Tecnologie per IoT – a.a. 2022/23

Esercitazione di Laboratorio 1

PARTE 1

Introduzione all'ambiente Arduino e primi semplici sketch

1.1) Pilotaggio di LED

Si scriva uno sketch in grado di pilotare due LED per farli lampeggiare con periodi indipendenti fra di loro. Lo sketch deve rispettare le seguenti specifiche:

- I pin a cui sono connessi i due led, e i (semi-)periodi di lampeggiamento devono essere definiti tramite costanti
- Per pilotare il primo LED, si utilizzi la funzione `delay()` all'interno della funzione `loop()`.
- Per pilotare il secondo LED, si utilizzi una Interrupt Service Routine collegata al Timer1 del MCU (mediante la libreria `MBED RPI PICO TimerInterrupt` oppure la libreria Scheduler).

Come prova, si utilizzino periodi di lampeggiamento non multipli tra di loro (ad esempio, 3s e 7s) e si colleghino i due LED ai pin D2 e D3 della board.

1.2) Comunicazione tramite Porta Seriale

Si modifichi lo sketch precedente in modo da trasmettere lo stato attuale dei LED tramite seriale (se richiesto). Lo sketch deve rispettare le seguenti specifiche:

- All'avvio, deve essere configurata una comunicazione Seriale con il PC con BaudRate = 9600 bit/s, e deve essere inviato un messaggio di "benvenuto" al PC dopo che la connessione è avvenuta
- In ogni iterazione della funzione `loop()`, lo sketch deve controllare la ricezione di un carattere tramite seriale. Se tale carattere corrisponde all'ASCII corrispondente al carattere 'R' (led rosso) o 'L' (led verde), lo sketch deve inviare un messaggio al PC contenente lo

stato del LED corrispondente. Se invece il carattere ricevuto non corrisponde ad uno dei due precedenti, deve essere stampato un messaggio di errore.

Si verifichi il corretto funzionamento del programma tramite il Serial Monitor della IDE Arduino. Nella fase di debug si tenga conto del fatto che la stampa tramite sulla seriale avviene in modo sincrono rispetto all'accensione/spegnimento di uno dei due LED (quello gestito nella funzione `loop()`), perciò il valore verrà inviato al PC sempre subito prima o subito dopo un'accensione/spegnimento.
Si faccia inoltre attenzione ad utilizzare il qualificatore `volatile` per le variabili utilizzate sia dalla ISR che dalla funzione `loop()` .

Extra: si provi ad utilizzare la libreria Scheduler vista all'esercizio precedente per rendere la stampa dello stato asincrona rispetto al loop principale.

Figura 1: Esempio di output su porta seriale per l'esercizio 1.2

1.3) Identificazione della presenza tramite Sensore PIR

Si scriva uno sketch in grado di contare il numero di persone che passano di fronte alla Arduino tramite sensore di movimento basato su tecnologia Passive InfraRed (PIR). Lo sketch deve rispettare le seguenti specifiche:

- L'output digitale del sensore PIR deve essere collegato ad un input digitale della Arduino. Lo sketch deve reagire ai cambi di valore sul pin tramite una ISR attiva su entrambi i fronti di salita e discesa del pin.

- La ISR deve leggere il valore del sensore e riprodurlo su un LED. Cioè il LED deve essere acceso quando il sensore PIR produce un valore alto, e spento altrimenti.

- Inoltre, la ISR deve salvare in una variabile il numero totale di eventi identificati dal sensore PIR. Per evento si intende un nuovo movimento registrato dal sensore.

- La funzione `loop()` deve inviare ogni 30s al PC un messaggio contenente il numero totale di eventi registrati.

Si verifichi il corretto funzionamento del programma tramite il serial monitor della IDE Arduino.

Figura 2: Esempio di output su porta seriale per l'esercizio 1.3

1.4) Controllo di un motore (ventola) tramite PWM

Si scriva uno sketch in grado di controllare la velocità di rotazione di un motore a corrente continua tramite Pulse-Width Modulation (PWM). Lo sketch deve rispettare le seguenti specifiche:

- Il pin di controllo della ventola deve essere collegato ad un output della Arduino che supporti il PWM.

- Al setup, lo sketch deve impostare una velocità di rotazione pari a 0, scrivendo il valore 0 (duty cycle = 0%) sul pin di controllo.

- La funzione `loop()` deve processare l'input della seriale, incrementando la velocità ogni volta che viene ricevuto il carattere '+' e decrementandola ogni volta che viene ricevuto il carattere '-'. Gli altri caratteri vanno segnalati come errori.

- Lo sketch deve predevere un numero fisso di "step" di velocità (ad esempio 10) ciascuno corrispondente ad un duty cycle crescente, fino ad un massimo pari al 100%,

corrispondente al valore 255 nel registro PWM. Ogni ricezione del carattere '+' o '-' incrementa o decrementa la velocità di uno step.

- Quando si raggiunge il duty cycle massimo/minimo, lo sketch deve rispondere ad un ulteriore ricezione del carattere '+'/'-' segnalando che la velocità massima/minima è già stata raggiunta.

Durante il test di questo sketch, si tenga conto che è possibile che per valori di duty cycle bassi (a seconda del numero di step scelto) il motore non produca una forza necessaria a muovere le pale della ventola se essa è inizialmente ferma (ma sufficiente a mantenerle in moto se lo erano già in precedenza). Questo è normale, e non va considerato come un errore del programma.

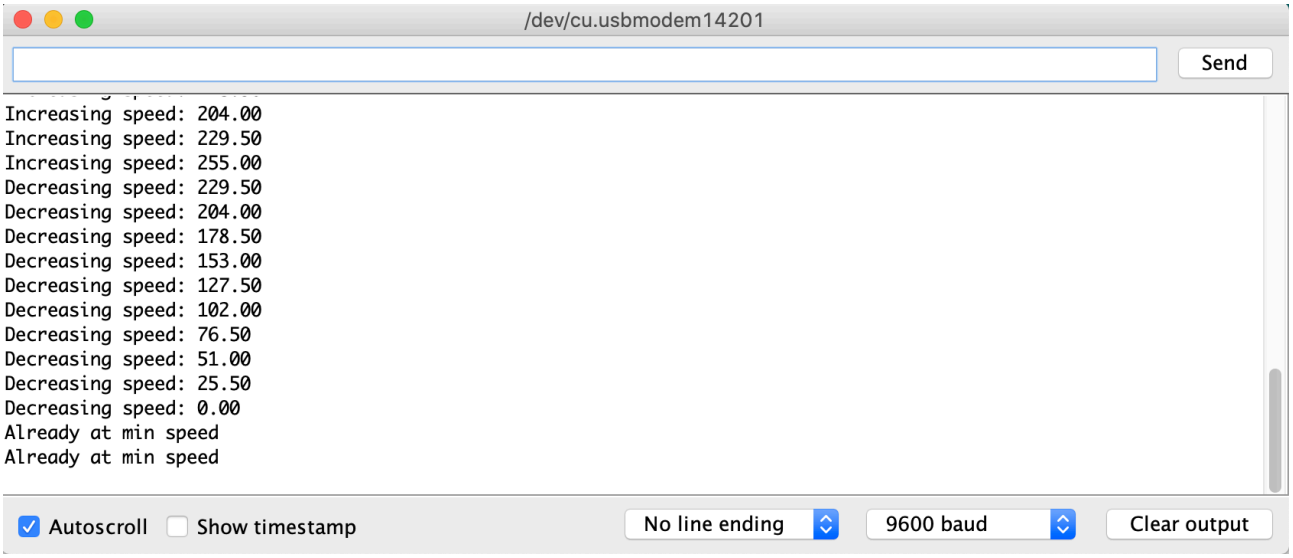

Figura 3: Esempio di output su porta seriale per l'esercizio 1.4

1.5) Lettura di valori analogici (temperatura)

Si scriva uno sketch in grado di monitorare periodicamente la temperatura dell'ambiente e inviare il valore al PC tramite porta seriale. Lo sketch deve rispettare le seguenti specifiche:

- Il pin di segnale del sensore di temperatura va collegato ad uno degli ingressi analogici della Arduino (ad esempio il pin A0).

- Il valore di tensione di tale pin va letto periodicamente (ad esempio ogni 10s), e convertito in una temperatura mediante le opportune funzioni riportate nel datasheet del sensore

- Il valore letto va inviato al PC mediante seriale.

Per controllare il funzionamento dello sketch, si può scaldare artificialmente il sensore (senza danneggiarlo!) ad esempio soffiandoci sopra.

Figura 4: Esempio di output su porta seriale per l'esercizio 1.5

1.6) Comunicazione con uno smart actuator (display LCD) tramite protocollo I2C

Si modifichi lo sketch precedente in modo da visualizzare le informazioni di temperatura su display LCD anziché trasmetterle al PC. Si utilizzi la libreria `LiquidCrystal_PCF8574`, scaricabile dal Library Manager della IDE Arduino per interfacciarsi con il display. Lo sketch deve rispettare le stesse specifiche del precedente, con l'unica differenza che le informazioni vanno visualizzate sul display. Si esplori il codice sorgente della libreria fornita per individuare il modo più efficiente per aggiornare la temperatura sul display, minimizzando la quantità di dati inviata sul bus I2C.

Figura 5: Esempio di output su LCD per l'esercizio 1.6