Udit Bansal

Batch - 1
Embedded Full Stack IOT analyst

DHT Control Using Telegram

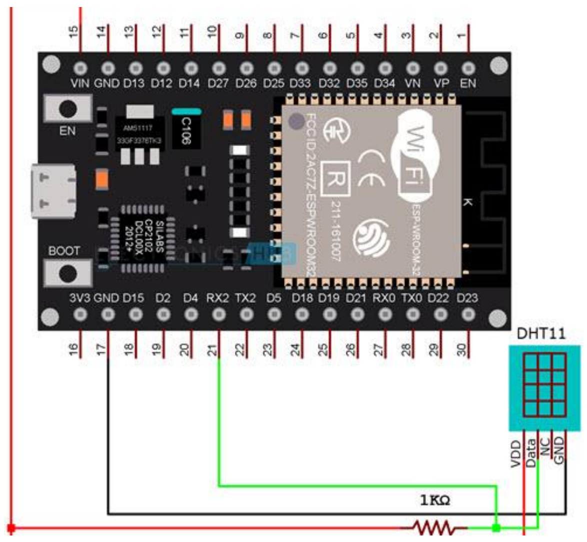
August 20, 2023

Project Overview

We are able to control DHT by sending a command to ESP32 from our telegram bot. Our main goal is to control our ESP32 from a simple mobile application.

Using this telegram bot, we are not only able to send commands to ESP32 to activate DHT but we can also receive DHT data from our ESP32 to our telegram Chat.

Same reading from the DHT when activated will be saved in the thingSpeak database and can send a command from telegram to receive a link of the ThingSpeak chart.


VIN GND D13 D12 D14 D27 D26 D25 D33 D32 D35 D34 VN VP EN
EN
BOOT
3V3 GND D15 D2 D4 RX2 TX2 D5 D18 D19 D21 RX0 TX0 D22 D23
DHT11
VDD Data NC GND
1KΩ

Apparatus Required

ESP32

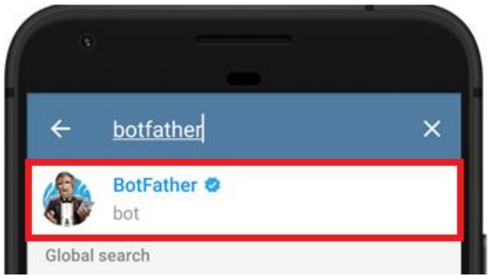
DHT11

BreadBoard

Connecting Wires

Creating a Telegram Bot

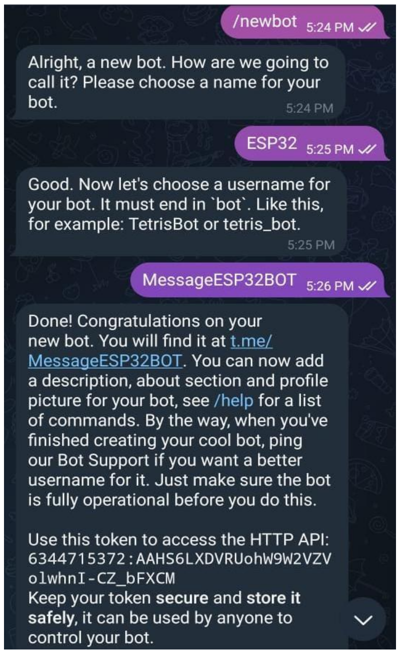
Using BotFather



First, search for “botfather” and click the BotFather as shown below. Or open this link t.me/botfather in your smartphone.

click the start button.

Type /newbot and follow the instructions to create your bot. Give it a name and username.



/newbot 5:24 PM

Alright, a new bot. How are we going to call it? Please choose a name for your bot. 5:24 PM

ESP32 5:25 PM

Good. Now let's choose a username for your bot. It must end in `bot`. Like this, for example: TetrisBot or tetris_bot. 5:25 PM

MessageESP32BOT 5:26 PM

Done! Congratulations on your new bot. You will find it at t.me/MessageESP32BOT. You can now add a description, about section and profile picture for your bot, see /help for a list of commands. By the way, when you've finished creating your cool bot, ping our Bot Support if you want a better username for it. Just make sure the bot is fully operational before you do this.

Use this token to access the HTTP API:
`6344715372:AAHS6LXDVRUohW9W2VZVolwhnI-CZ_bFXCM`
Keep your token **secure** and **store it safely**, it can be used by anyone to control your bot.

After Choosing a name and a username for your telegram bot.

You will receive an HTTP API token which we will use to access our bot from our program.

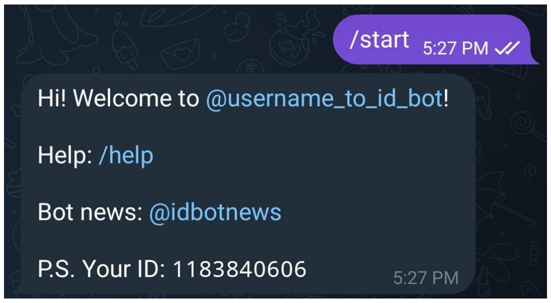
We will save our API as TOKEN in our microPython Code.

TOKEN = "6344715372:AAHS6LXDVRUohW9W2VZVolwhnI-CZ_bFXCM"

We can also add Menu to our bot by typing /mybots and selecting appropriate commands. For this, We have added 2 commands for /send and /link.

Using IDBot

Anyone that knows your bot username can interact with it. To make sure that we ignore messages that are not from our Telegram account, you can get your Telegram User ID.



/start 5:27 PM

Hi! Welcome to @username_to_id_bot!

Help: /help

Bot news: @idbotnews

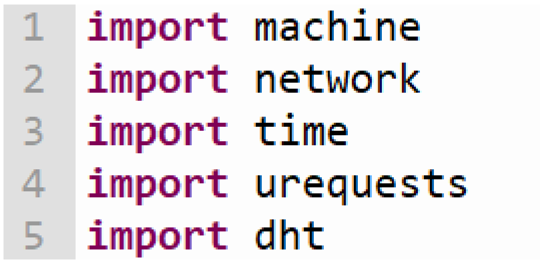
P.S. Your ID: 1183840606 5:27 PM

search for “IDBot” or open this link t.me/myidbot in your smartphone.

Type /start. You will receive a reply with your userID

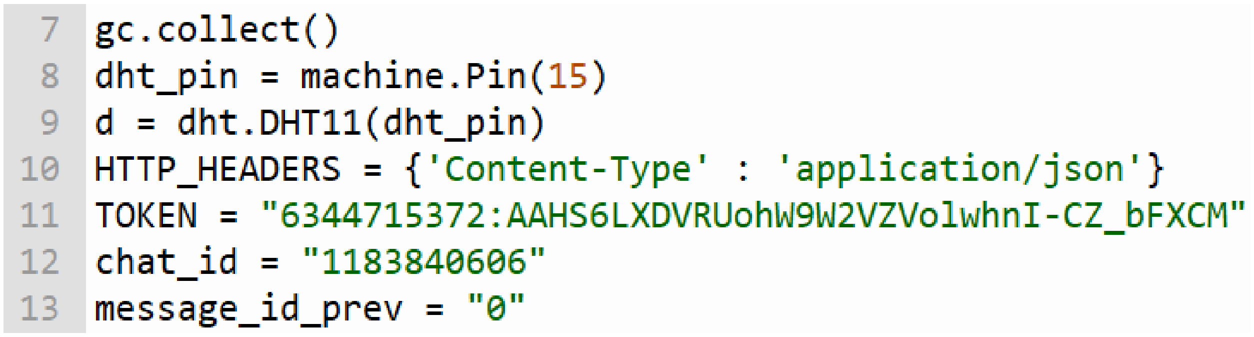
MicroPython Program

Import necessary modules



```
import machine
import network
import time
import urequests
import dht
```

DHT Sensor Object, Constants and Variables



```
gc.collect()
dht_pin = machine.Pin(15)
d = dht.DHT11(dht_pin)
HTTP_HEADERS = {'Content-Type' : 'application/json'}
TOKEN = "6344715372:AAHS6LXDVRUohW9W2VZVolwhnI-CZ_bFXCM"
chat_id = "1183840606"
message_id_prev = "0"
```

- DHT is set at pin number 15 in ESP32.
- HTTP_HEADERS is the type of data we are going to send to thingspeak
- TOKEN is the API key we received when we created the telegram bot
- Chat_id is the unique user id to whom we will send the data
- Message_id_prev is the unique id of the message we last received . Initially we take it as 0.

Function to Connect ESP32 with Wifi

```python
wlan = network.WLAN(network.STA_IF)
wlan.active(True)
def connect_wifi(ssid, password, timeout=10):
    wlan.connect(ssid, password)
    print("Connecting to network...")
    start_time = time.ticks_ms()
    while not wlan.isconnected() and (time.ticks_ms() - start_time < timeout * 1000):
        time.sleep(1)
    if wlan.isconnected():
        print("Connected to network:", wlan.ifconfig()[0])
    else:
        print("Connection failed!")
        machine.reset()
```

Handling JSON data send from Telegram

As per Telegram Bot API Documentation, We can access data send by User using the link:

https://api.telegram.org/bot6344715372:AAHS6LXDVRUohW9W2VZVolwhnI-CZ_bFXCM/getUpdates

This will provide data in JSON format.

We required two variables from this JSON data

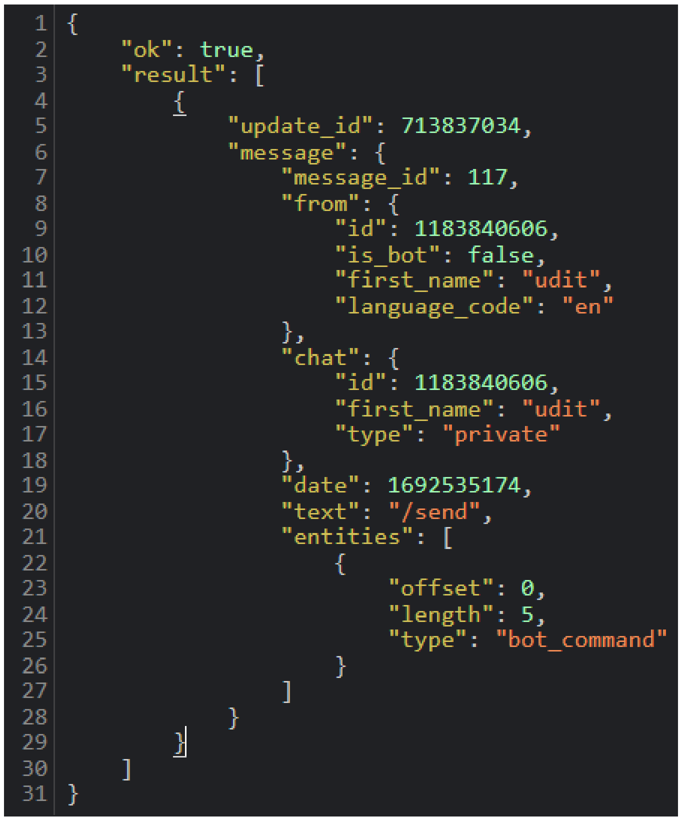
1. Update_id
2. text

We will save this JSON data in a variable “messages”

As we are required to access latest message we use messages[-1] and save in variable “latest_message”

latest_message.get("update_id", "")
latest_message.get("message", {}).get("text", "")

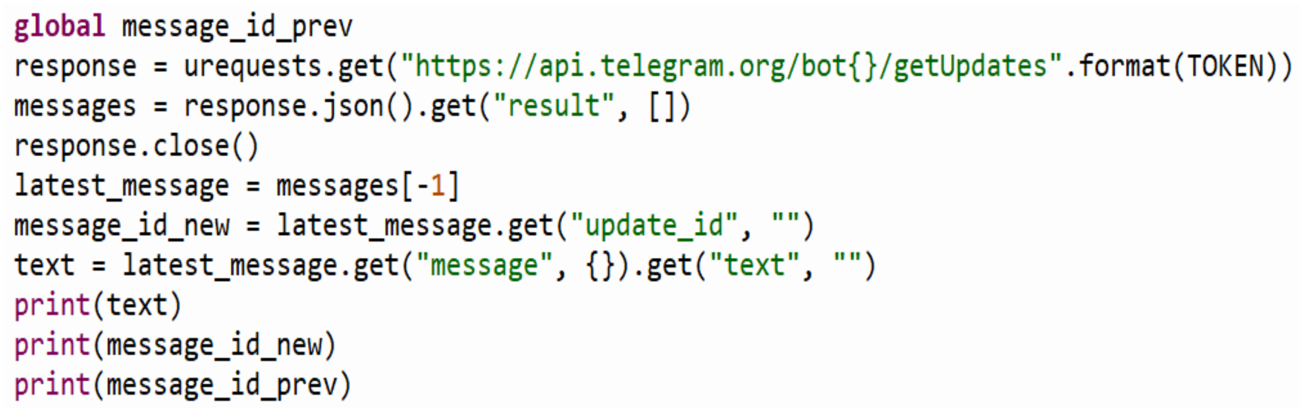
Using above to lines of code we can get update_id and the message that user sends



```json
{
    "ok": true,
    "result": [
        {
            "update_id": 713837034,
            "message": {
                "message_id": 117,
                "from": {
                    "id": 1183840606,
                    "is_bot": false,
                    "first_name": "udit",
                    "language_code": "en"
                },
                "chat": {
                    "id": 1183840606,
                    "first_name": "udit",
                    "type": "private"
                },
                "date": 1692535174,
                "text": "/send",
                "entities": [
                    {
                        "offset": 0,
                        "length": 5,
                        "type": "bot_command"
                    }
                ]
            }
        }
    ]
}
```

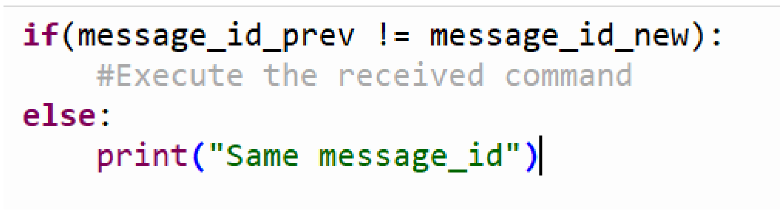
Function to receive and execute command send by the user

1. Our first step is to receive JSON data from telegram and fetch and store the latest update_id and message sent by the user.
   Update_id is stored in variable “message_id_new” and message in variable “text”



```
global message_id_prev
response = urequests.get("https://api.telegram.org/bot{}/getUpdates".format(TOKEN))
messages = response.json().get("result", [])
response.close()
latest_message = messages[-1]
message_id_new = latest_message.get("update_id", "")
text = latest_message.get("message", {}).get("text", "")
print(text)
print(message_id_new)
print(message_id_prev)
```

2. Now we compare the old message id with the new message id to check whether we are sending data repeatedly for the same last command. This will check whether we receive a new command or not.
   If the message id do not match with previous one then will act on the command accordingly otherwise we will do nothing or print “same message id” in terminal



```
if(message_id_prev != message_id_new):
    #Execute the received command
else:
    print("Same message_id")
```

3. Inside our if statement first we will take the reading from DHT11 and save reading in the variable “dht_reading” this data we will save in our ThingSpeak database which will create the chart for temperature and humidity for the same.
   We will use the API link of thingspeak to do the same.

```
d.measure()
t = d.temperature()
h = d.humidity()
dht_readings = {"Temp": t, "Humidity": h}

request = urequests.post(
        "https://api.thingspeak.com/update?api_key=" + "6B5UP2KHX6NISDZP",
        json=dht_readings,
        headers=HTTP_HEADERS,
)
request.close()
print(f"Temperature: {t}°C, Humidity: {h}%" )
```

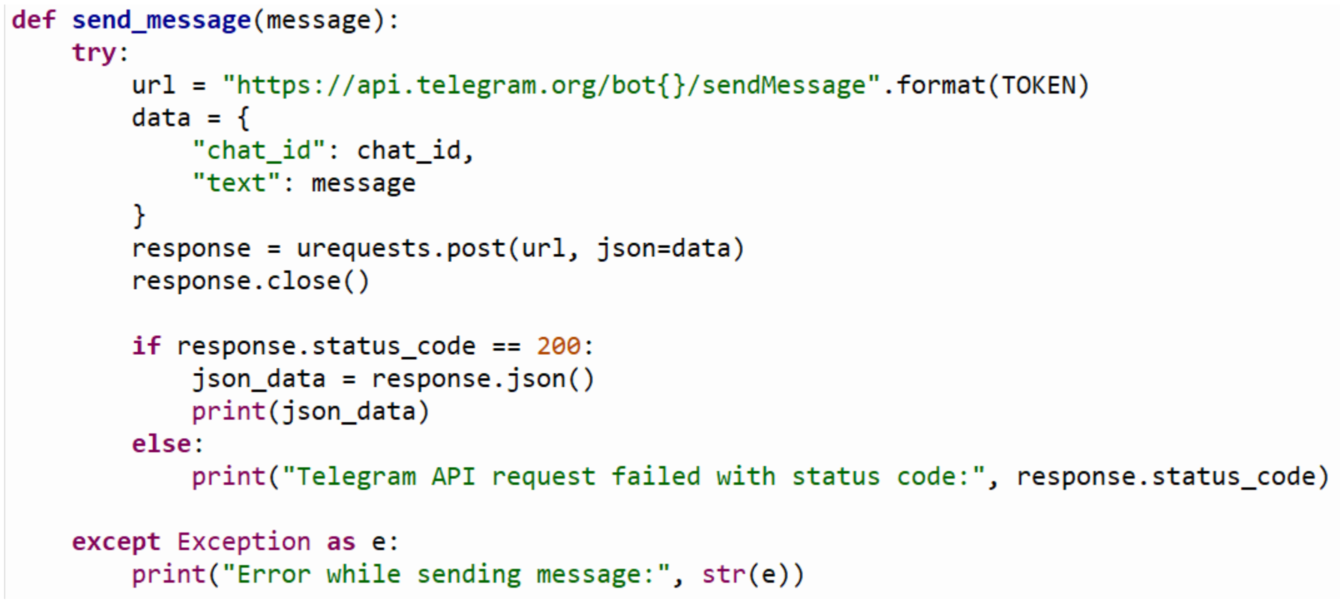
4. After successfully reading and saving the DHT values in the thingspeak database.
   We will now execute the telegram command send by the user
   If text == “\send” then we will send the DHT readings to the user using send_message() function
   If text == “\link” then we will send the link of thingspeak which when open will see a chart showing all the DHT readings
   If a user sends any other text or command it will be considered as invalid command and an “invalid command” message will be sent to the user.
   Previous message ID will be updated with the new one at the end for any case.

```
if text == "/send":
    send_message(mes)

elif text == "/link":
    send_message("https://api.thingspeak.com/channels/2240065/charts/1")
else :
    send_message("Invalid Command")
message_id_prev = message_id_new
```

Function to send message to user

As per Telegram Bot API Documentation, We can send a message to user using telegram api link:

https://api.telegram.org/bot6344715372:AAHS6LXDVRUohW9W2VZVolwhnI-CZ_bFXCM/sendMessage

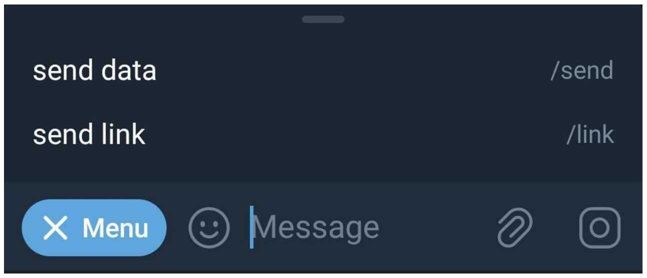


```python
def send_message(message):
    try:
        url = "https://api.telegram.org/bot{}/sendMessage".format(TOKEN)
        data = {
            "chat_id": chat_id,
            "text": message
        }
        response = urequests.post(url, json=data)
        response.close()

        if response.status_code == 200:
            json_data = response.json()
            print(json_data)
        else:
            print("Telegram API request failed with status code:", response.status_code)

    except Exception as e:
        print("Error while sending message:", str(e))
```

This Function takes one argument that is the message we wanted to send to the user.
The function first tries to create a URL for the Telegram API. The URL is made up of the base URL of the Telegram API, the bot token, and the endpoint for sending a message.
The function then creates a dictionary of data to be sent to the API. The data includes the chat ID of the recipient and the text of the message.
The function then makes a POST request to the API using the urequests library. The request includes the URL and the data.
If the response status code is 200, the function prints the JSON data from the response. Otherwise, the function prints an error message.

OUTPUT

Telegram Menu



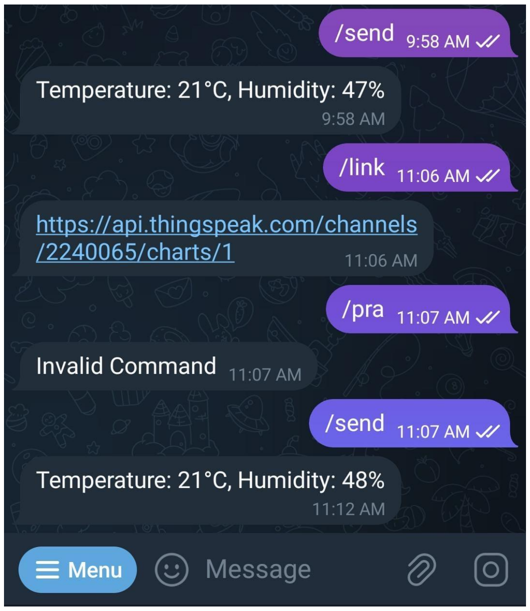
send data /send
send link /link

A menu is created using BotFather so that user can easily select the commands

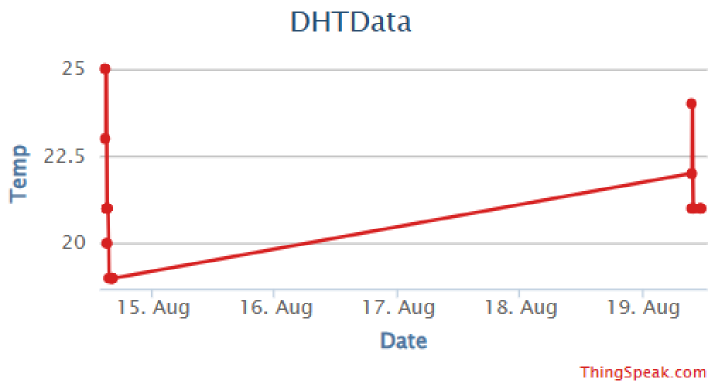
Testing all 3 cases

For a /send command ESP32 must read the value and send data only once
For a /link command We must receive a link to access ThingSpeak Chart
For any other message , we sent a message “invalid command”

When we open the ThingSpeak link we are able to see the chart created by ThingSpeak from the data we send to the database.



/send 9:58 AM
Temperature: 21°C, Humidity: 47% 9:58 AM
/link 11:06 AM
https://api.thingspeak.com/channels/2240065/charts/1 11:06 AM
/pra 11:07 AM
Invalid Command 11:07 AM
/send 11:07 AM
Temperature: 21°C, Humidity: 48% 11:12 AM


DHTData
Temp
15. Aug
16. Aug
17. Aug
18. Aug
19. Aug
Date
ThingSpeak.com