

Controllo automatico orientamento specchi forno solare

Scar. Francesco

Contenuti

1	Introduzione	2
1.1	Descrizione	2
1.2	Struttura generale	2
2	Schema a blocchi	3
3	Descrizione componenti	5
3.1	Microcontrollore	5
3.2	Motori	5
3.2.1	Controllo di potenza	6
3.3	Termocoppia	7
3.4	Sensore ottico	7
4	Condizionamento segnale sensore temperatura	9
5	Cenni matematici	11
5.1	Sistema coordinate cartesiane - polari	11
5.2	Relazioni trigonometriche	12
6	Algoritmo	14
6.1	Diagramma di flusso generale	14
6.1.1	Setup	14
6.1.2	Loop	15
6.2	Analisi funzioni principali	16
6.2.1	Controllo accelerazione motori	16
6.2.2	Protocollo sensore ottico	19
6.2.3	Edge detection	21
6.2.4	Discesa del gradiente	22

1 Introduzione

1.1 Descrizione

Si desidera realizzare un sistema di controllo automatico dell'orientamento e inclinazione degli specchi di un forno solare per garantire la massima efficienza al variare della posizione relativa del sole nell'arco della giornata. Si desidera inoltre monitorare la temperatura e lo stato di funzionamento generale del sistema mediante un'interfaccia web che permetta di visualizzare un grafico in tempo reale dell'andamento della temperatura.

1.2 Struttura generale

Il sistema è composto da un contenitore di materiale isolante con la faccia superiore di vetro trasparente che permette alla luce solare di irradiare energia all'interno. La luce solare viene inoltre riflessa da due specchi posti ai lati opposti della faccia superiore (in particolare il lato superiore e quello inferiore).

L'intera struttura è solidale ad una base girevole, controllata da un motore, che permette la regolazione dell'azimut, mentre l'inclinazione di ogni specchio è regolata singolarmente in funzione dell'elevazione del sole.

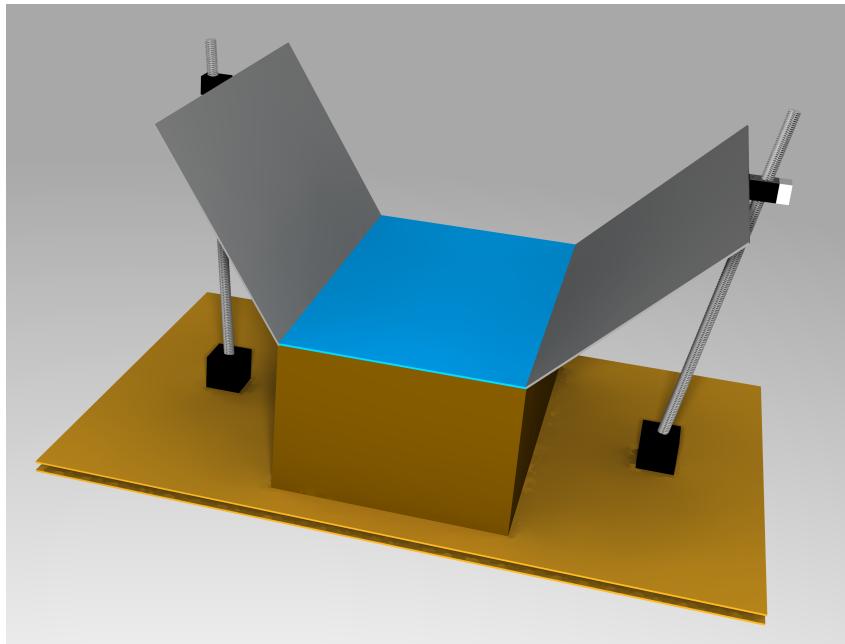


Fig. 1: Modello generico struttura di base

2 Schema a blocchi

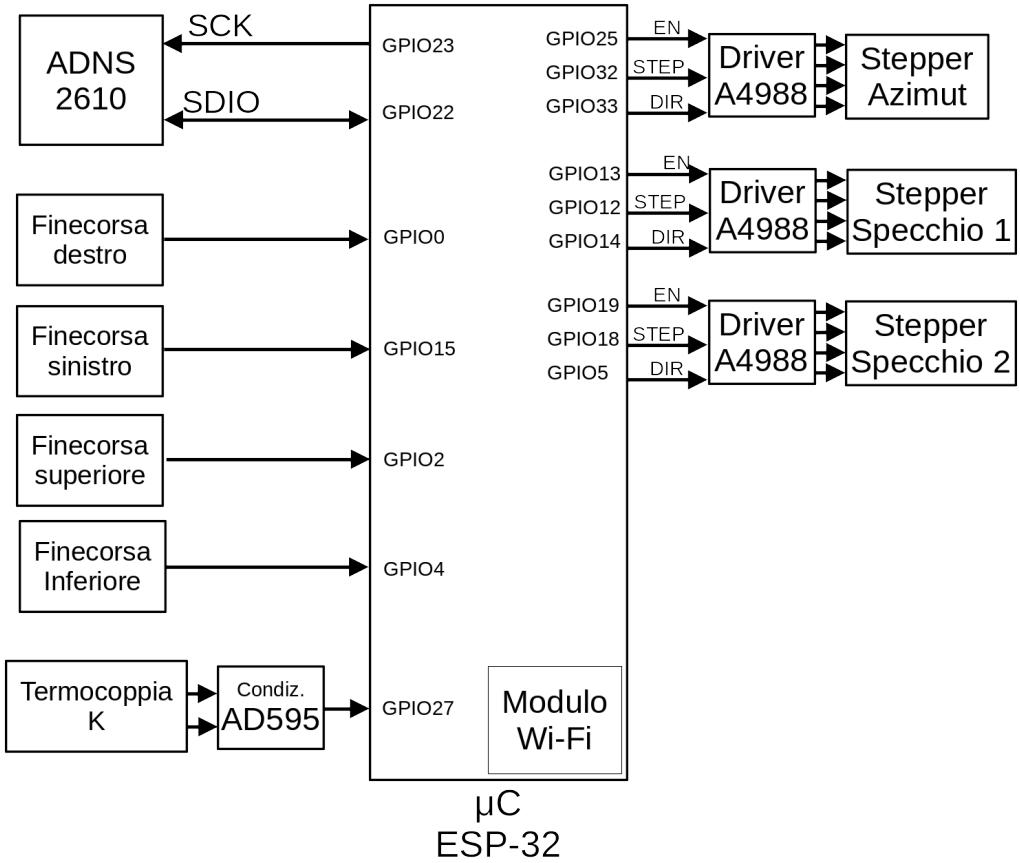


Fig. 2: Schema a blocchi

I pin indicati a destra nel diagramma (GPIO25, GPIO32, GPIO33, GPIO13, GPIO12, GPIO14, GPIO19, GPIO18 e GPIO5) sono uscite, mentre i pin indicati a sinistra (GPIO23, GPIO0, GPIO15, GPIO2, GPIO4, GPIO25) sono input, unica eccezione il GPIO 22, che, come trattato in seguito nella sezione 6.2.2, per la natura del protocollo seriale implementato dal sensore ADNS2610 richiede la possibilità di scambiare dati in modo bidirezionale. Tutti gli ingressi e le uscite utilizzate sono digitali, fatta eccezione del GPIO25 che è collegato internamente ad un ADC a 12 bit integrato nel ESP-32 (per cui non sono necessari ulteriori componenti esterni per effettuare la conversione).

Tutti i componenti direttamente utilizzati dall'ESP-32 sono compatibili con la logica 0V - 3.3V del microcontrollore e hanno assorbimenti molto limitati (inferiori a pochi mA).

Il modulo Wi-Fi è interno al chip dell'ESP-32 e dunque non richiede collegamenti con i pin esterni.

Lo schema circuitale completo è riportato di seguito.

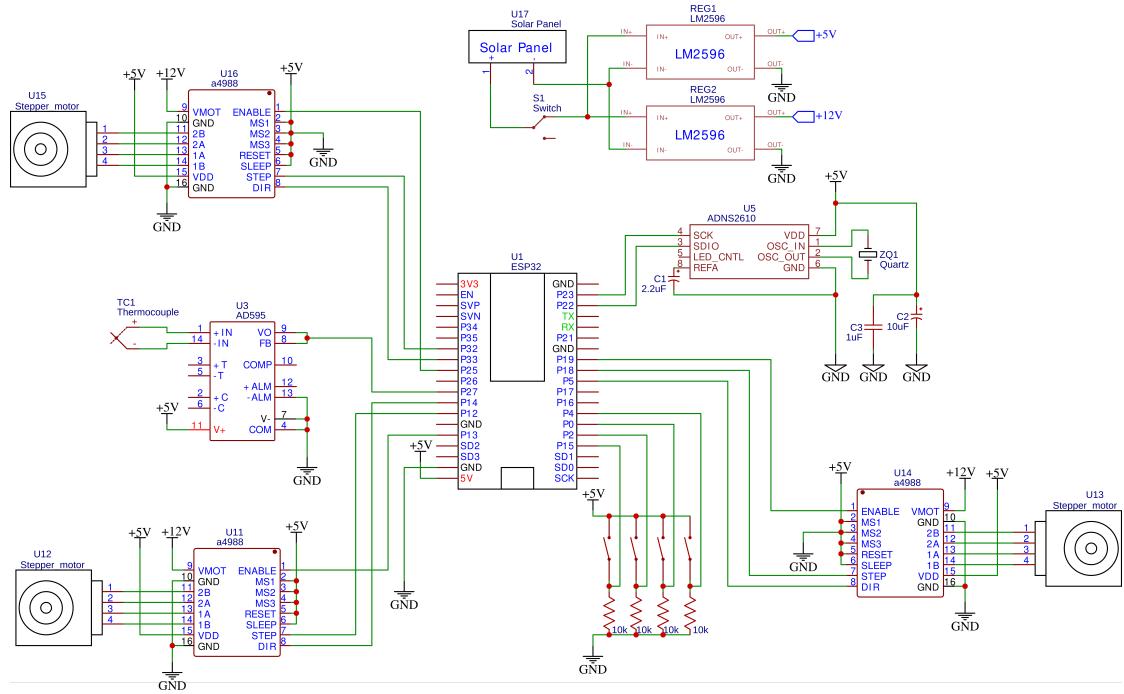


Fig. 3: Schema circuitale

L'alimentazione del sistema può essere fornita da un pannello solare per rendere il dispositivo indipendente dalla rete elettrica; la logica di controllo richiede una tensione di 5V, mentre i motori di 12V, per cui è stato necessario impiegare due regolatori di tensione LM2596. Essi sono convertitori step-down DC-DC regolabili, che permettono di ottenere in uscita una tensione inferiore a quella in ingresso in modo efficiente, limitando le dispersioni di calore.

3 Descrizione componenti

3.1 Microcontrollore

Per la realizzazione di questo sistema di controllo è stato scelto di utilizzare un ESP-32, un microcontrollore a 32 bit con ingressi e uscite compatibili con la logica TTL.

Questo microcontrollore (come il suo predecessore ESP8266) è spesso impiegato in applicazioni IoT, anche in ambito professionale, per la connessione wifi (e bluetooth per quanto riguarda l'ESP-32) integrata in un unico chip a basso costo, permettendo ai diversi dispositivi di comunicare tra loro.

Per semplicità costruttiva di seguito ci si riferirà alla specifica implementazione mediante la scheda di prototipazione rapida NodeMCU.

3.2 Motori

Dovendo regolare con accuratezza la rotazione dei motori, per il controllo di azimuth e inclinazione, si è deciso di impiegare dei motori passo-passo, o stepper, in particolare il modello NEMA-17.

Questi attuatori sono motori ad impulsi, in quanto composti da bobine disposte in modo tale che, se eccitate in ordine opportuno, permettono di produrre una rotazione dell'asse inferiore a 1.8° per ogni step.

Il principio di funzionamento è schematizzato nell'immagine seguente.

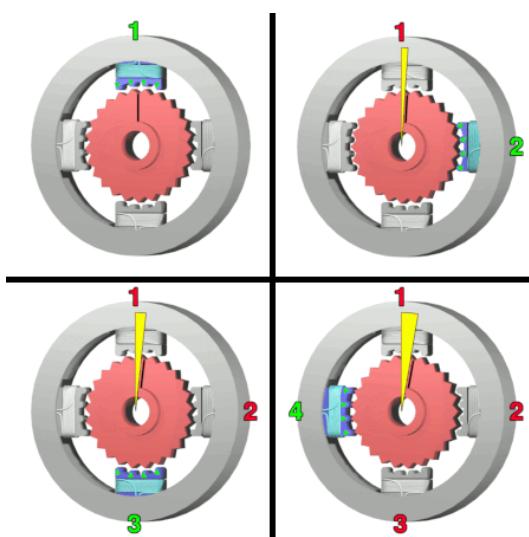


Fig. 4: Fasi motore passo-passo, Wikimedia Commons ¹

3.2.1 Controllo di potenza

Per il controllo dell'attivazione delle bobine si è scelto di utilizzare il modulo A4988 (molto diffuso nelle macchine a controllo numerico hobbistiche), che permette il controllo della rotazione del motore mediante un segnale di direzione (orario/antiorario) e un segnale di step.

La logica di attivazione degli avvolgimenti del motore è gestita dall'integrato mediante due ponti H a fet, uno per bobina, che permettono la regolazione della potenza ad ogni avvolgimento.

Il funzionamento interno, come mostrato dal costruttore nel datasheet, può essere schematizzato con il seguente diagramma a blocchi.

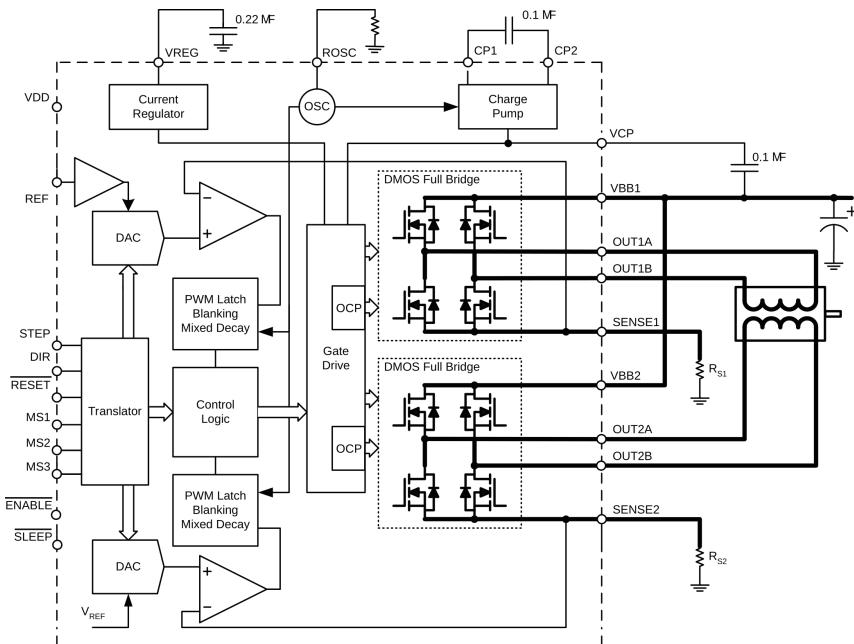


Fig. 5: Diagramma funzionamento interno A4988, datasheet

Gli ingressi MS_1 , MS_2 e MS_3 permettono la selezione del rapporto di microstepping, infatti oltre alla modalità full-step, in cui le singole bobine vengono eccitate con controllo ON-OFF in successione, è possibile realizzare un controllo microstepping con l'attivazione progressiva delle bobine consecutive, permettendo di mantenere il rotore tra due stati successivi, garantendo una maggiore fluidità di movimento, riducendo oscillazioni, vibrazioni e dunque sollecitazioni meccaniche

¹Questa immagine è tratta e modificata da Wikimedia Commons, informazioni riguardo l'autore originale, modifiche successive e licenza GNU Free Documentation License possono essere reperite al seguente link: commons.wikimedia.org/wiki/File:StepperMotor.gif

della struttura fisica.

Con riferimento all'immagine di Fig. 5 si può osservare che il dispositivo permette di regolare la coppia massima del motore mediante la regolazione di una tensione di riferimento che agisce sulla corrente massima che viene fornita alle bobine (misurata dalle resistenze di shunt R_{s1} e R_{s2} che forniscono un feedback al sistema di controllo).

3.3 Termocoppia

Per la rilevazione della temperatura si è scelto di utilizzare una termocoppia di tipo K a giunzione Chromel-Alumel (leghe metalliche a base di nichel), sensore molto diffuso che permette di misurare temperature nel range da circa -200 °C a oltre 1300 °C, con una sensibilità di $41 \frac{\mu V}{^{\circ}C}$.

Le termocoppe sfruttano l'effetto Seebeck, un effetto termoelettrico che produce una differenza di potenziale tra due punti di un materiale conduttore proporzionale alla differenza di temperatura tra i due punti (detti giunzione calda e giunzione fredda); l'intensità di questo effetto varia a seconda del materiale in considerazione e quindi, utilizzando due metalli differenti, è possibile calcolare la temperatura del giunto caldo nota la differenza di potenziale e la temperatura del giunto freddo. Per rendere apprezzabili tali variazioni di tensione è necessario impiegare un amplificatore da strumentazione, in particolare si è scelto l'AD595, che garantisce la compensazione interna della temperatura della giunzione fredda e fornisce in uscita una tensione proporzionale alla temperatura secondo la relazione:

$$V_{out} = T \cdot 10 \frac{mV}{^{\circ}C} \quad (1)$$

Nella sezione 4 si tratterà più approfonditamente il condizionamento di questo segnale per sfruttare l'intero range dell'ADC del microcontrollore.

3.4 Sensore ottico

Il sensore principale di questo sistema di controllo è l'ADNS-2610, che viene utilizzato dal microcontrollore per calcolare l'azimuth e l'elevazione relativi del sole.

Questo integrato nasce come sensore per mouse ottici, che dispongono di una fotocamera a bassa risoluzione e confrontano le immagini di due frame successivi per individuare lo spostamento relativo alla superficie sottostante, nella porzione



Fig. 6: Sensore ADNS-2610

di sinistra della figura 6 si osserva la matrice 18x18 degli elementi fotosensibili (l'immagine è stata realizzata con un ingrandimento e l'area effettiva coperta dalla griglia è di circa 2 mm^2).

Il principio di funzionamento è identico ad una camera a foro di spillo, in cui l'immagine viene proiettata sullo schermo fotosensibile attraverso un foro circolare di piccole dimensioni.

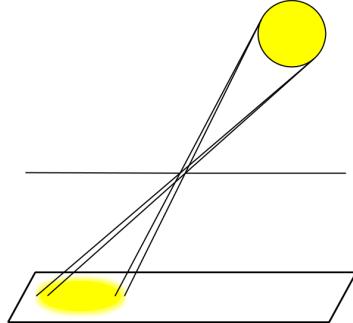


Fig. 7: Schema immagine stenoscopica

La posizione dell'immagine sullo schermo, a parità delle altre condizioni, varia al variare della distanza tra il foro e il sensore stesso, in particolare, aumentando tale distanza il dispositivo avrà un angolo di campo minore, viceversa, diminuendo la distanza foro-rivelatore, si otterrà un angolo di campo maggiore.

In cui la relazione tra x e l'angolo θ del raggio rispetto allo zenit è:

$$x = h \cdot \tan(\theta) \quad (2)$$

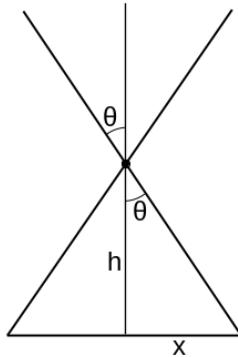


Fig. 8: Diagramma camera a foro di spillo

Maggiori dettagli riguardo l’interfacciamento in protocollo con il microcontrollore sono forniti nella sezione 6.2.2.

4 Condizionamento segnale sensore temperatura

Considerata l’uscita del AD595 descritta dall’equazione (1) nella sezione 3.3 e considerata la risoluzione di 12 bit degli ADC del microcontrollore, si ottiene una minima variazione apprezzabile (idealmente) inferiore a 0.12 °C. Per questo motivo non è necessario realizzare un secondo condizionamento del segnale, in quanto tale valore di incertezza è trascurabile rispetto alle temperature da misurare e alle altre fonti di errore presenti; per maggiore completezza si riporta comunque un possibile condizionamento di tale segnale.

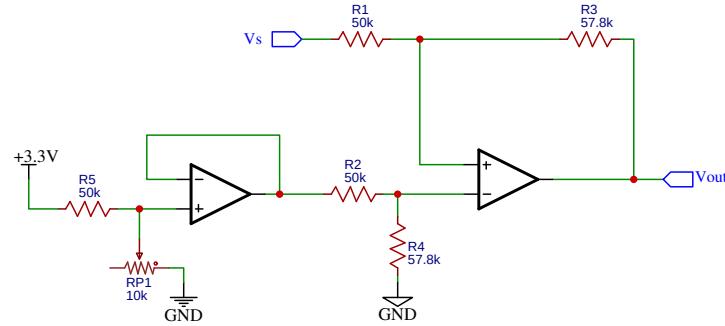
Si limiti il range di interesse da 15°C a 300°C in cui rientra la regolare temperatura di un forno solare. Secondo la relazione (1) l’uscita in tensione del sensore varierà all’interno dell’intervallo [150mV; 3.0V], mentre l’ADC integrato nel microcontrollore, con V_{ref} standard, ammette ingressi nel range [0V; 3.3V]; di seguito si suppongono tutti i dispositivi impiegati come ideali. È possibile applicare un offset e successivamente un’amplificazione opportunamente calcolati di seguito.

$$V_{offset} = V_{Smin} - V_{ADCmin} = 150mV \quad (3)$$

$$A = \frac{V_{ADCmax} - V_{ADCmin}}{V_{Smax} - V_{Smin}} \simeq 1.16 \quad (4)$$

Tali valori possono essere ottenuti utilizzando un amplificatore operazionale in

configurazione differenziale come segue.



In cui il potenziometro (o trimmer) RP_1 deve essere opportunamente regolato in fase di calibrazione per ottenere l'offset corretto.

Essendo $R_1 = R_2$ e $R_3 = R_4$ l'amplificazione risulta $A = \frac{R_3}{R_1} \simeq 1.16$, stesso valore ottenuto sostituendo i rispettivi valori numerici nell'equazione (4). Riguardo alle due resistenze da $57.8\text{k}\Omega$, tale può essere ottenuto dalla serie di una resistenza da $56\text{k}\Omega$ e da $1.8\text{k}\Omega$, entrambi valori appartenenti alla serie E12.

Combinando le equazioni (1), (3) e (4), la funzione di trasferimento dell'amplificatore differenziale e considerata la conversione dell'ADC a 12 bit dell'ESP-32 (valore da 0 a 4095), si ottiene:

$$Val_{ADC} = \left(T \cdot 10 \left[\frac{mV}{^{\circ}C} \right] - 150[mV] \right) \cdot \frac{57.8}{50} \cdot \frac{4095}{3.3[V]} \simeq \left(\frac{T}{1[^{\circ}C]} - 15 \right) \cdot 14.345 \quad (5)$$

chiaramente approssimato all'intero, da cui si ricava

$$T = \left(\frac{Val_{ADC}}{14.345} + 15 \right) [^{\circ}C] \quad (6)$$

L'aggiunta di questo condizionamento richiede inoltre ulteriori componenti per la generazione dell'alimentazione duale per l'operazionale, quindi, in aggiunta a quanto già discusso all'inizio di questa sezione, non è stato ritenuto necessario impiegare condizionamenti successivi all'integrato AD595.

5 Cenni matematici

5.1 Sistema coordinate cartesiane - polari

Si ricorda che è possibile definire in modo univoco un punto A sul piano cartesiano mediante le sue coordinate (x_A, y_A) . Lo stesso punto A può anche essere descritto univocamente dalla distanza r dal centro e dall'angolo α che esso forma con il centro e il semiasse positivo delle ascisse. È possibile ottenere la rappresentazione cartesiana dalle coordinate polari e viceversa secondo le relazioni:

$$\begin{aligned} x_A &= r \cdot \cos \alpha \\ y_A &= r \cdot \sin \alpha \end{aligned} \tag{7}$$

$$\begin{aligned} r &= \sqrt{x_A^2 + y_A^2} \\ \alpha &= \text{atan} \left(\frac{y_A}{x_A} \right) \end{aligned} \tag{8}$$

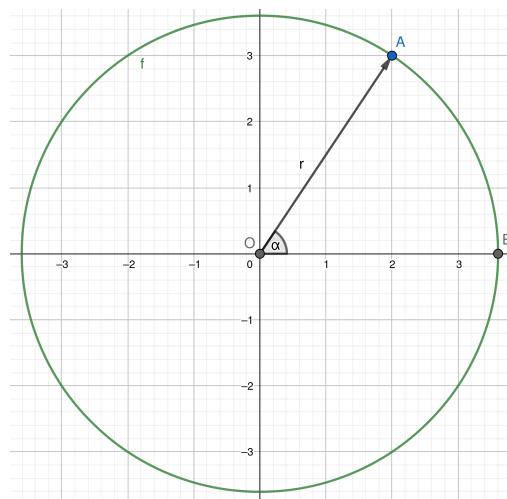


Fig. 9: Rappresentazione coordinate polari su piano cartesiano, geogebra

In particolare, una volta determinata la posizione del centro del sole nell'immagine acquisita, ed espresso tale punto in coordinate polari (r, α) , r è direttamente legato all'elevazione effettiva del sole secondo la relazione (2), mentre α indica la diffidenza tra l'azimuth del sole e la posizione attuale della base girevole. Si osserva, infatti, che le due quantità possono essere trattate indipendentemente, in quanto ruotando la base varierà α , mentre r sarà, idealmente, costante (il centro si muove lungo una

circonferenza di raggio r); viceversa, una variazione della sola elevazione non varia il valore di α (il punto si muove su uno stesso raggio della circonferenza originale).

5.2 Relazioni trigonometriche

Per ottimizzare la potenza assorbita dal forno si vuole che la totalità della luce che investe gli specchi venga riflessa all'interno del forno, attraverso il vetro trasparente posto sulla faccia superiore della scatola. La situazione può essere schematizzata come mostrato nella figura 10.

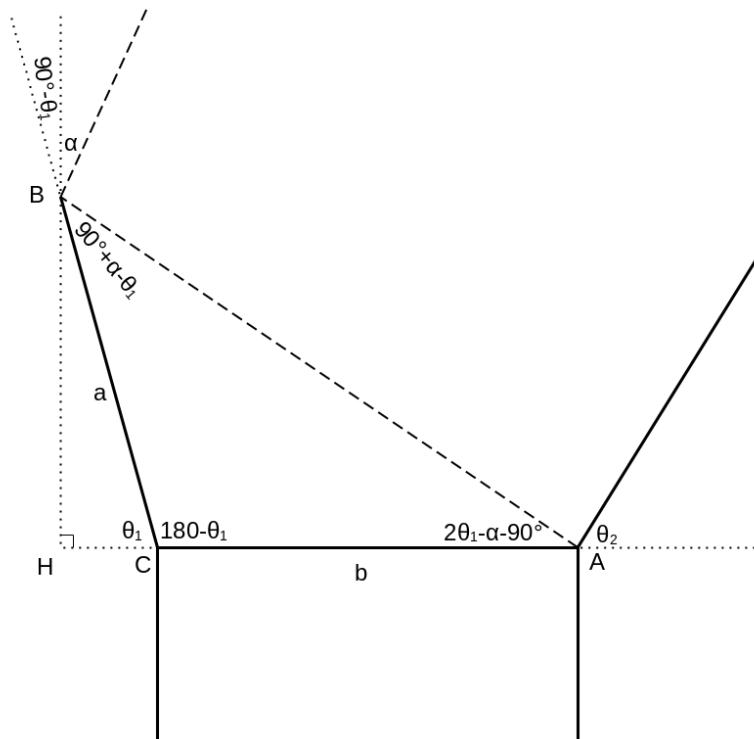


Fig. 10: Schema riflessioni raggi solari

Dal rispetto del teorema di Carnot e dalle relazioni trigonometriche nei triangoli rettangoli si ottiene:

$$\sqrt{a^2 + b^2 + 2ab \cos(\theta_1)} \cdot \cos(2\theta_1 - \alpha) + a \cdot \sin(\theta_1) = 0 \quad (9)$$

In modo equivalente si ottiene

$$\sqrt{a^2 + b^2 + 2ab \cos(\theta_2)} \cdot \cos(2\theta_2 + \alpha) + a \cdot \sin(\theta_2) = 0 \quad (10)$$

Risolvere algebricamente queste equazioni risulta particolarmente laborioso, sebbene

non difficile, e considerato che il range di interesse si limita a $\alpha \in [0, \frac{\pi}{2})$ e $\theta_1, \theta_2 \in (0, \pi)$ si può valutare di risolvere per approssimazione tali uguaglianze per trovare il valore degli angoli cercati per uno specifico valore di α misurato.

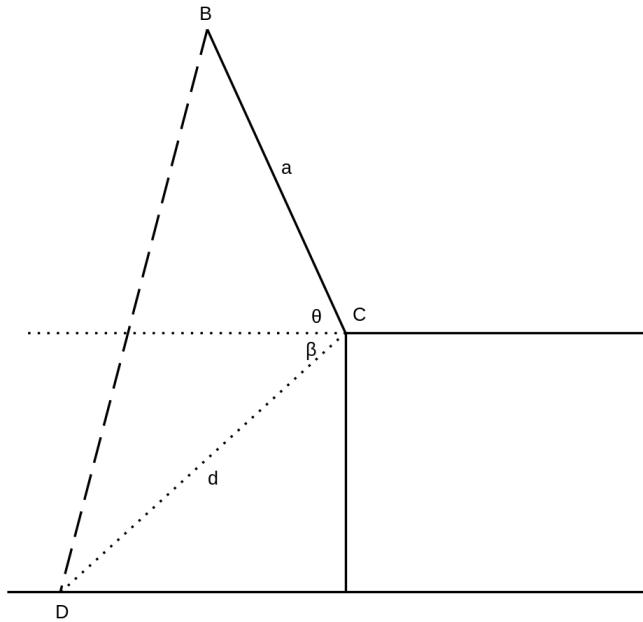


Fig. 11: Schema inclinazione specchi

L'inclinazione degli specchi viene regolata mediante un sistema vite-dado, che può essere schematizzato come mostrato nella figura 11.

In cui sia d che β sono noti, in quanto determinati dalla costruzione e θ è l'angolo calcolato in precedenza. È quindi possibile calcolare la distanza \overline{BD} , dipendente dalla rotazione del motore, applicando nuovamente il teorema di Carnot:

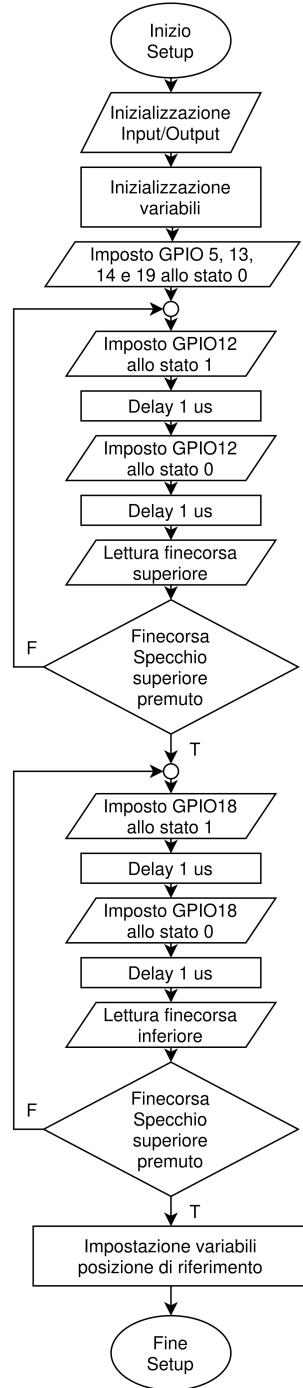
$$\overline{BD} = \sqrt{a^2 + d^2 - 2ab \cdot \cos(\beta + \theta)} \quad (11)$$

Infine, nota tale distanza, è possibile calcolare la rotazione del motore per portarlo alla posizione voluta, ma è necessario conoscere anche la posizione attuale assoluta dello specchio, per questo motivo è necessario un finecorsa che all'accensione permetta al controllore di avere un riferimento assoluto (analogia alla procedura di homing per le macchine a controllo numerico).

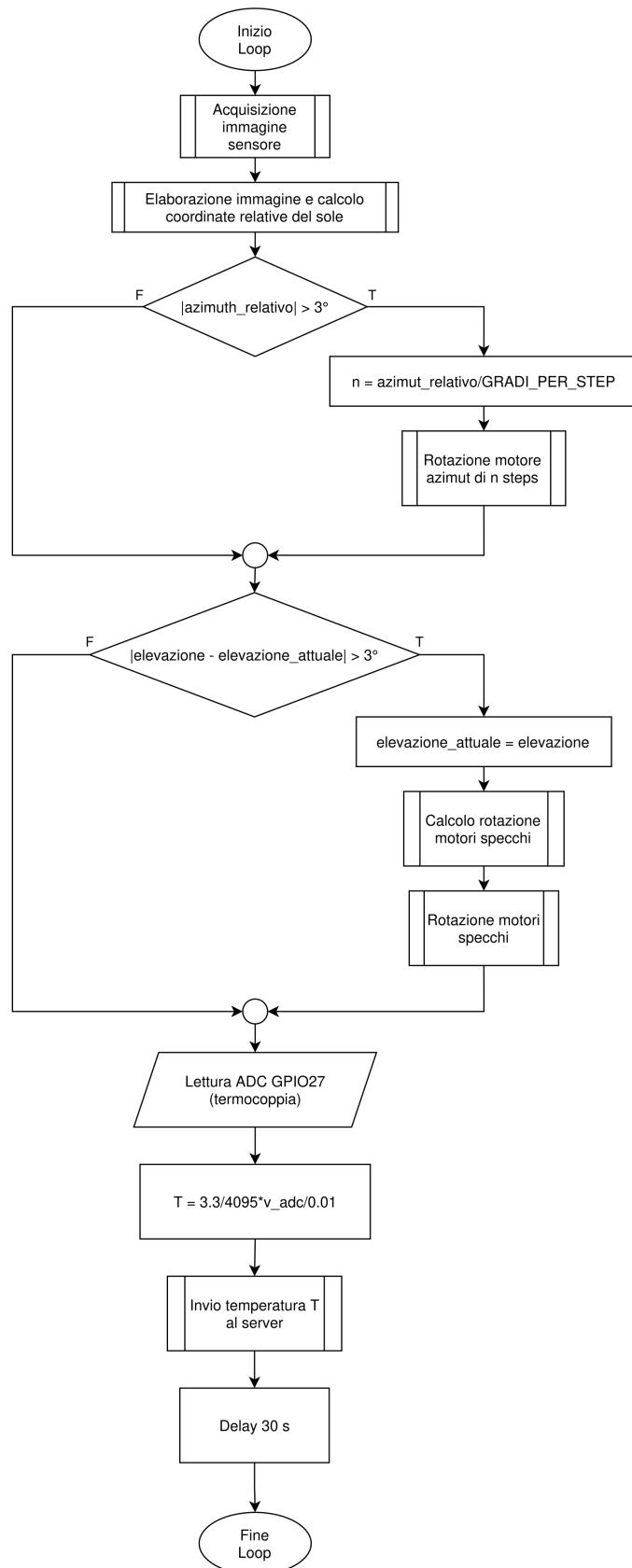
6 Algoritmo

6.1 Diagramma di flusso generale

6.1.1 Setup



6.1.2 Loop



6.2 Analisi funzioni principali

6.2.1 Controllo accelerazione motori

Il driver A4988 permette il controllo della rotazione dei motori mediante un segnale di direzione e uno di step: ogni fronte di salita del segnale di step comporta un incremento della posizione del motore nella direzione selezionata dal segnale di direzione, tale incremento dipende, oltre che dalle caratteristiche strutturali del motore in questione, dal rapporto di micro-stepping.

La rotazione è quindi composta da brevi scatti successivi, maggiore è il numero di microstep per step e maggiore sarà la fluidità del movimento.

Un semplice controllo ON-OFF, con accensione dei motori direttamente ad una velocità definita e costante, provoca elevate accelerazioni che portano ad una maggiore usura dei componenti meccanici, oltre che possibili oscillazioni attorno alla posizione desiderata, per questo motivo risulta opportuno variare la velocità gradualmente per ottenere accelerazioni e decelerazioni costanti.

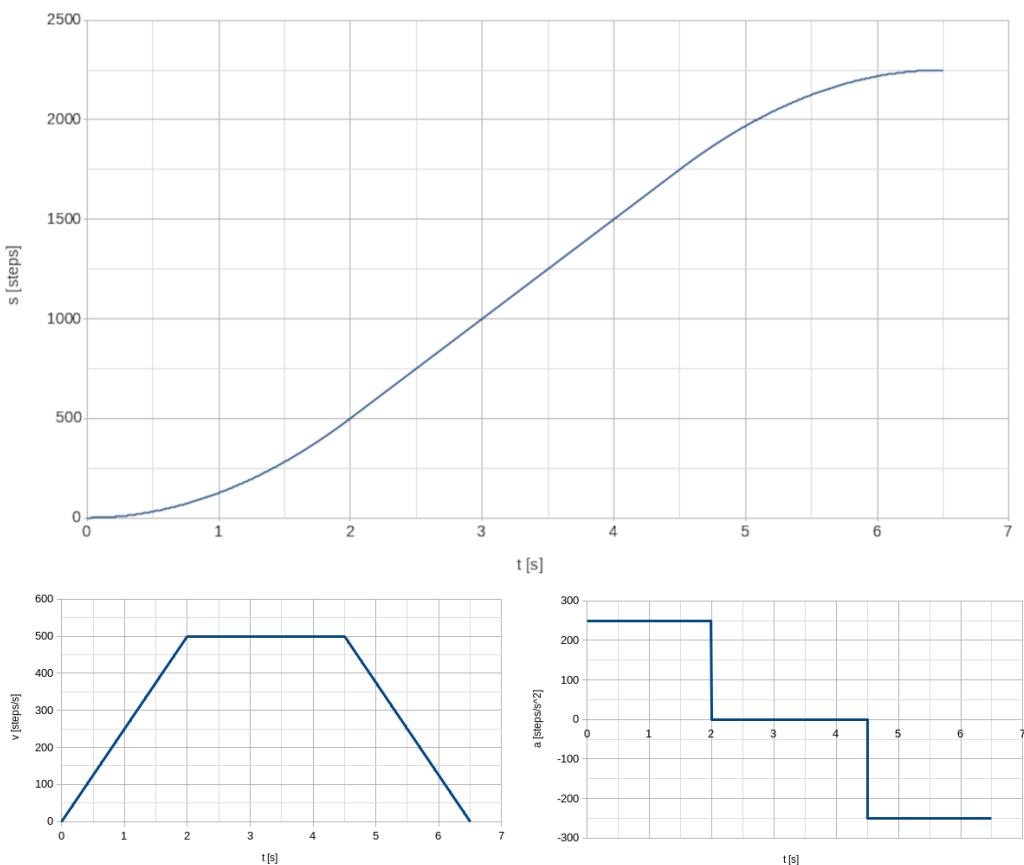


Fig. 12: Andamento steps nel tempo

L'andamento nel tempo mostrato in figura 12 può essere suddiviso in 3 fasi: una iniziale di accelerazione (ad accelerazione costante), un'eventuale fase centrale a velocità costante ed una finale di decelerazione (con accelerazione costante e opposta a quella iniziale).

Ricordando le equazioni del moto uniformemente accelerato, si ha:

$$s(t) = s_0 + v_0 \cdot t + \frac{1}{2} \cdot a \cdot t^2 \quad (12)$$

$$v(t) = v_0 + a \cdot t \quad (13)$$

Una possibile implementazione per il microcontrollore in considerazione può essere la seguente².

²Questa funzione fa parte del programma completo nel suo complesso, per cui vi sono riferimenti a variabili globali e costanti non definite all'interno di questa funzione.

```

30     condition = ((( microsteps / MICROSTEPPING_RATIO - (abs(n_steps) - acceleration_limit_steps)) * 1000000) > (speed * (t - t_offset) - 0.5 * acceleration * ((t - t_offset) / 1000.0) * ((t - t_offset) / 1000.0)));
31     if (condition) {
32         if (t - t_offset > speed / acceleration * 1000000) {
33             condition = false;
34         }
35     }
36 }
37 } while (condition);
38
39 if (!digitalRead(endstop)) {
40     digitalWrite(step_pin, HIGH);
41     delayMicroseconds(2); // Pulse must be at least 1us (from A4988 datasheet)
42     digitalWrite(step_pin, LOW);
43     delayMicroseconds(1); // Pulse must be at least 1us (from A4988 datasheet)
44     if (constant_phase) {
45         if (microsteps / MICROSTEPPING_RATIO >= abs(n_steps) - acceleration_limit_steps) {
46             constant_phase = false;
47             deceleration_phase = true;
48             t_offset = micros();
49         }
50     } else if (acceleration_phase) {
51         if (microsteps / MICROSTEPPING_RATIO >= acceleration_limit_steps) {
52             acceleration_phase = false;
53             constant_phase = true;
54             t_offset = micros();
55         }
56     }
57 } else {
58     if (disable_motor) {
59         digitalWrite(enable_pin, HIGH);
60     }
61     return microsteps;
62 }
63 }
64 if (disable_motor) {
65     digitalWrite(enable_pin, HIGH);
66 }
67 }
```

Codice 1: Controllo velocità motori

6.2.2 Protocollo sensore ottico

Il sensore ADNS2610 utilizza un protocollo digitale sincrono che permette la lettura e scrittura di registri interni con diverse funzioni. Per l'applicazione in questione è rilevante il registro 0x08, che alla i -esima lettura dell'ultima scrittura in questo registro fornisce un byte riguardante l' i -esimo pixel (ripartendo dal primo quando si raggiunge l'ultimo pixel): il bit più significativo indica se il pixel è o meno il primo dell'immagine; il bit successivo è posto a 1 se e solo se il dato è corretto e può essere letto; i restanti 6 bit rappresentano la luminosità del pixel in questione. Ogni comunicazione è iniziata dal master, che controlla il clock, e trasmette un bit che indica se intende scrivere o leggere un registro, seguito da 7 bit indicanti l'indirizzo del registro stesso (MSB-first).



Fig. 13: Diagramma temporale protocollo comunicazione

Nel diagramma precedente si osserva chiaramente una prima comunicazione che scrive nel registro 0x08 (per resettare lo stato interno e leggere successivamente il primo pixel) seguita da una lettura dello stesso registro che ritorna la luminosità del primo pixel.

Un possibile esempio di implementazione può essere il seguente³.

```

1 int i = 0;
2 while (i < 18 * 18) {
3     delayMicroseconds(1);
4     uint8_t data = read(0x08);
5     if (data & 0b01000000) {
6         matrix[i / 18][i % 18] = data;
7         i++;
8     }
9 }
10
11

```

³Come nell'esempio precedente vi sono variabili globali non definite, in particolare la variabile "matrix" conterrà la matrice rappresentante l'immagine acquisita

```

12 void write(uint8_t register_addr, uint8_t data) {
13     register_addr |= 0b10000000; // Set MSB to 1
14     write_byte(register_addr);
15     delayMicroseconds(200);
16     write_byte(data);
17     delayMicroseconds(200);
18 }
19
20 uint8_t read(uint8_t register_addr) {
21     register_addr &= 0b01111111; // Set MSB to 0
22     write_byte(register_addr); // High-Z mode
23     pinMode(SDIO, INPUT); // T_hold >= 100us from specifications
24     delayMicroseconds(100);
25
26     uint8_t read_data = 0;
27     for (int8_t i = 7; i >= 0; i--) {
28         digitalWrite(SCK, LOW); // Start clock signal
29         delayMicroseconds(50); // Same as mouse application
30         digitalWrite(SCK, HIGH); // Data must be read on transition from 0 to 1
31         read_data += digitalRead(SDIO) << i; // Data line must be read here
32         delayMicroseconds(50); // Same as mouse application
33     }
34     return read_data;
35 }
36
37
38 void write_byte(uint8_t byte_val) {
39     pinMode(SDIO, OUTPUT);
40
41     for (int8_t i = 7; i >= 0; i--) {
42         digitalWrite(SCK, LOW); // Start clock signal
43         digitalWrite(SDIO, byte_val & (1 << i)); // Data line must be set here
44         delayMicroseconds(50); // Same as mouse application
45         digitalWrite(SCK, HIGH); // A2610 latchs data on transition from 0 to 1
46         delayMicroseconds(50); // Same as mouse application
47     }
48 }
```

Codice 2: Lettura immagine da sensore

6.2.3 Edge detection

Per determinare i contorni di una figura in un'immagine i sistemi di visione, solitamente, valutano la differenza di colore, o luminosità, tra pixel successivi, in quanto maggiore è tale differenza e più è netto il contorno composto da quei pixel.

Matematicamente possiamo definire per una matrice $I m \times n$, una seconda matrice G di dimensioni $(m - 2) \times (n - 2)$ tale che:

$$G_{y,x} = \sqrt{(I_{y,x} - I_{y,x+2})^2 + (I_{y,x} - I_{y+2,x})^2} \quad (14)$$

$$\forall \quad 1 \leq x \leq n - 2 \quad 1 \leq y \leq m - 2$$

Che, nel caso in cui I rappresenti la luminosità dei pixel dell'immagine, equivale al modulo del vettore composto da due componenti ortogonali costituite, rispettivamente, dalla differenza di luminosità tra il pixel precedente-successivo (E-C) e superiore-inferiore (B-D) a quello considerato (A in figura).

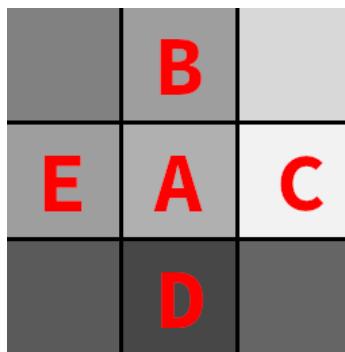


Fig. 14: Disposizione pixel

L'implementazione ⁴ di questa operazione è molto semplice, in quanto consiste solamente nel ciclare tra gli indici della matrice e applicare la formula (14).

```

1 void calculate_gradient_matrix () {
2     for (int8_t y = 0; y < 16; y++) {
3         for (int8_t x = 0; x < 16; x++) {
4             gradient_modules[y][x] = sqrt(pow(matrix[y + 2][x + 1] - matrix[y][x + 1], 2) + pow(matrix[y + 1][x + 2] - matrix[y + 1][x], 2)) + 0.5;
5         }
6     }
7 }
```

Codice 3: Calcolo matrice edge-detection

⁴Come del codice 2 la variabile globale "matrix" rappresenta i pixel dell'immagine acquisita, mentre "gradient_modules" conterrà il risultato dell'operazione

Nella seguente immagine si può osservare l'applicazione tale algoritmo ad una immagine acquisita dal sensore.

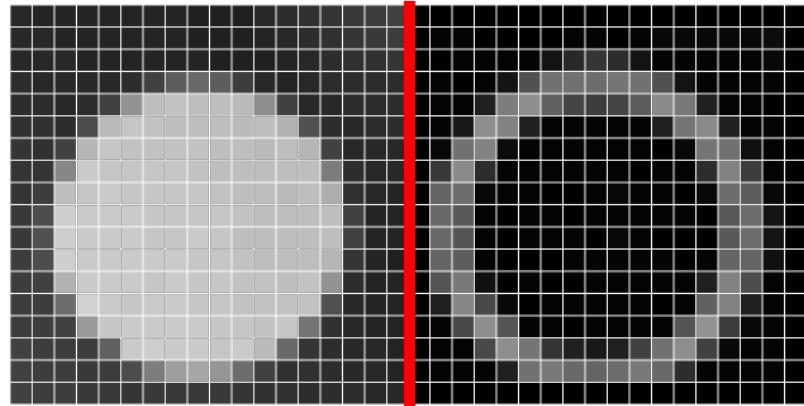


Fig. 15: Edge-detection su immagine acquisita dal sensore

6.2.4 Discesa del gradiente

Per ottenere il centro dell'immagine del sole, che approssimiamo ad un cerchio, è sufficiente trovare la circonferenza che meglio approssima il contorno esterno del cerchio, definito dai punti in cui si ha una differenza più netta tra pixels successivi. A tal fine si deve minimizzare la distanza geometrica tra la circonferenza e i punti con maggiore modulo del vettore trattato nella sezione 6.2.3; tale distanza è schematizzata nella seguente figura con il tratto e .

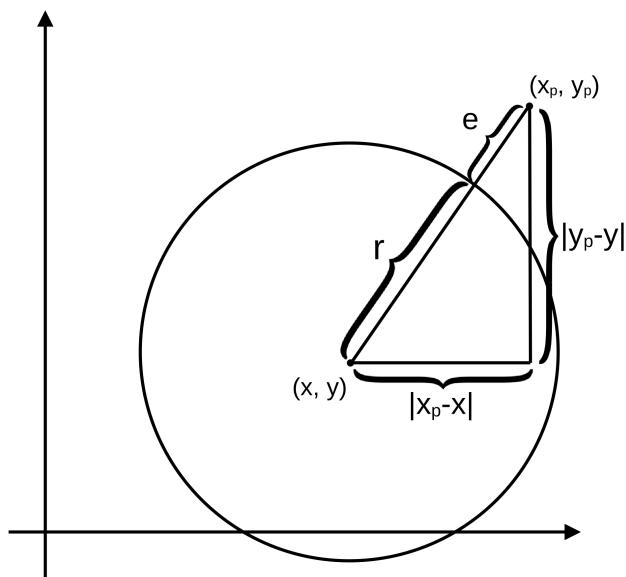


Fig. 16: Distanza punto - circonferenza

Applicando il teorema di pitagora, per differenza, si ottiene che la distanza e è definita in funzione delle coordinate del centro e del punto in questione, secondo la relazione

$$e = \left| \sqrt{(x_p - x)^2 + (y_p - y)^2} - r \right| \quad (15)$$

Possiamo quindi definire la seguente funzione⁵

$$f(x, y, r) = \sum_{y_p=0}^{15} \sum_{x_p=0}^{15} \left[G_{y_p, x_p} \cdot \left| \sqrt{(x_p - x)^2 + (y_p - y)^2} - r \right| \right] \quad (16)$$

in cui la matrice G_{y_p, x_p} rappresenta il modulo del gradiente dell'immagine del punto (x_p, y_p) , come trattato nella sezione 6.2.3, ed è quindi un coefficiente reale. Tale sommatoria risulta quindi una somma ponderata delle distanze tra la circonferenza in considerazione e tutti i punti della matrice discussa alla sezione precedente. Il modulo del gradiente sarà maggiore per i punti sul contorno dell'immagine del sole, che quindi avranno un peso maggiore rispetto agli altri.

Il problema risulta quindi equivalente alla ricerca del minimo di tale funzione, per questo scopo può essere utile considerare il caso più semplice di una funzione ad una sola variabile.

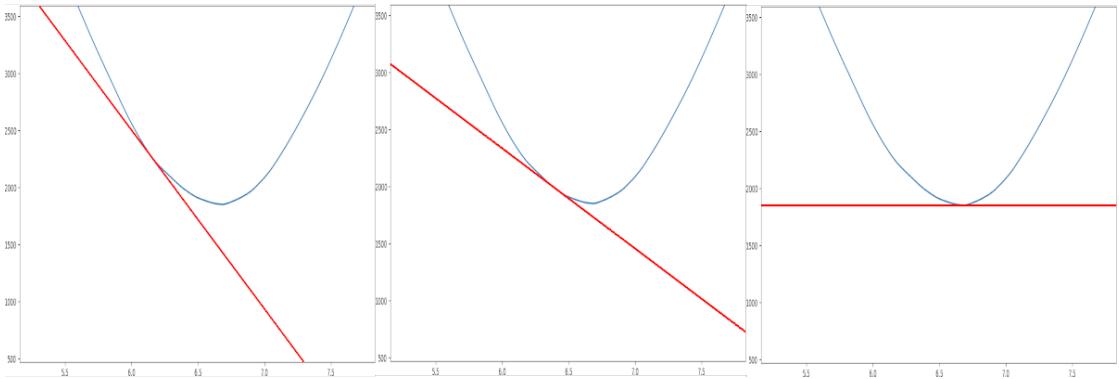


Fig. 17: Ricerca minimo - 2D

Nell'immagine 17 si osserva la funzione rappresentativa dell'errore al variare della variabile r , fissati x e y , nel caso dell'immagine rappresentata in figura 15, mentre la retta rossa indica la tangente alla curva. Si osservi che, partendo da un punto

⁵Per semplicità di seguito si assume che per una matrice $m \times n$ gli indici degli elementi assumano valori partendo da 0, ad esempio l'elemento della prima riga e colonna sarà $G_{0,0}$, mentre l'elemento dell'ultima riga e colonna sarà $G_{m-1,n-1}$. Questo perché nella maggior parte dei linguaggi di programmazione l'indice del primo elemento di un array è proprio 0.

iniziale sulla curva, è possibile individuare un minimo locale variando l'ascissa corrente di un valore $-k \cdot f'(r)$, con k scelto opportunamente (e positivo), e iterando questo procedimento fino ad ottenere l'approssimazione desiderata.

È possibile applicare lo stesso algoritmo per una funzione in due variabili, rappresentabile come una superficie in uno spazio tridimensionale.

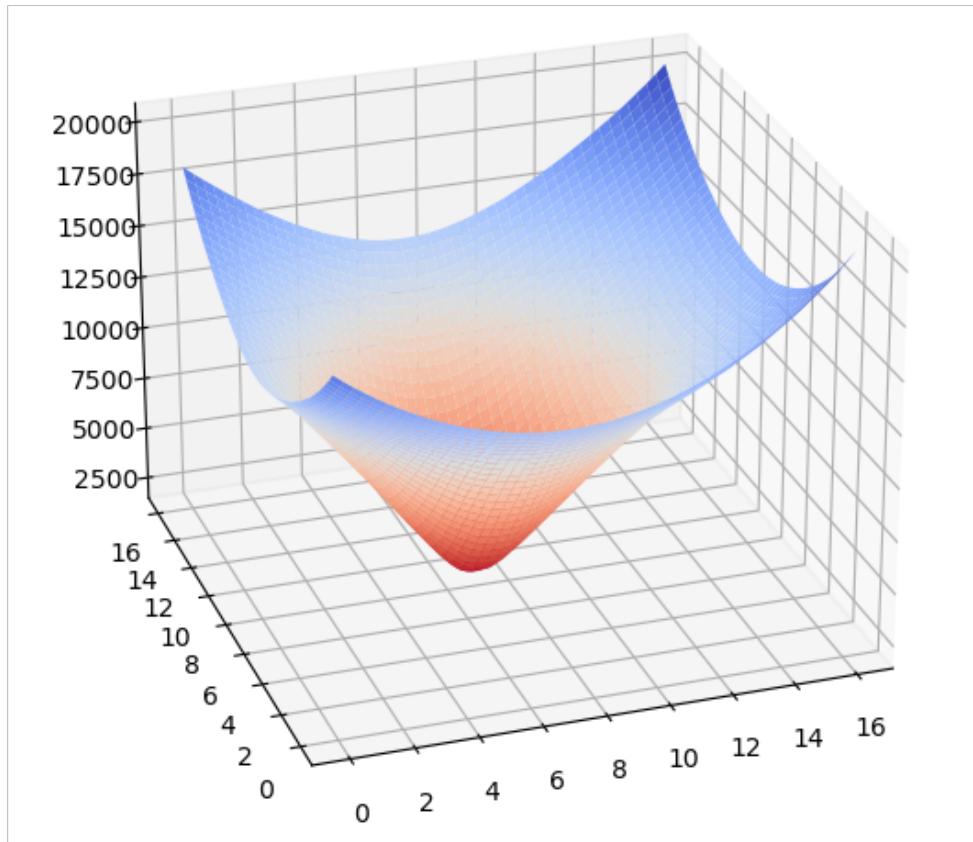


Fig. 18: Ricerca minimo - 3D

In cui il minimo può essere trovato nuovamente iterando più volte un procedimento analogo al precedente, considerando entrambe le derivate parziali della funzione.

Nella situazione in questione i le variabili sono tre (x , y e r), per cui la funzione può essere rappresentata geometricamente in uno spazio a 4 dimensioni, che associa ad ogni punto dello spazio tridimensionale un valore proporzionale all'errore, per questo motivo possiamo rappresentare il valore assunto dalla funzione mediante una scala cromatica e visualizzarne l'andamento.

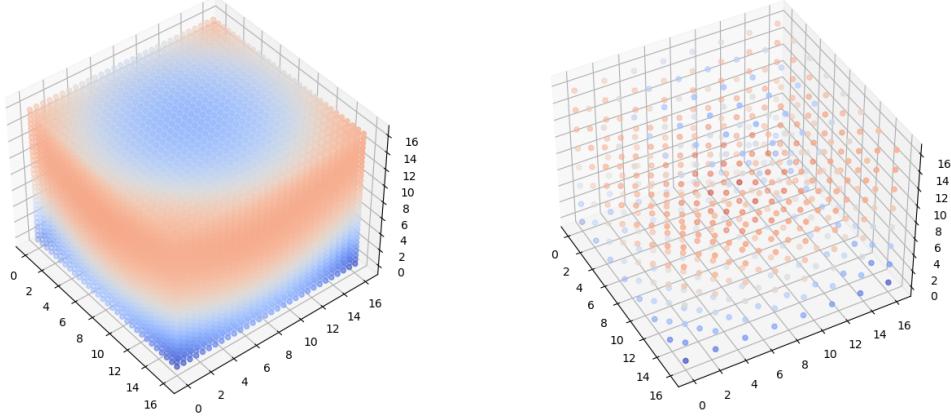


Fig. 19: Ricerca minimo - 4D

Calcoliamo quindi le derivate parziali di $f(x, y, r)$ rispetto alle tre variabili (non essendovi uno standard univoco, di seguito si è scelto di utilizzare la notazione di Leibniz per indicare derivate e derivate parziali)

$$\frac{\partial f}{\partial x} = \sum_{y_p=0}^{15} \sum_{x_p=0}^{15} \left[G_{y_p, x_p} \cdot sgn \left(\sqrt{(x_p - x)^2 + (y_p - y)^2} - r \right) \cdot \frac{x - x_p}{\sqrt{(x_p - x)^2 + (y_p - y)^2}} \right] \quad (17)$$

$$\frac{\partial f}{\partial y} = \sum_{y_p=0}^{15} \sum_{x_p=0}^{15} \left[G_{y_p, x_p} \cdot sgn \left(\sqrt{(x_p - x)^2 + (y_p - y)^2} - r \right) \cdot \frac{y - y_p}{\sqrt{(x_p - x)^2 + (y_p - y)^2}} \right] \quad (18)$$

$$\frac{\partial f}{\partial r} = \sum_{y_p=0}^{15} \sum_{x_p=0}^{15} \left[-G_{y_p, x_p} \cdot sgn \left(\sqrt{(x_p - x)^2 + (y_p - y)^2} - r \right) \right] \quad (19)$$

con $sgn(x)$ definita a tratti nel dominio $(-\infty; 0) \cup (0; +\infty)$:

$$sgn(x) = \begin{cases} -1 & \text{se } x < 0 \\ 1 & \text{se } x > 0 \end{cases} \quad (20)$$

Definiamo il gradiente calcolato in un punto (x_0, y_0, r_0) di una funzione $f(x)$ come la seguente funzione vettoriale

$$\nabla f(x_0, y_0, r_0) = \left(\frac{\partial f}{\partial x} \Big|_{x=x_0}, \frac{\partial f}{\partial y} \Big|_{y=y_0}, \frac{\partial f}{\partial r} \Big|_{r=r_0} \right) \quad (21)$$

Ad ogni iterazione dell'algoritmo si otterrà uno spostamento proporzionale a tale vettore, per cui alla i -esima ripetizione si avrà:

$$(x_i, y_i, r_i) = (x_{i-1}, y_{i-1}, r_{i-1}) - k \cdot \nabla f(x_{i-1}, y_{i-1}, r_{i-1}) \quad \text{con } k > 0 \quad (22)$$

Essendo un algoritmo euristico esso non garantisce di trovare con esattezza un punto di minimo, ma una sua approssimazione (che per il caso in esame è più che sufficiente); inoltre il punto trovato è un minimo locale, ma non è garantito che sia un minimo assoluto, come mostrato nell'immagine ??, per questo può essere opportuno ripetere il procedimento scegliendo diversi punti di partenza, oppure approssimare il centro con altri metodi e utilizzare il valore trovato come punto iniziale (quest'ultimo metodo risulta particolarmente efficace e può essere realizzato con una media ponderata delle coordinate dei pixel dell'immagine, pesati sulla loro luminosità).

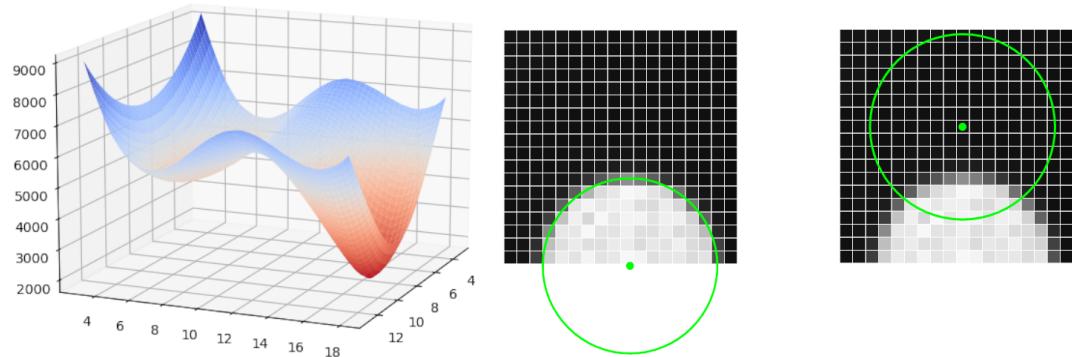


Fig. 20: Minimo locale e minimo assoluto

Il minimo locale e il minimo assoluto corrispondono rispettivamente alla circonferenza mostrata nell'immagine di destra e di sinistra.

Un possibile esempio di implementazione è il seguente⁶.

```

1 void gradient_descent (float x, float y, float r, uint16_t iterations) {
2     for (uint16_t recurse_n = 0; recurse_n < iterations; recurse_n++) {
3         float dx = 0.0, dy = 0.0, dr = 0.0;
4         for (uint8_t y_p = 0; y_p < 16; y_p++) {
5             for (uint8_t x_p = 0; x_p < 16; x_p++) {
6                 if (gradient_modules[y_p][x_p] > GRADIENT_THRESHOLD) {
7                     float x_diff = x_p - x;
8                     float y_diff = y_p - y;
9                     float x_diff_2 = pow(x_diff, 2);
10                    float y_diff_2 = pow(y_diff, 2);
11                    if ((x_diff_2 + y_diff_2) == 0) {
12                        continue;
13                    }
14                    dx -= (gradient_modules[y_p][x_p] - GRADIENT_THRESHOLD) * SGN(sqrt(x_diff_2 + y_diff_2) - r) *
15                        x_diff / sqrt(x_diff_2 + y_diff_2);
16                    dy -= (gradient_modules[y_p][x_p] - GRADIENT_THRESHOLD) * SGN(sqrt(x_diff_2 + y_diff_2) - r) *
17                        y_diff / sqrt(x_diff_2 + y_diff_2);
18                    dr -= (gradient_modules[y_p][x_p] - GRADIENT_THRESHOLD) * SGN(sqrt(x_diff_2 + y_diff_2) - r);
19                }
20            }
21        }
22        x -= dx * K;
23        y -= dy * K;
24        r -= dr * K;
25        progressive_approximations[recurse_n * 3] = x - 7.5;
26        progressive_approximations[recurse_n * 3 + 1] = y - 7.5;
27        progressive_approximations[recurse_n * 3 + 2] = r;
28    }
}

```

Codice 4: Ricerca minimi - discesa del gradiente

⁶Come per le porzioni di codice precedenti, esso fa riferimento a costanti e variabili globali non riportate in questo documento.