Progetto a Sorpresa

Corsi Arduino 2018

Lorenzo Prosseda

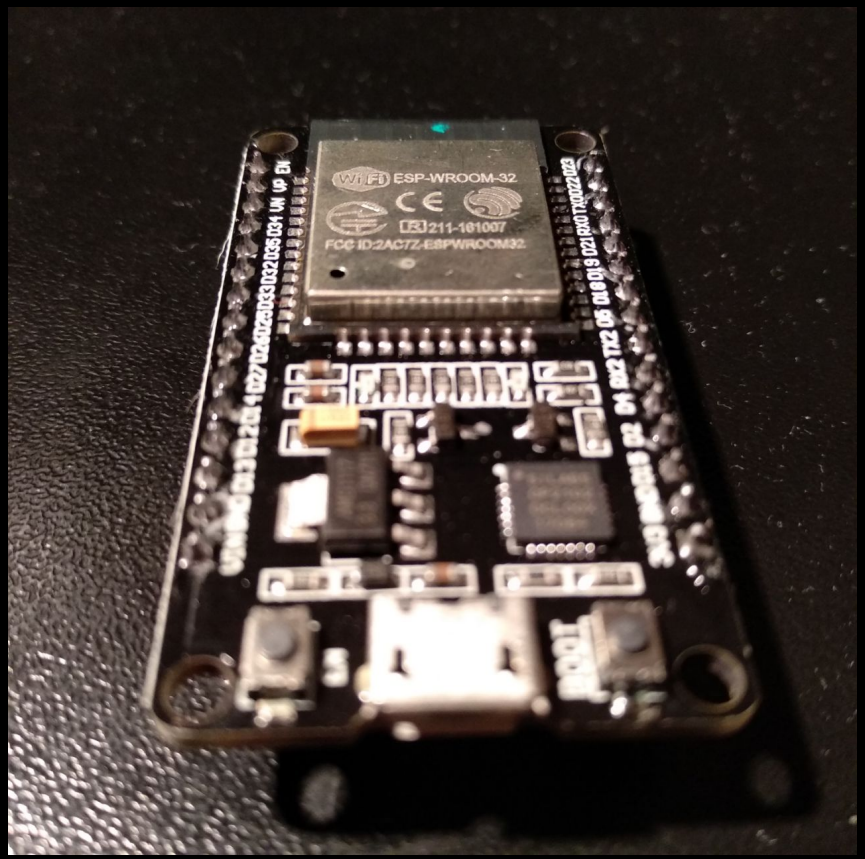
lerokamut@gmail.com

25 ottobre 2018

POLITECNICO OPEN
unix LABS

Non tutti gli MCU si chiamano Arduino

WiFi, Bluetooth, dual core a 32 bit, tanti MHz, tanta memoria flash, tanti GPIO, e tanta RAM...



Non tutti gli MCU si chiamano Arduino

Vogliamo realizzare un progetto secondo le seguenti specifiche

- connettere l'ESP32 a Internet via WiFi
- interfacciare l'ESP32 con Telegram tramite API
- usare OpenWeatherMap per ottenere informazioni sul meteo in formato JSON
- presentare informazioni meteo all'utente del Bot Telegram

In pratica...

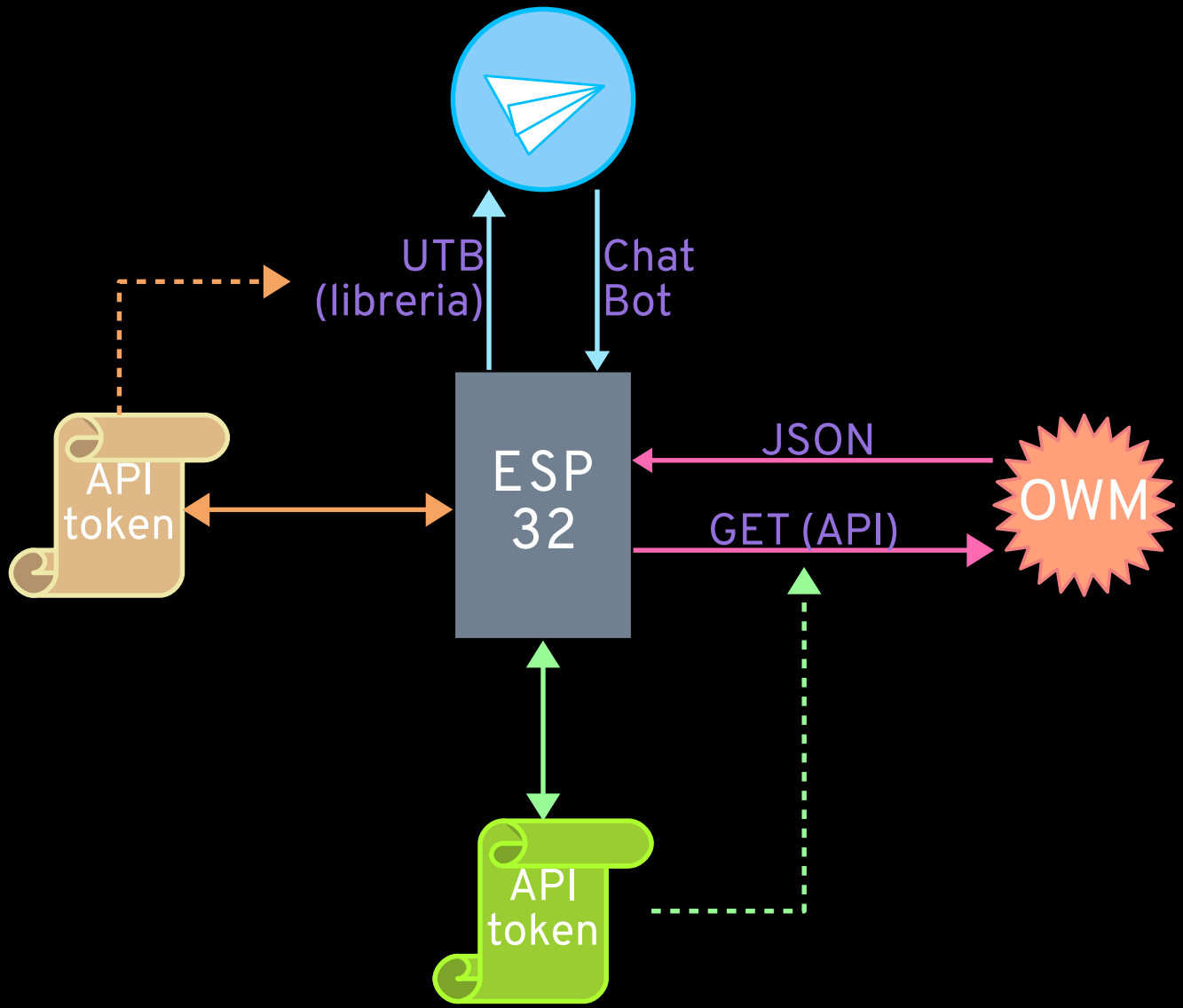
Questo progetto unirà teoria e pratica:

«Theory is when you know everything but nothing works.
Practice is when everything works but no one knows why.
In our lab, theory and practice are combined:
nothing works and no one knows why»

(da WeKnowMemes)

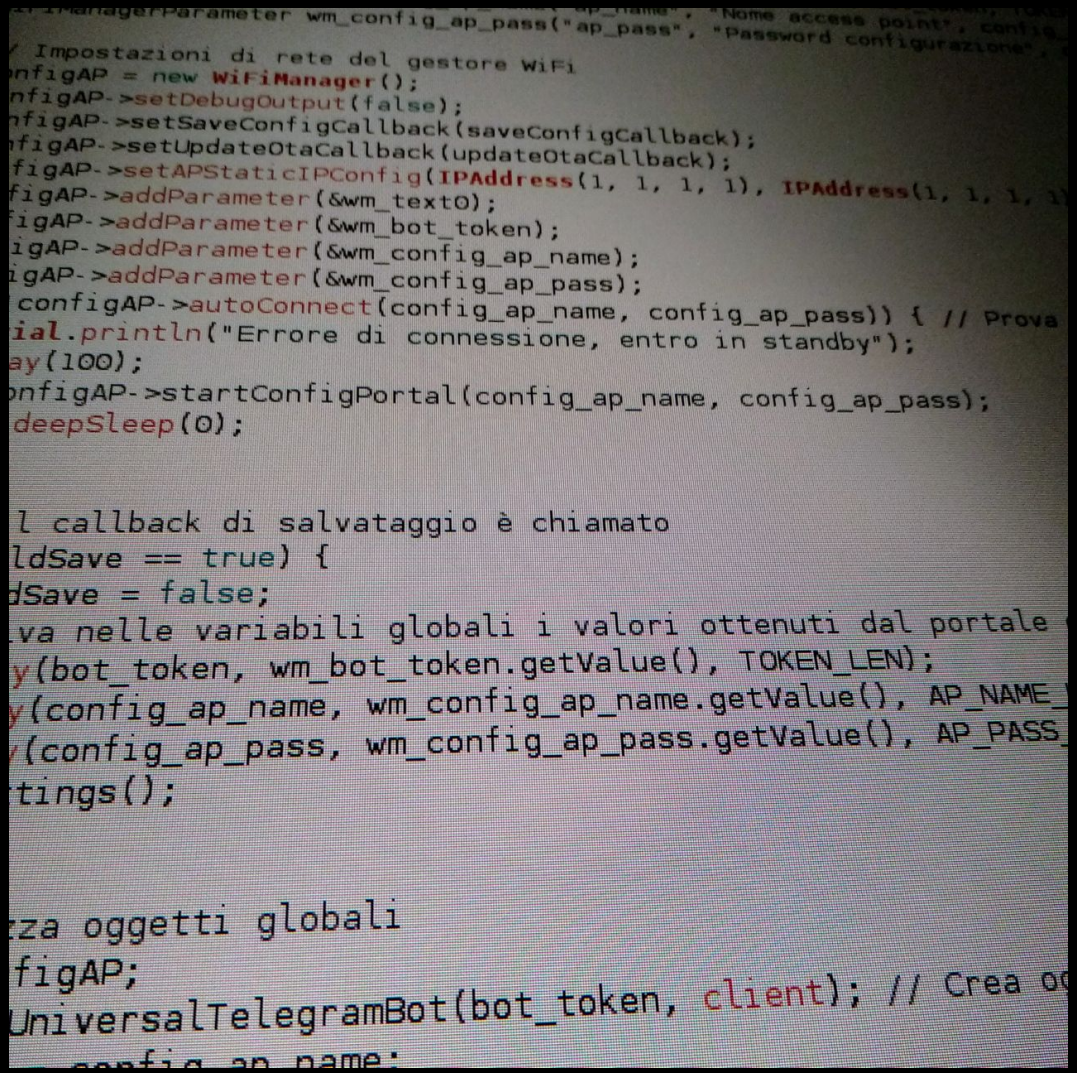
In pratica...

Il sistema da progettare potrebbe funzionare nel modo seguente:



In pratica...

... questo come tutti i progetti si realizza con le mani!



(scrivendo codice non troppo rotto)

Installare un SDK dal gestore schede

Arduino IDE contiene informazioni su come compilare e caricare il codice sulla scheda Arduino, ma è possibile istruire l'IDE sull'esistenza di altre piattaforme compatibili tramite il Gestore Schede:

- aprire le impostazioni, dal menu *File* o tramite la scorciatoia `CTRL + ,`
- incollare l'URL dell'SDK Espressif[1] per ESP32 nella casella «URL aggiuntive per il Gestore schede»
- salvare le impostazioni e navigare in *Strumenti* ▶ *Scheda: Arduino UNO* ▶ *Gestore Schede...*;
- cercare e installare la scheda «esp32»
- navigare in *Strumenti* ▶ *Scheda: Arduino UNO* e selezionare «NodeMCU - 32S»

[1] https://dl.espressif.com/dl/package_esp32_index.json (GitHub)

Installare le librerie necessarie

Arduino IDE contiene delle librerie preinstallate per gestire molte funzionalità della scheda Arduino, ma è possibile installarne delle altre dal Gestore Librerie:

- Navigare in *Sketch* ▶ *#include libreria* ▶ *Gestore librerie...* oppure usare la scorciatoia `CTRL + SHIFT + I`;
- Cercare e installare le seguenti librerie:
  - **Arduino Json** (versione 5.13.2)
  - **UniversalTelegramBot** (versione 1.1.0)

Ultimi magheggi con l'IDE

Arduino IDE è quasi pronto per gestire l'ESP32:

- Installiamo il modulo Python pyserial tramite pip:
  `sudo apt install python-pip`
  `pip install --user serial`
- Impostiamo l'utente nel gruppo giusto e installiamo il link all'IDE:
  nella cartella di Arduino IDE eseguire
  `./arduino-linux-setup.sh $USER`
- Per sicurezza, installiamo anche Python3 (se già non fosse presente):
  `sudo apt install python3`

Arduino JSON

Il JSON[2] è un formato leggero per scambiare dati legati a oggetti:

- Tramite la libreria ArduinoJson[3] è possibile «spacchettare» correttamente un oggetto in JSON
- Sul sito web di questa libreria è presente JsonAssistant, che genera automaticamente il codice C per gestire la ricezione e l'interpretazione di una stringa JSON

- Questi due strumenti permettono di lavorare con stringhe JSON:
- JSON Test permette di ricevere una stringa in JSON effettuando una richiesta GET senza TSL verso il sito
- JSON Editor Online permette di navigare e modificare comodamente una stringa JSON

[2] Introducing JSON
[3] ArduinoJson website

Universal Telegram Bot

Per collegare il microcontrollore all'API Telegram abbiamo bisogno di

- creare un Bot su Telegram, scrivendo a *BotFather* e copiando da parte il token API che sarà associato al bot
- inizializzare un client WiFi che usi TLS (HTTPS)
- invocare nel codice:
  `UniversalTelegramBot bot(BOTtoken, client);`

Esempi e informazioni significative si trovano sulla repository GitHub dell'autore della libreria (Link);

La pagina per l'API di Telegram per i Bot è situata al seguente Link

Noi vogliamo UaiFai

L'ESP32 comunica anche in Bluetooth 4.2, ma noi vogliamo il WiFi; inoltre visto che siamo bimbi diligenti vogliamo TLS (è richiesto per usare le API di Telegram)

- usare la libreria WiFiClientSecure, in particolare l'esempio omonimo per avere un codice di partenza
- ottenere il certificato della root CA per il sito a cui vogliamo far accedere l'ESP32, salvandolo in *ASCII Base 64*
- tramite questo script Python ottenere il codice per dichiarare una stringa contenente il certificato

Noi vogliamo UaiFai

Per usare il Bot Telegram sarà necessario inizializzare un client WiFi TLS: usare l'esempio EchoBot per avere un codice dal quale partire:

- si trova in *File* ▶ *Esempi* ▶ *UniversalTelegramBot* ▶ *ESP32* ▶ *EchoBot*
- sono necessari solo API token di Telegram, e credenziali di accesso per connettere l'ESP32 a un access point WiFi

Noi vogliamo UaiFai

Per usare il Bot Telegram sarà necessario inizializzare un client WiFi TLS: usare l'esempio EchoBot per avere un codice dal quale partire:

- si trova in *File* ▶ *Esempi* ▶ *UniversalTelegramBot* ▶ *ESP32* ▶ *EchoBot*
- sono necessari solo API token di Telegram, e credenziali di accesso per connettere l'ESP32 a un access point WiFi

Siamo nel 21esimo secolo: che ognuno si connetta tramite *hot spot* con lo smartphone!

Keep calm and parse JSON

Colleghiamoci al sito di OpenWeatherMap e creiamo un account per accedere all'API:

- effettuare una richiesta a Link e otteniamo il JSON per Milano
- usare il certificato rilasciato dal sito a seguito della righiesta GET nello sketch di esempio WiFiClientSecure
- cambiare le variabili `SSID`, `password` e `server` alle righe 11, 12 e 14 (`server = "api.openweathermap.org"`)

Keep calm and parse JSON

Impostiamo i seguenti parametri della richiesta HTTP nello sketch

- `GET Link HTTP/1.0` *(riga 82)*
- `Host: api.openweathermap.org` *(riga 83)*

Dopo aver sostituito CHIAVE_API con la propria chiave ottenuta da OpenWeatherMap, caricate lo sketch e nel monitor seriale otterrete la stringa JSON con il meteo attuale!

- tramite JsonAssistant otteniamo il codice per inizializzare il buffer e fare il *parsing* della stringa

Keep calm and parse JSON

Adesso un po' di code-fu:

- includere `ArduinoJson.h` nello sketch
- inserire alla riga 97 il codice ottenuto tramite JsonAssistant (al posto del ciclo while che stampa caratteri)
- inserire il seguente ciclo subito dopo la dichiarazione di `const size_t bufferSize`:

```
char json[bufferSize] = {0};
for (int i = 0; client.available() && i < bufferSize; i++)
{
      json[i] = client.read();
}
```

- rimuovere la dichiarazione di `const char* json`

Keep calm and parse JSON

Possiamo stampare tramite `Serial.print(VARIABILE)` uno degli elementi «spacchettati» dalla stringa JSON letta, come

```
Serial.print("Città:  "); Serial.println(name);
Serial.print("Condizioni:  ");
Serial.println(weather0_description);
```

Telegram API

Una volta creato un bot telegram e ottenuta una chiave API, possiamo aggiungere allo sketch per il parsing JSON: il seguente codice:

- una macro che includa la libreria UniversalTelegramBot e una col token API del proprio Bot (ottenuto da BotFather)

```
#include <UniversalTelegramBot.h>
#define BOTtoken "XXXXXXXXXXXXXXXXXXXXXXXXXXX"
```

- la dichiarazione di un oggetto Bot come

```
UniversalTelegramBot bot(BOTtoken, client);
```

- le seguenti variabili globali per gestire alcuni parametri del Bot

```
int Bot_mtbs = 1000;
long Bot_lasttime;
```

Telegram API

Processiamo i messaggi del Bot nel *loop()* col seguente ciclo

```
if (millis() > Bot_lasttime + Bot_mtbs) {
   int numNewMessages = bot.getUpdates(bot.last_message_received + 1);
   while (numNewMessages) {
      for (int i = 0; i < numNewMessages; i++) {
         String chat_id = bot.messages[i].chat_id;
         String text = bot.messages[i].text;
         String from_name = bot.messages[i].from_name;
         if (from_name == "") from_name = "Sconosciuto";
         Serial.print("Messaggio da "); Serial.print(from_name);
         if (text == "/meteo") {
            Serial.println("/meteo");
            String risposta = "Città " + (String)nome_citta +
"\nCondizioni:  " + (String)condizioni;
            bot.sendMessage(chat_id, risposta);
         }
      }
   numNewMessages = bot.getUpdates(bot.last_message_received + 1); }
   Bot_lasttime = millis();
}
```

Telegram API

Ciliegina sulla torta: bisogna aggiungere alla fine del certificato che già abbiamo per OpenWeatherMap, le stringhe per quello ottenuto dal sito di Telegram - oppure non potremo scaricare nessun mesaggio dalla chat del Bot!

Abbiate cura di approfondire

Il codice mostrato nelle slide e i suggerimenti sono da accettare con le dovute precauzioni

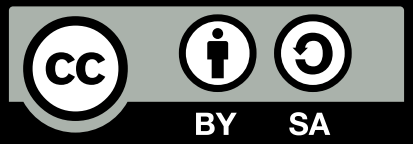
- il codice proposto è scritto per cercare di essere breve, NON sicuro
- anche se accennato, seguire alla lettera le slide non porta a scrivere uno sketch funzionante: è necessario usare le celluline grigie
- la piattaforma ESP32 è ancora *quasi* documentata per l'uso con Arduino IDE
- TUTTI i link in queste slide aiutano nel completare il progetto

Fonti

- ESP32 Datasheet
- ESP32 Technical Reference
- ESP32 FAQ
- Andreas Spiess (YouTube)

Fine

Grazie per l'attenzione!



`http://www.poul.org`