



UNIVERSITÀ DEGLI STUDI DI GENOVA

**DIPARTIMENTO DI INGEGNERIA NAVALE, ELETTRICA, ELETTRONICA E
DELLE TELECOMUNICAZIONI**

**CORSO DI STUDIO IN INGEGNERIA ELETTRONICA E TECNOLOGIE
DELL'INFORMAZIONE**

Tesi di Laurea Triennale

Giugno 2023

**Progetto ed implementazione di un sistema embedded per un astro-inseguitore
(Design and implementation of an embedded star tracker)**

Candidato: Alessio Bertini

Relatore: Prof. Riccardo Berta

Tutor: Luca Lazzaroni
Matteo Fresta

SOMMARIO

Con il presente scritto si vuole illustrare come è stato progettato e realizzato un sistema in grado di compensare il movimento di rotazione terrestre con l'obiettivo di consentire che una macchina fotografica abbia velocità angolare pari a quella terrestre ma in senso contrario.

Questo consentirà di avere la macchina fotografica idealmente ferma rispetto al cielo e questo permetterà di scattare fotografie agli astri notturni, che richiedono lunghi tempi di esposizione, senza che si noti l'effetto della rotazione terrestre.

Il sistema verrà realizzato programmando un Raspberry Pi 3 con installato il sistema operativo *Raspbian*[1], che controllerà un motore che permetterà il movimento di un tracciatore di modo che le stelle risultino fisse rispetto alla fotocamera. La fotocamera sarà controllata da remoto, anche questa attraverso il Raspberry Pi 3 per agevolare la procedura di scatto.

Verrà descritto anche qual è il procedimento che si utilizza in astrofotografia per cercare di ottenere immagini di buona qualità, che comprende, oltre alla ripresa delle immagini, anche una parte di post-elaborazione da fare separatamente con software appositi e già esistenti, quali *DeepSkyStacker*[2] e un editor di immagini come *Adobe Photoshop*[3] oppure *GIMP*[4].

INTRODUZIONE

La fotografia astronomica è quel settore della fotografia che si specializza nel riprendere immagini di corpi celesti. Volendo fotografare anche solo il cielo stellato si dovrà tenere conto di moltissimi fattori. Innanzitutto, sarà importante operare in un'area che sia la più buia possibile, poiché, in ambiente notturno, la luce che arriva sulla terra dalle stelle è molto debole. Già con i nostri occhi possiamo notare la grande differenza che c'è tra osservare la volta celeste da un centro città oppure da un luogo di campagna che sia particolarmente poco illuminato. Osserveremo che in una situazione in cui l'inquinamento luminoso è molto minore sarà possibile osservare una maggior quantità di stelle. Il sensore della macchina fotografica si comporta come i nostri occhi, avere poco inquinamento luminoso ci darà la possibilità di ottenere immagini con una migliore definizione. Inoltre, come noi riusciremo a cogliere maggiori dettagli rimanendo ad osservare per un lungo tempo la stessa porzione di cielo, così i tempi di apertura della nostra macchina fotografica ci permetteranno di catturare più luce dagli astri.

Una normale macchina fotografica, che sia analogica o digitale, funziona raccogliendo e registrando i fotoni emessi o riflessi dal soggetto che desideriamo catturare.

Per modificare la quantità di fotoni che vengono registrati si può operare su tre parametri principali: la sensibilità alla luce, il diametro di apertura dell'obiettivo e il tempo di esposizione.

La sensibilità alla luce, detta ISO, più è alta più la macchina sarà sensibile appunto alle più piccole fonti luminose, come effetto indesiderato però si ha che le foto risulteranno più rumorose, ed è per questo che sarà necessario utilizzare dei software che permettono di effettuare della post-elaborazione sulle immagini per ridurne il rumore. Comunque si cercherà un compromesso per tenere l'ISO più alta possibile senza avere eccessivo rumore.

L'apertura dell'obiettivo invece rappresenta l'ampiezza della sezione dell'obiettivo che può essere regolata per permettere a una certa quantità di luce di passare e colpire il sensore: più l'apertura è ampia maggiore sarà la luce che lascerà passare nell'unità di tempo. Non presenta alcun effetto negativo, quindi per le fotografie astronomiche sarà sempre tenuta al valore più aperto possibile. Il tempo di esposizione è quello che evidenzierà un problema di non semplicissima risoluzione. Nella normale fotografia diurna, il sensore della fotocamera richiede solo pochi centesimi o, in situazioni dove la luce è parecchia, addirittura pochi millesimi di secondo per catturare la giusta quantità di fotoni. Per gli oggetti celesti che sono molto deboli il sensore richiede diversi secondi, in alcuni casi anche minuti o ore, per raccogliere questi fotoni, questa è la lunga esposizione, ed è con essa che nasce il problema.

La rotazione terrestre è quasi impercettibile ai nostri occhi, possiamo notarla solo se osserviamo le stelle, o il sole in due momenti temporalmente abbastanza distanti fra di loro.

Abbiamo detto che la fotocamera dovrà tenere il tempo di esposizione molto alto, quindi l'effetto della rotazione terrestre diventerà rilevante. Poiché la macchina fotografica è posizionata sulla terra, sta ruotando insieme ad essa. Se abbiamo puntato la camera verso una stella, dopo alcuni minuti punterà altrove e non verso la stella che abbiamo inizialmente indicato. Come conseguenza la rotazione genererà delle strisce luminose sulla nostra immagine, invece che far apparire le stelle come puntiformi. Queste scie vengono comunemente chiamate "*star trails*" e per quanto possano in alcuni casi essere volute e risultare belle da vedere, non è esattamente quello che vogliamo se il nostro intento è avere un'immagine chiara e definita del cielo stellato o di una porzione di esso.

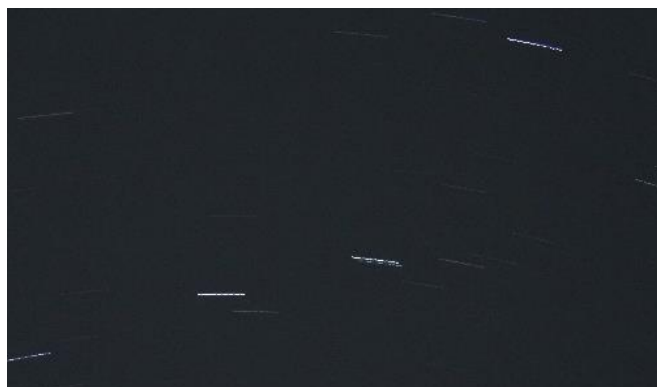


Figura 1: Star Trails

Per poter evitare questo effetto di movimento non voluto, senza strumenti specifici, si è vincolati a tenere i tempi di esposizione bassi. Per calcolare il massimo tempo di posa che evita le scie nell'immagine, si può utilizzare la "regola del 500", che consiste nell'applicare la formula: $500 / \text{lunghezza focale}$. La lunghezza focale di un obiettivo è la distanza (generalmente espressa in millimetri) tra il punto in cui si concentra la luce all'interno dell'obiettivo e il sensore della fotocamera. Dalla lunghezza focale dipende l'angolo di campo ovvero la porzione di realtà inquadrata nella fotografia. L'angolo di campo è inversamente proporzionale rispetto alla distanza focale maggiore sarà quest'ultima, minore sarà la porzione di realtà inquadrata. In termini più semplici possiamo dire che dalla lunghezza focale dipende l'ingrandimento dell'immagine. Con lunghezza focale in questo caso si intende la lunghezza focale equivalente, ovvero la lunghezza focale dell'obiettivo moltiplicata per un fattore 1,5 se si utilizza una fotocamera Nikon o Sony, oppure per un fattore 1,6 se si utilizza una Canon. Questo fattore moltiplicativo è dovuto al fatto che le moderne macchine fotografiche digitali hanno un sensore più piccolo del sensore delle vecchie macchine fotografiche analogiche, ovvero la pellicola. Nonostante la pellicola non venga più utilizzata, le misure di essa, ovvero 24*36 millimetri, rimangono la dimensione standard della superficie sensibile. Essendo i moderni sensori più piccoli di questa misura, l'effetto che si ottiene è pari ad avere un obiettivo con lunghezza focale maggiore su un sensore standard. Esiste anche la "regola del 600" che prevede la stessa formula citata in precedenza ma è un po' meno conservativa.

	1.5x Crop Sensor (Nikon / Sony)		1.6x Crop Sensor (Canon)		Full Frame Sensor	
Focal Length	500 rule	600 rule	500 rule	600 rule	500 rule	600 rule
8mm	42 s	50 s	39 s	47 s	63 s	75 s
14mm	24 s	29 s	22 s	27 s	36 s	43 s
17mm	19 s	23 s	18 s	22 s	29 s	35 s
24mm	14 s	17 s	13 s	16 s	21 s	25 s
35mm	9 s	11 s	9 s	11 s	14 s	17 s
50mm	7 s	8 s	6 s	7.5 s	10 s	12 s
70mm	5 s	6 s	4 s	5 s	7 s	9 s

Tabella 1: tempi massimi di esposizione

Esiste un procedimento che permette di ottenere fotografie discrete anche con tempi di esposizione così bassi: consiste nel realizzare molti scatti con un ISO più alta e successivamente sovrapporre questi scatti in un'unica immagine, questo permette di raccogliere una buona quantità di luce grazie alla sensibilità più alta e successivamente ridurre il maggior rumore prodotto sommando le immagini. In questo modo nell'immagine finale si sarà simulato un tempo di esposizione molto più alto. Ad esempio, raccogliendo 20 scatti da 2 secondi si otterrà un'immagine finale che avrà un tempo di esposizione totale di 40 secondi.

Questo metodo, tuttavia, oltre a essere abbastanza oneroso, poiché per una buona immagine finale sarebbe necessario scattare tra le 200 e le 300 fotografie, come minimo, non garantisce gli stessi risultati rispetto alla presa di meno immagini con tempo di esposizione molto più lungo. Queste ultime, infatti, risulteranno comunque più nitide e presenteranno più dettagli, dato che, in questo ultimo caso, la fotocamera riuscirà a raccogliere dettagli meno luminosi. Con il primo procedimento si può eliminare il rumore, ma non potranno mai essere presenti nella fotografia dettagli che la fotocamera non è riuscita a cogliere a causa del tempo di esposizione ridotto.

La soluzione migliore al problema sta nell'utilizzo di un tracciatore, o inseguitore di stelle.

La terra ruota con una velocità di circa 15 gradi all'ora; quindi, il tracciatore compenserà questa rotazione muovendosi in senso opposto con la stessa velocità. Lo scopo di questo star Tracker è quindi far muovere la nostra macchina fotografica in sincrono con la rotazione terrestre utilizzando la Stella Polare come riferimento per posizionare correttamente l'asse di rotazione del sistema.

Nell'emisfero nord, infatti, possiamo notare che questa stella, nella costellazione dell'Orsa Minore, rimane sempre fissa nella volta celeste, mentre tutte le altre assumeranno un movimento circolare attorno ad essa. Il risultato finale sarà che montando la fotocamera su questo tracciatore sarà possibile scattare fotografie con tempo di apertura molto lungo senza che le immagini risultino mosse o presentino scie.

Tuttavia, per calcolare la velocità in modo preciso bisogna sapere che per compiere una rotazione completa su sé stessa, la terra non impiega esattamente 24 ore ma 23 ore e 56 minuti; quindi, il tracciatore dovrà compensare una rotazione di 360° in 1436 minuti che corrisponde a una velocità angolare di: $360^\circ/1436\text{minuti} \approx 0,2507^\circ/\text{minuto} \approx 15,04^\circ/\text{ora}$. Nei nostri calcoli utilizzeremo questa misura più precisa rispetto a quella di 15 gradi all'ora citata in precedenza.

In questa tesi verrà progettato e realizzato un tracciatore a triangolo isoscele, sistema che è stato creato da George Haig, chiamato anche infatti "*Haig mount*", e pubblicato per la prima volta nel 1975. Il design originale è stato modificato da Dave Trott, che ne ha migliorato la precisione di tracciamento nel tempo, e venne pubblicato nel 1988. I disegni originali prevedono tutti l'attivazione manuale. Tuttavia, quasi tutti i tracciatori moderni, compreso quello che verrà realizzato in questo progetto, prevedono l'attivazione tramite un motore che possiamo controllare via software.

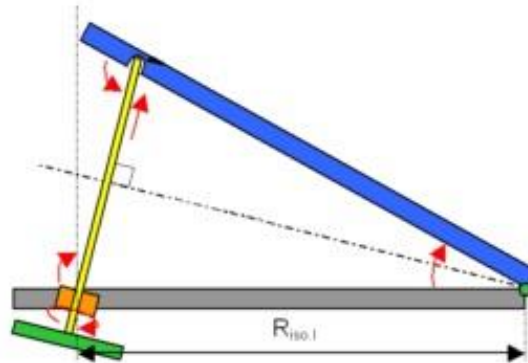


Figura 2: schema funzionamento

La rotazione dell'asta filettata (gialla) farà spostare verso l'alto la superficie superiore del tracciatore (blu), dove verrà montata la macchina fotografica. La superficie inferiore (grigia) rimarrà fissa e sarà possibilmente fissata su un cavalletto. Controllando con precisione la velocità, possiamo far ruotare la superficie superiore attorno alla cerniera alla stessa velocità della terra, ottenendo così il tracciamento. L'asse della cerniera dovrà essere puntato verso la Stella Polare in modo che la superficie superiore si muova da Est verso Ovest. Per agevolare il puntamento è previsto un piccolo cannocchiale che funzionerà da mirino e sarà fissato in modo da avere il suo asse parallelo a quello della cerniera.

La realizzazione del sistema prende ispirazione dal progetto di un tracciatore realizzato da *I2SDD – Ham Radio* con Arduino[5].

In questa tesi verrà utilizzato un Raspberry Pi 3 invece che Arduino e oltre al sistema di tracciamento si vuole controllare la macchina fotografica da remoto. Questo permette di avere dei controlli esterni per la fotocamera, in modo da poter scattare più fotografie alla volta senza dover premere il comando per ogni immagine. Un altro vantaggio di avere i controlli esterni è che si evita di dover utilizzare quelli sulla camera, il che potrebbe risultare in un movimento di questa, che per quanto minimo potrebbe tradursi, dato il lungo tempo di esposizione, in immagini mosse o sfocate.

METODI E STRUMENTI UTILIZZATI

Per la realizzazione del progetto si è operato su tre parti principali:

- Parte meccanica
- Parte hardware
- Parte software

Descrizione parte meccanica:

Comprende una serie di pezzi montati in modo da ottenere il triangolo illustrato in figura 2, è composta da:

- 11 Pezzi stampati in 3D
- 4 aste di collegamento tra la cerniera e l'alloggiamento di motore e asta filettata
- 1 asta filettata che permette l'apertura e chiusura del triangolo
- 6 cuscinetti per permettere il movimento della cerniera e delle parti dove sono posizionati motore e asta filettata

- Diverse viti e dadi per fissare tutto

Le stampe 3D sono state fatte grazie a una società genovese, la Astrati[6], e comprendono anche un piccolo cannocchiale con i relativi supporti da montare in asse con la cerniera per agevolare il puntamento dello strumento verso la stella polare. Le 2 parti, superiore e inferiore, che compongono la cerniera sono provviste di alcuni buchi per agevolare il fissaggio del sistema su di un cavalletto e della fotocamera. È stata realizzata anche una basetta in legno come collegamento tra il pezzo della cerniera superiore e uno snodo di un altro cavalletto, in modo da agevolare il puntamento della macchina fotografica.



Figura 3: Il triangolo per il tracciamento assemblato

Nella struttura finale la distanza esatta tra la cerniera e l'albero del motore è di 267mm. Questo dato ci sarà utile in seguito per calcolare l'esatta velocità di rotazione del motore per ottenere un tracciamento corretto.

Descrizione parte hardware:

Come hardware sono stati utilizzati:

- Un Raspberry Pi 3
- Un Motore 28BYJ-48 con relativo driver ULN2003 [7]
- Un LED con una resistenza da 1 k Ω
- Un Push Button
- Uno Switch Button

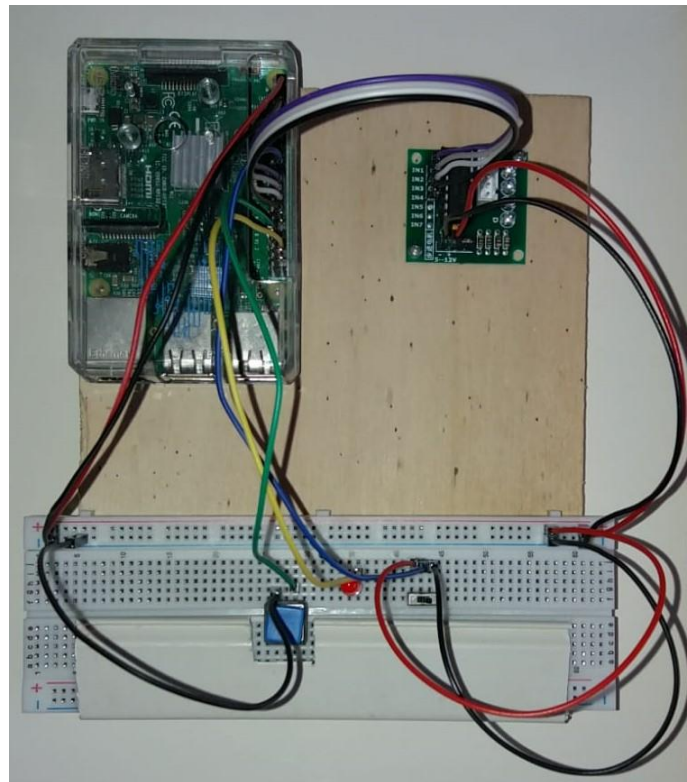


Figura 4: Il sistema hardware (motore non collegato)

Il motore è uno stepper, non è particolarmente veloce, ha una velocità di rotazione massima di circa 15 rotazioni al minuto, ma questo non è un problema per questa applicazione dato che la velocità che richiede il tracciamento è molto ridotta, come verrà illustrato in seguito. Essendo uno stepper, non abbiamo vincoli sulla velocità minima di rotazione, in quanto una rotazione è composta da una certa quantità di step e noi possiamo specificare il tempo che deve intercorrere tra uno step e il successivo. Nonostante sia di dimensioni ridotte, il motore può produrre una coppia di 34,3 mN*m, che è ampiamente sufficiente per il lavoro che deve svolgere, a patto che sul tracciatore non venga montata

una fotocamera particolarmente pesante. Il motore in ingresso ha 5 cavi: uno per l'alimentazione a 5V, e gli altri quattro che servono per pilotarlo, ognuno di questi infatti è legato a uno dei 4 avvolgimenti che, se azionati in successione, permettono il movimento del motore in un verso o nell'altro.

Al suo interno il motore ha ingranaggi per un rapporto di riduzione di circa 64:1, ovvero a 64 rotazioni del motore corrisponde una sola rotazione dell'albero motore in uscita. Infatti, sono presenti 4 ingranaggi con rapporti di riduzione di 32/9, 22/11, 26/9, 31/10, che, moltiplicati fra di loro danno $567424/8910 = 63.68395/1$.

Lo stepper può essere utilizzato in *"full-step mode"* oppure in *"half-step mode"*. La differenza sta nel fatto che nel primo caso il motore compie una rotazione ogni 32 steps (e quindi l'albero una ogni circa $32 \cdot 64 = 2048$ step), mentre nel secondo compie una rotazione ogni 64 steps (e quindi l'albero uno ogni circa $64 \cdot 64 = 4096$ step). In modalità *full-step*, vengono attivate le spire due alla volta, e, ipotizzando che nello stato iniziale siano attive le spire uno e due, al passo successivo verrà spenta la spira uno e accesa la spira tre per il movimento in senso orario, mentre verrà spenta la spira due e accesa la spira quattro per il movimento in senso antiorario (osservando l'albero motore dall'alto). La modalità *half-step* ha lo stesso principio di funzionamento ma fra un passo e il successivo ne aggiunge uno intermedio in cui viene lasciata attiva solo una spira. Facendo lo stesso esempio di prima, con la prima e la seconda spira attive, per arrivare allo stato in cui sono attive la seconda e la terza non avremo bisogno di un solo step ma di due, infatti il passo intermedio spegne la spira uno e non attiva ancora la spira tre e in questo frangente avremo uno stato in cui solo la spira due è attivata. Le seguenti due tabelle mostrano nel dettaglio l'attivazione delle spire del motore per una corretta rotazione di questo, sia in *"full-step mode"* che in *"half-step mode"*.

	SPIRA 1	SPIRA 2	SPIRA 3	SPIRA 4
PASSO 1	1	0	0	1
PASSO 2	1	1	0	0
PASSO 3	0	1	1	0
PASSO 4	0	0	1	1

Tabella 2: full-step mode

	SPIRA 1	SPIRA 2	SPIRA 3	SPIRA 4
PASSO 1	1	0	0	1
PASSO 2	1	0	0	0
PASSO 3	1	1	0	0
PASSO 4	0	1	0	0
PASSO 5	0	1	1	0
PASSO 6	0	0	1	0
PASSO 7	0	0	1	1
PASSO 8	0	0	0	1

Tabella 3: half-step mode

In questo progetto si è optato per utilizzo *"half-step"*, con il quale si può ottenere una rotazione più precisa. Inoltre, utilizzando questo metodo il motore si scalda meno, poiché in ogni ciclo avremo 8 passi di cui 4 con 2 avvolgimenti attivi e 4 con un solo avvolgimento attivo. Al contrario in *"full-step mode"* per ogni ciclo avremo 4 passi ma ci saranno sempre 2 avvolgimenti attivi. Quindi nello stesso periodo di tempo, in *"half-step mode"* avremo il 25% in meno di spire attraversate da corrente. La scelta è giustificata dal fatto che si vuole evitare il surriscaldamento del motore, che per questa applicazione dovrà rimanere attivato per molto tempo durante lo scatto delle fotografie.

Da questo momento in poi verrà fatto riferimento solo alla modalità *half-step*, che è di nostro interesse. Il fatto che per compiere una rotazione dell'albero siano necessari 4096 step significa che, potendo impostare il tempo di attesa tra uno step e il successivo, l'albero impiegherà 4096 volte questo tempo per completare un giro su sé stesso. Per portare un esempio pratico, se noi specificassimo un ritardo di un secondo tra gli step, otterremo che viene eseguita una rotazione ogni 4096 secondi, ovvero circa 68 minuti.

Osservando la tabella possiamo notare che le spire vengono attivate in sequenza dal passo uno al passo otto, che dovranno essere eseguiti in questa sequenza per muovere l'albero in senso orario (sempre se osservato dall'alto, come sopra). Al contrario dovremo eseguire i passi dall'ottavo al primo se vogliamo far muovere l'albero in senso antiorario.

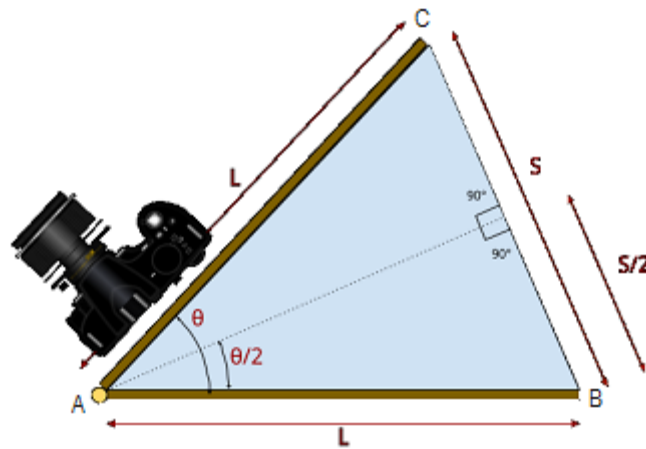
In questa realizzazione del tracciatore il motore pilota un'asta filettata che si avviterà o sviterà in un dado filettato inserito nella superficie superiore del tracciatore.

Quindi per far sì che le due superfici si allontanino abbiamo bisogno che l'asta, e quindi il motore, ruotino in senso orario, mentre, per riavvolgere e riportare il triangolo alla situazione iniziale, dovremo far ruotare l'asta e il motore in senso antiorario.

Ora che abbiamo chiarito il funzionamento del motore, possiamo calcolare la velocità necessaria al tracciamento.

CALCOLO VELOCITÀ DI ROTAZIONE DEL MOTORE:

Come visto nel paragrafo precedente la terra ruota di 15,04° all'ora attorno al suo asse, quindi l'obiettivo è ruotare il braccio su cui è fissata la fotocamera alla stessa velocità angolare.



Meccanicamente il movimento del braccio superiore si ottiene aumentando la lunghezza di "S", che rappresenta la nostra vite collegata al motore. Per gestire la velocità con cui aumenta l'angolo "θ" dobbiamo controllare l'aumento di S.

ABC è un triangolo isoscele in quanto la struttura è stata progettata in modo da avere i due bracci AC e AB della stessa lunghezza L.

Per semplificare i calcoli possiamo dividere in due parti uguali il triangolo isoscele, ottenendo così due triangoli rettangoli. Applicando alcuni semplici calcoli trigonometrici possiamo trovare S in funzione di L e θ:

$$\sin(\theta/2) = (S/2) / L \Rightarrow \sin(\theta/2) = S / (2 \cdot L) \Rightarrow S = 2 \cdot L \cdot \sin(\theta/2)$$

Calcoliamo ora approssimativamente l'aumento richiesto di S per ogni ora di tracciamento supponendo che all'inizio $S = 0\text{ mm}$ e quindi $\theta = 0^\circ$. Dopo un'ora, θ dovrebbe essere 15,04°.

$$S = 2 \cdot L \cdot \sin(15,04/2) \text{ all'ora} \Rightarrow S \approx 2 \cdot L \cdot 0.13087 \text{ all'ora} \Rightarrow S \approx 0,2617 \cdot L \text{ all'ora}$$

Nel nostro caso abbiamo visto nella descrizione della parte meccanica che L si attesta a 267mm, quindi S dovrà essere $S = 0,2617 \cdot 267\text{ mm all'ora} \Rightarrow S \approx 69,87 \text{ mm all'ora}$.

A questo punto dobbiamo capire quante rotazioni della vite rappresentata da S dobbiamo eseguire per ottenere quell'incremento. La vite è un'asta filettata M8 standard, che ha un diametro di 8 mm e un passo di 1,25 mm, per cui ogni giro completo aumenterà S di 1,25 mm. Con questo ulteriore dato, troviamo che dobbiamo eseguire $69,87\text{ mm} / 1,25\text{ mm} \approx 55,9$ rotazioni all'ora, che permette di calcolare quanti secondi dobbiamo impiegare per compiere un giro: $3600\text{ s} / 55,9 \text{ rotazioni} \approx 64,4 \text{ s/rotazione}$.

Come detto nella descrizione del motore, utilizzando la modalità "half-step" sappiamo che il motore impiega 4096 step per completare un giro dell'albero e quindi della vite. Possiamo infine calcolare il ritardo che dovremo inserire tra uno step e il successivo: $64,4 \text{ s} / 4096 \text{ step} \approx 0.015723 \text{ s/step}$.

Siccome la funzione che permette di ottenere il ritardo tra gli step vuole il valore del tempo in microsecondi, dovremo impostare questo valore a 15723.

Calcolando la velocità del motore in questo modo, tuttavia, genera un problema. Infatti, possiamo notare che la velocità che impostiamo è fissata e non varia nel tempo, quindi S, ovvero la base del triangolo isoscele, aumenterà linearmente, ma questo comporta il fatto che l'angolo non aumenti linearmente. Questo problema è noto come errore tangente e la non linearità deriva dal fatto che nel calcolo viene usata la funzione seno, che è non lineare.

L'effetto finale dell'errore tangente è che avremo un errore nella velocità del tracciatore e ciò diventa evidente dopo circa 8/10 minuti di esposizione. Per risolvere l'errore tangente si può agire principalmente in due modi:

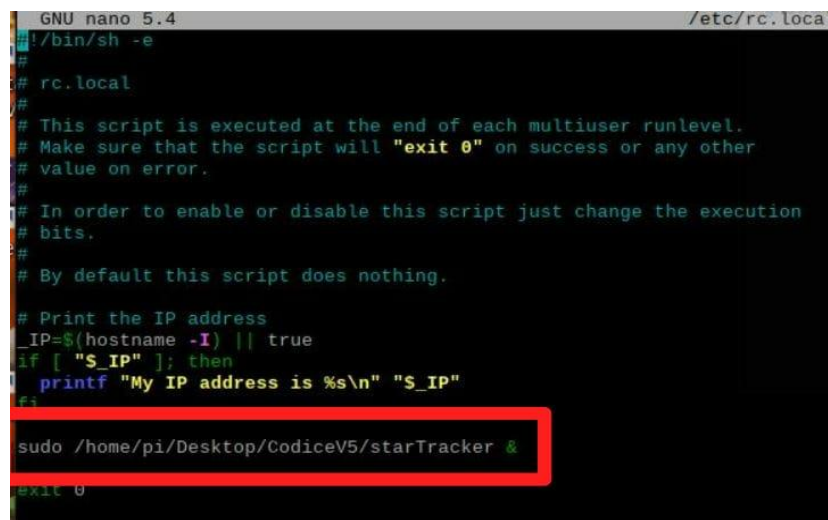
- Metodo meccanico: al posto di un'asta filettata dritta, rappresentata da S, si può inserire un'asta curva, ma questo comporterebbe enormi complicazioni nella rotazione della stessa.
- Metodo software: si realizzano delle tabelle in base al tempo di tracciamento e si programma il motore in modo che segua queste tabelle e vari la sua velocità nel tempo. È un metodo che non complica la struttura meccanica ma richiede comunque una notevole complessità nel codice.

In questo progetto l'errore tangente non viene corretto poiché sarebbe inutile. È molto difficile avere bisogno di tempi di esposizione così lunghi, inoltre sono veramente poche le macchine fotografiche che li supportano. Ad esempio, la macchina utilizzata in questo caso ha un massimo tempo di esposizione di 30 secondi. In secondo luogo, il sistema di puntamento non è preciso, infatti, per puntare la Stella Polare, è previsto un sistema di puntamento a occhio, tramite un piccolo cannocchiale, quindi è molto difficile puntare la struttura con assoluta precisione, per cui dopo un po' di tempo avremo comunque, anche senza l'errore tangente, un errore di tracciamento dovuto all'allineamento non perfetto della cerniera con la Polare.

Descrizione parte software:

Per controllare sia tracciatore che macchina fotografica viene utilizzato un Raspberry Pi 3 con sistema operativo Raspbian[1]. Il motore dell'inseguitore viene controllato tramite GPIO, che permette anche la lettura di pulsante e interruttore e il pilotaggio del led, mentre la macchina fotografica viene comandata attraverso un cavo USB.

Il sistema può essere alimentato da un powerbank a 5 volt e il programma di controllo viene lanciato automaticamente all'avvio del Raspberry, in modo che sia possibile utilizzarlo anche senza avere a disposizione uno schermo e una tastiera per operare. Questo risulta particolarmente utile nel caso si scegliesse di scattare le fotografie in un luogo lontano da prese di corrente. L'avvio del programma all'accensione del Raspberry è ottenuto tramite il file di sistema *"rc.local"*, viene elaborato all'avvio del sistema operativo Linux. Modificando questo file, è possibile inserire, prima dell'istruzione *"exit 0"* una chiamata ad uno script o ad un programma, seguita immediatamente dal comando *"&"*. Quest'ultimo permette l'esecuzione del processo in background e procede con l'esecuzione degli altri comandi e termina l'avvio del sistema. Nel file *"rc.local"* del sistema operativo, viene chiamato il programma *"starTrack"*, attraverso il suo percorso assoluto e, per questo programma in particolare, è fondamentale l'aggiunta del comando *"&"*, poiché verrà illustrato che l'esecuzione prevede l'utilizzo di cicli infiniti, quindi il rischio, se non si eseguisse in background, è che non verrebbe mai completato l'avvio del sistema operativo.



```
GNU nano 5.4 /etc/rc.local
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.
#
# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi
sudo /home/pi/Desktop/CodiceV5/starTracker &
exit 0
```

Figura 5: Il file *rc.local* con il comando per avviare lo *starTracker*

Sempre per garantire l'utilizzo del tracciatore senza l'ausilio di periferiche quali schermo e tastiera il motore e la camera possono essere attivati rispettivamente da uno switch e da un push button. I due sono indipendenti fra di loro, quindi si può utilizzare l'inseguitore senza avere la camera collegata e viceversa si possono scattare fotografie senza azionare il motore.

Per il controllo remoto della camera, è stata utilizzata la libreria *"libgphoto2"* [8], che fornisce anche una lista di compatibilità con tutte le camere che possono essere riconosciute e controllate via software. Per il fatto che non tutte le fotocamere possono essere controllate dal Raspberry, la scelta di mantenere i controlli separati permette di avere più versatilità del sistema. In particolare, questa architettura risulta utile anche nel caso in cui non si avesse a disposizione una macchina fotografica compatibile, poiché si può comunque utilizzare la funzione di tracciamento e al contempo scattare le fotografie dai normali comandi della camera.

Il software che si occupa di controllare tutto è suddiviso in 4 files, di cui un file “.c”, che è il programma vero e proprio, e altri 3 files “.sh” che eseguono dei comandi tramite la shell Linux di Raspbian. È presente anche un file “*compile.sh*”, che può essere utile se volesse modificare il programma. Questo file, infatti, contiene due comandi: il primo termina il programma se è in esecuzione (il che è molto probabile dato che viene lanciato all’avvio), mentre il secondo ricompila il programma.

File “*setupCamera.sh*”: È uno shell script, ovvero un file che esegue una serie di comandi Linux la cui sintassi è la stessa di quando vengono eseguiti da linea di comando, ed è molto semplice. All’avvio del Raspberry, il software “*gPhoto2*”, programma associato alla libreria “*libgphoto2*”, fa partire dei processi nel più comune caso in cui si voglia collegare una fotocamera con il solo scopo di trasferire immagini dalla camera alla memoria del computer. Tuttavia, questi processi possono creare dei problemi quando si tratta di lanciare i comandi per il controllo da remoto. In pratica il risultato in alcuni casi è che viene notificato un errore per cui si è impossibilitati a controllare la camera poiché è in uso in altri processi (quelli lanciati all’avvio e descritti sopra appunto). Questo file si occupa semplicemente di terminare i processi avviati di default da “*gPhoto2*” in modo da avere possibilità di controllo sulla fotocamera senza problemi.

File “*setParameters.sh*”: Altro shell script e si occupa di modificarne i parametri di scatto impostandoli in modo che siano adatti al tipo di fotografia che siamo per catturare, ovvero astrofotografia. Tuttavia, non tutti i soggetti celesti richiedono i parametri impostati allo stesso modo, quindi questo file può essere editato in modo da avere la miglior impostazione possibile per il soggetto che si vuole fotografare. In questo caso si è deciso di rimanere su regolazioni abbastanza standard, che vanno bene per molti corpi celesti e in particolare sono ottimi per scattare foto al cielo stellato. Viene impostato il massimo tempo di apertura dell’obiettivo per questa fotocamera, ovvero 30 secondi, l’ISO a 1600, valore abbastanza alto ma non troppo, per evitare rumore eccessivo, il bilanciamento del bianco viene messo come se si stesse scattando una foto alla luce del sole e infine, viene impostata la qualità dell’immagine su RAW, ovvero il formato non compresso. Questo formato non compresso ha l’unico svantaggio di avere grandi dimensioni, ma verrà successivamente compresso in jpeg dopo il lavoro di post-produzione sulle immagini.

Questo file viene chiamato quando viene premuto il pulsante per lo scatto delle foto dal Raspberry. Questa scelta, invece che chiamarlo all’avvio del programma, è stata dettata dal fatto che all’avvio la fotocamera potrebbe non essere ancora collegata al sistema e quindi, quando si andrebbe a scattare, non verrebbero impostati correttamente i parametri e le fotografie verrebbero effettuate con le impostazioni presenti in precedenza sulla camera.

Il file *setParameters.sh* può essere modificato in base alle necessità che abbiamo. Volessimo, ad esempio, fotografare un corpo celeste in particolare potremmo avere bisogno di impostazioni più specifiche, che possono essere fissate in questo file specificandole.

1. Tempo di esposizione: può essere impostato inserendo il tempo in secondi che desideriamo duri l’esposizione. Per tempi di esposizione inferiori al secondo (non di nostro interesse per fotografie astronomiche), va specificata la frazione di secondo che si desidera (es. 1/10, 1/20...). La fotocamera utilizzata in questo caso (Nikon-1 J5) consente questa regolazione per tempi compresi tra 30 secondi e 1/16000 di secondo.
2. ISO: va impostato inserendo il valore desiderato, a patto che sia un valore supportato dalla macchina fotografica. Per quella in uso in questo progetto i valori possibili sono: 160, 200, 400, 800, 1600, 3200, 6400, 12800. Il valore dell’ISO più è basso e meno il sensore sarà sensibile alla luce e quindi le foto saranno meno luminose. Per immagini astronomiche andrebbe impostato come minimo a 800.
3. Formato dell’immagine: Il parametro *imagequality2* consente 5 diversi valori di cui 4 corrispondono a formati diversi tra loro mentre uno salva le immagini in due formati diversi:
 - a. JPEG normal: è il formato più comune e quello che solitamente si trova preimpostato al primo avvio della macchina fotografica. Ha un dettaglio medio-buono dell’immagine in rapporto alla compressione. Il valore corrispondente da impostare è *imagequality2=0*.
 - b. JPEG fine: Ha un ottimo risultato in termini di qualità, ma è comunque un formato compresso, che non andrà bene se vorremo fare della post-elaborazione sulle

immagini. Contestualmente occuperà anche più spazio in memoria. Il valore corrispondente da impostare è *imagequality2=1*.

- c. JPEG basic: è il formato più povero, più compresso e che quindi occuperà meno spazio ma avrà anche la qualità minore rispetto agli altri. Il valore corrispondente da impostare è *imagequality2=2*.
- d. NEF + Fine: l'immagine viene memorizzata in entrambi i formati. Il formato RAW in questo caso avrà estensione ".nef" ed è un formato non compresso, quindi non ci saranno perdite di informazione dovute alla compressione, tuttavia occupa molta memoria, per dare un'idea, solitamente il formato RAW occupa 5 volte lo spazio dello stesso scatto in formato JPEG Fine. Il valore corrispondente da impostare è *imagequality2=3*.
- e. NEF (raw): l'immagine viene salvata soltanto in formato non compresso. Il valore corrispondente da impostare è *imagequality2=4*.

- 4. Bilanciamento del bianco: anche questo come i primi due può essere inserito specificando direttamente il valore desiderato, che si può scegliere tra *Auto*, *Tungsten*, *Fluorescent*, *Daylight*, *Flash*, *Cloudy* e *Shade*.

File "capture.sh": Terzo e ultimo shell script che si occupa della procedura di scatto. Contiene 2 comandi: il primo imposta la periferica dove andrà salvata la fotografia che viene scattata, il secondo è il comando che fa effettivamente partire lo scatto della camera.

Il primo comando imposta come periferica la scheda micro-SD presente nella macchina fotografica.

Se non si specificasse la posizione di salvataggio, l'immagine non verrebbe salvata.

Con il secondo comando viene innescato lo scatto e la seconda parte del comando specifica che prima di proseguire va atteso l'evento "*FILEADDED*", che si verifica quando la fotografia viene memorizzata.

File "starTracker.c": Questo file contiene il programma di tracking vero e proprio, è scritto in C e inizialmente si era pensato di dividerlo in due parti: una per la gestione del motore e l'altra per la gestione della macchina fotografica. Non è stato purtroppo possibile continuare con questo approccio poiché entrambe le parti di gestione hanno bisogno di controllare i pin GPIO e ciò può essere fatto da un solo programma alla volta. Volendo comunque mantenere fotocamera e motore separati si è trovata un'altra soluzione che verrà descritta in seguito.

Le librerie necessarie per il funzionamento del codice sono 3. per il controllo dei pin GPIO è stata usata la libreria "*pigpio.h*". Le altre due sono la "*stdlib.h*", che permette l'utilizzo della funzione "*system()*" per eseguire comandi che altrimenti andrebbero eseguiti dal terminale, e la "*unistd.h*" che permette la chiamata alla funzione "*fork()*" per creare processi diversi che lavorano in parallelo. Prima del "*main()*" vengono definite delle costanti, vengono definiti infatti: i PIN della GPIO che controlleranno il motore, il PIN con cui verrà rilevato lo stato dello *switch button* per attivare il motore, il PIN dedicato al *push button* che farà partire la ripresa delle immagini, il PIN di pilotaggio di un LED che si attiverà quando la fotocamera è impegnata a scattare, due costanti che definiranno il verso di rotazione del motore, ovvero "*TRACK*" per muovere il motore in senso orario e "*REWIND*" per muovere il motore in senso antiorario (se osservato dall'alto). Vengono a questo punto dichiarati i prototipi delle 4 funzioni che comporranno il programma: una per il controllo della fotocamera, una per il controllo del motore, una per muovere il motore in un senso o nell'altro e una per eseguire uno step del motore.

Infine, viene dichiarata una variabile intera globale "*step*" inizializzata a 0, che assumerà 8 valori, tra 0 e 7, e servirà per tenere traccia di quale step del motore si è raggiunto (vedi "*Tabella half-stepper*").

Il "*main()*" contiene poche istruzioni, infatti in questa parte di codice viene inizializzata la GPIO e vengono definite le modalità di utilizzo dei pin, ovvero se saranno pin di input o di output: abbiamo 5 pin di output, 4 per il motore e uno per il LED, e due pin di input, per i pulsanti.

A questo punto viene chiamata una "*fork()*" e ciò rappresenta la soluzione al problema iniziale, ovvero quello di voler controllare motore e camera separatamente. La "*fork()*" genera un processo figlio e restituisce il PID (identificativo di processo) del figlio nel caso in cui ci troviamo nel processo padre, mentre restituisce zero nel caso in cui ci troviamo nel processo figlio. Questo è sfruttato, con un semplice costrutto "*if-else*", per chiamare la funzione che gestisce la macchina fotografica nel processo padre e la funzione che gestisce il motorino nel processo figlio.

Grazie a questo procedimento abbiamo comunque un unico "*file.c*" ma abbiamo la possibilità di azionare separatamente e in modo indipendente fra di loro motore e camera.

L'indipendenza del motore dalla fotocamera e viceversa porta diversi benefici. Il più importante è il fatto che viene evitato che i comandi della fotocamera impediscano al motore di funzionare; infatti, la chiamata del comando per catturare la fotografia è bloccante per il programma e questo non può andare avanti sinché la foto non è stata scattata e salvata. Quindi si avrebbe l'inconveniente che durante la cattura il motore non muove la camera, che è il motivo per cui si è realizzato tutto il progetto, inoltre la procedura di cattura sarebbe molto lenta, poiché, come viene descritto all'inizio, abbiamo bisogno di tempi di esposizione molto lunghi per poter raccogliere più luce possibile, inoltre verrà scattata più di una fotografia; quindi, il tempo in cui il programma rimarrebbe bloccato sarebbe molto lungo.

Il secondo vantaggio, non essenziale come il primo ma comunque utile, si ottiene in questo modo quella versatilità già descritta in precedenza, grazie alla quale si possono utilizzare anche macchine fotografiche non controllabili da remoto.

Infine, nel *"main()"*, dopo aver chiamato le due funzioni di controllo, viene eseguita una funzione per terminare l'utilizzo della GPIO, che tuttavia, se non si verificano errori, non verrà chiamata.

```

33 int main() {
34     gpioInitialise();
35
36     //Defining pin modes
37     gpioSetMode(PIN_1, PI_OUTPUT);
38     gpioSetMode(PIN_2, PI_OUTPUT);
39     gpioSetMode(PIN_3, PI_OUTPUT);
40     gpioSetMode(PIN_4, PI_OUTPUT);
41     gpioSetMode(SWITCH, PI_INPUT);
42     gpioSetMode(BUTTON, PI_INPUT);
43     gpioSetMode(LED, PI_OUTPUT);
44
45     if(fork() == 0)
46         controlMotor();
47     else
48         controlCamera();
49
50     gpioTerminate();
51     return 0;
52 }
```

Figura 6: La funzione *main()*

- Funzione di controllo della camera: come detto in precedenza viene eseguita nel processo padre e al suo interno presenta un ciclo *"while(true)"* che verrà ripetuto all'infinito. Ad ogni ripetizione del ciclo viene controllato lo stato di un *push button*, non viene fatto nulla se questo non è premuto. In caso contrario parte la procedura di scatto: viene chiamato lo script *"setupCamera.sh"* per terminare i processi di default di *gPhoto2*, viene chiamato anche lo script *"setParameters.sh"* per impostare correttamente i parametri della fotocamera e successivamente viene fatto partire un ciclo for in cui, ad ogni iterazione, viene chiamato *"capture.sh"* per la cattura della fotografia. Il ciclo for non è necessario, senza quest'ultimo verrebbe scattata una sola immagine, tuttavia, è comodo avere più riprese in sequenza in modo da non doverlo fare manualmente. Queste prese multiple verranno utilizzate tutte per elaborare un'unica immagine finale in seguito, dato che per ottenere un'immagine con la migliore qualità possibile si è soliti, in ambito astrofotografico, combinare più immagini, soprattutto per eliminare il rumore prodotto dalla macchina fotografica. Questo procedimento di post-elaborazione verrà comunque approfondito in seguito.

In questo caso si è deciso di inserire un ciclo for da 5 iterazioni; quindi, a ogni pressione del pulsante, vengono scattate 5 fotografie, per cui alla fine si avrà comunque un multiplo di 5 di immagini salvate. Come già accennato in precedenza il procedimento di scatto della camera è bloccante, per cui una volta che si è premuto il pulsante non si ha modo di interromperlo e in questo frangente la pressione del bottone non ha alcun effetto.

Nel caso in cui si presentasse la necessità di interrompere la cattura delle istantanee l'unica soluzione è scollegare la fotocamera dal sistema oppure, se il Raspberry è collegato a schermo e tastiera e si ha la possibilità di accedere al terminale, terminando il programma. Quest'ultima opzione termina tutto il programma, anche la parte relativa alla gestione del motore.

Alla pressione del *push button* viene anche acceso un LED, che viene spento quando tutta la procedura sarà terminata, per notificare che la macchina fotografica ha iniziato a scattare e per informare quando i 5 scatti sono stati salvati e la fotocamera è pronta per iniziare un altro ciclo o per essere scollegata.

```

78 void controlCamera(){
79     while(1){
80         int button_state = gpioRead(BUTTON);
81         gpioWrite(LED, 0); //Turn off led when finished shooting
82
83         if(button_state == 0){
84             gpioWrite(LED, 1); //Turn on led while shooting photos
85             system("sudo /home/pi/Desktop/StarTrackercode/setupCamera.sh");
86             system("sudo /home/pi/Desktop/StarTrackerCode/setParameters.sh");
87
88             for(int i=0; i<10; i++){ //To make 5 photos in a row
89                 system("sudo /home/pi/Desktop/StarTrackerCode/capture.sh"); //Shoot a photo
90             }
91         }
92     }
93 }

```

Figura 7: La funzione controlCamera()

- Funzione di controllo del motore: viene eseguita nel processo figlio e come per la funzione di controllo della fotocamera entrerà in un loop infinito. Prima però vengono definite alcune variabili, tutte di tipo intero:

- countStep: Tiene conto di tutti gli step che vengono compiuti dal motore in fase di tracciamento, in modo che successivamente si possa riportare il motore, e quindi la struttura del tracciatore, alla posizione iniziale. Questa variabile è necessaria in quanto se si riavvolgesse il motore senza tenere conto del punto in cui fermarlo, si rischierebbe che, una volta che la struttura è arrivata a fine corsa, il motore si sforzi inutilmente. Viene inizializzata a zero quando viene dichiarata.

- trackDelay: Variabile che viene utilizzata per memorizzare il ritardo in microsecondi da applicare tra uno step e il successivo del motore per avere la corretta velocità di tracciamento, ovvero circa 15,04° di apertura ogni ora.

Il calcolo esatto di questo ritardo è già stato trattato ed è stato trovato che deve essere di 15765µs

- rewindDelay: Simile alla variabile precedente ma questa rappresenta il ritardo tra uno step e l'altro in fase di riavvolgimento del motore. Per comodità è stata fissata in modo da ottenere la massima velocità possibile del motore; quindi, la pausa tra gli step sarà di 900µs e il motore a compiere un giro impiegherà $900 \cdot 4096 \approx 3,68$ secondi, che corrispondono a una velocità di circa 15 giri al minuto.

```

54 void controlMotor(){
55
56     int countStep = 0; //Counts steps done while tracking
57     //to reset motor later
58     int trackDelay = 15723; //Almost 64 steps per second (15723 us per step)
59     int rewindDelay = 900; //max velocity
60
61     while(1){
62         //if switch button is on move motor to track.
63         if(gpioRead(SWITCH)){
64             moveMotor(TRACK);
65             countStep++;
66             gpioDelay(trackDelay);
67         }
68         //if switch button is off move motor to rewind,
69         //then wait to track again
70         else if(countStep > 0){
71             moveMotor(REWIND);
72             countStep--;
73             gpioDelay(rewindDelay);
74         }
75     }
76 }

```

Figura 8: La funzione controlMotor()

A questo punto si entra nel loop e a ogni ripetizione del ciclo viene controllato lo stato dello *switch button*. Se l'interruttore è acceso viene chiamata la funzione per muovere il motore nel verso corretto per il tracciamento, ovvero in senso orario, e viene incrementata la variabile "countStep", in modo da tenere traccia della quantità di passi eseguita, infine viene aggiunto il tempo da attendere prima di eseguire la prossima iterazione.

Se l'interruttore è spento viene controllato il valore di "countStep" e, se quest'ultimo non è zero, viene chiamata nuovamente la funzione per muovere il motore ma questa volta in senso opposto, ovvero antiorario, e viene decrementata la variabile. Il riavvolgimento continua sinché la variabile non diventa

zero, il che significa che si è tornati alla posizione iniziale, oppure sinché non viene attivato nuovamente lo *switch button*. In quest'ultimo caso si interrompe il riavvolgimento e il motore cambia direzione per riprendere a tracciare il cielo.

Infine, se l'interruttore è inattivo e la variabile *"countStep"* è già a zero, non viene fatto nulla; quindi, il motore rimane fermo e attende l'attivazione.

Funzione *"moveMotor"* per gestire il movimento del motore: su occupa di muovere il motore nella direzione corretta. In ingresso avrà un parametro direzione. Se quest'ultimo è *"TRACK"*, che dalle direttive iniziali si traduce con un 1, dovrà muovere il motore in senso orario, quindi viene eseguito uno step, tramite una funzione che fornisce l'output ai pin che pilotano il motore, e viene decrementato il valore della variabile globale *"step"*, in modo che all'iterazione successiva vengano dati in output i valori corretti secondo la *"Tabella 3"*. Dopo aver decrementato la variabile viene controllato che questa non sia diventata negativa (si ricorda che deve variare tra 0 e 7) e in questo caso viene ripristinata a 7. Se il parametro direzione è *"REWIND"* viene eseguita la stessa procedura appena descritta solo che la variabile *"step"* viene incrementata invece che decrementata. L'effetto è che il motore si muoverà nella direzione opposta. Anche qua la variabile viene controllata dopo essere stata aumentata e, in caso abbia superato il valore *"7"* viene ristabilita a 0.

L'ultima funzione, ovvero la *"doStep"* è composta solo da uno *"switch-case"* che riceve in ingresso la variabile *"step"*, che rappresenta lo step che si deve eseguire, e manda in output ai pin del motore i valori per eseguire quello step. Lo *"switch-case"* ha anche un caso di default, in cui ricade se si verifica qualche errore per cui la variabile *"step"* non ha nessuno dei valori compresi tra 0 e 7. Nel caso *"default"* i pin vengono messi tutti bassi, quindi non avremo nessuna delle 4 spire del motore attraversata da corrente e il motore rimarrà fermo nella posizione in cui si trovava allo step precedente.

Nel file del programma C, cioè l'ultimo descritto, ogni chiamata ai file esterni viene fatta utilizzando il percorso assoluto di ognuno di questi file, altrimenti, utilizzando il percorso relativo, i file non verrebbero trovati perché verrebbero cercati nella cartella *"/home"* del Raspberry Pi.

SPERIMENTAZIONE E RISULTATI

Per la scelta della fotocamera da utilizzare si è tenuto conto di diversi fattori: come prima cosa era necessario che questa fosse controllabile da remoto attraverso una libreria software, in secondo luogo doveva avere dei parametri che fossero soddisfacenti per la cattura di un cielo notturno.

Come già descritto nel paragrafo 2, la libreria utilizzata per controllare la fotocamera dal Raspberry Pi è la *"libgphoto2"*, scritta in C, ed è alla base di un'interfaccia, chiamata *"gPhoto2"*, che permette di utilizzare la libreria tramite la *command line* di Linux. La pagina del software fornisce una lista di fotocamere compatibili, di cui molte sono solo rilevabili dalla libreria per il trasferimento dei file immagine, mentre è specificato che alcune hanno la possibilità di essere comandate per lo scatto e di cui possono essere modificate le impostazioni dalla *command line*.

Considerata la compatibilità e il fatto che la macchina non poteva essere troppo pesante, per evitare che il motore facesse troppa fatica, è stata scelta la *"Nikon-1 J5"*, una fotocamera mirrorless dalle dimensioni contenute con buone qualità dal punto di vista della scelta dei parametri. Ha infatti la possibilità, in modalità manuale, di estendere il tempo di apertura del sensore fino a 30 secondi, ha una sensibilità ISO fino a 12800 e fornisce una risoluzione di 20,8 megapixel, con la possibilità di scattare fotografie in formato RAW, ovvero non compresso, oppure in JPEG, o anche entrambi. Il formato RAW in questo caso è utile perché permette di non perdere nessuna informazione nelle immagini e questo è fondamentale perché il programma che verrà usato per la post-elaborazione dell'immagine lavori correttamente.

La fotocamera è prevista di un obiettivo zoom, ovvero a lunghezza focale variabile, che permette di variare quest'ultima tra i 10 e i 30mm. L'obiettivo zoom dà la possibilità di fotografare grandi porzioni di cielo se tenuto a una lunghezza focale di 10mm, mentre con 30mm permette fotografie più mirate a qualche corpo celeste più specifico, anche se comunque non consente uno zoom così accentuato da immortalare particolari piccoli del cielo.

Per quanto riguarda i parametri da impostare, abbiamo visto che il formato delle immagini, l'ISO, il tempo di esposizione e il bilanciamento del bianco vengono impostati automaticamente dal programma. Tuttavia, alcuni parametri non sono modificabili via software quindi dovremo avere cura di impostarli correttamente prima di collegare la macchina fotografica al sistema.

Come prima cosa dobbiamo assicurarci di essere in modalità manuale, in modo da poter modificare i parametri a nostro piacimento. Nella pagina delle impostazioni della camera è necessario spegnere la

riduzione automatica del rumore sia per le lunghe esposizioni sia per gli alti valori di ISO. Questo accorgimento è fondamentale poiché queste impostazioni impiegano parecchio tempo di elaborazione prima di salvare la foto e il collegamento con il Raspberry andrebbe in timeout, provocando diversi errori che possono anche bloccare il programma. Comunque, il rumore verrà eliminato con un veloce lavoro di post-elaborazione.

Sarà necessario anche scegliere il valore di zoom dell'obiettivo e impostare la modalità di messa a fuoco manuale. Generalmente va bene impostare il valore della messa a fuoco su "infinito".

Per iniziare a fotografare il cielo abbiamo bisogno come prima cosa di utilizzare il cannocchiale del tracciatore per puntare la stella polare, poi possiamo montare la macchina fotografica sul braccio superiore e inquadrare il soggetto di nostro interesse o la porzione di cielo che vogliamo ritrarre. Solo a questo punto possiamo collegare con un normale cavo USB la camera al Raspberry Pi. Non conviene farlo prima perché nel momento in cui si effettua il collegamento il display della macchina fotografica si spegne e non ci dà la possibilità di vedere quello che stiamo inquadrando né di fare qualsiasi cosa utilizzando i comandi della camera.

A questo punto possiamo far partire il motore attraverso lo switch e poi premere il bottone per lo scatto delle fotografie. Verranno scattate 10 immagini da 30 secondi, quindi, considerando anche che ci vuole qualche secondo perché vengano memorizzate dopo lo scatto, ogni set di foto che facciamo impiegherà tra i 5 minuti e mezzo e i 6 minuti.

Per verificare il funzionamento della struttura e soprattutto per verificare che la velocità a cui è stato impostato il motore fosse corretta sono stati fatti 2 scatti alla stessa porzione di cielo, senza cambiare inquadratura, entrambi da 30 secondi di esposizione, ma il primo senza azionare il motore, quindi senza l'utilizzo del tracciatore, mentre il secondo attivando il motore e quindi il tracciamento.



Figura 9: Fotografia senza l'uso del tracciatore

Nella prima immagine (Figura 6), fatta senza lo Star Tracker, possiamo vedere come le stelle non appaiono puntiformi ma si originano gli *star trails* di cui si è già parlato nel paragrafo "introduzione". Nella seconda immagine (Figura 7), possiamo invece apprezzare come l'utilizzo del tracker abbia dato i risultati sperati in quanto le stelle appaiono puntiformi e definite. A questo punto possiamo constatare che il sistema funziona e che la velocità del motore è corretta. Le immagini qui riportate sono state scattate in un'area comunque abbastanza luminosa, in periferia della città di Genova, nel quartiere di apparizione, che non ha lo stesso inquinamento luminoso del centro città, ma non offre neanche un paesaggio particolarmente buio. Inoltre, a queste due immagini non è stato applicato nessun metodo di riduzione del rumore e non sono state elaborate in alcun modo. Con la post-elaborazione si possono ottenere risultati molto più soddisfacenti.



Figura 10: Fotografia con l'utilizzo del tracciatore

POST-ELABORAZIONE DELLE IMMAGINI:

Il lavoro di post-produzione permette di ottenere immagini molto più pulite, senza rumore e consente di ridurre notevolmente l'inquinamento luminoso che è chiaramente visibile nelle fotografie non lavorate.

Per un buon miglioramento delle immagini abbiamo bisogno di:

- **Bias Frames:** una trentina di immagini che devono essere scattate completamente al buio, se possibile anche con l'obiettivo della macchina fotografica coperto, e impostando il tempo di esposizione minore possibile, per la fotocamera che abbiamo usato in questo caso, si parla di 1/16000 secondi.
- **Flat Frames:** una trentina di immagini scattate contro una superficie piatta e bianca, l'esposizione dovrà in questo caso essere tale da avere l'indicatore dell'esposizione a 0.
- **Dark frames:** sempre una trentina, dovranno, come i primi, essere scattati completamente al buio, possibilmente anche con l'obiettivo coperto, ma questa volta dovranno avere lo stesso tempo di esposizione che utilizzeremo per fotografare le stelle e dovranno essere scattate alla stessa temperatura di quella che avremo quando fotograferemo le stelle. Per comodità si possono realizzare poco prima di quelle che faremo al cielo.
- **Light Frames:** sono finalmente le immagini delle stelle. Ne andranno scattate tra le 50 e le 100. Per confronto, senza utilizzare un tracciatore per un risultato simile, dovremo avere anche più di 1000 di queste immagini, dato il minor tempo di esposizione.



Figura 11: In questo caso per i flat frames impostiamo 1/10 di secondo

Tutti i tipi di immagini elencati sopra dovranno essere scattati utilizzando la stessa ISO, la stessa impostazione di bilanciamento del bianco e dovranno essere tutte in formato RAW. Inoltre, i *flat* e i *dark frames* dovranno avere anche lo stesso valore di zoom dei *light frames*.

Una volta ottenute le fotografie possiamo procedere con la sovrapposizione. Le immagini, infatti, dovranno essere sommate per ottenere più definizione e più luminosità delle stelle e per ridurre il rumore originato dal sensore della camera. Essendo parecchie le immagini da sovrapporre, eseguire il

lavoro a mano sarebbe molto lungo e impegnativo, dato che si tratterebbe di allineare le immagini una ad una e poi sovrapporle, quindi possiamo fare uso di un programma apposito, ovvero *“Deep Sky Stacker”*[2], dove possiamo caricare tutti i tipi di frames che abbiamo fatto. Il software utilizzerà *dark*, *flat*, e *bias frames* per individuare dove viene originato il rumore dal sensore della macchina fotografica e si servirà di queste informazioni per eliminarlo dai light frames, che allo stesso tempo verranno sovrapposti automaticamente per ottenere un'unica immagine.

L'immagine che otterremo avrà a questo punto una dimensione considerevole, anche più di 200MB, e sarà in formato *“.tif”*. Potrà essere caricata su un editor foto per l'elaborazione finale. Quest'ultimo passo è facoltativo, se siamo già soddisfatti del risultato ottenuto possiamo utilizzare l'editor foto solo per esportare l'immagine in un formato compresso, come il JPEG, in modo che non occupi così tanto spazio in memoria. Come editor foto viene usato *“Gimp”*[4], ma può andare benissimo anche *“Photoshop”*[3]. Sull'editor possiamo visualizzare l'istogramma della fotografia e noteremo che presenterà un picco verso il margine sinistro.

Grazie all'editor possiamo utilizzare gli indicatori dell'istogramma per far sì che il picco sia più esteso e più centrale. In questo modo avremo un'immagine ancora più definita e riusciremo ad eliminare completamente la luminosità dovuta all'inquinamento luminoso.

Se si volesse una spiegazione più dettagliata, è consigliata la visione di un video sul post-processing delle foto della nebulosa di Orione di *“Nebula Photos”* [9].

SPERIMENTAZIONE FINALE:

Per verificare le potenzialità del sistema terminato è si è provato a scattare un'immagine più elaborata di quelle mostrate in precedenza, che erano utili più che altro a verificare che il tracciamento avvenisse correttamente. Come soggetto della nostra fotografia si è scelta una porzione della Via Lattea.

Per una buona post-elaborazione sono stati acquisiti: 34 *Bias Frames*, 31 *Flat Frames*, 28 *Dark Frames* e 60 *Light Frames*.

I *Light Frames* sono stati scattati in una zona relativamente buia, sulla strada provinciale del Monte Fasce, sulle alture di Genova.

Avendo la città vicino era comunque presente un po' di inquinamento luminoso specie vicino all'orizzonte. Per questo è stata scelta una porzione della Via Lattea che fosse il più lontano dall'orizzonte possibile.

Le immagini sono state scattate il 23 maggio 2023, tra le 2 e le 3 e 30 del mattino, per un'esposizione complessiva di 30 minuti. I parametri sono quelli già illustrati in precedenza quindi 30 secondi di esposizione per ogni immagine, con ISO a 1600 e bilanciamento del bianco per la luce diurna.

Le immagini sono poi state poi sovrapposte utilizzando il software già citato, ovvero *Deep Sky Stacker*, e successivamente il risultato è stato elaborato in *GIMP* per rendere più chiare le parti meno luminose e per eliminare più possibile l'inquinamento luminoso presente. L'immagine finale risulta comunque molto nitida.

Si possono notare in particolare le regioni della Via Lattea *“The Great Rift”* e *“Nube Stella Di Cigno”*, che nell'immagine sono abbastanza centrali, la prima sarà un po' spostata in alto a destra e la



Figura 12: Esempio Istogramma iniziale

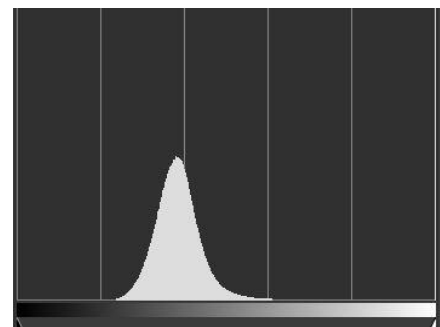


Figura 13: Esempio Istogramma modificato

seconda in basso a sinistra, e spiccano le stelle “Vega” e “Altair”, rispettivamente in alto a sinistra e in basso a destra della Via Lattea, che sono le due più luminose.



Figura 14: L'immagine finale della Via Lattea

CONTRIBUTO PERSONALE E CONSIDERAZIONI CONCLUSIVE

Come già accennato nel capitolo “Introduzione” l’idea del progetto è stata presa da I2SDD – Ham Radio[1], che ha realizzato uno Star-Tracker grazie ad Arduino. Nella descrizione della sua realizzazione è illustrata in modo dettagliato la parte meccanica, con anche un elenco di tutti i componenti necessari. Rispetto a quelli indicati in questa lista, sono stati utilizzati in più una basetta in legno autocostruita per fissare uno snodo (dove viene fissata la macchina fotografica) al braccio superiore, con le relative 4 viti e 4 dadi M6, e una vite da 6mm di diametro e un passo compatibile con quello standard delle macchine fotografiche con 2 dadi e 2 rondelle per fissare tutta la struttura al cavalletto tramite il braccio inferiore.

Sul sito di I2SDD sono presenti anche i file in formato “.stl” per le parti da stampare in 3D. Tra questi c’è un pezzo per l’alloggiamento del motore e tra i file ci sono alloggiamenti per motori diversi, in particolare per il 28BYJ-48, per il NEMA14 e per il NEMA17. Per la nostra realizzazione è stato stampato solo l’alloggiamento per il 28BYJ-48, dato che il motore utilizzato è quello.

Il problema più grosso che si è dovuto affrontare è stato dato dal fatto che le parti stampate avevano qualche difetto, in parte per la non perfetta precisione della stampante 3D e in parte perché le misure fornite nei disegni non erano corrette. Si è dovuto fare un lavoro di adattamento prestando molta attenzione sul fatto che le misure finali del tracciatore dovevano comunque essere precise, per evitare che si bloccasse durante il funzionamento e per garantire il corretto movimento del triangolo.

Questo lavoro è consistito nel:

- Allargare i fori dei due supporti del cannocchiale, che erano troppo stretti di circa 2mm ciascuno
- Allargare gli alloggiamenti dei cuscinetti di circa 1 mm per permettere ai cuscinetti di entrare nella loro sede.
- Assottigliare di circa 1 mm il diametro delle 4 aste metalliche che collegano la parte della cerniera agli elementi basculanti di motore e asta filettata alle estremità (per 40mm circa lato

cerniera e per 25mm circa lato supporti basculanti) per permettere che entrassero nei fori delle parti stampate.

L'ultimo problema di tipo meccanico è stato trovare un modo per trasferire il movimento del motore all'asta filettata. Questo è stato risolto praticando un incastro rettangolare in una delle due estremità dell'asta filettata in modo che l'albero del motore ci potesse essere inserito in maniera precisa. Questo incastro è stato ottenuto praticando un foro sull'asse dell'asta filettata, che è stato successivamente reso rettangolare allargandolo solamente in due direzioni.



Figura 15: Il foro dove viene inserito l'albero motore

Per preparare la parte software e i collegamenti hardware per far funzionare la parte elettronica è stato necessario come prima cosa studiare il funzionamento e le modalità di programmazione del Raspberry Pi 3, dopodiché è stato installato il sistema operativo *Raspbian* e si è cominciato a scrivere il codice C.

Per il software è stata scritta tutta la parte relativa alla gestione del motore, mentre per la gestione della fotocamera è stata installata e utilizzata una libreria (la "*libgphoto2*"). È stata anche installata una libreria per la gestione della GPIO del Raspberry Pi (la "*piGPIO*") con cui vengono pilotati i pin che controllano il motore e il led e con cui vengono letti gli input provenienti dai due bottoni.

Avendo interamente scritto il codice è stato necessario anche realizzare da zero tutti i collegamenti hardware associati.

Per comodità e per rendere meno probabile la rottura dei componenti che formano l'hardware il Raspberry, la basetta e il driver del motore sono stati fissati a una tavoletta in legno. Il Raspberry e la basetta si possono rimuovere e ri-fissare alla basetta senza problemi, mentre il driver del motore è avvitato alla tavoletta, in quanto non dovrebbe essere necessario rimuoverlo, se servono gli altri componenti si può scollegare e lasciare avvitato alla tavoletta.

Infine, mi sento di poter concludere dicendo che la realizzazione di questo progetto mi ha reso evidente come sia importante la parte di studio e conoscenza dei componenti da utilizzare (in questo caso in particolare il Raspberry Pi 3 e il motore), oltre a tutte le nozioni già acquisite durante i corsi, e quanto tempo si debba impiegare nella risoluzione dei problemi non previsti che si rendono evidenti solo durante la realizzazione. Ad esempio, non ci si aspettava di dover modificare così pesantemente alcune parti stampate in 3D.

RIFERIMENTI BIBLIOGRAFICI

- [1] <https://www.raspberrypi.com/software/>
- [2] <http://deepskystacker.free.fr/english/index.html>
- [3] https://www.adobe.com/it/products/photoshop/landpb.html?gclid=CjwKCAjw1MajBhAcEiwAagW9MTuNXwS4tE_tcggb3z_2gecsIbBEyFXZEWJUbyCV711jJ0F6l6kYxoCrhQQAuD_BwE&mv=search&mv=search&sdid=LZ32SYVR&ef_id=CjwKCAjw1MajBhAcEiwAagW9MTuNXwS4tE_tcggb3z_2gecsIbBEyFXZEWJUbyCV711jJ0F6l6kYxoCrhQQAuD_BwE:G:s&s_kwcid=AL!3085!3!595517531759!e!!g!!photohop!1457478956!59242745680&qad=1
- [4] <https://www.gimp.org/>
- [5] <https://www.i2sdd.net/ARDUINO/STARTRACK/STARTRACKER.HTML>
- [6] <https://www.astrati.eu/>
- [7] <https://lastminuteengineers.com/28byj48-stepper-motor-arduino-tutorial/>
- [8] <http://www.gphoto.org/>
- [9] https://www.youtube.com/watch?v=4_a8XmC6H3I&list=PLrzbdmripj1c9p7hzt7ffgH3oN8vzCFI3&index=1&ab_channel=NebulaPhotos