Progetto Demetra
per sistemi di elettronica digitale

Alessandro Lemme
mat. 723923

October 24, 2024

1 Obiettivi del progetto

Programmare un circuito che tramite ESP32 DevKit e sensori sia in grado di:

- (ESP32) connettersi al wifi
- (DS1307) mantenere l'ora anche se disconnesso dalla rete o rimasto senza corrente
- che possa rilevare ed inviare dati ad un'api di un server in lan raccolti tramite:
  - (FC-28) umidità del suolo
  - (termistore) temperatura ambientale
  - (fotoresistenza) intensità della luce
- eventualmente connettersi ad un server esterno alla lan e recuperare l'ora usando i timestamp UNIX, per poi impostarla sul modulo RTC

2 Struttura del codice

Il codice che viene eseguito dal microcontrollore è diviso in due sezioni; una che viene eseguita all'accensione della scheda (definito dalla funzione `void setup()`) ed una periodica (`void loop()`).
All'accensione della scheda si ha la creazione dei servizi di base

2.1 Setup

Connessione al wifi `setupWifi()` Usando la libreria `WiFi.h` si imposta la modalità del wifi come client (`WiFi.mode(WIFI_STA)`) che si connette ad un router (al contrario di `WIFI_AP` che lo rende un access point o di `WIFI_AP_STA` che lo rende un ripetitore). Si tenta quindi per un numero di volte pari alla costante `ALLOWED_CONNECTION_ATTEMPTS` (1 in questo caso) a connettersi al wifi. Lo stato del wifi (ottenibile con il getter `WiFi.status()`) può essere

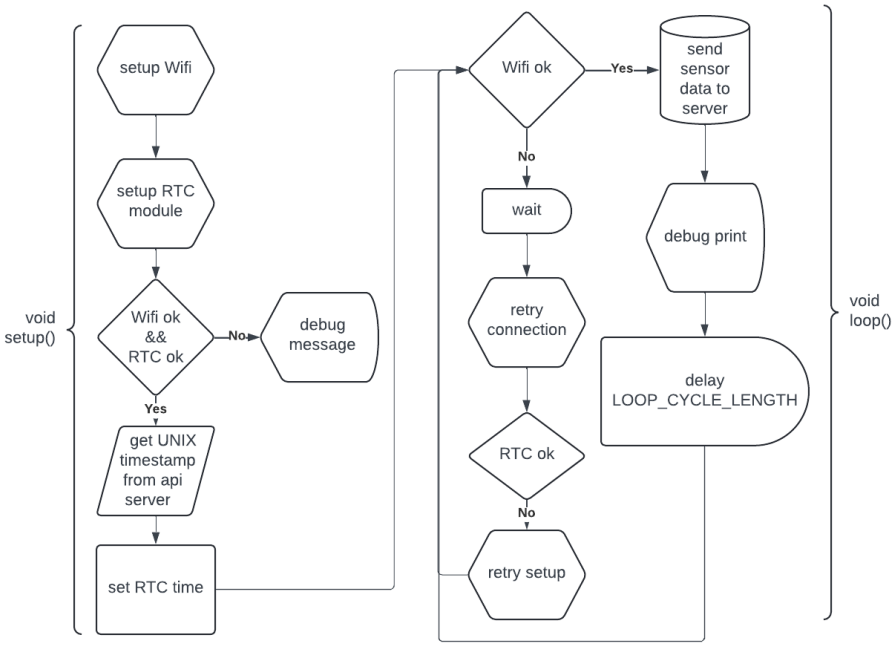


Figure 1: Flow chart di demetra.ino

- `WL_CONNECTED` se si è connesso alla SSID correttamente
- `WL_NO_SHIELD` se lo shield non è connesso correttamente alla scheda (difficile in questo caso)
- `WL_IDLE_STATUS` temporaneo, mentre si sta connettendo
- `WL_CONNECT_FAILED` se fallisce la connessione
- `WL_NO_SSID_AVAIL` se non ci sono SSID rilevabili
- `WL_SCAN_COMPLETED` al termine dello scan per delle reti
- `WL_CONNECTION_LOST` se cade la connessione precedentemente stabilita con successo
- `WL_DISCONNECTED` alla disconnessione da una rete

Nel caso la connessione abbia successo si stamperà sul monitor seriale una lista di dati; l'ip locale, la SSID a cui ci si è connessi e la qualità della connessione. Quest'ultima, la received signal strength information, il cui valore può essere letto tramite il metodo `WiFi.RSSI()` è un'unità di misura in decibell-milliwatts

(dBm o dB_{mW}), espressa usando una scala decibell rispetto ad un milliwatt; in formule, sia la potenza P se misurata in mW ed x se misurata in dBm, allora

$$x = 10 \log \frac{P}{10^{-3}W}$$
$$P = 10^{-3} \cdot 10^{\frac{x}{10}}$$

Inizializzazione del modulo RTC `RTC_DS1307::begin`() Il modulo DS1307 è un orologio/calendario digitale a rappresentazione FBCD che usa un oscillatore a cristallo da $32.768kHz$ ed il protocollo I^2C per la comunicazione.
FBCD (fully coded binary decimal) significa che invece di usare un unico intero per rappresentare un dato momento - come nei timestamp UNIX, dove l'intero a 32bit indica i secondi trascorsi dall'1 gennaio 1970 - usa 4bit per rappresentare una singola cifra decimale. Questo modulo in particolare monta una nvSRAM da 54bit, utili quindi per rappresentare 14 cifre; 1 per l'indirizzo da cui iniziare a leggere/scrivere (`0x00` secondi, `0x02` minuti, ...), 2 ciascuno per secondi, minuti ed ore, 1 per il giorno della settimana e poi di nuovo 2 ciascuno per giorno, mese ed anno.
Il protocollo I^2C (inter integrated ciruit) è un sistema di comunicazione seriale bifilare usato tra circuiti integrati. In questo caso ci sono un solo master ed un solo slave, e si può comunicare con l'orologio usando la libreria `Wire.h`. Personalmente ho preferito usare la `RTCLib.h`, ma nel caso impostare la data sarebbe semplice, qui sotto un esempio di codice

```
#include <Wire.h>

#define DS1307_ADDRESS 0x68

byte decToBcd(byte val) {
    return ( (val / 10 * 16) + (val % 10) );
}

void setDS1307Time(
    byte secondi,
    byte minuti,
    byte ore,
    byte giorno,
    byte data,
    byte mese,
    byte anno
) {
    Wire.beginTransmission(DS1307_ADDRESS);
    Wire.write(0);                  // Punto di partenza dell'array di registri
    Wire.write(decToBcd(secondi));  // Secondi (in BCD)
    Wire.write(decToBcd(minuti));   // Minuti (in BCD)
```

```
    Wire.write(decToBcd(ore));      // Ore (in BCD, formato 24 ore)
    Wire.write(decToBcd(giorno));   // Giorno della settimana (1=lunedì, 7=domenica)
    Wire.write(decToBcd(data));     // Giorno del mese (data)
    Wire.write(decToBcd(mese));     // Mese
    Wire.write(decToBcd(anno));     // Anno (0 = 2000, 21 = 2021)
    Wire.endTransmission();
}
```

L'oscillatore a cristallo è un componente composto da un cristallo di quarzo (biossido di silicio - SiO_2), materiale piezoelettrico che in seguito a deformazione meccanica genera una tensione periodica (onda quadra) in questo caso convenientemente da $32.768kHz$ poichè divide un secondo in esattamente 2^{15} parti. La deformazione del cristallo avviene immergendolo in un campo elettrico ottenuto applicando una tensione ad un elettrodo per un dato periodo, al che il cristallo nel tornare alla sua forma iniziale inizierà a vibrare emettendo il segnale.

Connessione all'api del server per ottenere la data `getServerTime`() e `RTC_DS1307::adjust(DateTime &dt)` Se l'orologio viene inizializzato correttamente, e la connessione alla rete ha successo, allora utilizzando la libreria `HTTPClient.h` l'ESP32 creerà un oggetto `HTTPClient` e tramite il metodo `HTTPClient::begin` farà una richiesta `GET` ad un server esterno definito nella variabile `timeApiUri`, se avrà successo si avrà una stringa contenente il timestamp UNIX, che una volta convertito in `int`, e successivamente in `DateTime` potrà essere impostata sul modulo RTC utilizzando il metodo I^2C di prima (anche se in `RTC_DS1307.cpp` si usa `Adafruit_I2CDevice` definita nella libreria omonima al posto di `Wire.h` come nel mio esempio di prima).

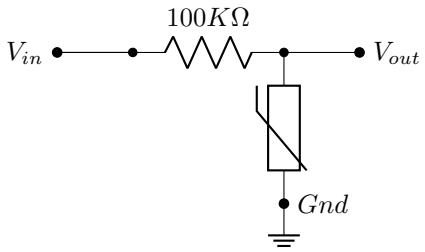
Connessione all'api per il salvataggio dei dati registrati `setupServerComunication`() Come ultima cosa del `setup()` se connesso alla lan il controllore tenterà di connettersi all'api HTTP di un server locale stabilendo una connessione TCP.

2.2 Loop

Terminato il setup si avrà il controllo periodico del funzionamento dell'orologio e dell'api, e nel caso di errore si tenterà di riconnettersi. Si avrà poi la raccolta dei dati da parte dei sensori quali;

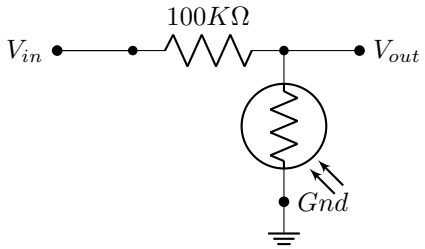
Temperatura con termistore Il termistore è un resistore la cui resistenza varia al variare della temperatura. In questo progetto è stato fatto un partitore di tensione con un resistore da $100K\Omega$ e calcolata la sua resistenza R_t con la forumula

$$R_t = \frac{v_{out} \cdot R}{v_{in} - v_{out}}$$



La tensione viene misurata nel codice chiamando la funzione `analogRead(uint8_t pin)` che ritornerà la tensione incognita v_{out} su quel pin.

Intensità della luce con fotoresistore Analogamente al termistore la resistenza del fotoresistore è inversamente proporzionale alla quantità di luce che lo colpisce, tanto che la funzione che ricava la resistenza dei componenti è la stessa - `get_voltage_divider_unknown_resistance(...)`.



Umidità del suolo Il sensore FC-28 è misuratore di umidità del terreno resistivo; tanto più il terreno è secco tanto maggiore sarà la sua resistenza. E' composto da due componenti;

- Forche, da inserire nel terreno. Volendo si potrebbe usare anche un paio di cavi spelati

- PCB con potenziometro ed LM393, quest'ultimo compara due tensioni in ingresso e rende un'uscita digitale basata su quale tensione è maggiore.

Di fatto segue lo stesso principio dei primi due sensori, ma ha l'extra dell' LM393. Al contrario un sensore basato su capacità è composto sempre da due elettrodi ma invece che misurare la tensione ai capi, con un NE555 o un TL555C si emette un'onda quadra; i due elettrodi saranno separati dal suolo che agirà da dielettrico, e tanto più sarà secco quanto più la sua capacità sarà minore.

Invio dei dati al server e controlli periodici Tramite la connessione TCP di prima, si inviano i vari dati al server (nominato Ade). Dopo di che si attenderanno `LOOP_CYCLE_LENGTH` millisecondi. Ogni esecuzione del loop comporta anche

un controllo sulla connessione alla rete, al server e al funzionamento dell'orologio. Ogni `CYCLES_INTER_WIF_CONNECTION_ATTEMPT` cicli senza connessione si riterà la connessione. Uguale per la connessione all'api ed all'orologio.

3 Extra: perché questi nomi?

Demetra era la dea greca della natura e dei raccolti. Questo è il nome del controllore che tiene controllate le piante. Ade era il dio greco dei morti, ma a me piace di più pensarlo come quello della memoria, dell'esperienza passata che non è più presente in un dato luogo; quando qualcosa non è più viene salvato da qualche altra parte perché non vada perso. Persefone era la figlia di Demetra e moglie di Ade, secondo il mito durante trascorreva i mesi autunnali ed invernali con suo marito, per poi essere restituita alla madre durante quelli primaverili ed estivi, per poi ripetere il ciclo. Questa figura mi sembrava un'analogia perfetta per gli scambi tra il server ed il controllore.