UNITÀ 5: SEGNALI ANALOGICI

OBIETTIVI

Fino ad ora abbiamo detto che Arduino utilizza solo segnali digitali con livelli di "1" o "0". I segnali di input possono provenire da pulsanti, interruttori, rilevatori e molte altre fonti, ma devono essere preceduti da un "1" o uno "0". I segnali di output possono alimentare luci a LED, interruttori a relè, motori e molti altri dispositivi, ma ancora una volta dobbiamo prevedere un livello "1" o uno "0". Persino i segnali PWM che hai usato per controllare la luminosità di una luce a LED o il posizionamento di un servomotore sono digitali proprio come le frequenze e i toni di un altoparlante.

Ma non tutti gli impulsi sono digitali. Ci sono anche quelli che sono chiamati segnali "analogici" e il loro valore o voltaggio può variare tra un minimo e un massimo in un periodo di tempo.

In questa unità studierai come funzionano questi segnali, come usarli e cosa puoi fare con loro. Imparerai a conoscere le funzioni in linguaggio Arduino che manipolano gli input analogici del controller.

SESSIONE TEORICA

- INTRODUZIONE
- CONVERSIONE DIGITALE
- RISOLUZIONE
    - E ora è il tuo turno
- FUNZIONI DI LINGUAGGIO ARDUINO
- PERIFERICHE ANALOGICHE
    - Potenziometri
    - Sensori fotografici
    - Sensori IR riflettenti
    - Condizionatori d'aria
- OUTPUT PSEUDO ANALOGICI
    - Che cosa sono i segnali PWM?
    - Per cosa sono usati?
    - Come sono generati?
        - La funzione analogWrite()

- Ulteriori funzioni in linguaggio Arduino
    - La funzione random()
    - La funzione randomSeed()

SESSIONE PRATICA

- ESEMPIO 1: Conversione ADC
- ESEMPIO 2: Soglie
- ESEMPIO 3: Comparatore analogico
- ESEMPIO 4: Controllo luminosità
- ESEMPIO 5: Timone elettrico
- ESEMPIO 6: Fotometro
- ESEMPIO 7: Controllo dell'illuminazione
- ESEMPIO 8: Misurazione dei riflessi
- ESEMPIO 9: Rilevamento dei colori
- ESEMPIO 10: Temperatura
- ESEMPIO 11: Condizionatore
- ESEMPIO 12: Segnale PWM
- ESEMPIO 13: Effetti ottici
- ESEMPIO 14: Regolazione manuela
- ESEMPIO 15: Luci random

MATERIALI PRATICI

-Laptop o computer da tavolo

-Arduino IDE ambiente di lavoro; questo dovrebbe includere il materiale supplementare già installato e configurato.

-Connettore Arduino UNO

-Un cavo USB

SOMMARIO

SEZIONE TEORICA

1. INTRODUZIONE

L'espressione "segnali di input analogici" potrebbe essere emersa durante il corso, ma cosa sono, per cosa sono utilizzati, dove sono e come vengono controllati? È giunto il momento di rispondere a queste domande.

Sei già consapevole che tutto nel mondo "digitale" funziona sul presupposto che ci siano solo due possibili valori o livelli: livello "1" e livello "0". Un pulsante, un interruttore o un rilevatore possono essere attivati ("1") o disattivati ("0"). È possibile accendere o spegnere un LED, un relè o un motore. Il suono non è altro che un segnale che va dal livello "1" al livello "0" a una data velocità o frequenza. Un segnale PWM è digitale e possiamo variare la lunghezza del livello "1" o il ciclo di lavoro e quindi regolare la tensione. La comunicazione seriale non è altro che il trasferimento di bit con livelli di "1" e "0".

Quando abbiamo un dato numero di bit, li mettiamo insieme in byte e li usiamo per creare numeri di dimensioni diverse, codificare caratteri e inviare messaggi. In ogni caso, questo è il modo in cui abbiamo lavorato fino ad ora.

Tuttavia, il mondo "reale" non è così. Troviamo quantità fisiche nel mondo naturale che possono avere più valori o altre caratteristiche.

Pensa ad esempio alla temperatura dell'ambiente. Questa è una quantità fisica che cambia costantemente. La temperatura non è la stessa al mattino come a mezzogiorno o alla sera. Se avessimo un sensore che potesse misurare la temperatura e generare una tensione proporzionale ad esso, vedremmo che varia costantemente nel tempo più o meno come mostrato nella figura più in basso.

La temperatura è una grandezza fisica analogica. Il sensore eroga una tensione di 2 V alle h 5.00 del mattino. Alle h 9.00 la temperatura sale e quindi anche la tensione; sale a 4 V. Alle h 15.00 la tensione raggiunge i 5 V e dalle h 16.00 in poi inizia a scendere con la temperatura. Tutto ciò che dobbiamo fare è cercare una relazione tra la temperatura e la tensione fornita dal sensore.

Ora pensa all'enorme numero di quantità fisiche analogiche che ci circondano. Usando opportuni sensori o "trasduttori" è possibile trasformare queste quantità fisiche in equivalenti tensioni analogiche:

- ✓ Umidità e / o umidità relativa. Questo ci consente di calcolare la quantità di vapore acqueo nell'atmosfera.
- ✓ Pressione atmosferica. Possiamo misurare la pressione che l'aria esercita sulla Terra usando un sensore adatto.

- ✓ Peso. Possiamo misurare la forza che un corpo esercita su un punto a riposo.
- ✓ Velocità. Col sensore adatto, possiamo misurare quanto velocemente si muove l'aria o quanto velocemente un veicolo si muove.
- ✓ Luce. Puoi misurare la luce ambientale o la luce che colpisce un oggetto. Ci sono sensori che rilevano la luce visibile, la luce infrarossa, la luce ultravioletta e altri.
- ✓ Sound. Siamo in grado di misurare, studiare e / o rilevare rumori, suoni ambientali, volume e altri fenomeni acustici.

Esistono molti altri sensori di area o "trasduttori" che possono trasformare le grandezze fisiche in equivalenti tensioni analogiche.

2. CONVERSIONE DIGITALE

Come puoi ben immaginare, ci sono diversi tipi di sensori e "trasduttori" che sono in grado di misurare e fornire una tensione analogica equivalente alla quantità fisica che stanno misurando. Sfortunatamente, nessun sistema digitale è in grado di gestire o elaborare direttamente queste correnti analogiche; nemmeno Arduino.

La prima cosa che dobbiamo fare è convertire le tensioni analogiche in valori digitali o binari equivalenti. Per eseguire questa conversione utilizziamo circuiti elettronici chiamati "convertitori analogico-digitali", abbreviati in "ADC".

Dai un'occhiata alla figura 1. Ad esempio, mostra i componenti necessari per elaborare una grandezza fisica analogica come la temperatura.

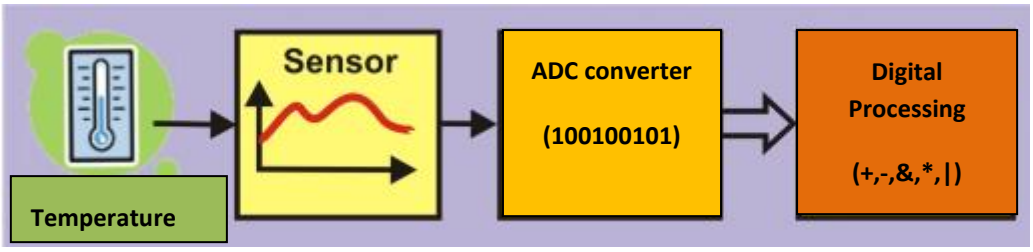

Figura 1

1. Il sensore rileva la quantità fisica da misurare; in questo caso è la temperatura. Generalmente genera una tensione proporzionale alla quantità fisica.

2. La tensione viene quindi inserita nel convertitore analogico-digitale (ADC). Il circuito emette un valore binario equivalente alla tensione di ingresso.

3. Il valore binario può ora essere letto da un controller come Arduino.

4. Il controller può eseguire qualsiasi tipo di processo con questo valore binario: memorizzarlo nella memoria, eseguire operazioni aritmetiche o logiche, visualizzarlo, trasferirlo su un altro sistema e molti altri.

Quasi tutti i moderni controller incluso Arduino hanno un convertitore ADC integrato. Tutto quello che devi fare è connettere il sensore o trasduttore appropriato al pin di ingresso analogico. Il tipo di sensore o trasduttore utilizzato dipende dalla quantità fisica che si intende misurare.

Inoltre, questi controller hanno in genere un numero di pin di ingresso per segnali analogici. Dai un'occhiata alla figura a sinistra. La maggior parte dei controller ha solo un circuito convertitore ADC con un numero di pin di ingresso o "canali analogici" collegati ad esso. Arduino UNO ha un singolo convertitore e sei pin di ingresso analogici collegati ad esso: numeri da A0 a A5. Ciò consente di prelevare campioni di tensioni analogiche da ben sei diversi sensori.

Ovviamente puoi solo prendere un campione alla volta. Utilizzando le funzioni adatte è possibile convertire una alla volta le tensioni su ciascun canale di ingresso analogico. Diciamo che i canali di input sono "multipli".

Dai un'occhiata da vicino alla figura 2. Ogni volta che richiedi una conversione, il convertitore ADC preleva un campione della tensione analogica sul canale di ingresso specificato. Il risultato, o il suo equivalente binario, è ottenuto in 100 µS (0,0001"), che è quanto tempo è necessario per eseguire la conversione. Arduino può eseguire circa 10.000 conversioni al secondo.

La tensione analogica è 2 V al primo istante. Il convertitore ADC genera un valore o numero binario equivalente a questa tensione. La tensione analogica è di 2,8 V al secondo istante, 3,5 V al terzo, 4,2 V al quarto e così via. Puoi vedere come è un segnale analogico: nell'esempio fluttua costantemente tra un minimo di 0 V e un massimo di 5 V.

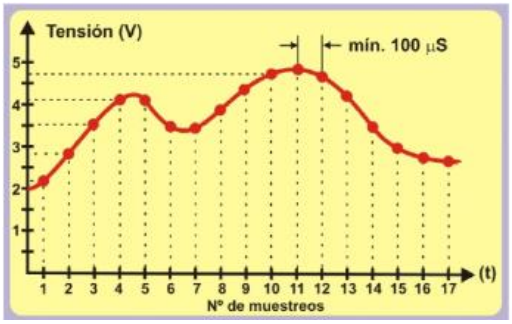

Figura 2

Ho già detto che Arduino UNO richiede circa 100 µS per eseguire una conversione. Questo è abbastanza veloce, ma ce ne sono altri che possono farlo molto più velocemente. Dai un'occhiata alla figura a sinistra. Mostra un segnale analogico con una frequenza, F, di 1000 Hz. La sua durata, T, è

1000 µS (1 mS). Ciò significa che Arduino sarebbe in grado di prelevare al massimo dieci campioni del segnale (1000 µS / 100 µS).

Se si avesse un segnale analogico con una frequenza, F, di 500 Hz e un tempo, T, di 2000 µS, si potrebbero ottenere un totale di venti campioni (2000 µS / 100 µS). Ciò significa che otterresti una "digitalizzazione" più precisa rispetto a quando la frequenza del segnale analogico era di 1000 Hz.

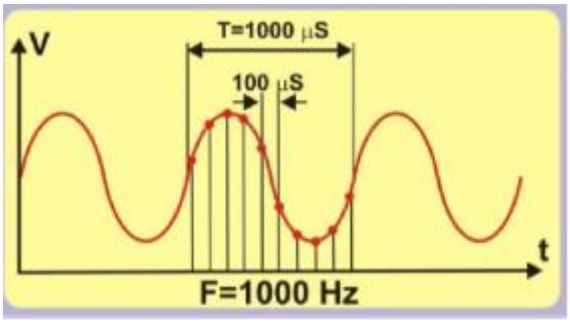

Figure 3

Non è sempre necessario prendere tanti campioni al secondo. Ricordi il sensore che misura la temperatura della stanza? Questa è una quantità fisica che non varia molto. La temperatura non sale da -10ºC a + 15ºC in 0,0001 secondi. La stessa cosa accade con la luce naturale. Non andiamo dalla notte al giorno a 100 µS. E per quanto riguarda il peso? Nulla pesa 75 kg un momento e 31 kg 100 µS dopo. Lo stesso vale per altre grandezze fisiche come velocità, pressione atmosferica, umidità e molte altre.

La velocità di conversione di Arduino UNO è più che sufficiente per misurare la maggior parte delle quantità fisiche analogiche!

3. RISOLUZIONE

Oltre alla velocità di conversione, un altro fattore importante in un circuito convertitore ADC è la sua precisione o "risoluzione". Diciamo che il convertitore incorporato in Arduino UNO ha una risoluzione di 10 bit. Ciò significa che il risultato di una conversione può avere 1024 valori binari possibili (210).

Come stabiliamo una relazione tra la tensione analogica e il valore binario? Abbiamo bisogno di conoscere una costante chiamata "tensione di riferimento" o VREF, che è la tensione utilizzata dal circuito convertitore per eseguire le sue operazioni interne. Calcoliamo la risoluzione per bit usando la

seguente equazione che dipende sia dal VREF sia dal numero di bit del convertitore. Supponendo che VREF = 5V:

$$Resolución = \frac{V_{REF}}{2^{10}} = \frac{5}{1024} = 0.0048V/Bit \cong 0.005V = 5mV$$

Se lo sai, puoi prevedere il valore di uscita binario che il convertitore ti fornisce in base alla tensione di ingresso analogica. Dai un'occhiata da vicino alla seguente tabella.

INPUT VOLTAGE	OUTPUT			**OUTPUT VOLTAGE**	OUTPUT		
	BINARY	**DEC.**	**HEX.**		**BINARY**	**DEC.**	**HEX.**
0.000	0000000000	0	0x000	**2.500**	1000000000	512	0x200
0.005	0000000001	1	0x001	**3.000**	1001100110	614	0x266
0.010	0000000010	2	0x002	**4.000**	1100110011	819	0x333
0.015	0000000011	3	0x003	**4.985**	1111111100	1020	0x3FC
0.020	0000000100	4	0x004	**4.990**	1111111101	1021	0x3FD
1.000	0011001100	204	0x0CC	**4.995**	1111111110	1022	0x3FE
2.000	0110011000	408	0x198	**5.000**	1111111111	1023	0x3FF

Tutto quello che devi fare è dividere la tensione di ingresso analogica dalla risoluzione bit del convertitore, che in questo caso è 0,0048.

A. E ORA E' IL TUO TURNO

Supponendo che la tensione di riferimento VREF sia 3 V e il convertitore abbia una risoluzione di 8 bit, completare la seguente tabella per ogni diversa tensione di ingresso analogica.

BIT RESOLUTION							
INPUT VOLTAGE	OUTPUT			**OUTPUT VOLTAGE**	OUTPUT		
	BINARY	**DEC.**	**HEX.**		**BINARY**	**DEC.**	**HEX.**

0.00				**1.50**			
0.01				**1.67**			
0.50				**1.85**			
0.65				**2.12**			
0.83				**2.57**			
0.95				**2.93**			
1.15				**3.00**			

.

QUANTO SEGUE È MOLTO IMPORTANTE! La tensione di ingresso analogica che si misura non deve MAI superare la tensione di riferimento VREF. Ad esempio, se VREF = 5 V, anche la tensione di ingresso analogica deve essere al massimo 5 V.

4. FUNZIONI IN LINGUAGGIO ARDUINO

L'utilizzo del convertitore ADC incorporato in Arduino UNO non potrebbe essere più semplice. Sono necessarie solo due funzioni del linguaggio di programmazione Arduino.

- **La funzione analogReference()**

Questo è importante perché una volta che sai di cosa si tratta, puoi calcolare la risoluzione per bit come hai fatto in precedenza.

La tensione VREF non deve superare la tensione che alimenta il controller UNO di Arduino (5 V). D'altra parte, la tensione di ingresso analogica convertita non può superare VREF.

Sintassi:

analogReference(type);

type: Configura il V_{REF} utilizzato per l'ingresso analogico (ovvero il valore utilizzato come la parte superiore dell'intervallo di input). Le opzioni sono:

PREDEFINITO: il riferimento analogico predefinito di 5 volt (su schede Arduino da 5 V) o 3,3 volt (su schede Arduino da 3,3 V).
INTERNO: questo è un V_{REF} generato all'interno del controller. Nel caso di Arduino UNO è 1,1 V.

ESTERNO: Il V_{REF} richiesto viene fornito sul codice AREF del controller.

Esempio:

analogReference(INTERNAL); **// Viene utilizzata la tensione interna di 1,1**

- **The analogRead() function**

Questa è la frase che userai quando vorrai eseguire una conversione da analogico a digitale. Ogni volta che viene eseguito, prende un campione della tensione sul pin o canale analogico specificato e quindi esegue la conversione.

Sintassi:

analogRead(pin);

pin: Il numero pin corrispondente al canale analogico che si desidera convertire. Nel caso di Arduino UNO questo potrebbe essere da A0 a A5.

Esempio:

int V;

V = analogRead(A2); *//Esegue la conversione della tensione su A2*

- **La funzione map()**

Anche se questa funzione non è espressamente progettata per la conversione ADC, potrebbe essere interessante per quello che stiamo facendo al momento. Permette di rimappare, riassegnare o ridefinire un valore tra un minimo e un massimo.

Vediamo; sai già che il convertitore ADC di Arduino ha una risoluzione di 10 bit e può esprimere un valore compreso tra 0 e 1023 (210). D'altra parte, i numeri con cui Arduino funziona sono pacchetti di byte (8 bit) o multipli di byte (int, unsigned int, long e unsigned long). A volte è meglio usare il risultato di una conversione (10 bit) come se fosse un byte compreso tra 0 e 255 (8 bit) o un int (16 bit) e così via. I byte o gli interi sono formati molto più standard.

Sintassi:

map(value, fromLow, fromHigh, toLow, toHigh);

value: il numero da mappare. Di solito è un int (16 bit) o un long (32 bit). I valori mobili non possono essere utilizzati

fromLow: Esprime il limite inferiore dell'intervallo del valore target.

fromHigh: Esprime il limite superiore dell'intervallo del valore corrente.

toLow: Esprime il limite inferiore dell'intervallo del valore target.

toHigh: Esprime il limite superiore dell'intervallo del valore target.

Esempio:

```
int V;
V = analogRead(A2);          //Esegue la conversion della tensione su A2
V = map(V,0,1023,0,255);     //Riassegna il valore letto e converte in un byte
```

Il valore analogico letto sul pin A2 si trova tra 0 e 1023 e viene memorizzato nella variabile "V". Questo valore è rimappato in un equivalente tra 0 e 255.

Secondo Arduino questa funzione è derivata dal seguente calcolo matematico; lo includiamo puramente per curiosità:

$$Resultado = \frac{(valor - min) * (Nmax - Nmin)}{(max - min) + Nmin}$$

5. PERIFERICHE ANALOGICHE

Esiste un'ampia gamma di sensori sul mercato in grado di fornire una tensione analogica tra un minimo e un massimo a seconda della quantità fisica in questione. Puoi considerarli come periferiche analogiche:

- ✓ **Potenziometri analogici**. Lo spostamento delle aste fornisce una tensione analogica variabile compresa tra 0 e 5 V. Sono collegati ai codici A0 e A1 e possono essere considerati come sensori di movimento analogici;
- ✓ **Sensore di luce**. Questo è collegato al pin A2. La tensione che genera dipende dalla quantità di luce ambientale che lo colpisce;
- ✓ **Sensore a infrarossi**. Questo è un sensore riflettente a luce infrarossa (IR) non visibile. È collegato al pin A3 e misura la luce IR che lo colpisce;
- ✓ **Sensore di temperatura**. Misura la temperatura ambiente e genera una tensione proporzionale. È collegato al pin A4.

A. POTENZIOMETRI

I potenziometri sono resistori variabili e puoi modificare il loro valore spostando l'asta o un controllo

chiamato "tergicristallo". Non ho dubbi che tu li abbia usati un sacco di volte senza rendertene conto; il controllo che usi per regolare il volume di una radio, televisione o sistema audio sono alcuni esempi.

Sono le periferiche analogiche più semplici ed economiche che troverai. Hanno diverse forme e dimensioni.

Ha tre terminali o pin. I pin 1 e 2 si trovano a ciascuna estremità del resistore. Rappresentano il suo valore totale. Il pin 3 è il tergicristallo. C'è un meccanismo che lo sposta da un'estremità del resistore all'altro variando il suo valore. Se dovessi posizionarlo proprio nel punto centrale del suo viaggio tra i codici 1 e 3 otterresti metà della resistenza. Se lo metti tra i codici 3 e 2 otterrai l'altra metà.

Ora guarda cosa sono le connessioni di entrambi i potenziometri. Le estremità sono collegate a 0 V e +5 V. Se si sposta il tergicristallo verso l'estremità inferiore, cioè a sinistra, la tensione analogica diminuirà fino a raggiungere 0 V. Se la si sposta nella direzione opposta, la tensione aumenterà fino a raggiungere un massimo di +5 V. I tergicristalli di entrambi i potenziometri sono collegati ai codici di ingresso analogici A0 e A1 sul controller.

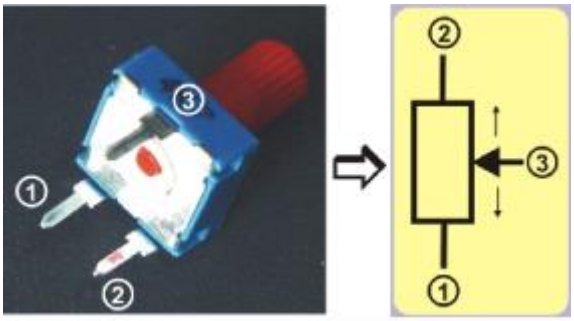

Figura 4

B. SENSORI FOTOGRAFICI

Si basano su un piccolo dispositivo chiamato "fototransistor". Dispensando con lunghe spiegazioni tecniche, potremmo dire che questo componente aumenta o diminuisce la quantità di corrente che attraversa in base alla quantità di luce che lo colpisce. Ci sono quelli che sono visibili o sensibili alla luce ambientale e altri che sono sensibili alla luce a infrarossi (IR). Sono disponibili in una gamma di forme e dimensioni.

Come mostrato nella figura a destra, possiamo usare una torcia per variare la quantità di luce che colpisce il sensore. Questo aumenta o diminuisce la corrente elettrica, I, che scorre attraverso di essa. Questa corrente, I, circola attraverso un resistore, R, e produce una tensione, V, che varia anche proporzionalmente: V = R * I. Per riassumere: quando la luce cambia, lo fa anche l'intensità della corrente, I, e questo a sua volta varia la tensione, V, tra 0 e 5 V. Questa tensione viene inviata all'ingresso analogico A2 dell'Arduino.

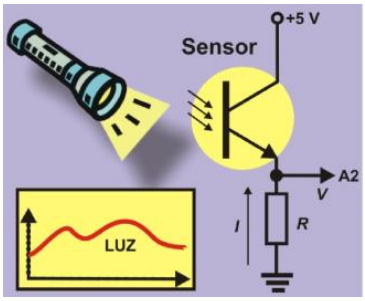

Figura 5

C. SENSORI IR RIFLETTENTI

Queso è un altro tipo di sensore di luce: rileva la luce a infrarossi (IR) non visibile all'occhio umano. Rileva la quantità di luce riflessa che colpisce. Il sensore comprende due componenti. La luce a LED o emettitore (E), che funziona a +5 V, emette un raggio costante di luce a infrarossi. Quando questo viene riflesso da un oggetto, colpisce il fotodiodo o il ricevitore (R) che assorbe la luce. A seconda della quantità di luce riflessa, il ricevitore aumenta o diminuisce l'intensità, I, si alimenta nel resistore, R, che a sua volta aumenta o diminuisce la tensione analogica, V. Ricorda: V = R * I. Per riassumere: più l'oggetto è vicino, maggiore è la quantità di luce riflessa. L'intensità, I, è maggiore e quindi anche la tensione V. Questa tensione viene applicata al pin di ingresso analogico A3 su Arduino UNO.

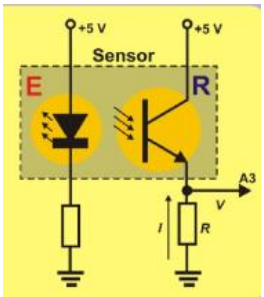

Figura 6

D. SENSORI DI TEMPERATURA

Dispositivo LM35, questo componente ha tre pin. Due di loro sono collegati alla tensione di alimentazione di 0 e +5 V. L'intensità, I, che circola attraverso il terzo pin, è direttamente proporzionale alla temperatura. Questa intensità attraversa il resistore, R, e questo crea una tensione, V (V = R * I). È collegato al pin di input analogico A4.

Secondo il produttore di questo dispositivo, la risoluzione del sensore è 10 m V / ºC. È possibile utilizzare la seguente equazione basata sulla tensione analogica (Va) per calcolare la temperatura ambiente in gradi centigradi

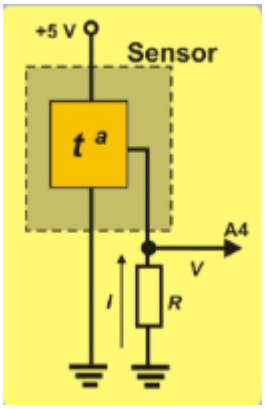

Figura 7

$$^{o}C = \frac{5 * Va * 100}{1024} = \frac{Va * 500}{1024}$$

6. OUTPUT PSEUDO ANALOGICO

La piattaforma Arduino si compone di input e output digitali. Abbiamo visto che la scheda di controllo di Arduino UNO ha 14 pin che possono essere configurati come input o output digitali. In effetti li abbiamo visti nella maggior parte degli esercizi finora.

Possiamo ricordare che 6 di questi pin possono funzionare anche come pin di output del segnale PWM. Dai un'occhiata alla figura, mi riferisco ai pin 3,5,6,9,10 e 11, quelli preceduti dal segno "~"; hanno una freccia che li punta.

In questa unità verranno usati come output del segnale PWM. Puoi farti una idea di cosa sembrano i pin quando sono connessi allo sviluppo della scheda guardando la figura.

Gli output del segnale PWM numeri 6, 9, 10 e 11 sono connessi ai led. I numeri 3 e 5, che sono anche pin di output del segnale PWM, rendono possibile la connessione tra due servomotori o un motore DC; in questa unità connetteremo due servomotori.

A. CHE COSA SONO I SEGNALI PWM?

The abbreviation "**PWM**" stands for "*Pulse Width Modulation*", an "*asymmetric*" periodic digital signal of "1"s and "0"s which is repeated constantly at the same frequency, **F** (there are X number of cycle repetitions in a second). This means that the time the signal is at level "1" may be completely different to the time the signal is at level "0".

L'abbreviazione PWM sta per "Pulse with modulation", un segnale periodico "asimmetrico" di 1 o 0, che è ripetuto costantemente alla stessa frequenza F (ci sono ciclo ripetuti in un secondo). Questo significa che il tempo in cui il segnale è a livello "1" può essere completamente diverso dal tempo in cui il segnale è a livello "2".

Dai una occhiata al segnale della figura 8; lavoriamo su questo. È un segnale digitale con un periodo T di 2 mS. in altre parole, ci sono 500 cicli identici in un secondo (1000mS/2). La frequenza F è perciò 500Hz (F=1/T).

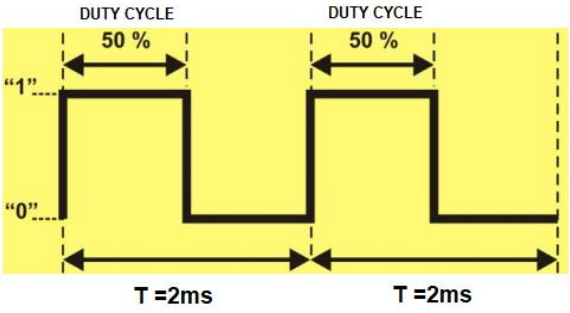

Figura 8

In questo caso il livello "1" di ogni ciclo dura lo stesso tempo del livello "1": 1 mS (se aggiungiamo un livello "1" e un livello "0" aumentano di 2m" il periodo T). Questo è chiamato segnale simmetrico. Il tempo del segnale che è sul livello "1" è chiamato "ciclo duty". Il suo valore è il 50% (1mS) del valore totale del periodo nell'esempio. È chiamato segnale PWM 50%.

Diamo un'occhiata ai quattro segnali PWM nella figura sotto. Hanno tutti lo stesso periodo T, o 2mS, o una frequenza F, di 500Hz, che è la stessa cosa. Comunque la durata del livello "1", che è il ciclo duty è diversa in tutti.

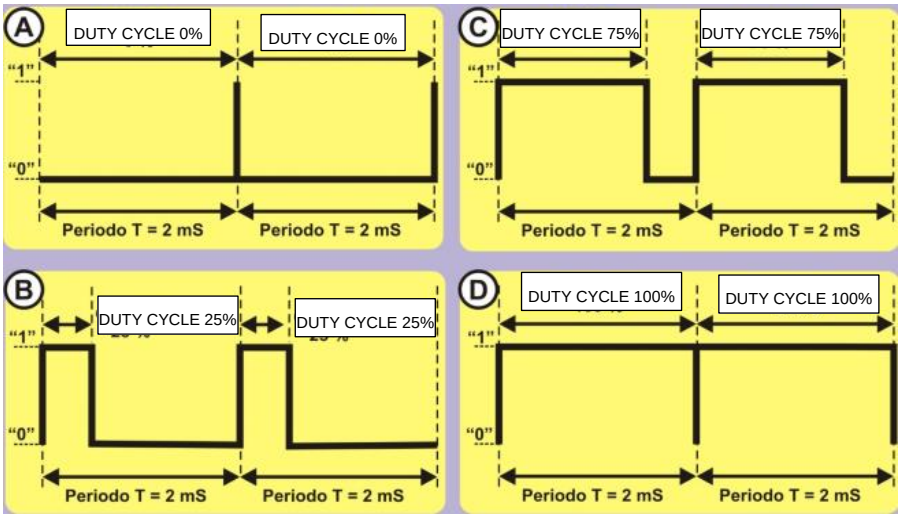

Figura 9

- ✓ **Segnale A**. questo è un segnale PWM con un ciclo duty con 0% di lunghezza del periodo. In altre parole, rimane al livello “1”per 0 mS (0*2 / 100) e a livello “0” per il 100% del tempo rimanente, che vuol dire, 2 mS (100% * 2 / 100).
- ✓ **Segnale B**. Questo è un segnale PWM con un ciclo duty del 25% di lunghezza del periodo. In altre parole, rimane a livello “1” per 0,5 mS (25% * 2 / 100) e a livello “0” per 75% del tempo rimanente, che significa 1,5 mS (75% * 2 / 100).
- ✓ **Segnale C**. Questo è un segnale PWM con un ciclo duty del 75%. Rimane a livello “1” per 1,5 mS (75% * 2 / 100) e a livello “0” per il 25% del tempo rimanente, che significa, 0,5 mS (25% * 2 / 100).
- ✓ **Segnale D**. Questo è un segnale PWM con un ciclo duty del 100%. Rimane a livello “1”per 2 mS (100% * 2 / 100) e a livello “0” per 0% del tempo restante, che è come dire, 0 mS (0% * 2 / 100).

Abbiamo già detto che la maggior parte dei controllori moderni sono in grado di generare uno o più segnali PWM su alcuni dei loro pin. Nel caso di Arduino UNO questi pin sono 3, 5, 6, 9, 10 e 11. Quando usi uno di loro come output PWM, devi conoscere la frequenza F dei segnali che Arduino genera. Guarda questa tabella:

PIN Nº	FREQUENZA, F	PERIODO, T
5 e 6	**980 Hz**	**1.02 mS o 1020 µS**
3, 9, 10 e 11	**490 Hz**	**2.04 mS o 2040 µS**

B. PER COSA SONO USATI?

Cerchiamo di evitare termini complicati. Possiamo controllare il tempo che rimane sul livello “1” per ciascun periodo con segnali PWM; questo è chiamato ciclo duty. Possiamo anche usarli per controllare e aggiustare il voltaggio che forniamo alle periferiche di output sicuro, come le lampadine, le luci LED, un motore, un servomotore e altri.

Immaginiamo un segnale PWM come quello della figura 10, connesso a una luce LED. Il ciclo duty è il 50% così il LED è acceso per il 50% del tempo e spento per l'altro 50%. Credici o no, il LED brillerà a metà luminosità per ogni periodo.

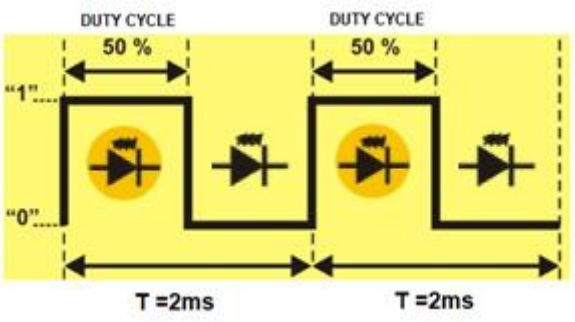

Figura 9

Per le stesse ragioni, se il ciclo duty è lo 0%, il segnale rimarrà sempre su livello "0". Il LED non brillerà. Se il ciclo duty è al 25%, il LED brillerà a un quarto della sua luminosità e se il ciclo duty è al 75%brillerà a tre quarti della sua luminosità. Se il ciclo duty è al 100% il segnale rimane sempre sul livello "1". Il LED brillerà al massimo. Questo ci dà un range tra 0% e 100% del segnale PWM per raggiungere la luminosità desiderata.

Allo stesso modo in cui puoi aggiustare la luminosità di una luce LED o di una lampada, puoi aggiustare la velocità di un motore o la posizione dell'albero di un servomotore. Il principio è lo stesso e lo potremo sperimentare a breve. Impareremo che ci sono tante periferiche che sono controllate attraverso segnali PWM.

C. COME SONO GENERATI?

I controllori in grado di generare segnali PWM hanno una serie di circuiti elettronici integrati, come oscillatori, timer, comparatori, registri e altri; questi sono piuttosto complessi. Attraverso la sincronizzazione e il lavoro condiviso di tutti questi strumenti, possiamo generare uno o più segnali di questo tipo. Comunque, è necessario avere molte competenze tecniche per fare tutto questo lavoro.

Per fortuna Arduino fa cose veramente semplici per noi. Dimentica I circuiti elettrici, come lavorano e come li usiamo. Ciò che ti serve è una singola funzione dal linguaggio di programmazione per generare un segnale PWM; Arduino si occupa del resto.

- **La funzione analogWrite()**

Questa funzione rende possibile aggiustare la lunghezza di un ciclo duty del segnale di output PWM.

Sintassi:

analogWrite(pin, value);

pin: pin da scrivere. Si riferisce al pin che deve generare il segnale PWM. Ricorda che il nostro controller Arduino UNO può usare numeri pin 3, 5, 6, 9, 10 e 11.

value: determines the length of the duty cycle. It's a byte number between 0 and 255. The value 0 represents 0% of the duty cycle, 127, a duty cycle of 50%, and 255 corresponds to a PWM signal with a duty cycle of 100%. Determina la lunghezza del ciclo. È un numero byte tra 0 e 255. Il valore 0 rappresenta un segnale PWM con lo 0% del ciclo duty, 127 rappresenta un segnale PWM un ciclo duty del 50% e 255 rappresenta un segnale PWM con un ciclo duty del 100%.

Esempio:

```
byte Voltage = 25;          //Indica la percentuale del voltaggio richiesto

int Cycle= Voltage*255/100; //Calcula un ciclo duty di lunghezza tra 0 e 255

analogWrite(6,Cycle);       //Genera un segnale PWM del 25% su pin 6
```

Non bisogna configurare il pin che genera il segnale PWM con un output; è già nella funzione. Il pin genera il segnale di output PWM indefinito fino a quando non si esegue di nuovo la funzione analogWrite() con un valore diverso o le funzioni digitalRead() o digitalWrite() sullo stesso pin. In tutti questi tre casi l'output del segnale PWM termina.

7. ULTERIORI FUNZIONI IN LINGUAGGIO ARDUINO

Sebbene le funzioni seguenti non hanno niente a che fare con I segnali PWM, diamo comunque un'occhiata affinché si possa allargare la conoscenza sul linguaggio; anche queste funzioni sono interessanti.

Riguardano la generazione di numeri casuali, che significa qualsiasi valore all'interno di un certo limite.

- **La funzione random()**

This function generates a random number between a minimum and a maximum. It returns a 32 bit signed integer value (long). Questa funzione genera un numero casual tra un Massimo e un minimo. Restituisce un valore intero (long)a 32bit.

Sintassi:

random(min, max);

min: stabilisce ed include il valore minimo di un numero casuale da generare; è opzionale. Se il valore minimo non è indicato è 0.

max: stabilisce ma esclude il valore massimo di un numero casuale da generare.

Esempio:

```
//Imita il rotolamento di due dadi

byte Die_1;

byte Die_2;

Die_1 = random(1,7);        //Genera un numero casuale tra 1 e 6
```

Die_2 = random(1,7); *//Genera un numero casuale tra 1 e 6*

- **La funzione randomSeed()**

Questo è un altro sistema che Arduino usa per generare numeri casuali. Consiste in una sequenza o lunga lista di numeri ed è sempre la stessa. Su può cominciare generando i numeri casuali da qualsiasi punto della lista o sequenza.

Sintassi:

randomSeed(value);

value: Questo può essere qualsiasi numero intero a 16 bit (int) o 32 bit (long). È usato come un "seme" per cominciare la sequenza di numeri casuali in qualsiasi punto. Quanto più il "seme" è casuale, tanto più casuale sarà la sequenza.

SEZIONE PRATICA

8. ESEMPIO 1: conversione ADC

Ecco i primi esercizi e sono facili come sembra. Tutto quello che devi fare è leggere il valore analogico che inserisci nel potenziometro collegato al pin A0. La conversione viene eseguita ogni volta che si preme il pulsante collegato al pin 4. L'idea di questo esercizio è rinforzare le conoscenze acquisite nelle unità precedenti in modo da utilizzare una comunicazione seriale e una gamma di formati per fornire il risultato della conversione al PC.

L'aspetto davvero nuovo di questo esempio è stato evidenziato nella figura seguente. La funzione **analogread(A0)** legge e converte il segnale analogico sull'ingresso A0 nel suo equivalente in codice binario e lo salva nella variabile "AN0".

```
while(digitalRead(4));
delay(20);                  //Esperar a recibir un pulso desde D4

AN0=analogRead(A0);         //Realiza la conversión de A0 (potenciom
V=AN0*0.00488;              //Calcula la tensión equivalente

Serial.print(AN0);          //Transmite la medida en decimal
Serial.print("\t");         //Tabulación
```

Figura 10

Questo valore viene convertito in tensione moltiplicandolo per la costante, 0,0048 (la risoluzione in bit), e viene salvato nella variabile "V". Infine, tutti i risultati vengono trasmessi utilizzando la comunicazione seriale in formato decimale, binario ed esadecimale e in tensione. La figura mostra un numero di conversioni. Inizia con il potenziometro completamente a sinistra (0 V) e spostalo gradualmente fino ad arrivare completamente a destra (5 V).

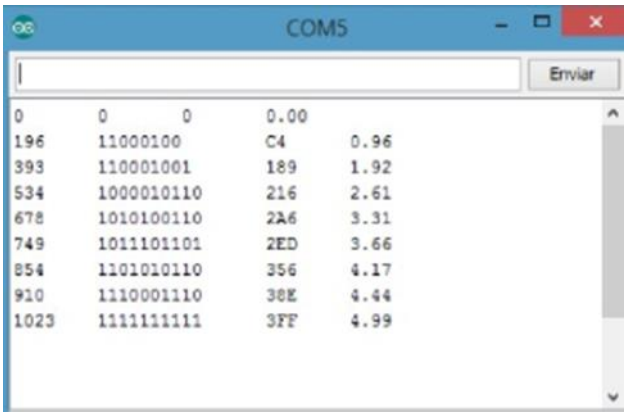

Figura 11

9. ESEMPIO 2: Soglie

Ovviamente puoi eseguire qualsiasi tipo di operazione con il valore binario di una conversione. Eseguirete due conversioni in questo esercizio: una con il potenziometro collegato all'ingresso analogico A0 e uno con il potenziometro collegato all'ingresso analogico A1. Se il risultato della conversione sull'ingresso A0 è superiore a 4 V, la spia rossa si accende. Se il risultato sull'ingresso A1 è superiore a 2,5 V, la luce bianca si accende. Se i risultati sono inferiori alle tensioni indicate, le spie LED rimangono spente.

Quando carichi il programma assicurati che la luce rossa si accenda quando girerai l'albero del potenziometro quasi completamente verso destra; accertati inoltre di ruotare l'albero del potenziometro a metà strada verso sinistra in modo che la luce bianca si accenda.

Ricorda che un potenziometro funziona come una sorta di joystick e puoi rilevare in che posizione si trova.

10. ESEMPIO 3: comparatore analogico

Sulla base dell'esercizio precedente, possiamo creare programmi che confrontano due tensioni o segnali e dire se uno è maggiore dell'altro o se sono entrambi uguali. Questo è ciò che vogliamo che tu faccia in questo esercizio.

Confrontiamo due tensioni analogiche, V1 e V2, collegate ai pin A0 e A1; entrambi provengono dai potenziometri. I LED di uscita si accendono come mostrato nella tabella di seguito.

IF...	LED
V1 = V2	Amber
V1 > V2	Red
V1 < V2	Green

11. ESEMPIO 4: controllo della luminosità

Devi avere già visto un sistema per controllare la luminosità dell'illuminazione in una stanza. Puoi illuminare o attenuare le luci e creare atmosfere diverse nella stanza usando il controllo. E questo è esattamente quello che farai in questo esercizio.

Utilizziamo la tensione analogica proveniente da uno dei potenziometri e passiamo al pin

A0 per generare un segnale PWM che regola la tensione che passa al LED bianco collegato al pin 6.

Have a look at the extract from the main body of the program (Figure 13). It's a pushover! The analogRead(A0) function reads the analog value on the A0 pin.

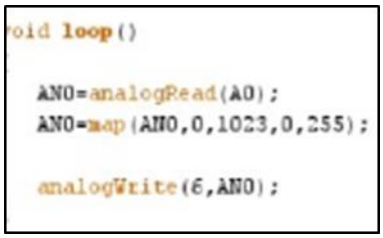

Figure 12

Come sai già, il risultato della conversione può essere compreso tra 0 e 1023. Usando la funzione **map()** puoi arrotondare quel valore ad un altro equivalente tra 0 e 255, che è il parametro che la funzione **analogWrite()** necessita per generare un segnale PWM sull'uscita 6 (il LED bianco).

Carica il programma e stupisci la tua famiglia e i tuoi amici. È possibile attenuare o illuminare la luce spostando il potenziometro da sinistra a destra o viceversa.

12. ESEMPIO 5: Timone elettrico

Useremo gli stessi principi dell'esercizio precedente. Supponiamo che tu stia navigando. Spostando il potenziometro, si sposta l'albero del servomotore che a sua volta sposta il timone della barca.

Se osservi attentamente il programma, vedrai che la funzione **map()** arrotonda il risultato della conversione a un valore compreso tra 0 e 180, il numero massimo di gradi che l'albero del servomotore può ruotare.

13. ESEMPIO 6: Fotometro

Ora lavoreremo con il sensore di luce e imiteremo un fotometro, o strumento per misurare la luce ambientale. Ogni volta che premiamo il pulsante collegato al pin 4, la tensione proveniente dal sensore di luce collegato al pin A2 viene convertita nel suo equivalente analogico.

Il programma misura il valore analogico sull'ingresso A2 in base alla luce che colpisce il sensore e quindi calcola la tensione. I risultati vengono trasmessi utilizzando la comunicazione seriale.

Puoi considerare questo esercizio come un programma sperimentale. Puoi fare un numero di misurazioni come quelle nella figura a destra. La prima misurazione eseguita è con il sensore al buio e l'ultima con la massima luminosità.

Puoi anche cercare una relazione tra queste misure e quelle fornite da uno strumento prodotto commercialmente. Sulla base di ciò che scopri, puoi creare il tuo strumento per misurare la luce in lux o in lumen.

14. ESEMPIO 7: Controllo dell'illuminazione

Questo esercizio non offre nulla di particolarmente nuovo, ma ha applicazioni pratiche nei sistemi di illuminazione pubblica, ad esempio. Non ho dubbi che sia usato per illuminare le strade della tua città.

Quando inizia a fare buio, le luci della strada si accendono. In questo esercizio, il sensore di luce collegato all'ingresso analogico A2 misura la luce ambientale. Ci sarà sicuramente un sensore simile nella tua strada; lo troverai in qualche modo fuori posto ma in una posizione sicura. Il LED collegato al pin 6 simula l'illuminazione stradale.

È interessante notare che ancora una volta è stata creata una funzione: **measurelight()**. Questa nuova funzione prende una serie di campioni di luce ambientale a intervalli regolari, quindi calcola e restituisce la media di tutti.

Questo è usato per "filtrare" il segnale analogico proveniente dal sensore di luce; questo evita variazioni molto piccole che fanno accendere e spegnere le luci della strada senza motivo.

Il programma principale chiama la nostra funzione **Measure_light()**e confronta la media con il valore minimo stabilito per accendere il LED bianco.

- **E ora è il tuo turno**

Pensa a questo esempio: è notte e le luci della strada sono accese. Cosa succede se c'è una tempesta e una striscia di fulmine colpisce il sensore? I lampioni probabilmente si spengono perché il nostro programma penserà che è giorno. Si riaccenderanno più tardi, naturalmente. Questo è un vero esempio; cose come questa accadono. Ma potremmo trovare un modo per evitarlo? Trova una soluzione che eviti questo problema e migliora il programma.

15. ESEMPIO 8: Misurazione dei riflessi

Questo esercizio è molto semplice e non offre nulla di particolarmente nuovo. L'idea è di misurare la luce riflessa rilevata dal sensore a infrarossi (IR) collegato all'ingresso analogico A3. Il risultato viene trasmesso utilizzando la comunicazione seriale.

- **E ora è il tuo turno**

Quando registri il programma, apri anche il monitor della porta seriale in modo da poter vedere le diverse misurazioni. Poiché il convertitore h una risoluzione di 10 bit, i valori saranno compresi tra 0 e 1023 (210). Ecco alcune cose da controllare:

- ✓ Se non metti niente davanti al sensore, noterai che la lettura è piuttosto piccola. La luce IR emessa si disperde, non rimbalza e il sensore rileva solo una piccola quantità di luce riflessa.
- ✓ Se si posiziona qualcosa di vivacemente colorato a una distanza di circa 10 mm dal sensore, si noterà che la lettura aumenta notevolmente.

I colori luminosi si riflettono meglio della luce IR; si girano e rimbalzano meglio verso il sensore.

- ✓ Prova lo stesso esperimento ma questa volta con un oggetto più scuro. La lettura sul sensore scende. I colori più scuri assorbono maggiormente la luce IR e il sensore riceve una quantità minore di luce riflessa.
- ✓ Puoi anche sperimentare spostando l'oggetto più vicino al sensore o più lontano. Noterai che la lettura cambia quando sposti l'oggetto più vicino al sensore o più lontano. Più l'oggetto è vicino al sensore, maggiore è la quantità di luce riflessa che riceve il sensore.

16. ESEMPIO 9: Rilevazione dei colori

Questo è un esercizio puramente sperimentale e ci sono modi per migliorarlo. Si tratta di rilevare il colore di un oggetto. Utilizza i valori sulla tabella dell'esercizio precedente come riferimento. In questo esercizio cercheremo di distinguere tra bianco e nero.

Il programma attende finché non si preme il pulsante collegato al pin 12. Quindi il sensore IR collegato all'ingresso analogico A3 esegue la misurazione. Se questo è inferiore a 300, trasmette il messaggio "NERO". Se il valore è uguale a 300 o superiore, trasmette il messaggio "BIANCO". Questi messaggi vengono trasmessi utilizzando la comunicazione seriale.

Come puoi vedere dalla figura sopra puoi creare un oggetto in bianco e nero con un semplice pezzo di cartone. Lo metti davanti al sensore ad una distanza di circa 15 mm e premi D12 per effettuare la misurazione. Il colore dell'oggetto apparirà sul monitor della porta seriale.

17. ESEMPIO 10: Sensori di temperatura

Questo esercizio è molto simile a quelli precedenti. Qui l'idea è di misurare e visualizzare la temperatura ambiente rilevata dal sensore.

La misurazione viene eseguita premendo il pulsante collegato al pin D4. Il risultato della conversione (AN4), il suo equivalente in tensione (V = AN4 * 0,0048) e la temperatura ambiente in ºC (T = AN4 * 500/1024) vengono trasmessi utilizzando la comunicazione seriale.

18. ESEMPIO 11: Condizionatori d'aria

Ecco un altro esempio con un'ovvia applicazione pratica. Simuleremo un semplice sistema di condizionamento d'aria. La figura seguente ti darà un'idea migliore.

Quando la temperatura supera un valore stabilito nella variabile "Max", la spia LED verde sul pin 9 si accende; questo per simulare che il sistema di raffreddamento è appena andato avanti. Se la temperatura ambiente scende al di sotto di un certo valore stabilito nella variabile "Min", la luce a LED rossa sul pin 11 si accende; questo è per simulare che il sistema di riscaldamento si è appena acceso. Se il sensore rileva una temperatura tra "Max" e "Min", entrambi i sistemi di raffreddamento e di riscaldamento si spengono. Si presume che questa sia la zona di temperatura confort.

19. ESEMPIO 12: Un segnale PWM

Questo è un esercizio molto semplice e una buona opportunità di usare la funzione analogWrite(). L'idea è di generare un segnale PWM sul pin 6 che è connesso al LED bianco sulla tavola di sviluppo.

Bisogna dichiarare la percentuale del ciclo duty o il voltaggio richiesto sulla variabile "Power". La variabile "Ciclo" calcola il valore dell'equivalente ciclo duty che sta tra 0 e 255.

La funzione setup() è vuota ma è obbligatorio includerla.

Il corpo maggiore del programma è nella funzione loop(). Genera un segnale PWM con il ciclo duty, richiesto sul pin 6 usando analogWrite().

cco un dettaglio importante: guardare bene la funzione while(1). Questo loop continua all'infinito finchè l'espressione dentro la parentesi () non diventa falsa. Qualcosa deve cambiare la variabile indicata altrimenti il loop non si interromperà mai. Formerà un loop infinito. Questo significa che il controller non eseguirà nessun altra funzione.

Il segnale PWM è ancora là sul pin 6. Perchè? Perchè il controllo dei circuiti elettronici interni genera segnali PWM sui pin 3, 5, 6, 9, 10 e 11. Una volta che la funzione analogWrite() comincia a generarli, questi continuano all'infinito fino ad un ordine contrario. Ci riferiamo a questi come segnali generati dall'hardware.

20. ESEMPIO 13: Effetti ottici

Questo esempio mostrerà chiaramente l'effetto ottico creato con la luminosità di una luce LED attraverso un segnale PWM con un ciclo duty che gradatamente aumenta e diminuisce.

Il primo loop for() aumenta il ciclo duty da 0 a 255 in 5 step. La luminosità del LED aumenterà da un minimo a un massimo.

Anche il secondo loop for() diminuisce il ciclo duty da 255 a 0 in 5 step. La luminosità della luce LED diminuirà da un massimo a un minimo.

21. ESEMPIO 14: Manuale di regolazione

Questo è veramente un eccellente esercizio pratico. Bisogna regolare manualmente la luminosità della luca Bianca LED connessa al pin 6. Si dovrà rendere la luce LED più luminosa aumentando il ciclo duty del segnale PWM con multipli di 5 usando l'interruttore a pulsante 4. Si potrà fare dissolvenza usando l'interruttore a pulsante 7 per ridurre la lunghezza del ciclo duty, ancora una volta con multipli di 5.

Ricordiamo che gli stessi principi di regolazione si possono applicare a un motore elettrico, per esempio; lo si potrà controllare variando la velocità (utilizzando lo stesso principio).

22. ESEMPIO 15: Luci casuali

Questo è un esercizio nuovo. Potrebbe aiutarvi a decorare l'albero di Natale, la vetrina, la camera da letto, il giardino o qualsiasi cosa voi vogliate. Andremo a generare 4 segnali PWM in simultanea sui pin 6, 9, 10 e 11.

The random() function we studied in this unit will randomize the length of each PWM signal's duty cycle. This in turn randomizes the voltage received by the Led lights.

La funzione random () che abbiamo studiato in questa unità renderà casuale la lunghezza di ciascun ciclo duty del segnale PWM. Questo porrà casualità nel voltaggio ricevuto dalle luci LED.

RIFERIMENTI

PUBBLICAZIONI

[1]. **EXPLORING ARDUINO**, Jeremy Blum, Ed.Willey
[2]. Practical ARDUINO, Jonathan Oxer & Hugh Blemings, Ed.Technology in action
[3]. Programing Arduino, Next Steps, Simon Monk
[4]. Sensores y actuadores, Aplicaciones con ARDUINO, Leonel G.Corona Ramirez, Griselda S. Abarca Jiménez, Jesús Mares Carreño, Ed.Patria
[5]. ARDUINO: Curso práctico de formación, Oscar Torrente Artero, Ed.RC libros
[6]. 30 ARDUINO PROJECTS FOR THE EVIL GENIUS, Simon Monk, Ed. TAB
[7]. Beginning C for ARDUINO, Jack Purdum, Ph.D., Ed.Technology in action
[8]. ARDUINO programing notebook, Brian W.Evans

SITI WEB

[1]. https://www.arduino.cc/
[2]. https://www.prometec.net/
[3]. http://blog.bricogeek.com/
[4]. https://aprendiendoarduino.wordpress.com/
[5]. https://www.sparkfun.com