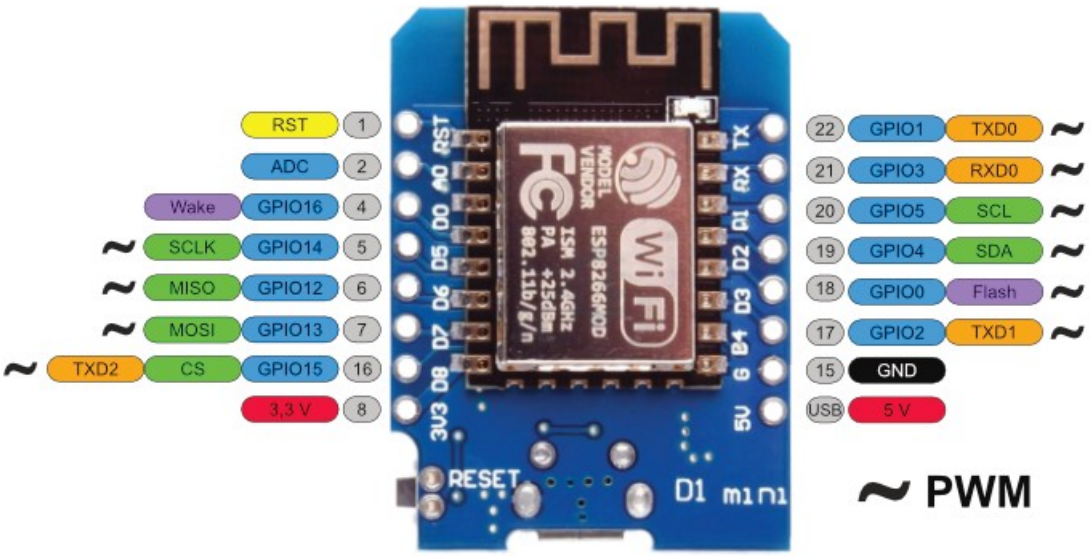
WEMOS D1 MINI E TELEGRAM

La scheda Wemos D1mini è una piccola ma potente scheda con microcontrollore a 32 bit ESP8266 con 4MB di memoria Flash ed 80 MHz di clock..

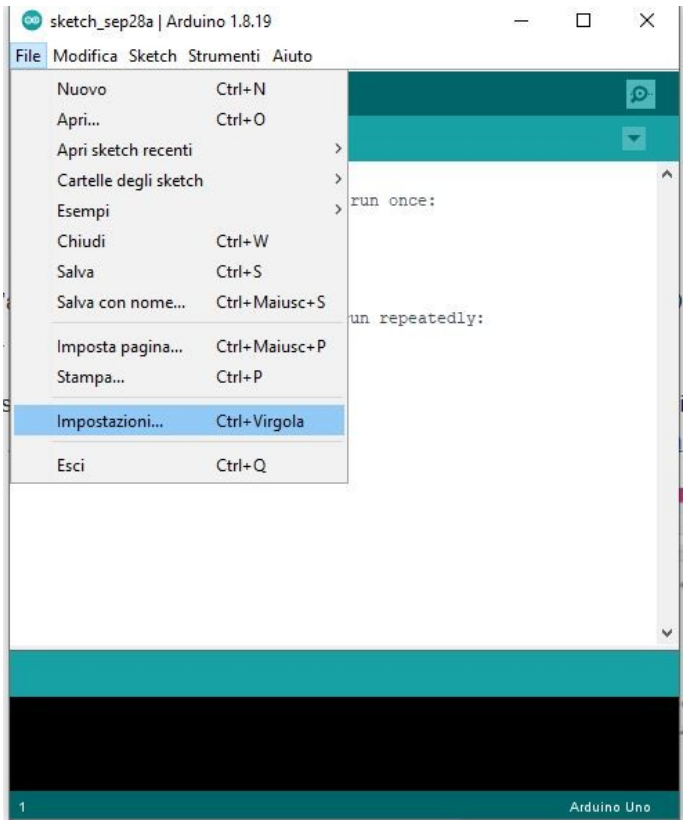
Questo micro è dotato di supporto wifi per la connessione ad una qualsiasi rete.



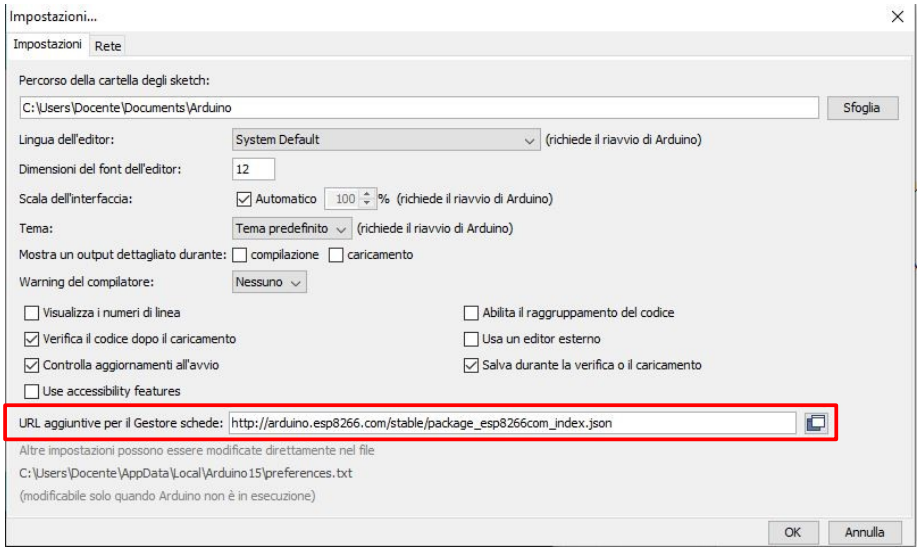
La scheda è programmabile con la piattaforma ARDUINO.

Per la sua programmazione, occorre innanzitutto installare la scheda sulla piattaforma, per farlo seguire i seguenti passi:

1) Sul menù file andare su IMPOSTAZIONI

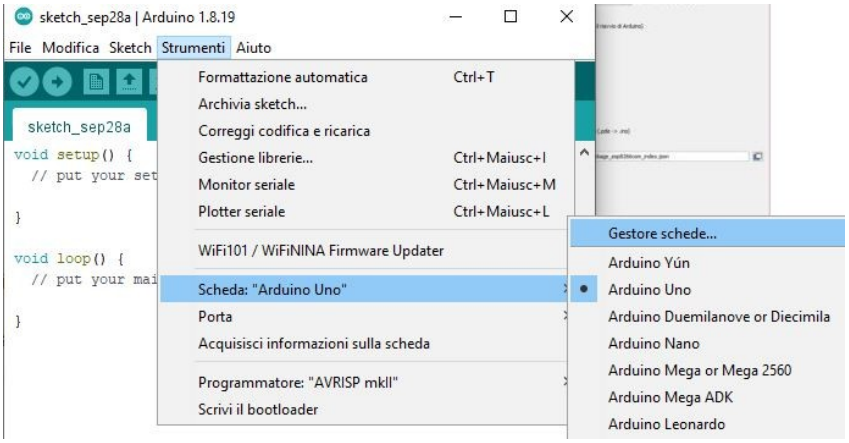


2) Sulla sezione **URL aggiuntive** scrivere il link evidenziato in figura:

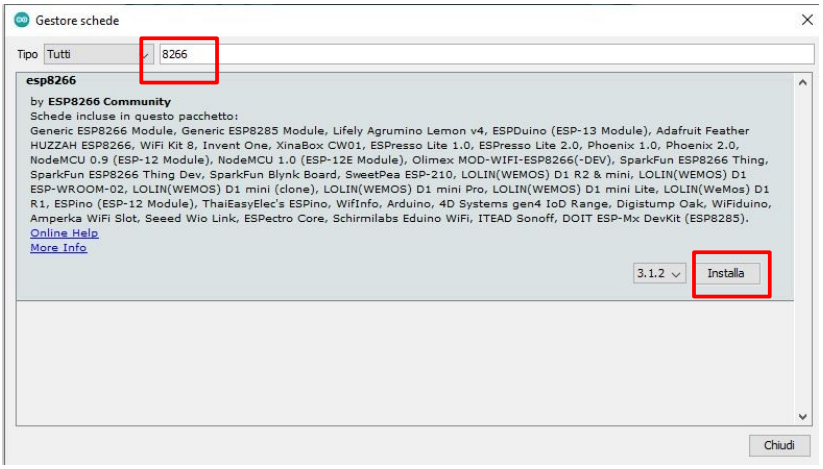


http://arduino.esp8266.com/stable/package_esp8266com_index.json

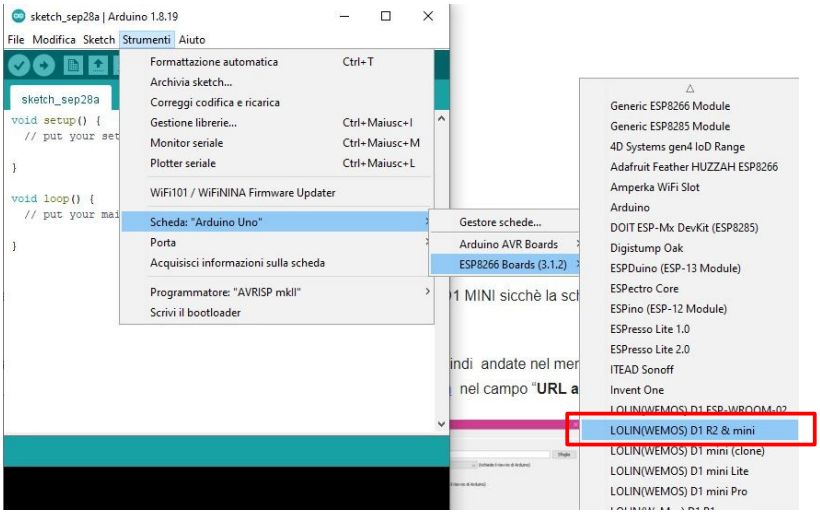
3) A questo punto sulla sezione **STRUMENTI**, selezionare **GESTIONE SCHEDE**



4) Inserire sul filtro di ricerca **8266** o anche **ESP8266** e verrà visualizzata la libreria **ESP8266**, fare click su **INSTALLA**

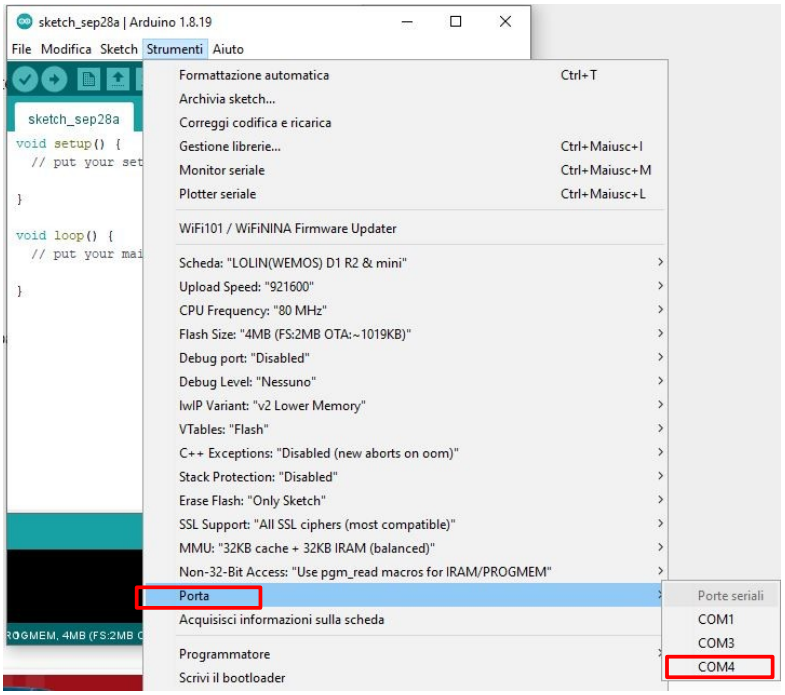


5) Una volta installata la scheda, selezionare su **STRUMENTI** il tipo di scheda **LOLIN WEMOS D1 R2 & mini**

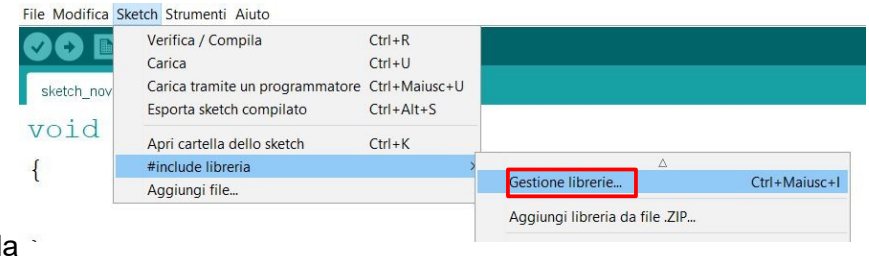


6) Collegare la scheda ed impostare la porta **COM** che appare sul menu **STRUMENTI-PORTA**

La porta COM da selezionare dipende dal PC e può essere differente da quella indicata in figura.



7) Installare la libreria Thingspeak. Andare sul menù **SKETCH** poi **Gestione librerie.**



Inserire nel filtro la parola **UniversalTelegramBot** per cercare la libreria.
Installare la libreria, al termine chiudere la finestra.

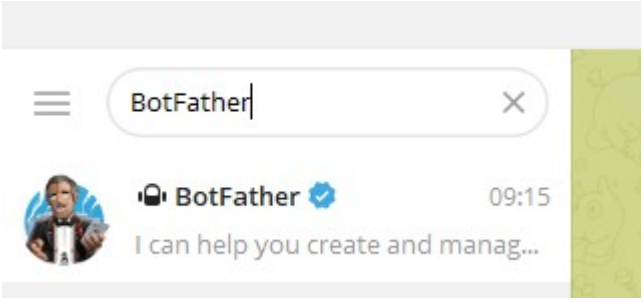
CREAZIONE BOT TELEGRAM

Ora occorre creare un BOT sull'App Telegram, per poter comunicare con la scheda.

BOT in informatica è l'abbreviazione di Robot. Un BOT Telegram non è altro che un'applicazione all'interno del client Telegram, con cui poter scambiare messaggi.

Per creare un BOT occorre utilizzare il "Padre" di tutti i BOT, e cioè il BotFather.

La prima operazione sarà perciò di lanciare Telegram e cercare la parola **BotFather**.



Entrando nella chat con il BotFather vi verranno dati due link per la manualistica, ed il primo comando per aprire la lista comandi cioè **/start**

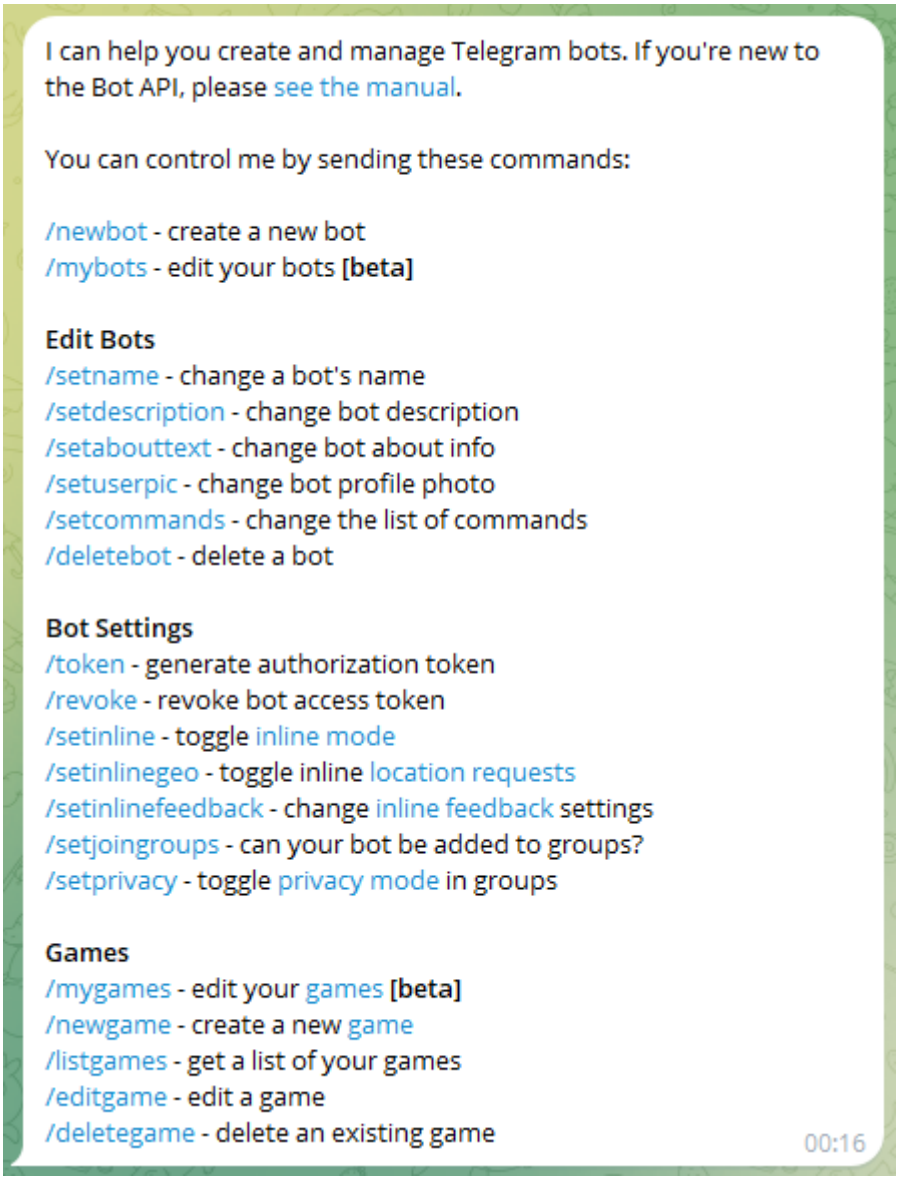
Cliccando su **/start** appariranno i possibili comandi che il BotFather può eseguire.

Per eseguire un comando è sufficiente cliccarci sopra, o scrivere direttamente il comando compreso il simbolo **slash /**

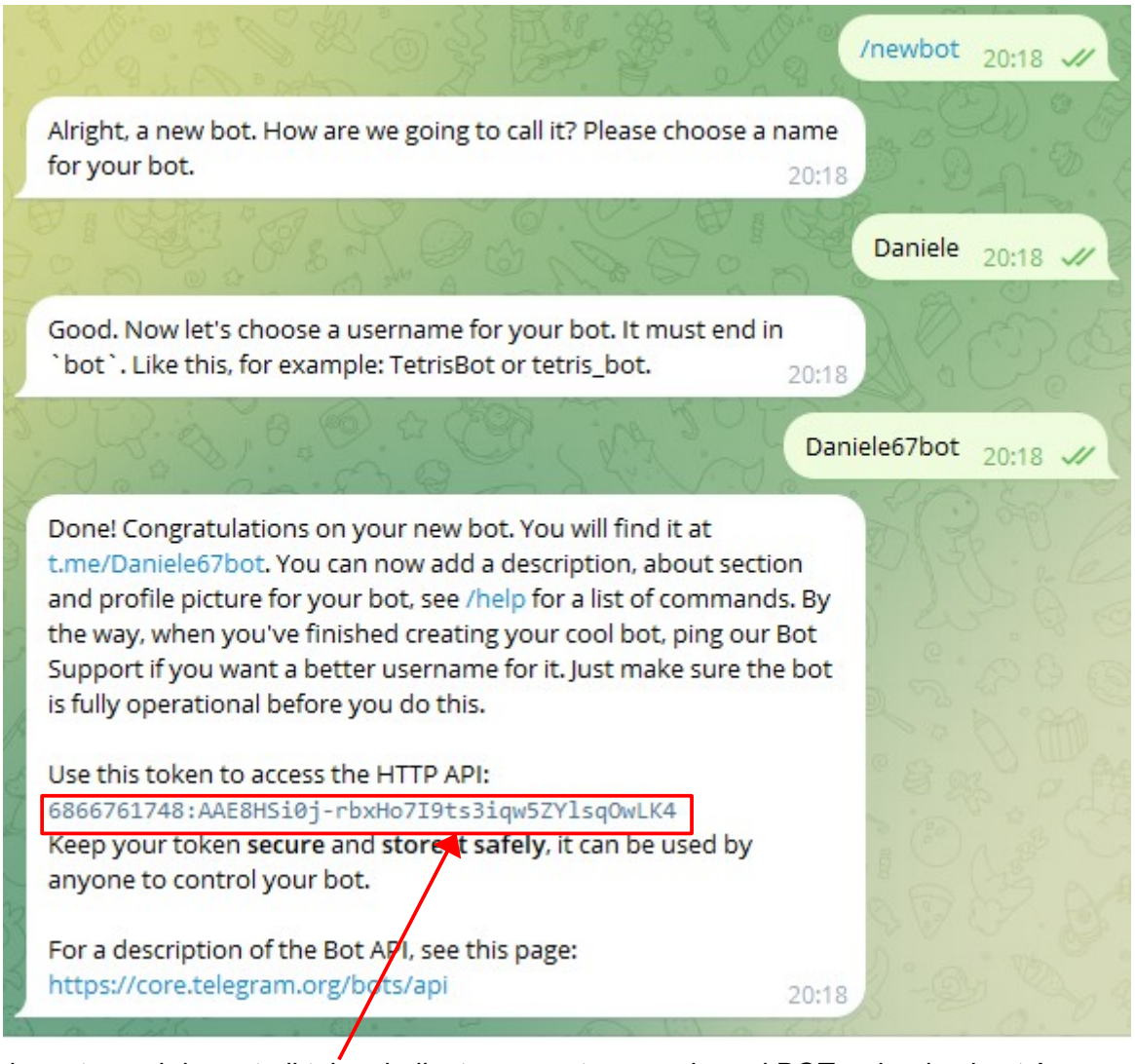
I comandi ci consentono di creare un BOT, di caricare la sua descrizione la sua immagine e tante altre cose oltre ovviamente alla cancellazione del BOT creato.

I comandi che ci utilizzeremo saranno essenzialmente 2:

- /newbot
- /setuserpic

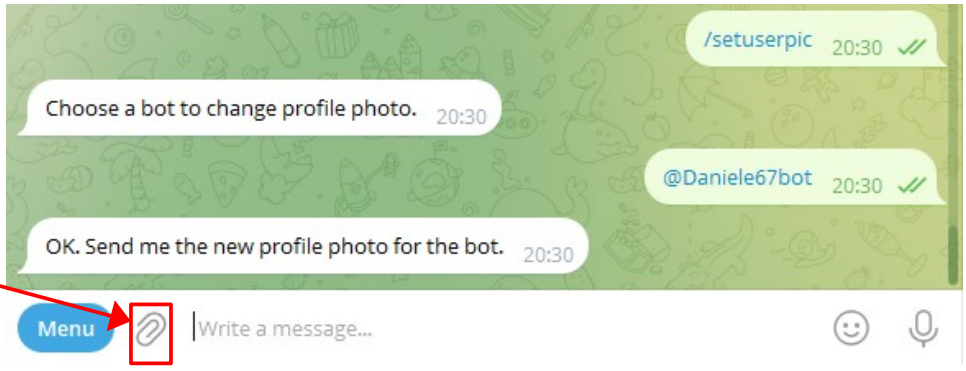


Innanzitutto creiamo il bot con il comanto /newbot. Ci verrà chiesto di dare un nome ed uno username. Diamo come nome del BOT il nostro nome e come username il nome seguito dalla data di nascita e dalla scritta bot, in quanto **lo username deve sempre terminare con bot**.
Nel mio caso ho inserito come nome del bot **Daniele**, e come username **Daniele67bot**.
Lo username è unico e se è già utilizzato vi verrà segnalato dal BotFather.



Dobbiamo tenerci da parte il token indicato, per poter accedere al BOT, volendo si potrà comunque generare un nuovo token con l'apposito comando /token.
Una volta creato il bot possiamo anche caricare un'immagine sul suo profilo con il comando /setuserpic.

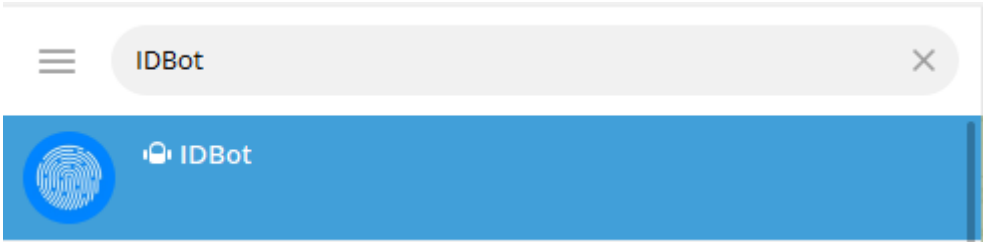
In questo caso ci verrà chiesto di selezionare un'immagine o di scattare una foto.
Tramite l'apposito pulsante si accede alla galleria o alla fotocamera.



Una volta creato il BOT ed aver memorizzato il token, abbiamo bisogno di un'altra informazione per poter comunicare e cioè del nostro **Id**.

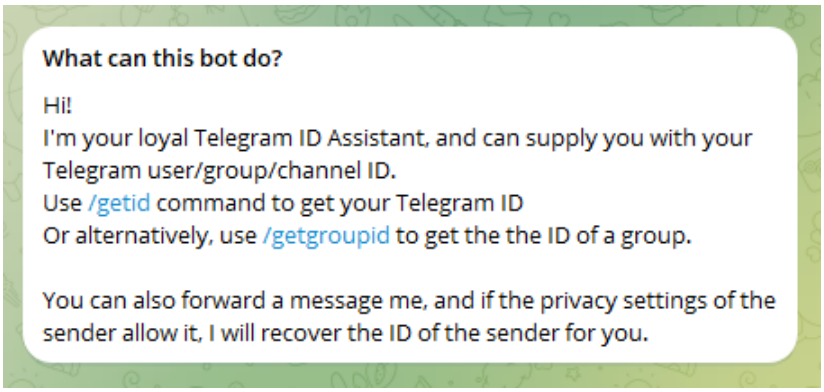
L'**Id** è un numero che identifica il nostro utente Telegram, ed è ottenbile mediante un altro BOT chiamato **IDBot**.

Come fatto prima con il BotFather, andiamo a cercare IDBot ed apriamo la chat.

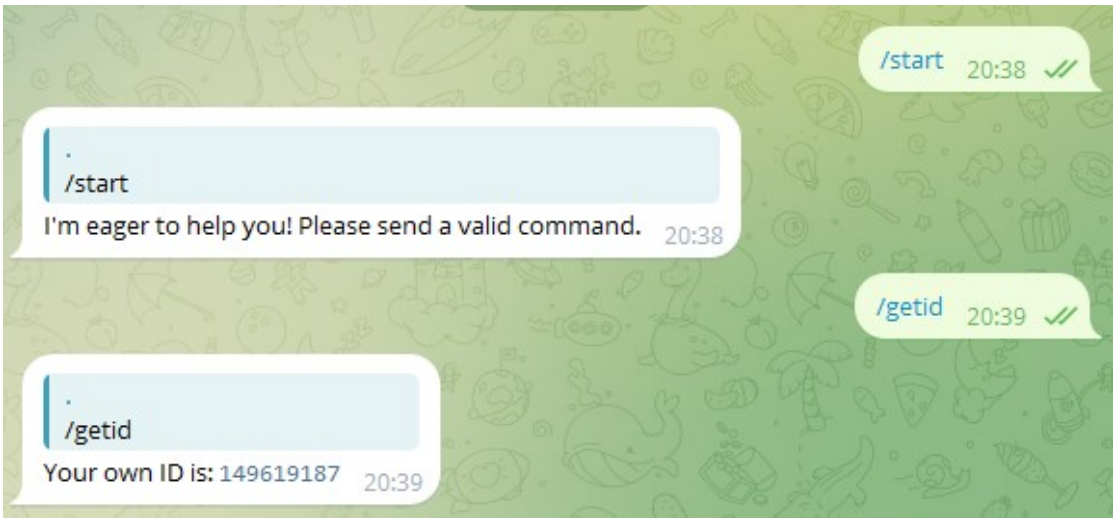


All'interno di questa chat, troveremo i due comanid possibili e cioè **/getid** e **/getgroupid**.

A noi servirà il primo dei due.



Prima di lanciare il comando /getid occorre avviarlo con l'apposito comando /start.



A questo punto abbiamo le due informazioni necessarie per poter comunicare con il nostro BOT; il Token visto prima e l'ID.

PROGRAMMAZIONE SCHEDA

Facciamo prima una breve premessa sui moduli WIFI montati a bordo delle schede.
I due moduli wifi più utilizzati sono l'**ESP32** e l'**ESP8266**. In entrambi i casi i moduli hanno un processore a 32 bit, ma il primo è un'evoluzione del secondo e pertanto ha caratteristiche migliori.
Vediamo le principali differenze tra i due.

ESP8266	ESP32
CPU Xtensa single core 80 MHz 32 bit	*CPU Xtensa dual core 160-240MHz 32 bit*
WIFI 802.11 b/g/n HT20 (banda canale 20Mhz)	*WIFI 802.11 b/g/n HT40 (banda canale 40Mhz)*
SRAM 160 Kbytes (RAM Statica no refresh)	*SRAM 512 Kbytes (RAM Statica no refresh)*
FLASH Memory 16 MBytes	*FLASH Memory 16 MBytes*
17 GPIO	*36 GPIO*
8 PWM Software	*1 PWM Hardware 16 PWM Software*
2 SPI - 1 I2C - 2 I2S – 2 UART	*4 SPI - 2 I2C - 2 I2S – 2 UART*
Convertitore ADC 10bit	*Convertitore ADC 12 bit*
----------	*1 CAN*
----------	*1 Ethernet*
----------	*1 Interfaccia Sensore Touch*
----------	*1 Sensore di Temperatura*

Nella scheda Arduino Wifi rev.4 troviamo un modulo ESP32, mentre invece nel modulo WEMOS D1 mini troviamo un ESP8266.
Tornando alla programmazione, il primo passo sarà quello di importare le librerie con i seguenti comandi.

N.B. una libreria è un insieme di funzioni che vengono messe a disposizione nel programma in cui essa viene inclusa.

```
#ifdef Esp32
   #include <WiFi.h>          //libreria per il chip ESP32
#else
   #include<ESP8266WiFi.h>   //libreria per il chip ESP8266
#endif

#include<WiFiClientSecure.h>
#include<UniversalTelegramBot.h>
#include<ArduinoJson.h>
```

Il primo **ifdef,** permette di caricare la libreria adatta al tipo di modulo utilizzato, nel nostro caso potremmo anche includere direttamente la libreria **ESP8266.h.**

Successivamente andiamo a definire delle costanti per il compilatore.
Con la define infatti non viene definita una vera e propria costante, questa è una direttiva per il compilatore, in pratica viene associato un valore numerico o stringa ad un nome.
Utilizziamo le define per dare al compilatore le stringhe del BOTtoken e dell'ID che abbiamo visto prima.

```
//token del BOT creato
#define BOTtoken "6866761748:AAE8HSi0j-rbxHo7I9ts3iqw5ZYlsqOwLK4"
//chat_id della chat ottenuto con myidbot
#define CHAT_ID "149619187"
```

Andiamo ora a definire le variabili e le costanti necessarie per il programma, tra cui il nome della rete wifi e la sua password.

```
//intervallo di verifica della presenza del messaggio, 1 secondo
int delayRichiestaBot = 1000;
unsigned long lastTimeBotRan;
int valido;
//nome rete e password
const char* ssid = "Lenovo";
const char* password = "prova123";
```

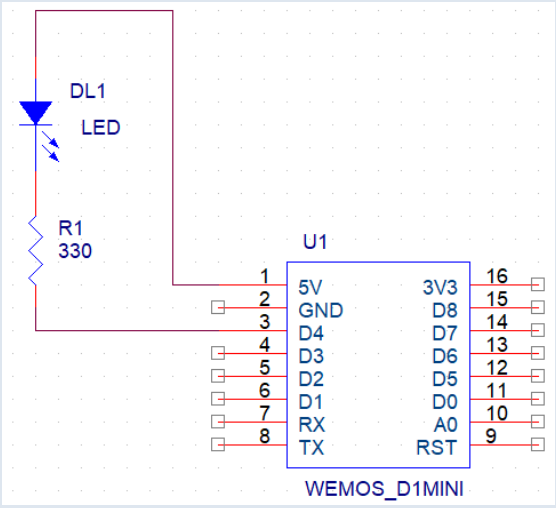
Il passo successivo è quello di instanziare un oggetto per creare una connessione sicura HTTPS, con la libreria **WifiClientSecure.h** e di creare un oggetto bot con la libreria **UniversalTelegramBot.**

N.B. In maniera estremamente semplificata, nella programmazione ad oggetti "istanziare un oggetto" significa creare in una zona di memoria un insieme di funzioni che hanno come modello l'oggetto base chiamato classe.

Wificliensecure.h è una libreria che ci permette di instaurare connessioni sicure con il protocollo https.
UniversaTelegramBot.h è una libreria che contiene le funzioni necessarie per dialogare con il BOT di Telegram.

```
//instaurare una connessione client sicura in https
WiFiClientSecure client;
//creare un oggetto bot, con la libreria UniversalTelegramBot
UniversalTelegramBot bot(BOTtoken, client);
```

N.B.
Sulla scheda Wemos D1 mini è presente un LED collegato al pin D4.
Il PIN D4 può essere chiamato anche LED_BUILTIN ma va comunque inizializzato nel setup del programma.
Essendo il LED connesso come in figura si accenderà se D4=0 e si spegnerà se D4=1.



Andiamo ora a vedere il SETUP.

```
void setup() {
  valido=0;
  Serial.begin(115200);    //inizializzo il monitor seriale alla velocità di 115200 bps
  pinMode(D4, OUTPUT);     //imposto il bit a bordo della scheda come output
  digitalWrite(D4, HIGH); //accendo l'uscita e di conseguenza spengo il led collegato ad essa

  #ifdef ESP8266
    client.setInsecure(); //impedisce i controlli di validazione del server
  #endif                  //e consente richieste HTTPs senza certificato

  WiFi.mode(WIFI_STA);     //imposto il modulo wifi in modalità Station
  WiFi.begin(ssid, password);   //inizializzo il modulo wifi con i dati della rete wifi

  while (WiFi.status() != WL_CONNECTED) {  //attendo la connessione alla rete
    delay(1000);
    Serial.println("Connessione in corso...");
  }//while

  Serial.println(WiFi.localIP());    //stampa dell'indirizzo IP della scheda
}
```

Nel setup viene inizializzato il monitor seriale con la velocità di 115.200 bps, viene inizializzato il pin a cui è connesso il LED interno.
Se il modulo è un ESP8266 (come nel nostro caso) viene anche consentita una connessione sicura ma senza l'utilizzo di certificati.

N.B.
Un certificato digitale serve a garantire l'affidabilità di un sito con cui realizzare una connessione criptata secondo il protocollo SSL.

Nel setup inoltre viene attivata la scheda wifi in modo Station, cioè in modo che possa connettersi ad una rete wifi. Successivamente vengono inseriti il nome della rete (SSID) e la sua password ed effettuata la connessione.

Nella funzione loop invece viene testato il BOT di Telegram ad intervalli di tempo regolari.

```
void loop() {
  if(millis() > lastTimeBotRan + delayRichiestaBot) {
    Serial.print("leggo i nuovi ");
    valido=1;
    int numNuovoMessaggio = bot.getUpdates(bot.last_message_received + 1);
    Serial.println(numNuovoMessaggio);
    while(numNuovoMessaggio){
       Serial.print("Rispondo al messaggio ricevuto.");
       gestisciMessaggioBot(numNuovoMessaggio);
       numNuovoMessaggio = bot.getUpdates(bot.last_message_received + 1);
    }//while
    //aggiornamento del tempo trascorso
    lastTimeBotRan = millis();
  }//if
}
```

Con millis abbiamo il tempo attuale dall'avvio del programma, se si è superato un tempo indicato in **delayRichiestaBot** , allora viene effettuata la lettura di eventuali messaggi provenienti dal BOT.

Quando viene testato il BOT (all'interno del ciclo if) la prima operazione è vedere quanti messaggi sono arrivati.
Questo valore viene ottenuto con la funzione ***bot.getUpdates(bot.last_message_received+1)***
bot.last_message_received è un indice che si incrementa ad ogni messaggio ricevuto.
All'interno del ciclo while, viene richiamata la funzione **gestisciMessaggioBot**, dove si va a valutare il contenuto del messaggio ricevuto.

Alla funzione **gestisciMessaggioBot** viene passato il valore **numNuovoMessaggio** che contiene il numero di messaggi da valutare.

All'interno di questa funzione possiamo evidenziare le seguenti istruzioni:

- ***chat_id = String(bot.messages[i].chat_id);*** per leggere l'ID della chat
- ***bot.sendMessage(chat_id, "stringa da mandare");*** per inviare una stringa di risposta al bot
- ***String text = bot.messages[i].text;*** per leggere il contenuto del messaggio ricevuto
- ***bot.messages[i].from_name;*** per leggere il nome dell'utente Telegram che ha inviato il messaggio

Di seguito la funzione **gestisciMessaggioBot** completa.

```
//funzione per gestire ogni messaggio ricevuto
void gestisciMessaggioBot(int numNuovoMessaggio){
    Serial.println("gestisciMessaggioBot");
    Serial.println(String(numNuovoMessaggio));

    for (int i=0; i<numNuovoMessaggio; i++){
      //lettura del chat_id del messaggio che stiamo analizzando
      String chat_id = String(bot.messages[i].chat_id);
      if(chat_id != CHAT_ID){
        bot.sendMessage(chat_id, "Ciao non sei autorizzato ad usare i miei servizi");
        continue;
      }//if

      //Stampa a monitor seriale il testo del messaggio che stiamo analizzando
      String text = bot.messages[i].text;
      Serial.println(text);
      //Lettura del nome utente dell'utente stesso che ha inviato il messaggio al bot
      String nome_utente = bot.messages[i].from_name;

      if(text== "/start")
       {
        valido=0;
        String messaggio_benvenuto = "Benvenuto, " + nome_utente + "\n";
        messaggio_benvenuto += "Prova i seguenti comandi\n";
        messaggio_benvenuto += "/ON\n";
        messaggio_benvenuto += "/OFF\n";
        //Invio del messaggio
        bot.sendMessage(chat_id, messaggio_benvenuto, "");
       }//if

       if(text== "/ON")
        {
         valido=0;
         digitalWrite(D4,LOW);  //accendo l'uscita e di conseguente spengo il LED
         String messaggio = "Ho acceso il led\n";
         bot.sendMessage(chat_id, messaggio, ""); //Invio del messaggio
        }

       if(text== "/OFF")
        {
         valido=0;
         digitalWrite(D4,HIGH);  //spengo l'uscita e di conseguente accendo il LED
         String messaggio = "Ho spento il led\n";
         bot.sendMessage(chat_id, messaggio, ""); //Invio del messaggio
        }

        if(valido==1){
          valido=0;
          String messaggio = nome_utente +  " comando non riconosciuto. ";
          bot.sendMessage(chat_id, messaggio, "");  //Invio del messaggio
        }
     }//for
 }//void gestisciMessaggioBot
```

Di seguito il programma completo.

```
#ifdef Esp32
  #include <WiFi.h>
#else
  #include<ESP8266WiFi.h>
#endif
#include<WiFiClientSecure.h>
#include<UniversalTelegramBot.h>
//token del BOT creato
#define BOTtoken "6866761748:AAE8HSi0j-rbxHo7I9ts3iqw5ZYlsqOwLK4"
//chat_id della chat ottenuto con myidbot
#define CHAT_ID "149619187"
//intervallo di verifica della presenza del messaggio, 1 secondo
int delayRichiestaBot = 1000;
unsigned long lastTimeBotRan;
int valido;
//nome rete e password
const char* ssid = "Lenovo";
const char* password = "prova123";
//instaurare una connessione client sicura in https
WiFiClientSecure client;
//creare un oggetto bot, con la libreria UniversalTelegramBot
UniversalTelegramBot bot(BOTtoken, client);

void setup() {
 valido=0;
 Serial.begin(115200);   //inizializzo il monitor seriale alla velocità di 115200 bps
 pinMode(D4, OUTPUT);    //imposto il bit a bordo della scheda come output
 digitalWrite(D4, HIGH);    //accendo l'uscita e di conseguenza spengo il led collegato ad essa
 #ifdef ESP8266
   client.setInsecure(); //impedisce i controlli di validazione del server
 #endif                        //e consente richieste HTTPs senza certificato
 WiFi.mode(WIFI_STA);    //imposto il modulo wifi in modalità Station
 WiFi.begin(ssid, password);   //inizializzo il modulo wifi con i dati della rete wifi
 while (WiFi.status() != WL_CONNECTED) {  //attendo la connessione alla rete
   delay(1000);
   Serial.println("Connessione in corso...");
 }//while
 Serial.println(WiFi.localIP());   //stampa dell'indirizzo IP della scheda
}
```

```
//funzione per gestire ogni messaggio ricevuto
void gestisciMessaggioBot(int numNuovoMessaggio){
   Serial.println("gestisciMessaggioBot");
   Serial.println(String(numNuovoMessaggio));
   for (int i=0; i<numNuovoMessaggio; i++){
    //lettura del chat_id del messaggio che stiamo analizzando
    String chat_id = String(bot.messages[i].chat_id);
    if(chat_id != CHAT_ID){
      bot.sendMessage(chat_id, "Ciao non sei autorizzato ad usare i miei servizi");
      continue;  //esce dal ciclo for
    }//if
   //Stampa a monitor seriale il testo del messaggio che stiamo analizzando
   String text = bot.messages[i].text;
    Serial.println(text);
    //Lettura del nome utente dell'utente stesso che ha inviato il messaggio al bot
    String nome_utente = bot.messages[i].from_name;

    if(text== "/start")  {
      valido=0;
      String messaggio_benvenuto = "Benvenuto, " + nome_utente + "\n";
      messaggio_benvenuto += "Prova i seguenti comandi\n";
      messaggio_benvenuto += "/ON\n";
      messaggio_benvenuto += "/OFF\n";
      //Invio del messaggio
      bot.sendMessage(chat_id, messaggio_benvenuto, "");
    }//if

    if(text== "/ON")  {
      valido=0;
      digitalWrite(D4,LOW);  //accendo l'uscita e di conseguente spengo il LED
      String messaggio = "Ho acceso il led\n";
      bot.sendMessage(chat_id, messaggio, ""); //Invio del messaggio
    }

    if(text== "/OFF")  {
      valido=0;
      digitalWrite(D4,HIGH);  //spengo l'uscita e di conseguente accendo il LED
      String messaggio = "Ho spento il led\n";
      bot.sendMessage(chat_id, messaggio, ""); //Invio del messaggio
    }
```

```
      if(valido==1){
        valido=0;
        String messaggio = nome_utente +  " comando non riconosciuto. ";
        bot.sendMessage(chat_id, messaggio, "");  //Invio del messaggio
      }
    }//for
}//void gestisciMessaggioBot

void loop() {
  if(millis() > lastTimeBotRan + delayRichiestaBot) {
    Serial.print("leggo i nuovi ");
    valido=1;
    int numNuovoMessaggio = bot.getUpdates(bot.last_message_received + 1);
    Serial.println(numNuovoMessaggio);
    while(numNuovoMessaggio){
      Serial.print("Rispondo al messaggio ricevuto.");
      gestisciMessaggioBot(numNuovoMessaggio);
      numNuovoMessaggio = bot.getUpdates(bot.last_message_received + 1);
    }//while
    //aggiornamento del tempo trascorso
    lastTimeBotRan = millis();
  }//if
}
```

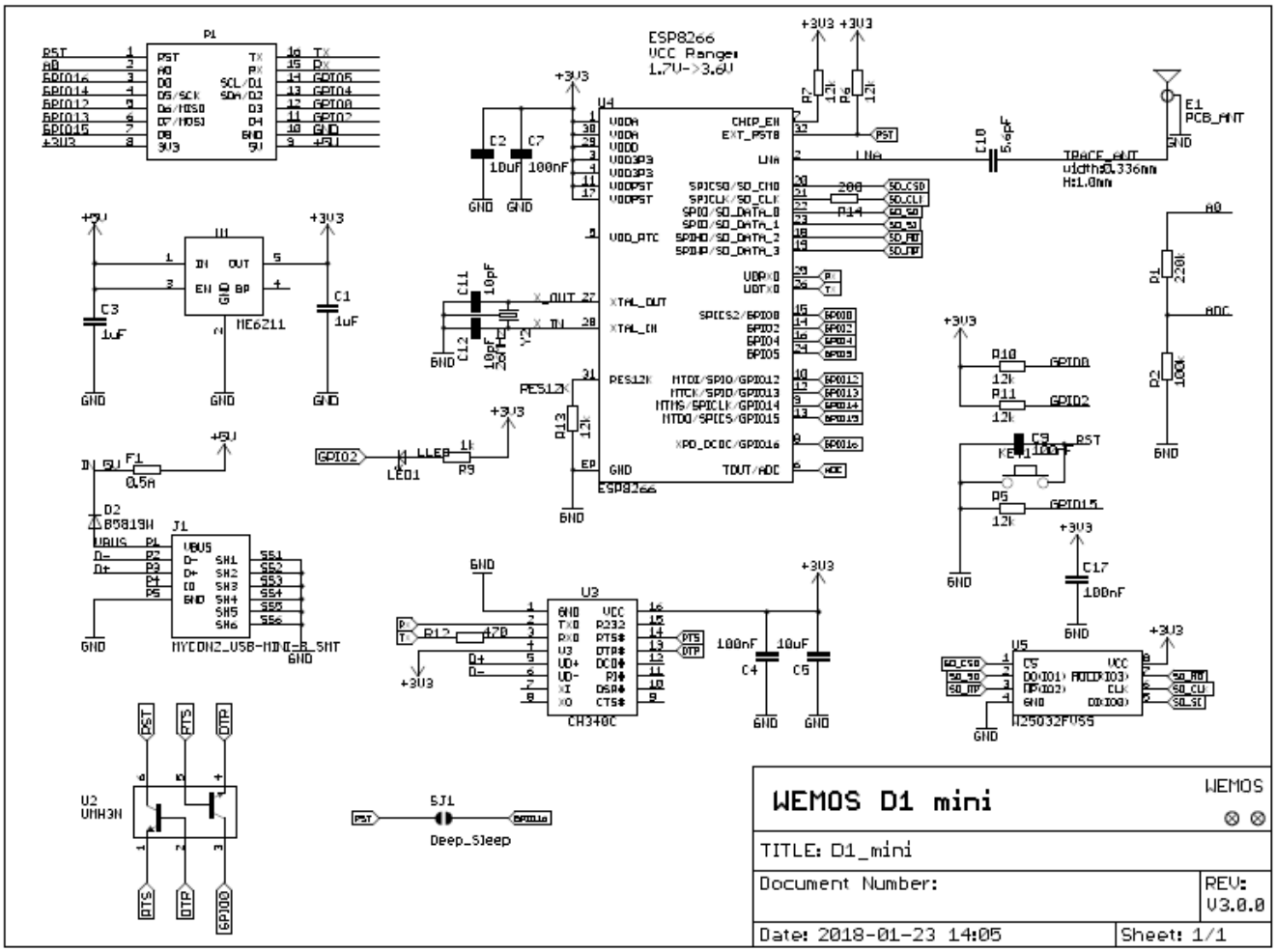
APPENDICE

Funzioni della libreria **UniversalTelegramBot.h** *https://github.com/witnessmenow/Universal-Arduino-Telegram-Bot*

Feature	Description	Usage	Example
Receiving Messages	Your bot can read messages that are sent to it. This is useful for sending commands to your arduino such as toggle and LED	`int getUpdates(long offset)` Gets any pending messages from Telegram and stores them in **bot.messages** . Offset should be set to **bot.last_message_received** + 1. Returns the numbers new messages received.	FlashLED or any other example
Sending messages	Your bot can send messages to any Telegram or group. This can be useful to get the arduino to notify you of an event e.g. Button pressed etc (Note: bots can only message you if you messaged them first)	`bool sendMessage(String chat_id, String text, String parse_mode = "")` Sends the message to the chat_id. Returns if the message sent or not.	EchoBot or any other example
Reply Keyboards	Your bot can send reply keyboards that can be used as a type of menu.	`bool sendMessageWithReplyKeyboard(String chat_id, String text, String parse_mode, String keyboard, bool resize = false, bool oneTime = false, bool selective = false)` Send a keyboard to the specified chat_id. parse_mode can be left blank. Will return true if the message sends successfully.	ReplyKeyboard
Inline Keyboards	Your bot can send inline keyboards. Note: URLS & callbacks are supported currently	`bool sendMessageWithInlineKeyboard(String chat_id, String text, String parse_mode, String keyboard)` Send a keyboard to the specified chat_id. parse_mode can be left blank. Will return true if the message sends successfully.	InlineKeyboard

Send Photos	It is possible to send phtos from your bot. You can send images from the web or from the arduino directly (Only sending from an SD card has been tested, but it should be able to send from a camera module)	Check the examples for more info	From URL Binary from SD From File Id
Chat Actions	Your bot can send chat actions, such as *typing* or *sending photo* to let the user know that the bot is doing something.	`bool sendChatAction(String chat_id, String chat_action)` Send a the chat action to the specified chat_id. There is a set list of chat actions that Telegram support, see the example for details. Will return true if the chat actions sends successfully.	
Location	Your bot can receive location data, either from a single location data point or live location data.	Check the example.	Location
Channel Post	Reads posts from channels.	Check the example.	ChannelPost
Long Poll	Set how long the bot will wait checking for a new message before returning now messages. This will decrease the amount of requests and data used by the bot, but it will tie up the arduino while it waits for messages	`bot.longPoll = 60;` Where 60 is the amount of seconds it should wait	LongPoll

Update Firmware and SPIFFS	You can update firmware and spiffs area through send files as a normal file with a specific caption.	`update firmware` or `update spiffs` These are captions for example.	telegramOTA
Set bot's commands	You can set bot commands programmatically from your code. The commands will be shown in a special place in the text input area	`bot.setMyCommands("[{\"command\":\"help\", \"description\":\"get help\"}, {\"command\":\"start\",\"description\":\"start conversation\"}]");` . See examples	SetMyCommands

---

Schema Wemos D1 mini