

# Une Télécommande WIFI Pypilot bâtie sur un microcontrôleur ESP32

Pourquoi une telle télécommande ?

Les télécommandes Pypilot communiquant vers un module récepteur radio 433 Mhz (généralement RXB6 sur le TinyPilot) sont pas toujours fiables (souvent bruits parasites) et surtout communiquent les ordres dans un seul sens sans retour d'acquiescement de la commande tapée sur un bouton (par exemple boutons  $-10^\circ$ ,  $-1^\circ$ ,  $+1^\circ$ ,  $+10^\circ$ )

Ces télécommandes radio ont l'avantage de consommer très peu mais peuvent être relativement imprécises mais étant intégrées au TinyPilot elle ont le mérite d'exister pour le TinyPilot....

Si on prend l'option comme moi d'utiliser un PI 5 relié directement à la carte contrôleur moteur sans TinyPilot, le plugin Pypilot dans OpenCpn qui permet de contrôler le pilote automatique en navigation n'est pas confortable car rattaché à l'écran du PI 5 qui est souvent protégé à l'intérieur. On peut toutefois utiliser une tablette pour une prise de main sur cette écran OpenCpn / Pypilot via RealVNC (voir article précédent) mais cela reste relativement imprécis en mode tactile et mieux vaut avoir une souris Bluetooth rattachée à cette tablette, dans ce cas à l'extérieur cela n'est pas trop pratique non plus.

Donc reste la solution d'une vraie télécommande de pilote automatique dialoguant avec le service Pypilot du PI 5.

**Avantages de cette solution d'une télécommande utilisant un microcontrôleur ESP32 :**

**Précision** car là on communique sur l'adresse IP fixe du PI 5 qui est point d'accès WIFI sur le port TCP 23322 du service Pypilot qui est à l'écoute des commandes tapées **mais qui peut aussi renvoyer l'acquiescement de ces commandes**. Cette communication bidirectionnelle nous garantit donc la vérification et l'acquiescement de chaque commande tapée. Selon moi pour un pilote automatique c'est non négligeable.....

**Ergonomie** avec affichage des informations de navigation les plus importantes sur un petit écran rattaché à cette télécommande (exemple : Cap demané, Cap réel, mais aussi vitesse et direction GPS : SOG COG, ou vitesse et angle vent apparent : AWS AWA)

Le seul inconvénient de cette télécommande est sa consommation basée sur une communication WIFI permanente quand on utilise Pypilot. Mais là si on utilise un module d'alimentation « intelligent » adapté à l'ESP32 on contourne ce problème et on arrive à des autonomies de quelques jours.... dans un prochain document, je ferai des mesures de consommation plus précises de l'ESP32 avec l'alimentation que je détaillerai plus loin.

## Principales fonctions de cette télécommande :

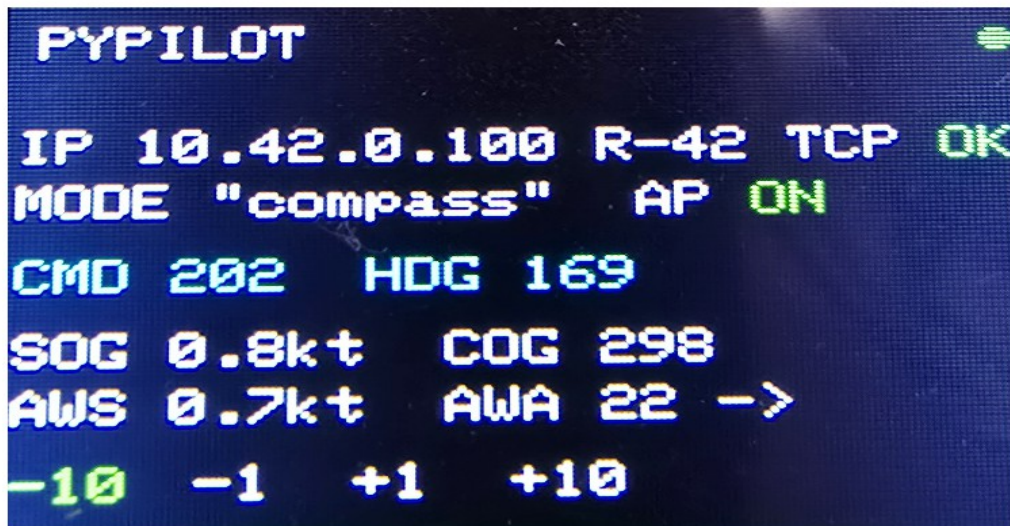
Cette télécommande possède 6 boutons de contrôle du pilote auto

Bouton Autopilot Pypilot ON

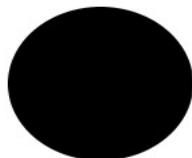
Bouton Autopilot Pypilot OFF

Si Autopilot Pypilot ON

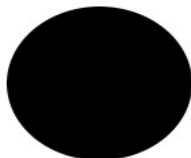
les 4 Boutons de cap : -10, -1 , +1, +10 peuvent être utilisés



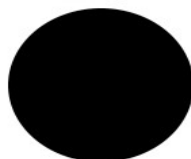
ON



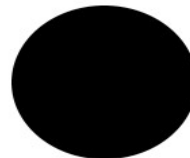
-1



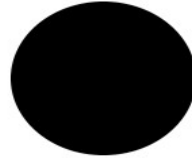
-10



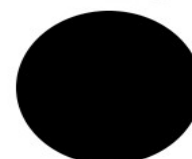
OFF



+1

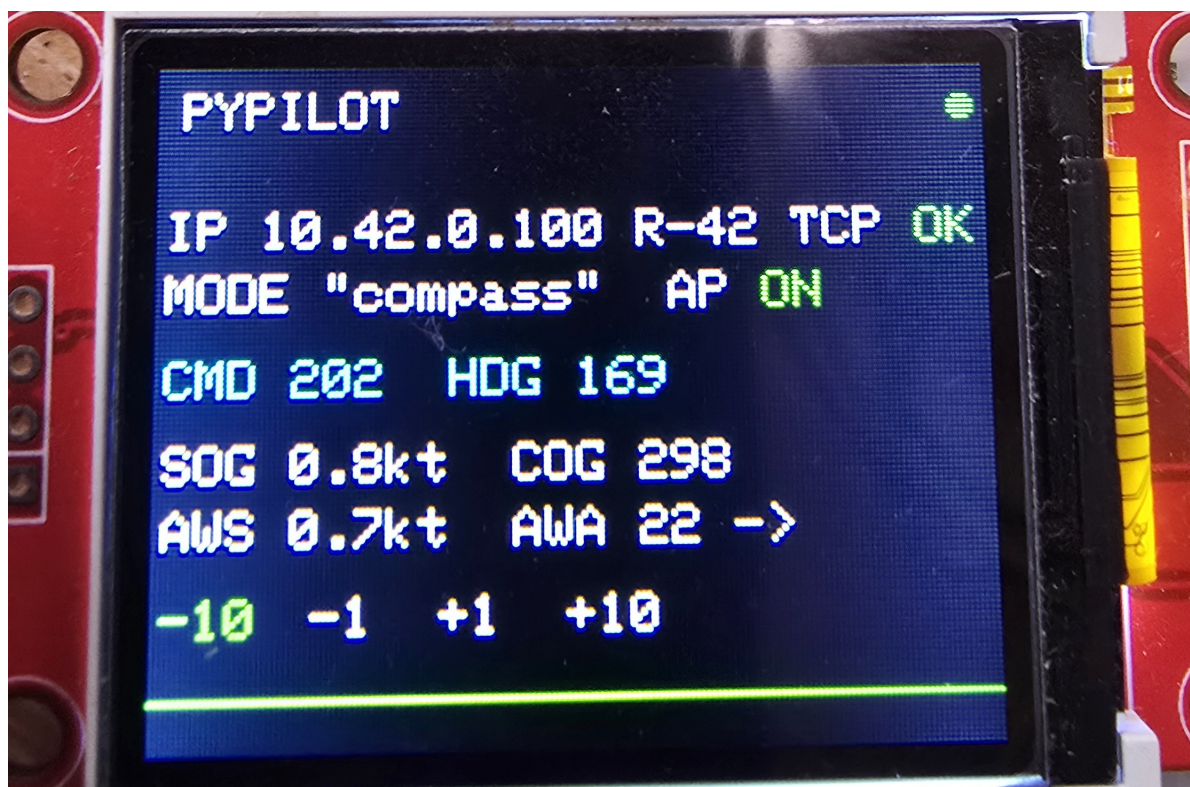


+10



Pour le moment cette télécommande est à l'état de prototype et le schéma ci dessus avec l'agencement des 6 boutons devrait correspondre au modèle définitif.  
(je rajouterai probablement un bouton mode :compas ou vent plus tard)

Explication écran de contrôle :



Petite pastille verte en haut à droite clignotante indique transmission de data (émission ou réception)

Ligne 1: statut connexion adresse ip : adresse ip de l'esp32, R : qualité signal RSSI (-30 à -50: Excellent, -50 à -60: Très bon, -60 à -60: Bon, -67 à -70: Acceptable) TCP liaison TCP sur le port 23322 de Pypiloty OK (en vert) ou KO (en rouge)

Ligne 2: MODE (là toujours par défaut compas) et AP AutoPilot **ON** ou **OFF** (si appui sur bouton on ou off)

Ligne 3 en bleu: CMD cap demandé au pilote, HDG cap réel

Ligne 4: SOG et COG vitesse et cap GPS

Ligne 6: AWS et AWA vitesse et angle vent apparent (pour AWA angle à bâbord = <-, tribord=>)

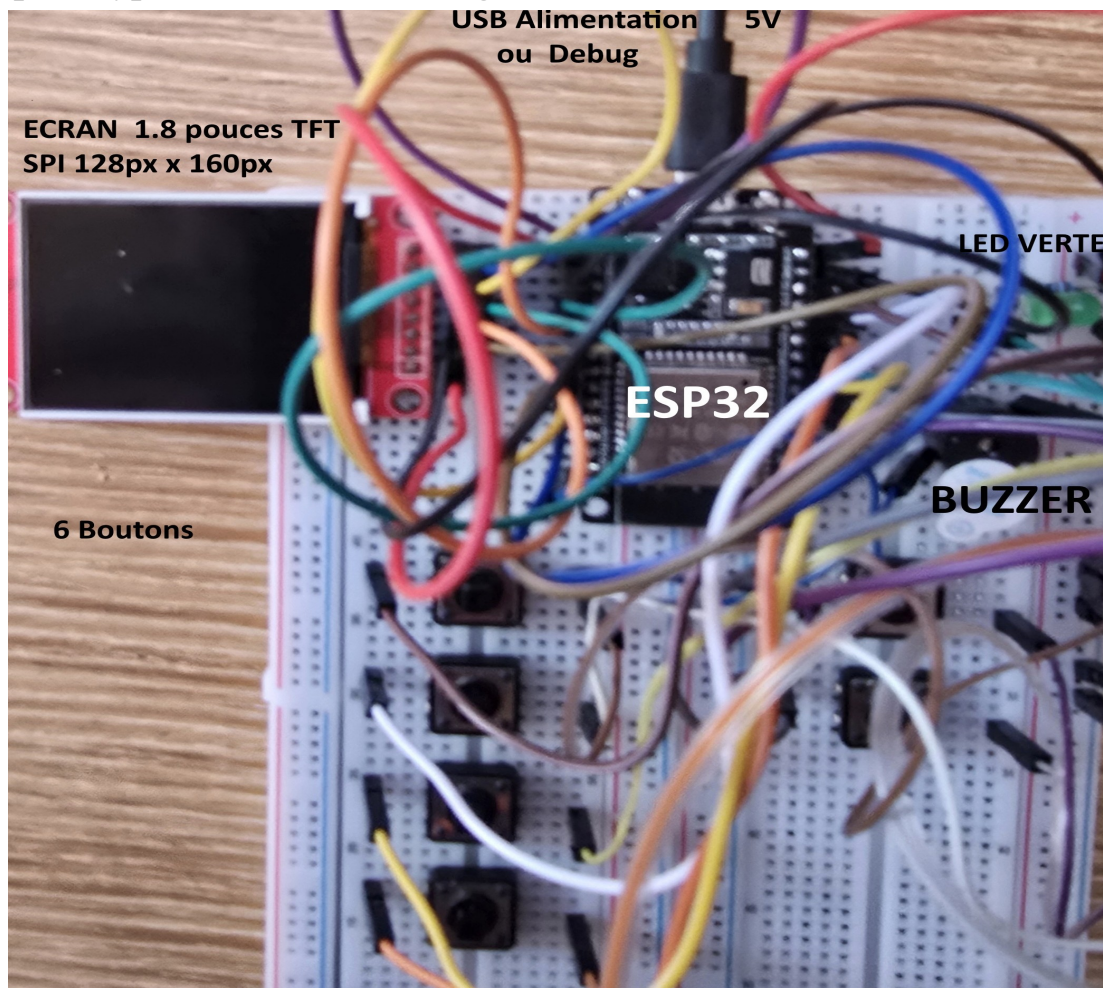
Ligne 7: dernière commande de cap appuyée sur bouton en vert -10 -1 +1 +10

On voit là que cette télécommande présente l'énorme avantage d'afficher la confirmation de la prise en compte de chaque commande AP **ON** ou **OFF** et le dernier bouton de cap pris en compte par Pypilot affiché en vert exemple ici **-10**

De plus a chaque appui un beep de confirmation de commande sonne. Pour AP ON : un beep long et led verte allumée, Pour AP OFF un beep et led verte éteinte, Pour boutons de cap un beep court.



## Le prototype et schéma de câblage



Une fois les boutons étanches enfermés dans une boîte IP 67 avec écran et bloc d'alimentation adapté, cela ressemblera à une vraie télécommande (voir schéma de la télécommande pages précédentes)

Le schéma est très simple (je fournirai ultérieurement un schéma kickad)

**LED AP ON** ----> **GPIO 26** (avec une résistance 320 ohms)

**BUZZER** ----> **GPIO 23**

**BOUTON ON** ----> **GPIO 33**

**BOUTON OFF** ----> **GPIO 32**

**BOUTON -10** ----> **GPIO 13**

**BOUTON -1** ----> **GPIO 14**

**BOUTON +10** ----> **GPIO 25**

**BOUTON +1** ----> **GPIO 27**

Ce schéma étant très simple on peut éventuellement se passer d'un circuit imprimé en raccordant directement chaque fils à chaque PIN GPIO de l'ESP32 et à un GND commun

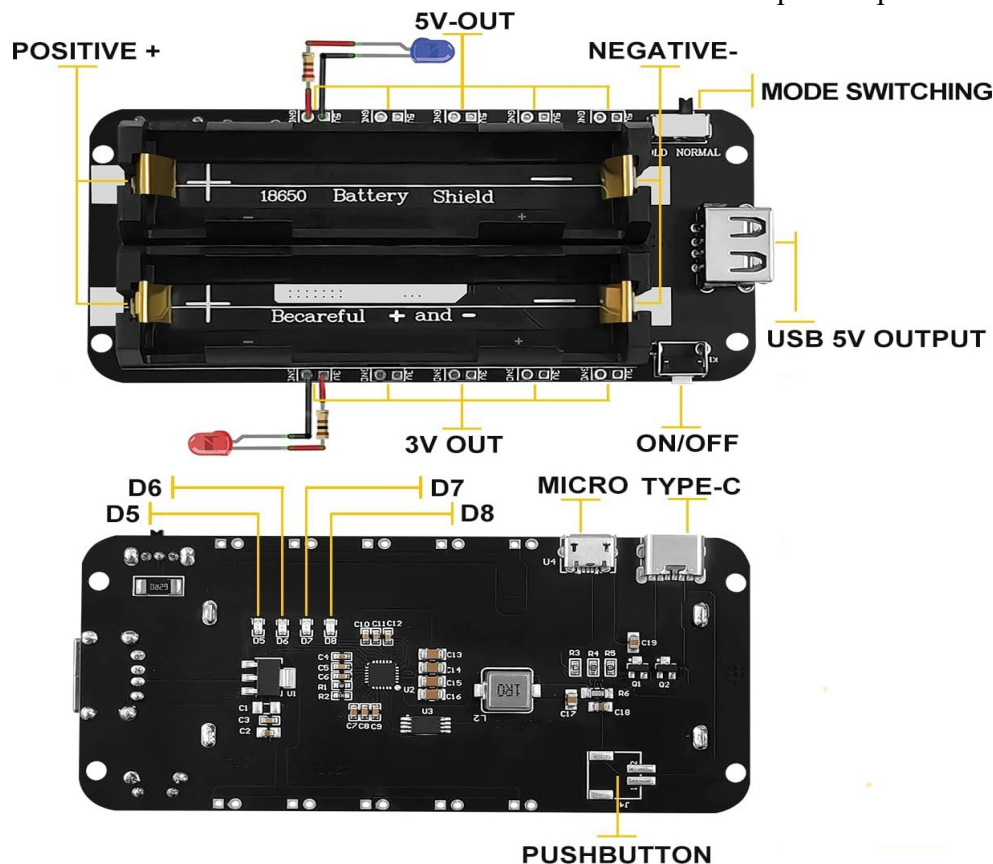
## Alimentation

Bloc d'alimentation régulé 5V avec des batteries rechargeables Li-ion type 18650 (normalement 1000 cycles de décharge)



J'effectuerai des tests de durée de ces batteries en longue navigation. Je possède un autre jeu de 2 batteries identiques pour pouvoir éventuellement en charger un jeu en parallèle.

Détails de cette alimentation VooGenezk Mediatek MT8125 que l'on peut trouver sur amazon.fr:



When the power is 25%, D5 lights red;  
When the power is 50%, D5 and D6 will be red;  
When the power is 75%, D5, D6, D7 will be red.  
When the power is 100%, D5, D6, D7, D8 will be red.



## Sources

J'utilise pour l'ESP32 l'environnement de développement standard **Arduino IDE** avec les librairies ESP32 et ses librairies Esp23 standard : WiFi.h, WiFiClient.h et pour l'écran : SPI.h, Adafruit\_GFX.h, Adafruit\_ST7735.h

[Lien vers les sources TelPypilotEsp32.ino dans mon GoogleDrive](#)

### Sources bientôt disponibles dans Github...

Pour le moment étant en phase de tests, pour adapter cette télécommande au point d'accès WIFI de votre PI 5 il suffit de modifier les paramètres au debut du source : WIFI\_SSID, WIFI\_PASSWORD, PYPILOT\_HOST (prochainement fichier de paramètres séparé)

La télécommande définitive « étanche » réalisée avec les moyens du bord sur mon voilier :



A noter la Led verte sur côté droit du mini écran LED qui est allumée quand le Pypilot est sur ON

## Conclusion

Ma philosophie pour pypilot se résume ainsi avec une configuration propre **HORS Tinypilot** :  
Un pi5 suffisamment puissant pour héberger le service **pypilot installé proprement via son script d'installation** (donc sans toute cette couche OpenPlotter qui masque pas mal de choses...)  
+ open cpn et son plugin pypilot, éventuellement signalk exclusivement à l'écoute sur les ports tcp de pypilot 20220 23322 et gpsd 2947 pour restituer des jolis écrans via ses pages web locales sur tablette via kip  
+ une télécommande digne de ce nom donc basée sur un esp32 bien adaptée au fonctionnement du service pypilot sur le pi5.

Parti depuis 4 ans en balade sur son voilier, c'est ma vision des choses de vieux développeur mais je ne détiens pas la vérité, cette configuration est une configuration PyPilot parmi tant d'autres tant cette solution est ouverte et surtout Open Source....