19297) {
     Idc1 = (65536 - cable_report.v_cc1_pu) / i_r.RPU_CC1;
     I2c1 = cable_report.v_cc1_pu / i_r.R2_CC1;
     I1c1 = Idc1 - I2c1;
     R_cc1_pd = (cable_report.v_cc1_pu - (i_r.R1_CC1 * I1c1)) / I1c1;
     R_cc1_pu = INFINITY;
   } else{
   //ERROR
   R_cc1_pu = NAN;
   R_cc1_pd = NAN;
   }
}
```

**Figura 3.5:** Istruzioni del caso CC1 con generatore esterno

Dopo questa suddivisione si iniziano a considerare tutti i casi con il generatore di tensione cominciando dal pin CC1 (figura 3.5). Prima si verifica la presenza della resistenza di pull-up, ovvero quando c'è la tensione sul nodo in no-pull collegato all'ADC, detto $v\_cc1\_np$. Inoltre, quest'ultima deve essere inferiore alla tensione che

[4] Normalizzato a 10923
[5] Scollegato dall'alimentazione interna

può raggiungere al limite dettato da $v\_max\_cc1\_pu$[6]. Altrimenti, si controlla se c'è una resistenza di pull-down, verificando che la tensione sul sul nodo in pull-up con l'ADC, detto $v\_cc1\_pu$, sia compresa nel range tra 0.530V e 0.7V[7]. Se non si verifica nessuna delle due condizioni allora c'è un errore. Gli stessi controlli vengono poi ripetuti per CC2.

Passando al caso dell'assenza del generatore esterno, come prima cosa, viene impostato a *NAN* il valore della tensione data dal generatore esterno (figura 3.6). Diversamente dalle casistiche con generatore esterno, si inizia con il doppio pull-up, dove il valore delle tensioni ai nodi in no-pull ai pin CC1 e CC2 registrato dall'ADC sono correlate e di conseguenza il limite massimo impostato di tensione diminuisce a 0.244V[8]. Affinché ci sia il doppio pull-up è richiesto che sia presente tensione in entrambi i nodi in no-pull e che questa sia inferiore al limite massimo sopra stabilito. Verranno quindi applicate le equazioni e trovate le resistenze per entrambi i pin.

```
/*
No external generator case
*/
    else {
      V_ext = NAN;
      /*
      Double pull-up case
      */
      if (cable_report.v_cc1_np>100 && cable_report.v_cc2_np>100 ) {
          if(cable_report.v_cc1_np < 8900 && cable_report.v_cc2_np < 8900){
    //100=2.747mV; 8900=0.244 V
        Ic1 = cable_report.v_cc1_np / i_r.R2_CC1 ;
        Ic2 = cable_report.v_cc2_np / i_r.R2_CC2 ;
        Ir1b = Ic1 + Ic2;
        Vbus = 65496 - (Ir1b * i_r.R1_VBUS);

        R_cc1_pu =(Vbus * i_r.R2_CC1) / cable_report.v_cc1_np - i_r.R2_CC1
                - i_r.R1_CC1;
        R_cc2_pu =(Vbus * i_r.R2_CC2) / cable_report.v_cc2_np - i_r.R2_CC2
                - i_r.R1_CC2 ;
        R_cc1_pd = INFINITY;
        R_cc2_pd = INFINITY;
          }
```

**Figura 3.6:** Istruzioni del caso di doppio pull-up

Poi si considerano i restanti casi come fatto per la macro area del generatore esterno, partendo da CC1 (figura 3.7). Si inizia con la verifica della presenza della resistenza di pull-up. Le condizioni sono che, nel nodo in no-pull, l'ADC registri un valore di tensione e che questo sia inferiore a 0.340V. Se non si verifica, si prova il caso della resistenza in pull-down e, la tensione registrata dall'ADC nel nodo in pull-up, deve essere compreso nel range da 0.530V a 0.700V.

Se nessuna delle due condizioni è verificata allora c'è un errore. Si ripetono poi le stesse istruzioni per CC2 con le rispettive variabili.

[6]Il massimo è calcolato quando è presente un cortocircuito tra pin CC1 e VBUS ed è influenzato dal valore del generatore esterno

[7]Normalizzati rispettivamente a 19297 e a 25486

[8]Normalizzato a 8900

```
      /*
      CC1 pull-up
      */
      if(cable_report.v_cc1_np>100) {
     //100=2.747mV; 12379=0.340 V
         if (cable_report.v_cc1_np <= 12379) {
         R_cc1_pu = ((65496 / cable_report.v_cc1_np) * i_r.R2_CC1)
                 - i_r.R1_VBUS - i_r.R1_CC1 - i_r.R2_CC1;
         R_cc1_pd = INFINITY;

       } else {
//ERROR
          R_cc1_pu = NAN;
          R_cc1_pd = NAN;
         }
       } else{
         /*
         // CC1 pull-down
         */
         // 25486=0.7 V  and 19297= 0.530 V
           if (25486 >= cable_report.v_cc1_pu && cable_report.v_cc1_pu >= 19297) {
             Idc1 = (65536 - cable_report.v_cc1_pu) / i_r.RPU_CC1;
             I2c1 = cable_report.v_cc1_pu / i_r.R2_CC1;
             I1c1 = Idc1 - I2c1;
             R_cc1_pd = (cable_report.v_cc1_pu - (i_r.R1_CC1 * I1c1)) / I1c1;
             R_cc1_pu = INFINITY;

           } else {
//ERROR
          R_cc1_pu = NAN;
          R_cc1_pd = NAN;
           }
       }
```

**Figura 3.7:** Istruzioni del caso di CC1 senza generatore esterno

### 3.3.3 Immagazzinamento dei dati

Una volta finiti i controlli per il riconoscimento delle resistenze viste in ingresso tutti le variabili dove sono immagazzinate momentaneamente i valori vengono divise per 1000 (figura 3.8). L'obiettivo di questo passaggio è quello di diminuire il numero delle cifre utilizzate per indicare i valori delle resistenze. Di conseguenza cambierà anche l'unità di misura di quest'ultime da ohm a kiloohm. L'unica a non venire divisa è quella contenente il valore in tensione del generatore esterno.

```
  // from ohm to kiloohm
  R_cc1_pd = R_cc1_pd / 1000;
  R_cc1_pu = R_cc1_pu / 1000;
  R_cc2_pd = R_cc2_pd / 1000;
  R_cc2_pu = R_cc2_pu / 1000;

// conversion from float to half:
  cable_resistance.r_cc1_pd = float_to_half(R_cc1_pd);
  cable_resistance.r_cc1_pu = float_to_half(R_cc1_pu);
  cable_resistance.r_cc2_pd = float_to_half(R_cc2_pd);
  cable_resistance.r_cc2_pu = float_to_half(R_cc2_pu);
  cable_resistance.v_ext = float_to_half(V_ext);
```

**Figura 3.8:** Istruzioni per il cambio di variabile e l'immagazzinamento dei dati su *cable_resistance*

La diminuzione del numero delle cifre è necessaria per la successiva conversione per il cambio di variabile che le fa passare da float a half-float, o meglio *uint16_t*. Questa conversione è fatta tramite la funzione "float_to_half" che lavora solo con

numeri inferiori a 262144 e viene descritta in seguito nella sezione Funzioni ausiliari 3.3.5. I dati così modificati vengono poi salvati nella struct *cable_resistance* e la funzione termina.

### 3.3.4 Invio dei dati tramite bluetooth

Tutte queste conversioni vengono fatte per far diventare i valori su "cable_resistance" più facili da inviare tramite bluetooth. Per sfruttare al meglio la connessione bluetooth che offre il microcontrollore nRF52840 è necessario ridurre lo spazio che occupano i dati, per poterli inviare simultaneamente in un solo evento di connessione piuttosto che due. La conversione rende i valori a 2 bytes ciascuno anziché 4 bytes, passando da un totale di 20 bytes a 10 bytes. L'invio con bluetooth low energy è fatto nella funzione "computer_power_status" insieme agli altri invii, tramite la funzione "ble_sys_send_notification" (figura 3.9). Quest'ultima assume la forma di *ble_sys_send_notification(3, &cable_report, sizeof cable_report)*, dove *3* è il canale, *&cable_report* è l'indirizzo di memoria e *sizeof cable_report* è la lunghezza.

```
power_report.t_cpu = t;
ble_sys_send_notification(3, &power_report, sizeof power_report);
ble_sys_send_notification(3, &cable_report, sizeof cable_report);

find_resistance();
ble_sys_send_notification(3, &cable_resistance, sizeof cable_resistance);
```

**Figura 3.9:** Istruzioni per l'invio dei dati tramite bluetooth

Insieme ai valori delle resistenze viste in ingresso vengono inviati anche i valori immagazzinati in cable_report e power_report. Tutti i valori vengono inviati in successione e sono distinguibili in base all'ordine con cui vengono ricevuti e la quantità di bytes occupati. Quelli immagazzinati su *cable_resistance* si riconoscono perché in totale hanno 16 bit, sono 5 e sono successivi.

### 3.3.5 Funzioni ausiliari

Il programma per il riconoscimento delle resistenze in ingresso utilizza altre due funziono: "uint32_t as_uint (const float x)" e "uint16_t float_to_half (const float x)". Con l'ausilio di entrambe viene eseguita la conversione da float a *uint16_t* dei valori delle resistenze. Sono state inserite all'interno di power.c, inizializzata dentro power.h e poi richiamate nella funzione "find_resistance".

Le due funzioni sono state realizzate da ProjectPhysX (6) e rivisitate insieme all'aiuto del professor Giorgio Biagetti per farle funzionare anche nel caso di *NAN* e *INFINITY*.

```
// half to float conversion

// IEEE-754 16-bit floating-point format: 1/5/10 bits, Â±131008.0,
// Â±6.1035156E-5, Â±5.9604645E-8, 3.311 digits
uint32_t as_uint(const float x) { return *(uint32_t *)&x; }
// Code thanks to "ProjectPhysX"
// https://stackoverflow.com/questions/1659440/32-bit-to-16-bit-floating-point-co
// Adapted by Giorgio Biagetti su support INFINITY and NANs.

uint16_t float_to_half(const float x) {
  const uint32_t b =
      as_uint(x) +
      0x00001000; // round-to-nearest-even: add last bit after truncated
                  // mantissa TODO: check correctness: can't it alter exponent
                  // in an odd way, exp. for subnormals?
  const uint32_t e = (b & 0x7F800000) >> 23; // exponent
  const uint32_t m =
      b &
      0x007FFFFF; // mantissa; in line below: 0x007FF000 = 0x00800000-0x00001000
                  // = decimal indicator flag - initial rounding
  const uint16_t h =
      (b & 0x80000000) >> 16 | // sign
      (e > 101 && e < 113) *
          ((((0x007FF000 + m) >> (125 - e)) + 1) >> 1) |     // subnormal
      (e > 112) * ((((e - 112) << 10) & 0x7C00) | m >> 13) | // normal
      (e > 142 && e < 255) * 0x7FFF |                        // saturate
      (e == 255) * 0x4000;                                   // NaN & INF
  // printf("%08X -> %04X\n", b, h);
  return h;
}
```

**Figura 3.10:** Le due funzioni che servono per fare il cambio di variabile

## 3.4 Problematiche riscontrate

Durante le prove per la lettura dei dati è stato riscontrato un problema relativo al collegamento della resistenza di pull-up interna propria dell'ADC. Quando viene fatto il collegamento tra la resistenza e il nodo dei pin CC1 o CC2 o VBUS dopo pochi microsecondi si scollega automaticamente, non permettendo alla capacità presente nel cavo usb di caricarsi correttamente. Inoltre, sono presenti dei diodi di protezione che si attivano anche per tensioni molto basse, circa 0,2V, che ne rallentano ancora di più la carica. C'è la possibilità di impostare il tempo di collegamento tramite la regolazione dei bit di configurazione dell'ADC con un valore che va da 1 a 5 aumentandone la durata al massimo a 40 microsecondi, ma sarebbe comunque troppo poco per far avvenire la carica. Questa problematica rende molto difficile la distinzione delle resistenze viste in ingresso in quanto il valori registrati sarebbero fin troppo simili tra loro.

CAPITOLO 4

# Misurazioni delle resistenze interne al dispositivo nRF52840

*I valori sono forniti dai produttori del microcontrollore nRF52840 e di conseguenza possono subire variazioni al momento della costruzione o tramite il deterioramento dato dall'utilizzo. Si è ritenuto, pertanto, necessario condurre uno studio approfondito per calcolarne il valore reale così da avere una precisione maggiore durante la distinzione dei dispositivi in ingresso.*

## 4.1 L'idea iniziale

Dopo aver programmato la funzione "find_resistance", si è deciso di valutare l'esatto valore delle resistenze interne al microprocessore in quanto quelli utilizzati fin' ora per fare i calcoli erano dei valori ideali, forniti dai produttori del dispositivo. L'idea è quella di inserire in ingresso al chip, nella porta usb-c, un dispositivo che funga da cortocircuito tra i pin CC1, CC2 e VBUS e gli altri pin della porta che non sono connessi. Tramite questo collegamento si forma una connessione tra le resistenze interne ai pin e il multiplexer collegato a una serie di componenti tra cui un DAC[1] dal quale è possibile impostare un valore di tensione a piacimento. In questo modo è possibile ricavare i valori delle resistenze interne R1, R2 e di pull-up.

Per ricavare il valore delle resistenze bisogna operare sull'hardware del microcontrollore andando a configurarne i componenti: il DAC; l'ADC; il multiplexer; i pin della porta USB-C. Queste istruzioni avvengono tramite uno *switch case* che passa da uno stato all'altro tramite un interrupt dettato dall'hardware stesso, come una macchina a stati. Nella macchina a stati sono già presenti otto *case* che servono a fare la calibrazione dei vari componenti e, per fare il calcolo delle resistenze interne, ne sono stati programmati altri 6.

Sarebbe possibile calcolare anche il valore della resistenza interna di pull-down, ma non essendo utilizzata ai fini del riconoscimento del dispositivo in ingresso, sarebbe solo un rallentamento del programma in quanto bisognerebbe creare altrettanti *case* che rallenterebbero inutilmente l'avvio del dispositivo.

[1] Convertitore digitale analogico

## 4.2 La realizzazione

### 4.2.1 I collegamenti realizzati

Il corto circuito tale da collegare i pin è stato creato ad hoc saldando assieme i cavi di un connettore USB-C .

I collegamenti risultanti sono i seguenti:

- corto circuito tra il pin CC1 e il pin A7
- corto circuito tra il pin CC2 e il pin B6
- corto circuito tra il pin VBUS e il pin B2

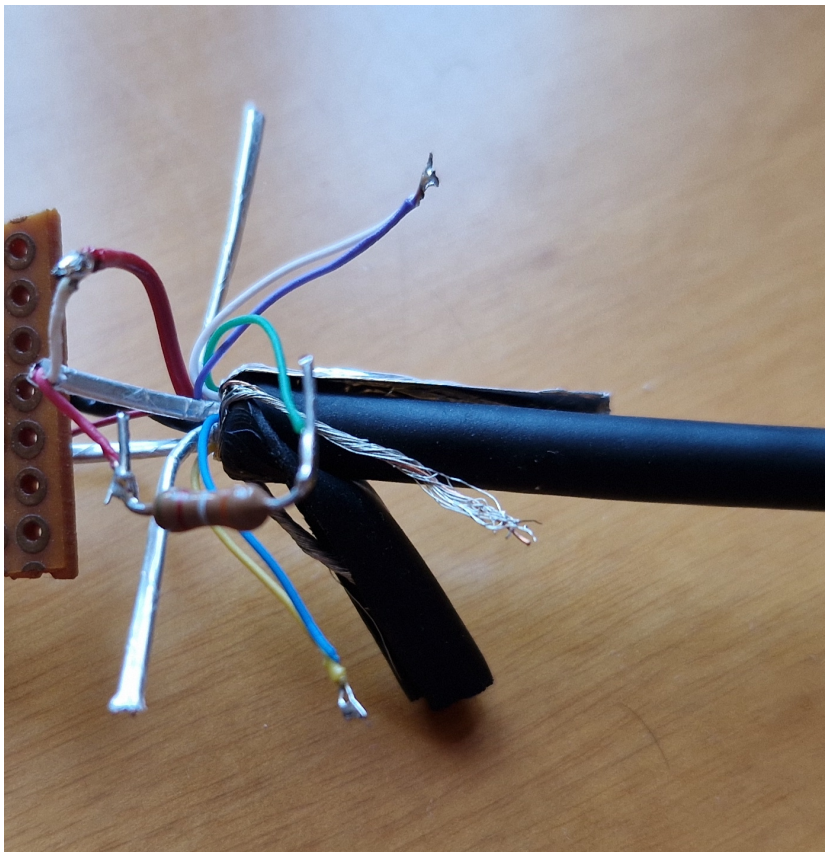

**Figura 4.1:** Il cortocircuito realizzato

Affinché ci sia il collegamento tramite multiplexer, deve essere impostato il giusto canale che cambia in base al pin connesso. Per CC1 il collegamento con il pin in cortocircuito avviene nel canale MUX 3, per CC2 avviene nel canale MUX 9 ed infine per VBUS nel MUX 11. Una volta impostato il canale del multiplexer bisogna solo disattivare l'ADC e scollegarlo dai pin così da poter iniziare le misure.

Data la presenza delle due resistenze R1 e R2 viste in *serie* l'una con l'altra, è necessario andare a configurare il nodo dove avviene il loro collegamento. Bisogna impostarlo una volta in alta impedenza, così da misurare entrambe le resistenze, un'altra in bassa impedenza, così da misurare solo la prima in quanto viene forzato un collegamento a massa del nodo.

I casi considerati sono riportati nella tabella 4.1

| Pin interno | Pin collegato | Canale MUX | Tipo impedenza | Resistenza registrata |
|---|---|---|---|---|
| CC1 | A7 | 3 | alta impedenza | $R_{eq}CC1$ |
| CC1 | A7 | 3 | bassa impedenza | $R1c1$ |
| CC2 | B6 | 9 | alta impedenza | $R_{eq}CC2$ |
| CC2 | B6 | 9 | bassa impedenza | $R1c2$ |
| VBUS | B2 | 11 | alta impedenza | $R_{eq}VBUS$ |
| VBUS | B2 | 11 | bassa impedenza | $R1b$ |

**Tabella 4.1:** Le resistenze ottenute in base al tipo di collegamento

### 4.2.2 Circuito equivalente

Il circuito a cui viene collegato il pin tramite il multiplexer è composto da un mosfet collegato con l'uscita dell'op amp nel gate, con il multiplexer nel source e con la resistenza Rs (o resistore di sensing) nel drain (figura 4.2). I terminali di Rs sono collegati ad un ADC che ne misura la caduta di tensione, mentre nell'op amp l'ingresso invertente è collegato con un DAC e l'ingresso non invertente con l'ingresso del multiplexer. Il DAC connesso ad op amp serve a fornire una tensione scelta dall'utente, nel caso in questione 1.0 Volt. Il mosfet fa sì che ai suoi capi di source e drain scorra la stessa corrente e di conseguenza è possibile fare un rapporto tra la tensione ai capi di Rs e la tensione ai capi delle resistenze interne per via della legge di Ohm. Infine, grazie al multiplexer è possibile regolare su quale pin far passare la corrente, così da risolvere il circuito equivalente pin per pin, prima CC1, poi CC1 ed infine VBUS.

Il rapporto ottenuto è $\frac{RS}{Tensione_{\text{ADC}}} = \frac{R_{\text{equivalente}}}{Tensione_{\text{DAC}}}$

### 4.2.3 I calcoli svolti

Dai case si ricavano Rtot = R1 + R2 e R1 di tutti e tre i pin, che vengono immagazzinati in maniera "grezza" all'interno della struct *internal_resistance*. Infatti, per calcolare il valore reale delle resistenze interne, è necessario il valore della resistenza Rs per poter applicare il rapporto. Dato l'ordine con viene svolta la misurazione degli elementi del circuito, il valore della resistenza Rs viene calcolato solo dopo la fine della macchina a stati. Di conseguenza ciò che viene fatto è immagazzinare nelle variabili contenenti i valori delle resistenze solo una parte dell'equazione. Quest'ultime vengono poi completate all'interno della funzione "find_resistor_divider", la quale viene chiamata dall'interno della funzione "measure_finished". All'interno di "find_resistor_divider" sono presenti i calcoli per ottenere i valori completi di Rtot e R1 e una volta trovati viene fatta la differenza tra i due, ovvero Rtot - R1 per ricavare R2. Infine, si ricava quella di pull-up dei tre pin, risolvendo il circuito con i nuovi valori di R1 e R2 nel caso non ci sia nulla collegato.

La divisione delle operazioni, in parte su measure.c e in parte su power.c, per trovare le resistenze interne è fatta perché i pin devono essere connessi prima in analogico e poi in digitale. Nella prima parte è necessario che i pin siano connessi alla parte analogica, così da poter calcolare le resistenze tramite i valori impostati dal DAC. L'utilizzo delle casistiche serve al microcontrollore per avere il tempo di

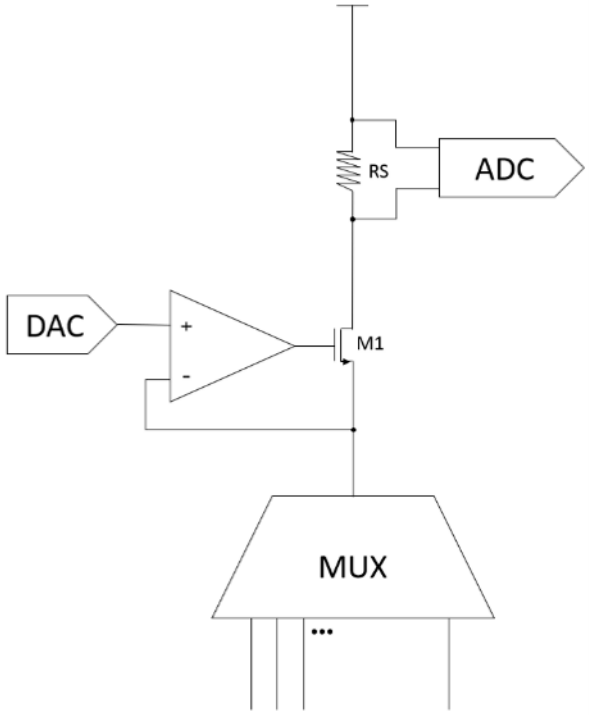


**Figura 4.2:** Circuito del DAC immagine realizzata con Microsoft PowerPoint®

modificare collegamento e valori in tensione del DAC all'occorrenza, andando alla casistica successiva tramite una interrupt analogico. Nella seconda parte è necessario che i pin siano connessi in digitale così da poter utilizzare i valori delle tensioni misurate dall'ADC dei pin.

## 4.3 Descrizione della programmazione

### 4.3.1 Breve descrizione

La programmazione effettuata, diversamente da quanto fatto per la distinzione delle resistenze esterne, è stata fatta attraverso una serie di istruzioni all'interno della macchina a stati e funzioni più piccole per lo svolgimento dei calcoli. Poiché i casi da considerare sono fissi, non è necessario un albero decisionale in quanto tutte le operazioni da eseguire sono sequenziali.

### 4.3.2 Le variabili utilizzate

La principale variabile globale utilizzata è una struct contenente tutti i valori delle resistenze interne dei pin (figura 4.3).

```
struct internal_resistance
{
    float R1_CC1;
    float R2_CC1;
    float R1_CC2;
    float R2_CC2;
    float R1_VBUS;
    float R2_VBUS;
    float RPU_CC1;
    float RPU_CC2;
    float RPU_VBUS;
    float R_TOT_CC1;
    float R_TOT_CC2;
    float R_TOT_VBUS;
} extern i_r;
```

**Figura 4.3:** La variabile struct esterna contenente le resistenze interne ai pin

Le variabili con nome *R_TOT_CC1*, *R_TOT_CC2* e *R_TOT_VBUS* servono ad immagazzinare la *serie* delle resistenze R1 e R2 e non vengono utilizzate nella funzione per il riconoscimento dei dispositivi collegati.

La funzione necessita di due parametri in ingresso: *average*, che rappresenta il valore registrato dall'ADC ai capi di RS in numero intero; *clipping*, che è un valore booleano che serve ad indicare l'overflow del dato sulla misura della variabile *average*

Le altre variabili interne utilizzate servono per le impostazioni del multiplexer e dell'ADC. La variabile *out* serve per impostare il canale del mux per il collegamento tra il circuito con il DAC e quello del pin. Le variabili *ok*1 e *ok*2 sono booleane e servono per la configurazione e l'avvio dell'ADC.

### 4.3.3 La macchina a stati

La macchina a stati per la calibrazione è realizzata all'interno della funzione "adc_cal_ready" (figura 4.4). Al suo interno si procede con l'inizializzazione delle variabili *ok*1 e *ok*2 e subito dopo con lo switch case che caratterizza questa funzione. I *case* totali sono 14, dal primo all'ottavo viene fatta la calibrazione dell'ADC, mentre dalla seconda parte dell'ottavo compreso fino all'ultimo vengono raccolti i dati per lo studio delle variabili interne. La parte relativa alla calibrazione era già presente e di fatto sono stati aggiunti solo gli ultimi sei.

Le prime istruzioni per il calcolo delle resistenze interne sono scritte all'interno dell'ottavo *case* (figura 4.5). Si inizia con la configurazione dei pin gpio, in questo caso il pin CC1, tramite la chiamata della funzione "nrf_gpio_cfg". I parametri inseriti andranno a modificarne i collegamenti, le resistenze dell'inverter e il modo in

```
static void adc_cal_ready (int32_t average, bool clipping)
{
    bool ok1, ok2;
    switch (calibration_phase++) {
        case 0:
            // reset calibration registers to default values:
            adc_calibrate(ADC_CAL_DEFAULT, ADC_CAL_DEFAULT);
            dac_calibrate(0, 0);
            dac_set_drain(DAC_CAL_LOW_POINT);
            ok1 = adc_config(ADC_CH_2, ADC_PGA_1, ADC_OFFSET_0, adc_cal_ready);
            ok2 = adc_start_read(PRECISE_OVERSAMPLE);
            break;
```

**Figura 4.4:** Le prime istruzioni della funzione "adc_cal_ready" per l'avvio della macchina a stati

cui viene utilizzato il pin. Queste impostazioni sono fatte per portare CC1 in alta impedenza, così da poter calcolare la *serie* delle due resistenze interne.

I parametri in ingresso per CC1 sono:

- USBC_PIN_CH_CC1, per scegliere CC1 come pin da configurare
- NRF_GPIO_PIN_DIR_INPUT, per considerare il pin come input
- NRF_GPIO_PIN_INPUT_CONNECT, per considerare il pin come connesso
- NRF_GPIO_PIN_NOPULL, per impostare la connessione in no-pull
- NRF_GPIO_PIN_S0D1, per impostare le resistenze dell'inverter. In questo caso S0 imposta il livello basso a standard e D1 imposta il livello alto a disconnesso (open drain)
- NRF_GPIO_PIN_SENSE_LOW, per abilitare il pin per risvegliare il controllore sul livello basso

In seguito, tramite la funzione[2] "dac_set_drain(DAC_CAL_TEST_POINT)", viene impostato il valore del DAC a 1 V.

Infine sono presenti i comandi per configurare l'ADC, *ok*1, e avviarne la lettura, *ok*2. Il parametro ADC_CH_1 serve a scegliere il canale uno, ADC_PGA_1 per avere guadagno unitario e ADC_OFFSET_0 per non considerare l'offset.

[2]DAC_CAL_TEST_POINT è una costante impostata per avere il valore di un volt normalizzata con il fondo scala analogico e digitale dell'ADC, il primo a 1,25 V e il secondo è il numero puro $2^{(16)}$

```
// set MUX, channel 3 :
uint32_t out = (chans[2] + 8) << MUX_CH_SHIFT;
MUX_PORT->OUTSET = out & MUX_CH_MASK;
MUX_PORT->OUTSET = out & MUX_EN_MASK;

//PIN CC1 high impedence
nrf_gpio_cfg(USBC_PIN_CH_CC1,NRF_GPIO_PIN_DIR_INPUT,NRF_GPIO_PIN_INPUT_CONNECT,
    NRF_GPIO_PIN_NOPULL,NRF_GPIO_PIN_S0D1,NRF_GPIO_PIN_SENSE_LOW);

dac_set_drain(DAC_CAL_TEST_POINT);
ok1 = adc_config(ADC_CH_1, ADC_PGA_1, ADC_OFFSET_0, adc_cal_ready);
ok2 = adc_start_read(PRECISE_OVERSAMPLE);
break;
```

**Figura 4.5:** Il case 8, viene impostato il canale del mux, configurato il pin CC1 e avviato l'ADC per la lettura dei dati

Si può passare al *case* 9 solamente tramite l'interrupt hardware che viene fatto una volta che la configurazione del pin CC1 e dell'ADC è completa ed è quindi disponibile la lettura della tensione ai capi di Rs nella variabile *average* (figura 4.6). Subito dopo viene applicata la formula inversa del rapporto ottenuto risolvendo il circuito nella sezione 4.2.1. Tuttavia l'equazione non è completa perché manca la moltiplicazione per la RS, dato che quest'ultima verrà calcolata nella funzione "cal_compute" che viene svolta successivamente. Diventa necessario dividere equazione in due parti e salvare i dati "grezzi" all'interno della variabile globale *internal_resistance* per poi completarne il calcolo in un secondo momento.

```
case 9:
    i_r.R_TOT_CC1=DAC_CAL_TEST_POINT*(1<<16)/average;

    //PIN CC1 low impedence
    nrf_gpio_pin_write(USBC_PIN_CH_CC1,0);
    nrf_gpio_cfg(USBC_PIN_CH_CC1,NRF_GPIO_PIN_DIR_OUTPUT,NRF_GPIO_PIN_INPUT_DISCONNECT,
        NRF_GPIO_PIN_NOPULL,NRF_GPIO_PIN_H0S1,NRF_GPIO_PIN_NOSENSE);

    dac_set_drain(DAC_CAL_TEST_POINT);
    ok1 = adc_config(ADC_CH_1, ADC_PGA_1, ADC_OFFSET_0, adc_cal_ready);
    ok2 = adc_start_read(PRECISE_OVERSAMPLE);
    break;
```

**Figura 4.6:** Il case 9, dove avviene la prima lettura e le configurazioni per l'avvio della seconda

Una volta salvato il valore della equazione semi-calcolata all'interno di *R_TOT_CC*1, iniziano le istruzioni per la nuova configurazione del microcontrollore. La prima funzione, "nrf_gpio_pin_write", viene utilizzata per settare il valore del pin CC1, mentre "nrf_gpio_cfg" viene utilizzata per farne la configurazione.

I parametri in ingresso per CC1 sono:

- USBC_PIN_CH_CC1, per scegliere CC1 come pin da configurare
- NRF_GPIO_PIN_DIR_OUTPUT, per considerare il pin come output
- NRF_GPIO_PIN_INPUT_DISCONNECT, per disconnettere il pin
- NRF_GPIO_PIN_NOPULL, per impostare la connessione in no-pull

- NRF_GPIO_PIN_H0S1, per impostare le resistenze dell'inverter. In questo caso S0 imposta la prima a high e S1 imposta la seconda a standard
- NRF_GPIO_PIN_NOSENSE, per disabilitare il pin per avviare il microcontrollore

Queste impostazioni servono per portare il pin in bassa impedenza, così da poter escludere la seconda resistenza interna. Il nodo che collega la $R1c1$ e la $R2c1$ va a massa tramite un resistore di circa 50 ohm e quindi non scorre più corrente su $R2c1$.

Successivamente viene nuovamente impostato a 1 V la tensione sul DAC tramite la funzione "dac_set_drain" ed infine è configurato ed avviato l'ADC tramite $ok1$ e $ok2$. All'inizio del *case* 10 finisce la configurazione iniziata in quello precedete e si applica la formula inversa del rapporto del circuito equivalente per trovare la $R1c1$, qui chiamata $R1\_CC1$ (figura 4.7).

```
case 10:
    i_r.R1_CC1=DAC_CAL_TEST_POINT*(1<<16)/average;

    // set MUX, channel 9 :
    out = (chans[4] + 8) << MUX_CH_SHIFT;
    MUX_PORT->OUTSET = out & MUX_CH_MASK;
    MUX_PORT->OUTSET = out & MUX_EN_MASK;

    //PIN CC2 high impedence
    nrf_gpio_cfg(USBC_PIN_CH_CC2,NRF_GPIO_PIN_DIR_INPUT,NRF_GPIO_PIN_INPUT_CONNECT,
        NRF_GPIO_PIN_NOPULL,NRF_GPIO_PIN_S0D1,NRF_GPIO_PIN_SENSE_LOW);

    dac_set_drain(DAC_CAL_TEST_POINT);
    ok1 = adc_config(ADC_CH_1, ADC_PGA_1, ADC_OFFSET_0, adc_cal_ready);
    ok2 = adc_start_read(PRECISE_OVERSAMPLE);
    break;
```

**Figura 4.7:** Il case 10, dove avviene la seconda lettura e le configurazioni per il pin CC2

Ancora una volta il calcolo non è completato per la mancanza del valore Rs e verrà ultimato successivamente in un'altra funzione insieme ai restanti valori semi-calcolati.

Le successive istruzioni sono le stesse rispetto ai *case* 8 e 9, con la differenza che si prende in considerazione il pin CC2 e viene cambiato il canale del multiplexer utilizzato. Lo stesso viene fatto per i *case* relativi allo studio delle resistenze interne del pin VBUS. L'unica differenza è che nell'ultimo, il quattordicesimo, non viene più fatta la configurazione del pin successivo visto che lo studio è terminato (figura 4.8). Invece, come ultima serie di istruzioni, viene disabilitato il collegamento tramite multiplexer.

```
case 14:
    i_r.R1_VBUS=DAC_CAL_TEST_POINT*(1<<16)/average;

    // disable MUX:
    MUX_PORT->OUTCLR = MUX_EN_MASK;
    MUX_PORT->OUTCLR = MUX_CH_MASK;
```

**Figura 4.8:** Il case 14, dove avviene l'ultima lettura e viene scollegato il multiplexer

Infine la macchina a stati si conclude con lo stato *default*, dove l'ADC e il DAC vengono disabilitati tramite le funzioni "adc_disable" e "dac_disable", mentre vengono avviate le successive misure da fare tramite la funzione "do_measure" (figura 4.9).

```
default:
    // disable everything:
    adc_disable();
    dac_disable();
    do_measure();
```

**Figura 4.9:** Viene raggiunto questo stato una volta completati tutti gli altri e la funzione "adc_cal_ready" termina

### 4.3.4 La funzione per il calcolo delle resistenze interne

Il calcolo delle resistenze interne viene fatto dentro power.c tramite la funzione "find_resistor_divider" (figura 4.10) .

```
void find_resistor_divider (void)

{
//
float R_TOT;
//CC1
R_TOT=i_r.R_TOT_CC1*dat_calibration_register.cal_sense;
i_r.R1_CC1=i_r.R1_CC1*dat_calibration_register.cal_sense;
i_r.R2_CC1=R_TOT-i_r.R1_CC1;
//CC2
R_TOT=i_r.R_TOT_CC2*dat_calibration_register.cal_sense;
i_r.R1_CC2=i_r.R1_CC2*dat_calibration_register.cal_sense;
i_r.R2_CC2=R_TOT-i_r.R1_CC2;
//VBUS
R_TOT=i_r.R_TOT_VBUS*dat_calibration_register.cal_sense;
i_r.R1_VBUS=i_r.R1_VBUS*dat_calibration_register.cal_sense;
i_r.R2_VBUS=-i_r.R1_VBUS;
}
```

**Figura 4.10:** Funzione utilizzata per ultimare i calcoli delle resistenze interne

Al suo interno viene inizializzata la variabile di appoggio *R_TOT* per contenere il valore della resistenza equivalente della serie dei pin dopo essere stata moltiplicata per Rs. Il valore di Rs è immagazzinato all'interno della variabile *dat_calibration_register.cal_sense* e serve per ultimare l'equazione moltiplicandolo con i valori "grezzi" precedentemente calcolati nella macchina a stati. Si inizia la risoluzione dalle resistenze del pin CC1, andando a moltiplicare il valore di Rs per *R1_CC1* e *R_TOT_CC1*. Poi si procede con la sottrazione della resistenza R1 su quella equivalente e si trova la *R2_CC1*. Gli stessi calcoli vengono poi effettuati per trovare tutte le resistenze di CC2 e VBUS.

Infine vengono calcolate le resistenze di pull-up interne tramite il valore in tensione registrato dall'ADC del microcontrollore. Per fare questo calcolo è necessario che non ci sia nulla collegato al microcontrollore. Viene utilizzata la caduta di tensione

ai capi della resistenza di pull-up per calcolarne i valori e salvarli all'interno della struct *internal_resistance* (figura 4.11).

```
i_r.RPU_CC1= (( 65536.0 -cable_report.v_cc1_pu)/cable_report.v_cc1_pu)*i_r.R2_CC1;
i_r.RPU_CC2= (( 65536.0 -cable_report.v_cc2_pu)/cable_report.v_cc2_pu)*i_r.R2_CC2;
i_r.RPU_VBUS= ((  65536.0 -cable_report.v_bus_pu)/cable_report.v_bus_pu)*i_r.R2_VBUS;
```

**Figura 4.11:** Le equazioni per il calcolo delle resistenze interne di pull-up

## 4.4 Interfaccia utente

Il programma è munito di interfaccia utente che permette la visualizzazione dei dati raccolti e la configurazione del microcontrollore (figura 4.12). Sono presenti tre sezioni, quella per il setup, quella per le misure e l'ultima per il debug. In seguito sono riportati alcuni esempi riguardo la finestra di debug dove sono presenti, oltre che ai valori delle tensioni e delle resistenze registrate, i tasti per effettuare la calibrazione. Il pulsante "ADC DIV" serve per fare il calcolo dei divisori resistivi interni al microcontrollore, quindi R1 e R2 dei tre pin CC1, CC2 e VBUS, mentre "ADC RES" serve per il calcolo delle resistenze di pull-up interne dei pin.

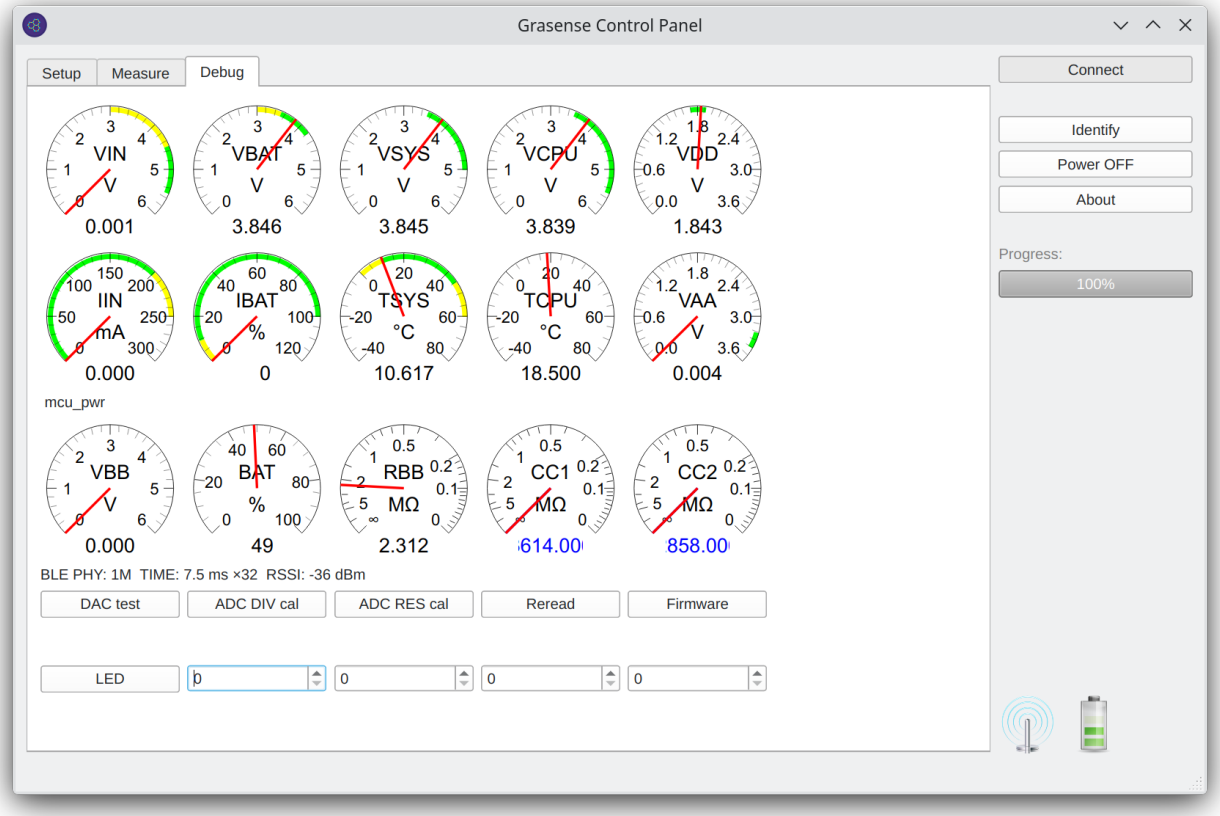


**Figura 4.12:** L'interfaccia utente nella sezione debug

Per effettuare la calibrazione delle resistenze interne è necessario collegare al microcontrollore il cortocircuito realizzato tramite cavo USB-C (figura 4.1). Invece

per la calibrazione delle resistenze interne di pull-up è necessario che non ci sia nulla collegato così da poter escludere la resistenza interna R1 e semplificare i calcoli. Diventa quindi necessario un intervento dell'utente per effettuare questi passaggi.

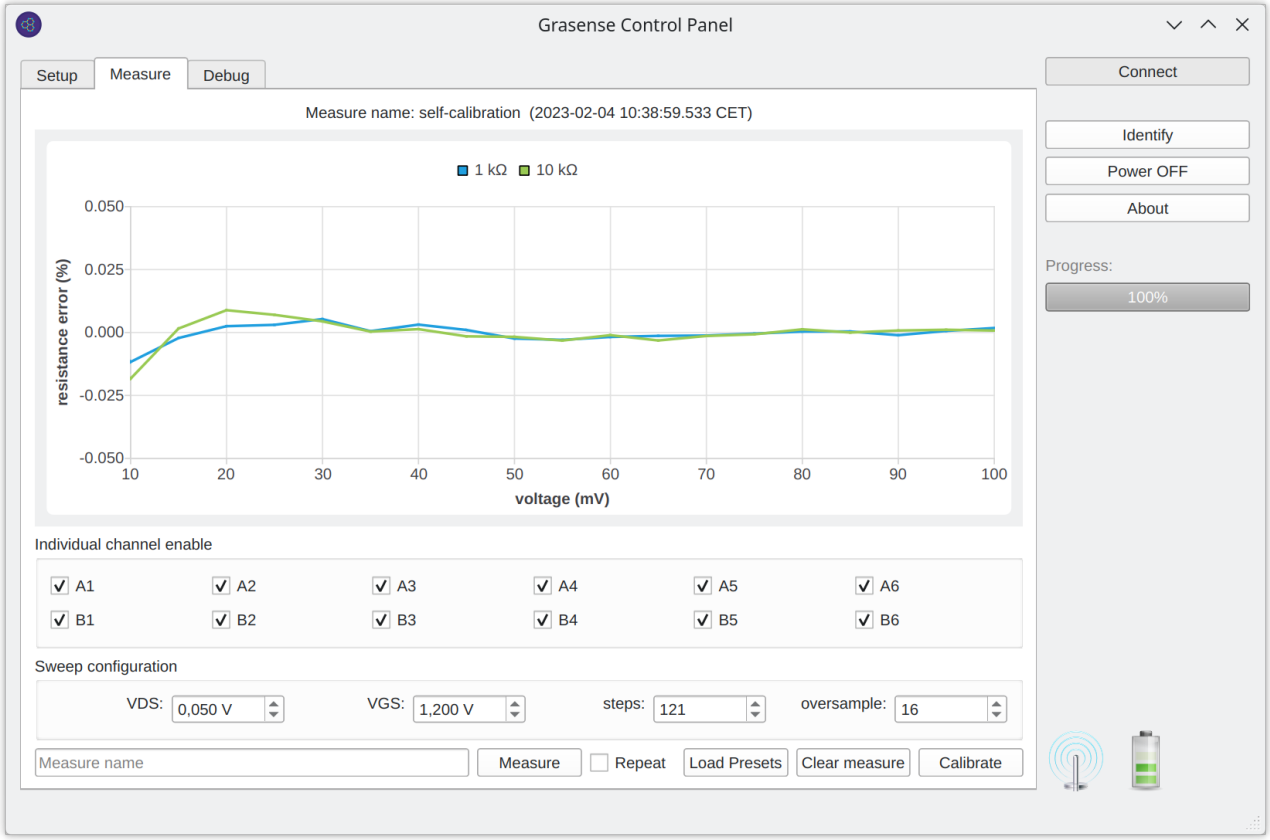


**Figura 4.13:** La precisione in percentuale del calcolo delle resistenze esterne

Infine si può osservare dalla figura 4.13 la precisione con cui vengono calcolate le resistenze esterne al microcontrollore. Più la tensione aumenta più aumenta la precisione della misurazione della resistenza.

# CAPITOLO 5

# Raccolta dati

Di seguito vengono esposte ed analizzate tutte le misurazioni fatte durante lo studio del dispositivo nRF52840. Si parte dalla trattazione delle resistenze più facilmente distinguibili entro i range di tensione ammissibili, per poi passare alla raccolta di misure fatte per le resistenze interne ed esterne al microcontrollore.

## 5.1 Range delle resistenze facilmente misurabili

Tramite l'ausilio di MATLAB sono stati realizzati dei grafici per rendere più comprensibile il range delle resistenze misurabili. Nell'asse delle ascisse del grafico è posto il valore delle resistenze individuate e nell'asse delle ordinate il relativo valore in tensione. L'ADC interno al microcontrollore ha una precisione in tensione del millivolt, di conseguenza sono stati realizzati degli array contenenti tutti i valori delle tensioni ammissibili nei vari casi con passo del millivolt. Sono stati calcolati i range per i casi resistenza di pull-up e pull down di CC1. Non è servito calcolarli per i pin CC2 e VBUS in quanto le resistenze interne al microcontrollore sono le stesse per tutti i pin. Di conseguenza anche i limiti minimi e massimi di tensione ammissibili nell'ADC sono gli stessi.

Il primo range analizzato è quello per le resistenze di pull-down (figura 5.1).

I limiti sono stati ricavati dalla simulazione del circuito tramite LTspice XVII, quello inferiore, pari a 0,542 V, è stato calcolato sostituendo la resistenza esterna con un cortocircuito, mentre quello superiore, pari a 0,680 V, sostituendo la resistenza con un circuito aperto. Quando la curva assume un andamento verticale è più facile distinguere i valori delle resistenze e di conseguenza si riescono a identificare in modo più chiaro quelle con valore compreso tra 10 $k\Omega$ a 450 $k\Omega$.

Il secondo range analizzato è quello delle resistenze di pull-up (figura 5.2).

Per questo caso i limiti vengono ricavati simulando il nuovo circuito, considerando come limite inferiore 0 V, quando c'è un circuito aperto e come limite superiore 0,330 V, quando c'è un cortocircuito. Anche in questo caso le resistenze più facilmente distinguibili si possono riconoscere quando la curva assume un andamento verticale e il loro range è compreso tra 9 $k\Omega$ e 300 $k\Omega$.

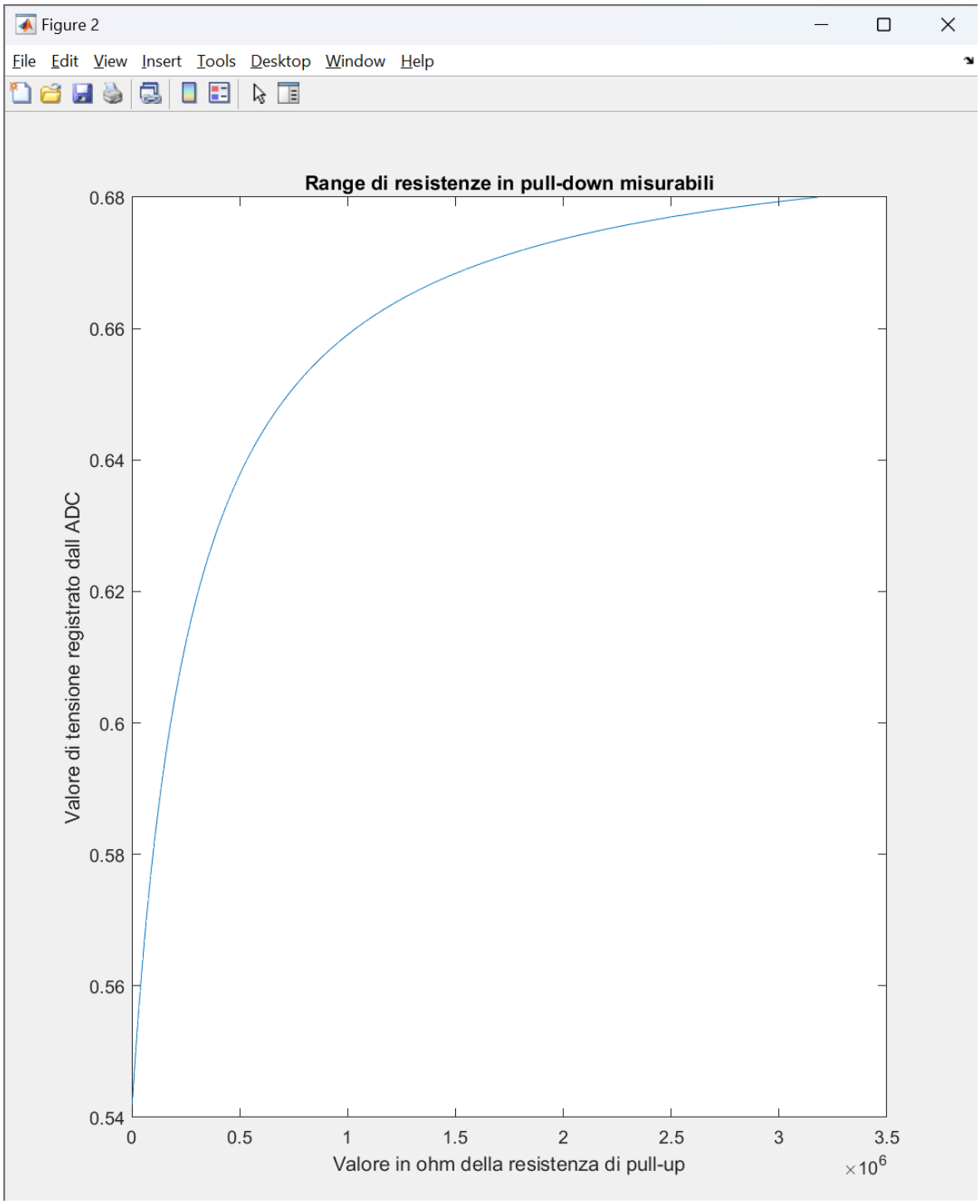


**Figura 5.1:** Il range per resistenza di pull-down, realizzato tramite MATLAB

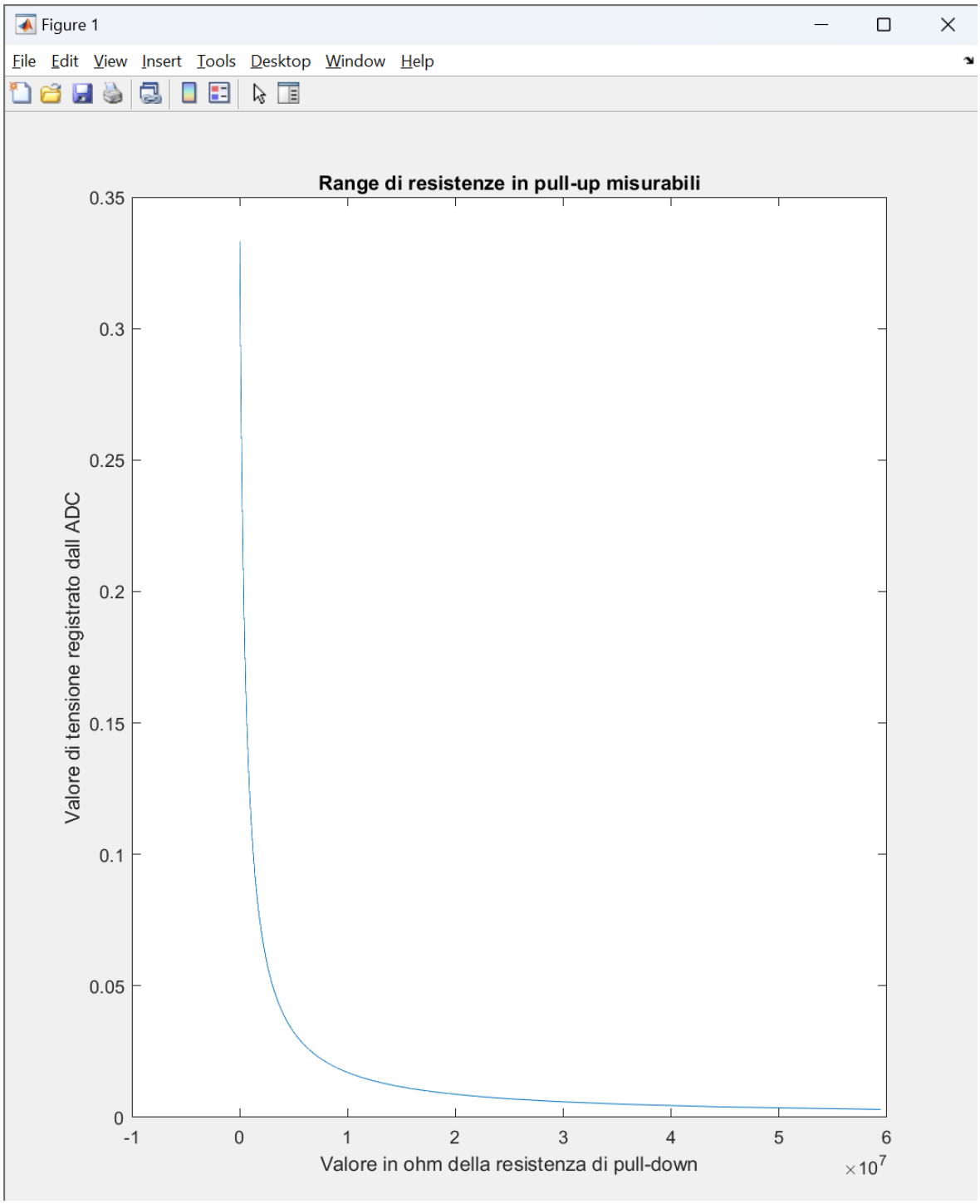


**Figura 5.2:** Il range per resistenza di pull-up, realizzato tramite MATLAB

## 5.2 Misure delle resistenze interne al dispositivo

### 5.2.1 Resistenze dei divisori R1 e R2

Durante la calibrazione fatta con la macchina a stadi vengono utilizzati i valori immagazzinati nella variabile *average* per calcolare le resistenze interne del microcontrollore. Data la grande precisione di queste misure, come visto nel capitolo Interfaccia Utente 4.4, è sufficiente farne solo una per ottenere un valore estremamente affidabile.

Dalla documentazione del microcontrollore nRF52840, tutte le resistenze dei divisori R1 sono circa 220 $k\Omega$,mentre quelle di R2 sono circa 100 $k\Omega$. Questi valori valgono per tutti i pin CC1, CC2 e VBUS. Facendo la misurazione tramite la macchina a stati sono state riportate le seguenti misure, elencate nella tabella 5.1:

| **Pin** | *average* | **Resistenza trovata** | **Valore in $\Omega$** |
|---|---|---|---|
| CC1 | 78222663 | $R1\_CC1$ | 219098 |
| CC1 | 53774689 | $R2\_CC1$ | 99682 |
| CC2 | 78100731 | $R1\_CC2$ | 219440 |
| CC2 | 53650322 | $R2\_CC2$ | 100079 |
| VBUS | 78144000 | $R1\_VBUS$ | 219318 |
| VBUS | 53722519 | $R2\_VBUS$ | 99771 |

**Tabella 5.1:** Elenco delle resistenze ricavate in base al valore registrato da *average*

Dal secondo valore di *average* di ogni pin in realtà si otterrebbe la serie delle resistenze, ma per semplificare la lettura del grafico si è scelto di indicare la resistenza R2 che si ottiene sottraendo a quella totale la resistenza R1.

Come si può osservare dalla tabella i valori delle resistenze misurate non si discostano molto dal valore ideale della documentazione. Queste misurazioni sono state comunque necessarie per verificarne la veridicità e serviranno in futuro per fare nuovamente la calibrazione vista l'usura nel tempo.

### 5.2.2 Tensioni misurate dall'ADC per il calcolo delle resistenze interne di pull-up

Una volta calcolate le resistenze interne R1 e R2, sono state fatte delle misurazioni per ricavare il valore delle resistenze di pull-up interne. Le prime sono state fatte collegando il cortocircuito realizzato con il cavo USB-C e impostandolo in alta impedenza. Tuttavia le impostazioni in alta impedenza non sono tali da riuscire a registrare valori abbastanza alti di tensione per misurare le resistenze correttamente. Di conseguenza è stato necessario rifare altre misurazioni senza avere nulla collegato al dispositivo così da avere un circuito aperto. Tutti i valori registrati sono stati immagazzinati su MATLAB all'interno di tre array, uno per pin, e ne sono poi state calcolate la media e la deviazione standard. Tramite queste due serie di misurazioni è stato possibile fare un confronto tra i valori registrati in caso di dispositivo collegato in alta impedenza e in caso di circuito aperto, come mostrato nella tabella 5.2

Come si può osservare la misura fatta sul pin VBUS è quella che più risente della precisione dell'ADC. In entrambi i modi in cui sono stati misurati i valori la

| Pin | Tipo | Numero misure | Valore medio | Deviazione standard |
|---|---|---|---|---|
| CC1 | alta impedenza | 17 | 22161.54 | 33.69 |
| CC1 | circuito aperto | 22 | 24851.13 | 31.97 |
| CC2 | alta impedenza | 17 | 22283.29 | 37.10 |
| CC2 | circuito aperto | 22 | 24799.65 | 27.87 |
| VBUS | alta impedenza | 17 | 19389.17 | 80.04 |
| VBUS | circuito aperto | 22 | 24733.09 | 81.23 |

**Tabella 5.2:** Elenco dei valori registrati dall'ADC

deviazione standard è molto più grande rispetto a quelle dei pin CC1 e CC2. Inoltre il valore medio nel caso del circuito aperto è molto diverso rispetto al caso di alta impedenza. Ciò conferma che è necessario tenere il dispositivo scollegato quando si deve fare la calibrazione delle resistenze interne di pull-up.

### 5.2.3 Resistenze interne di pull-up

I valori delle tensioni utilizzate per il calcolo delle resistenze di pull-up interne sono quelle relative al caso del circuito aperto. Tramite quei valori in tensione sono state applicate le equazioni per trovare le resistenze. Sono stati fatti due campionamenti, di undici misurazioni ciascuno, con i valori precedentemente ottenuti. Il primo campionamento è stato fatto usando i valori ideali delle resistenze interne dei divisori, il secondo usando quelli calcolati con il collegamento in cortocircuito.

| Pin | Valori resistenze | Valore medio | Deviazione standard |
|---|---|---|---|
| CC1 | ideale | 165250.40 | 247.64 |
| CC1 | misurata | 165365.30 | 429.10 |
| CC2 | ideale | 165387.60 | 247.64 |
| CC2 | misurata | 165251.10 | 253.02 |
| VBUS | ideale | 168133.40 | 535.82 |
| VBUS | misurata | 168197.10 | 618.99 |

**Tabella 5.3:** Elenco delle misure fatte per le resistenze interne di pull-up

Come si può osservare dalla tabella 5.3, tutte le resistenze sono molto simili tra loro, soprattutto le doppie misure fatte per ogni pin. Di conseguenza si può dedurre che le misure fatte sono corrette e le resistenze sono abbastanza simili a quelli indicati dal produttore del dispositivo.

## 5.3 Misure delle resistenze esterne

Per fare una prova di misurazione delle resistenze esterne è stato scelto il caso di pull-up con generatore di tensione eterno. Gli altri casi sarebbero stati difficili da distinguere per via dei problemi con il collegamento della resistenza di pull-up interna dell'ADC che non permette il completo caricamento delle capacità parassite

impedendo di fatto di misurare un valore veritiero. Invece, per questo caso considerato, serve la tensione registrata dall'ADC nel collegamento in no-pull e di conseguenza le misure non vengono influenzate dalle capacità.

Le prove effettuate sono state in totale tre, una sul pin CC1 e due sul pin CC2. La tensione del generatore esterno è stata impostata a circa 5 V, mentre sono state scelte delle resistenze da $27k\Omega$ e da $150k\Omega$. L'obiettivo di queste misure è quello di verificare il corretto funzionamento dell'albero decisionale del programma ed anche l'accuratezza con cui riesce a identificare le resistenze e i generatori esterni.

Nella variabile *cable_report* vengono registrati i valori di tensione nei nodi in pull-up e no-pull dei tre pin, quindi in totale sei misure. In seguito, nella tabella 5.4 sono riportati i valori registrati, i valori sono stati normalizzati[1]:

| **VPU_CC1** | **VPU_CC2** | **VPU_VBUS** | **VNP_CC1** | **VNP_CC2** | **VNP_VBUS** |
|---|---|---|---|---|---|
| Misure fatte per la resistenza di pull-up su CC1 a $27k\Omega$ | | | | | |
| 56304 | 24832 | 59232 | 51824 | 0 | 56176 |
| 56240 | 24880 | 59296 | 51968 | 0 | 56304 |
| 56432 | 24832 | 59168 | 51760 | 0 | 56224 |
| 56256 | 24688 | 59232 | 51760 | 0 | 56288 |
| 56384 | 24832 | 59152 | 51872 | 0 | 56224 |
| Misure fatte per la resistenza di pull-up su CC2 a $27k\Omega$ | | | | | |
| 24864 | 56288 | 59200 | 0 | 51760 | 56272 |
| 24880 | 56192 | 59184 | 0 | 51840 | 56304 |
| 24848 | 56288 | 59264 | 0 | 51776 | 56288 |
| 24944 | 56112 | 59248 | 0 | 51728 | 56208 |
| 24832 | 56112 | 59104 | 0 | 51824 | 56256 |
| Misure fatte per la resistenza di pull-up su CC2 a $150k\Omega$ | | | | | |
| 24768 | 47232 | 59248 | 0 | 38160 | 56272 |
| 24896 | 47232 | 59264 | 0 | 38240 | 56240 |
| 24816 | 47216 | 59200 | 0 | 38176 | 56288 |
| 24960 | 47216 | 59136 | 0 | 38208 | 56192 |
| 24800 | 47280 | 59168 | 0 | 38160 | 56144 |

**Tabella 5.4:** Elenco delle misure riportate su cable_Report

[1]La formula della normalizzazione applicata è: $n = \frac{V_{\text{pin}} \cdot N\_Fs}{V\_FS}$
dove n=numero intero presente in tabella; $V_{\text{pin}}$= valore in tensione; N_Fs e V_Fs fondo scala digitale e analogico

Con "VPU" si indica la tensione nel nodo in pull-up, mentre con "VNP" si indica la tensione nel nodo in no-pull del rispettivo pin. Questo grafico è di difficile comprensione per via dei valori delle tensioni normalizzate, ma si può osservare che nel caso di resistenza CC1 non c'è tensione nel nodo in no-pull del pin CC2, mentre nel caso di resistenza CC2 non c'è nel nodo in no-pull di CC1. Nel primo caso significa che il generatore esterno è collegato al pin CC1, mentre nel secondo che è collegato al pin CC2. Questo permette all'albero decisionale di considerare il caso del generatore esterno e poi di calcolare la relativa resistenza. Una volta ultimati i calcoli i valori registrati sulla variabile *cable_resistance* vengono inviati tramite bluetooth al computer. Tali dati vengono mostrati nella tabella 5.5, l'unità di misura utilizzata è il kiloohm:

| **R_CC1_PU** | **R_CC1_PD** | **R_CC2_PU** | **R_CC2_PD** | **V_EXT** |
|---|---|---|---|---|
| Valori delle resistenze ai capi di CC1 nel caso di resistenza di pull-up 27$k\Omega$ | | | | |
| 26.873 | inf | inf | 3864 | 4.937 |
| 26.702 | inf | inf | 4476 | 4.943 |
| 27.599 | inf | inf | 3864 | 4.941 |
| 26.337 | inf | inf | 3538 | 4.941 |
| 26.476 | inf | inf | 4050 | 4.933 |
| Valori delle resistenze ai capi di CC2 nel caso di resistenza di pull-up 27$k\Omega$ | | | | |
| inf | 3538 | inf | 27.892 | 4.945 |
| inf | 4252 | inf | 27.896 | 4.943 |
| inf | 4476 | inf | 27.564 | 4.949 |
| inf | 4050 | inf | 27.896 | 4.945 |
| inf | 5636 | inf | 27.712 | 4.932 |
| Valori delle resistenze ai capi di CC2 nel caso di resistenza di pull-up 150$k\Omega$ | | | | |
| inf | 3258 | inf | 151.875 | 4.945 |
| inf | 4720 | inf | 150.625 | 4.941 |
| inf | 3694 | inf | 151.875 | 4.944 |
| inf | 6020 | inf | 150.625 | 4.937 |
| inf | 3538 | inf | 153.125 | 4.949 |

**Tabella 5.5:** Elenco delle misure riportate su cable_Report

Come si può osservare dalla tabella le resistenze calcolate sono molto simili al valore ideale e si riesce bene a fare la distinzione tra la resistenza di 27 k$\Omega$ e 150 k$\Omega$. Anche la tensione rilevata in tutte le misure fatte del generatore esterno è molto simile a 5 V, mentre tutte le resistenze di pull-down sono impostate a inf per indicare la loro assenza. Infine, sono presenti dei valori molto alti di resistenze di qualche megaohm che rappresentano la presenza di un circuito aperto ai capi del pin non connesso con la resistenza.

# CAPITOLO 6

# Conclusioni

In questa tesi è stato trattato lo studio di un sistema in grado di riconoscere i cavi e gli accessori collegati al dispositivo nRF52840. All'inizio è stata fatta una panoramica sul dispositivo e le tecnologie utilizzate, tra cui la connessione tramite connettore USB-C e i sistemi embedded. Queste tecnologie negli ultimi anni hanno trovato parecchi utilizzi e sono diffuse in ogni ambito. Per il dispositivo Grasense preso in considerazione è stato realizzato un sistema embedded in grado di identificare la presenza di un virus all'interno di una soluzione posta in un sensore al grafene. C'è inoltre la possibilità di cambiare sensore e quindi è necessaria la presenza di un sistema di riconoscimento per farne la distinzione.

Dato che i cavi e gli accessori in ingresso possono essere visti come delle resistenze, si è partiti andando ad analizzare i circuiti equivalenti alle connessioni dei cavi USB-C al dispositivo, simulandoli tramite il software LTspice XVII. Sono quindi state ricavate le equazioni relative al circuito per trovare le resistenze viste in ingresso. Inoltre, tramite la simulazione dei circuiti, è stato possibile ricavare i limiti dei valori di tensione da applicare all'albero decisionale per la distinzione dei circuiti.

L'albero decisionale è stato trasposto in linguaggio C per fare la programmazione del microcontrollore sull'IDE Eclipse. Tramite una serie di "if" sono state risolte le condizioni per la decisione del tipo di resistenza vista in ingresso e sono state applicate le equazioni in base al relativo circuito. Infine, tramite una macchina a stati sono stati misurati anche i resistori interni al microcontrollore utilizzati per svolgere le equazioni dei circuiti equivalenti. Questo passaggio è stato fatto per via delle possibili variazioni del valore dei resistori interni a causa del deterioramento o delle condizioni ambientali, come temperatura e umidità. Le misure fatte hanno dato risultati molto positivi e sono stati registrati una serie di valori simili tra loro con una deviazione standard bassa.

Il problema principale riscontrato nello studio è stato il tempo di collegamento della resistenza interna di pull-up dell'ADC integrata nel microcontrollore. Normalmente, negli altri microcontrollori, la resistenza si collega e svolge le misurazioni fino a quando il programmatore non decide di scollegarla ma nel caso del microcontrollore nRF52840 il collegamento e scollegamento avviene automaticamente. Si

può soltanto impostarne il tempo, al massimo 40 microsecondi, che è comunque troppo poco per poter caricare completamente le capacità presenti sul cavo USB-C, non permettendo quindi la corretta misura delle resistenze in ingresso. D'altro canto è stato possibile effettuare delle misure per i casi che non comprendevano l'utilizzo dei valori misurati tramite il collegamento della resistenza di pull-up interna che hanno dato esito positivo. Infatti, è stato possibile distinguere correttamente i valori delle diverse resistenze collegate.

Delle possibili applicazioni future posso essere la misura fatta su diversi istanti temporali, regolati in base al tempo di connessione dell'ADC, per individuare l'andamento della tensione e stimarne una curva tramite l'andamento della tensione relativa alla carica del condensatore. Nel caso questa stima non fosse abbastanza accurata un'altra possibilità potrebbe essere l'integrazione di un ADC collegato ai pin del microcontrollore. Tramite questo collegamento sarebbe possibile calcolare tutte le tipologie di resistenze viste in ingresso andando a cambiare la configurazione dell'ADC. Infine, data la versatilità dei sistemi embedded c'è la possibilità di utilizzare la stessa programmazione per un diverso dispositivo al fine di svolgere lo stesso compito, andando a modificare le parti di codice specifiche del dispositivo.

# Bibliografia


ANDRAMUÑO, J., VEGA, N. e PARRA, P. (2019), «Industry 4.0 Embedded Systems Network», in «2019 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)», p. 1–6, IEEE.

BRANDOLESE, C. e FORNACIARI, W. (2009), «Sistemi Embedded: caratteristiche, tecnologie e mercato», .

COVID, C., TEAM, R., BURRER, S. L., DE PERIO, M. A., HUGHES, M. M., KUHAR, D. T., LUCKHAUPT, S. E., MCDANIEL, C. J., PORTER, R. M., SILK, B. e OTHERS (2020), «Characteristics of health care personnel with COVID-19—United States, February 12–April 9, 2020», *Morbidity and Mortality Weekly Report*, vol. 69 (15), p. 477.

FORNACIARI, W. e BRANDOLESE, C. (2007), *Sistemi embedded. Sviluppo hardware e software per sistemi dedicati*, Pearson Italia Spa.

GAMBARINI, F. (2020), «Embedded automatic transfer switch on low voltage circuit breakers with IEC61850 communication», .

MARWEDEL, P. (2021), *Embedded system design: embedded systems foundations of cyber-physical systems, and the internet of things*, Springer Nature.

MUKHERJEE, S., KUMAR, A. e MAKSIMOVIĆ, D. (2021), «Efficiency-optimized current-source resonant converter for USB-C power delivery», in «2021 IEEE Applied Power Electronics Conference and Exposition (APEC)», p. 500–505, IEEE.

REMIGIO, R. (2020), «A methodology to reconstruct information and enable static analysis in raw binaries», .

ROMAGNOLI, A., D'AGOSTINO, M., PAVONI, E., ARDICCIONI, C., MOTTA, S., CRIPPA, P., BIAGETTI, G., NOTARSTEFANO, V., REXHA, J., PERTA, N., BAROCCI, S., COSTABILE, B. K., COLASURDO, G., CAUCCI, S., MENCARELLI, D., TURCHETTI, C., FARINA, M., PIERANTONI, L., LA TEANA, A., AL HADI, R., CICCONARDI, F., CHINAPPI, M., TRUCCHI, E., MANCIA, F., MENZO, S., MOROZZO DELLA ROCCA, B.,

D'ANNESSA, I. e DI MARINO, D. (2023), «SARS-CoV-2 multi-variant rapid detector based on graphene transistor functionalized with an engineered dimeric ACE2 receptor», *Nano Today*, vol. 48, p. 101 729, URL https://www.sciencedirect.com/science/article/pii/S1748013222003577. (Cited at pages iv, 1 e 4)

SEMICONDUCTORS, N. (2021), «nRF52840», *Nordic Semiconductor. Rev*, vol. 7.

TUCHTENHAGEN, L. (2022), «More Easy Synchronous Transmissions: Expanding Baloo to the nRF52840», *Online-ArXiV Preprint or similar*.

# Sitografia

- Wikipedia – www.wikipedia.org
- Nordic semiconductor – https://infocenter.nordicsemi.com
- Microchip develop helper - https://microchipdeveloper.com/usb:tc-pins
- Sistemi integrati – https://www.sistemi-integrati.net/
- Elettro amici – https://www.elettroamici.org/
- ScienceDirect – https://www.sciencedirect.com/science/article/pii/S1748013222003577
- Informatica e ingegneria online – https://vitolavecchia.altervista.org/le-principali-caratteristiche-di-un-sistema-embedded/
- Stackoverflow – https://stackoverflow.com/questions/1659440/

# Ringraziamenti

Devo ringraziare soprattutto mia madre e mio padre per avermi sempre sostenuto nel percorso universitario, credendo sempre nelle mie potenzialità e rassicurandomi quando avevo dei dubbi.

Ringrazio il Prof. Biagetti per avermi accompagnato nel lavoro fatto durante il tirocinio e la tesi, grazie ai suoi consigli e alla sua disponibilità.

Infine voglio ringraziare tutti i miei amici: Ciabi, Jado, Luca, Momo, Nando, Raso e Sofi che dalle superiori mi hanno fatto sempre portato il sorriso e mi sono stati a fianco.