Sergi Mariano Rosales

DISSENY I IMPLEMENTACIÓ D'UN SISTEMA DE CONTROL REMOT PER A EMBARCACIONS AMB ESP32

TREBALL DE FI DE GRAU

dirigit pel Sr. Enrique Fernando Cantó Navarro

Grau de Enginyeria Electrònica Industrial i Automàtica

UNIVERSITAT ROVIRA I VIRGILI

Tarragona
2024

Resum

El projecte es basa en dissenyar, programar, muntar i provar un sistema en el qual es pugui tenir un control remot de qualsevol embarcació. La idea es que l'usuari pugui tenir accés a una sèrie de dades i controls ja siguin de seguretat (estat de bateries, aigua a la sentina, vent, fum a la sala de màquines, etc.) com de confort (control dels aires condicionats, hèlix de proa i/o popa, llums, etc.).

L'objectiu del projecte és que sigui modulable i es pugui adaptar a les necessitats de cada client o del tipus d'embarcació.

Resumen

El proyecto se basa en el diseño, programación, montaje y pruebas de un sistema en el cual se pueda tener un control remoto de cualquier embarcación. La idea es que el usuario pueda tener acceso a una serie de datos y controles ya sean de seguridad (estado de las baterías, agua en la sentina, viento, humo en la sala de máquinas, etc.) como de comfort (control de los aires acondicionados, hélice de proa i/o popa, luces, etc.)

El objetivo de este proyecto es que sea modulable y se pueda adaptar a las necesidades de cada cliente o tipo de barco.

Abstract

This project is based on the design, programming, assembly and testing a system that allows remote control of any boat. The idea is that the user can have access to data and controls, whether for safety (battery status, water in the bilge, wind, smoke in the engine room, etc.) or comfort (control of air conditioning, bow and/or stern thruster, lights, etc.)

The objective of this project is to be modular and adaptable to the needs of each client or type of boat.

ÍNDEX

ÍNDEX D'IL·LUSTRACIONS

ÍNDEX DE CODIS

ÍNDEX D'ESQUEMES

1 INTRODUCCIÓ

1.1 Antecedents i motivació

Soc de la Ràpita, un poble costaner en un entorn privilegiat al delta de l'Ebre. Des de petit, la meva vida ha estat profundament connectada amb el mar; als 8 anys, acompanyava al meu pare amb la seva llanxa, que feia de taxi pel delta. Als 14, vaig començar a practicar rem olímpic on vaig arribar a participar a un mundial a Lituània al 2017. A més, tots els treballs que he tingut també han estat vinculats amb l'àmbit marítim: he treballat al Club Nàutic, a una empresa de lloguer d'embarcacions, als Remolcadors de Barcelona i, actualment, a Tallers Cornet, un taller naval.

La decisió d'estudiar enginyeria electrònica va sortir del meu desig de poder aplicar aquests coneixements al mon naval. Fa uns anys, els vaixells portaven una mecànica molt bàsica, però avui en dia, els sistemes electrònics han adquirit una importància fonamental.

La motivació per desenvolupar aquest projecte sorgeix d'una experiència personal. El meu pare i jo fèiem el manteniment a un iot motora model *Princess 45* de 15 metres d'eslora. Durant una temporada, ens vam trobar que cada cop que anàvem ens trobàvem el diferencial elèctric baixat. Això generava un greu inconvenient ja que al desconnectar-se de la toma de terra les bateries es deixaven de carregar i quan es volia engegar els motors per sortir a navegar no es podia per baix nivell de bateries. Mentre no es trobava el problema, vaig pensar una solució ràpida que va ser utilitzar una placa d'Arduino per a connectar-la al sistema elèctric de 220 Vac del IoT de forma que, utilitzant la aplicació de Blynk, rebés una notificació cada cop que es perdés la connexió, fet que indicaria que ha saltat la llum. Aquest sistema, tot i que no era pràctic ja que havies d'anar cada cop a l'embarcació per tornar a connectar la llum, va resultar ser molt efectiu, fet que va cridar l'atenció al propietari. Aquest, em va demanar si es podria expandir la idea per a controlar altres aspectes del iot de forma remota. Aquesta sol·licitud va ser el desencadenant que va fer que es pogués desenvolupar un sistema més complert i avançat.

1.2 Objectius

L'objectiu principal d'aquest projecte es desenvolupar i implementar un sistema integral per a controlar i monitoritzar de forma remota diversos aspectes d'una embarcació de la forma més econòmica possible. Aquest objectius es poden desglossar de la següent forma:

1. Aconseguir una **connexió WiFi estable, còmoda per a l'usuari i funcional** per a poder garantir el correcte funcionament de tot el sistema.
2. Crear **un *dashboard* amb el software de Blynk** on l'usuari pugui tenir la millor experiència possible.
3. Crear un **sistema modulable** on es pugui adaptar a diferents necessitats de possibles clients i models d'embarcacions.
4. Poder **dissenyar, programar, muntar i provar** diferents aspectes com:
    a. Control de l'estat de les **bateries**.
    b. Control de la **temperatura** per poder utilitzar-ho en temperatura ambient, sala de màquines, equip electrònic, etc.
    c. Control de les **hèlix de proa i/o popa**.
    d. Comptatge de les **hores dels motors** amb avisos cada certes hores preestablertes per millorar el manteniment.

    e. Control de presència d'**aigua a les sentines** amb avisos per poder fer actuacions preventives.
    f. Control dels **aires condicionats** de bord, amb control automàtic i manual.
5. Fer el tot el sistema **el més econòmic possible** degut als escassos recursos econòmics dels que es disposen i pensant en que sigui el més rentable possible per a una possible venta.
6. **Valoració final i propostes de millora.**
7. **Valoració de les propostes de millora**: perquè son millors i quins problemes hi poden haver.

2 DESCRIPCIÓ DEL SISTEMA

2.1 Anatomia d'una embarcació i principis de navegació

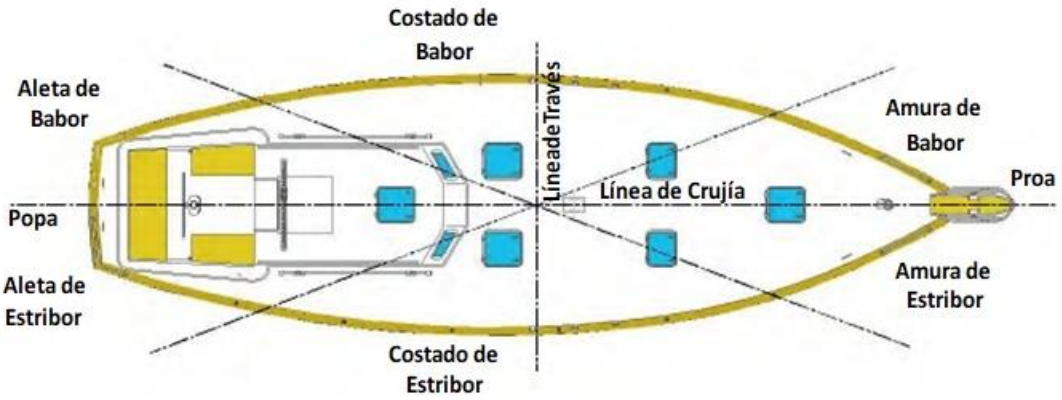
Per entendre aquest projecte és essencial que es tingui un coneixement bàsic de l'estructura d'una embarcació i les seves parts principals.

A continuació es descriuran les parts més bàsiques d'una embarcació. No s'endinsarà en molta profunditat, només s'explicaran aquelles parts que sigui interessant conèixer per poder entendre el conjunt del projecte.

2.1.1 Nomenclatura bàsica

La nomenclatura nàutica difereix significativament de la utilitzada a terra. En l'àmbit marítim, no es fan servir termes com ara dreta i esquerra, davant i darrere, corda o costat; al seu lloc, es fa servir un vocabulari específic. A continuació, es descriuen algunes de les parts bàsiques que s'esmentaran al llarg d'aquest projecte.

- **PROA**: Part davantera del vaixell.
- **POPA**: Part del darrere del vaixell.
- **BABOR**: Part de l'esquerra del vaixell (mirant a la proa).
- **ESTRIBOR**: Part de la dreta del vaixell (mirant a la proa).
- **CASC**: Estructura principal.
- **TIMÓ**: Dispositiu utilitzat per canviar la direcció.
- **AMURA**: Part del costat de davant.
- **ALETA**: Part del costat del darrere.
- **CAP**: Corda o soga utilitzada per amarrar o maniobrar.
- **ESLORA**: Mida del vaixell des del punt més a proa fins al punt més a popa.
- **MANGA:** Mida del vaixell des del punt mes a babor fins al punt mes a estribor.



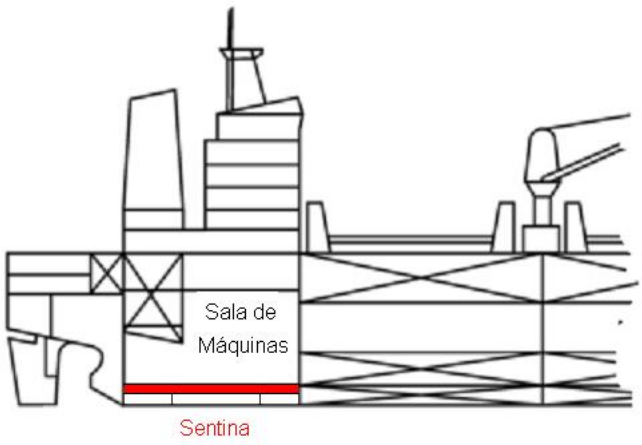
Il·lustració 1. Descripció gràfica de les parts d'un vaixell (Obarti, 2020)

2.1.2 Sentines

Les sentines son la part més baixa de la sala de màquines. La seva funció principal és recollir tots els líquids que es puguin acumular, ja sigui entrada d'aigua salada, pèrdua d'oli dels motors, pèrdua d'aigua dolça d'alguna bomba o junta, entre d'altres.

En aquest punt, normalment hi sol anar instal·lada una bomba de 12 V amb sensor que expulsa immediatament l'aigua quan es detecta la seva presencia. En condicions normals, no hi hauria d'haver aigua a la sentina o només una quantitat mínima. Per això, és molt recomanable instal·lar també algun indicador lluminós o acústic que alerti quan s'estigui expulsant aigua ja que pot ser indicador de que hi ha algun problema.

Aquest sistema és vital per a la seguretat del vaixell, ja que durant la navegació, el trencament d'una canonada, un forat al casc o algun altre problema podria causar una via d'aigua. Si no es reacciona a temps, això podria portar a l'enfonsament de l'embarcació. Per tant, un sistema de monitorització eficient a les sentines és essencial per prevenir emergències i garantir una navegació segura.



Il·lustració 2. Esquema bàsic de la ubicació de les sentines al vaixell. (Wikipedia, Sentina, 2021)

Generalment, l'entrada d'aigua a la sentina només s'indica amb algun led o alarma sonora encara que moltes embarcacions no tenen ni alarma. Això no és un problema si es viu tot l'any al vaixell, ja sigui per motius professionals, com en el cas d'un remolcador, o perquè els propietaris hi resideixen (encara que hi segueix sent recomanable). Tanmateix, això només representa un percentatge mínim, ja que la majoria dels vaixells estan amarrats en algun port sense ningú a dintre per sentir les alarmes.

Per això, és molt important poder tenir la tranquil·litat de rebre una notificació al telèfon mòbil en cas que hi hagi aigua a la sentina independentment de on estigui el propietari per poder avisar de forma immediata al club nàutic o a la marina on hi estigui atracat poder actuar a temps.

2.1.3 Sistema elèctric

El sistema elèctric pot variar molt depenent la mida de l'embarcació, any de construcció, tipus, manteniment, etc.
A continuació es descriuran els sistemes elèctrics típic que s'haurien de trobar, tot i que no sempre és així.

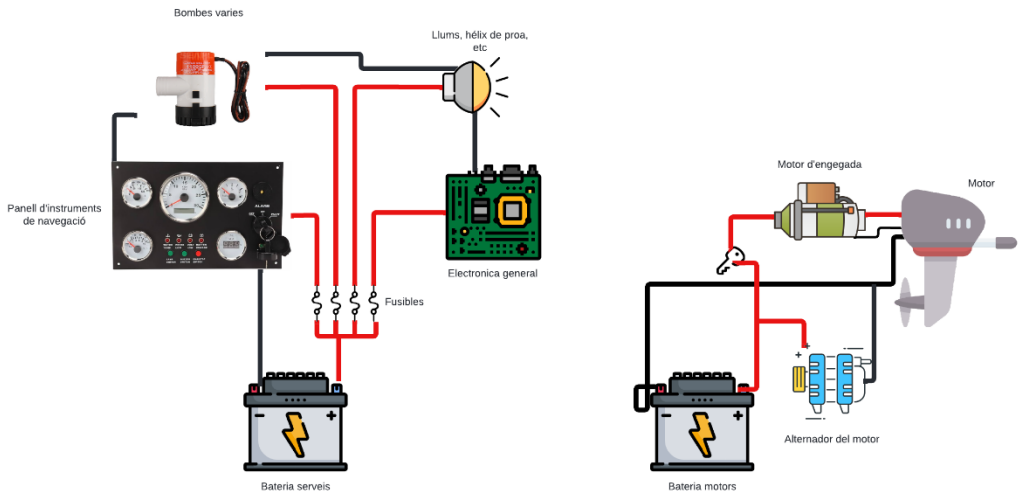
2.1.3.1 Sistema de corrent continu (12 V/24 V DC)

El sistema elèctric de corrent continu és el sistema d'alimentació més important de l'embarcació; si aquest falla, tot el vaixell queda inoperatiu. Segons la mida de l'embarcació, podem trobar des d'un únic bloc de bateries fins a diversos blocs separats: un per als motors i un altre per als serveis essencials. Fins i tot, en algunes embarcacions, hi pot haver un tercer bloc de bateries AGM dedicat exclusivament a les hèlix de proa i/o popa ja que aquestes estan mes preparades per a suportar grans quantitats de càrrega.
Generalment a totes les embarcacions on el motor s'hagi d'engegar amb el motor d'engegada podrem trobar una o un bloc de bateries de motors. Aquestes bateries són exclusivament per engegar el motor o els motors. Els motors d'engegada dels motors principals funcionen a 12 V o a 24 V depenent del model i és per això que es necessiten aquestes bateries.

També hauríem de trobar una altra bateria o bloc de bateries per a serveis. Aquestes estan més dedicades a tots els instruments de navegació, bombes d'aigua, algunes llums, etc.

Es possible que hi hagi cops que ens trobem que aquestes dues bateries no estan diferenciades i les de motor també alimenten als serveis, sobretot per a embarcacions de menys eslora o antigues. Aquesta pràctica no es recomanable ja que si s'està fondejat a algun lloc sense estar connectat i s'utilitzen alguns serveis com les llums, aquestes podrien fer esgotar la bateria i podria fer que no es pogués engegar els motors per tornar a port.

Un esquema genèric que ens podríem trobar en un vaixell tipus seria el següent:



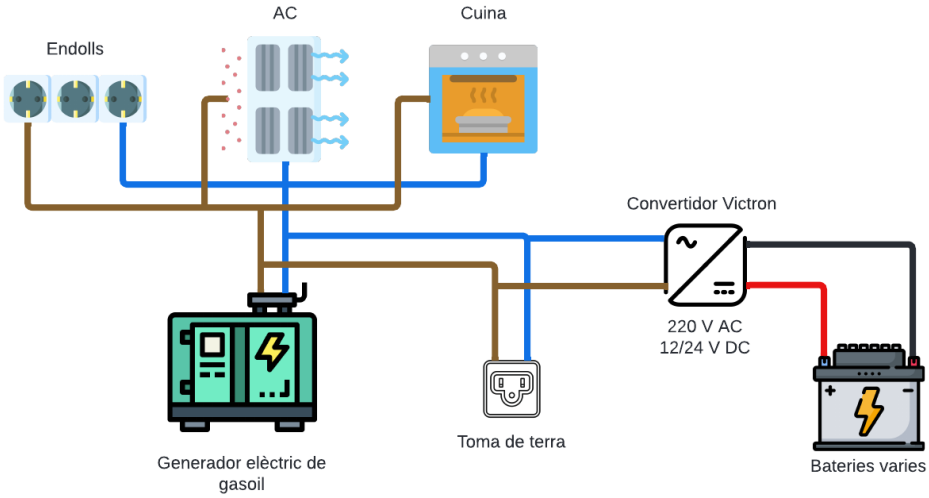
Il·lustració 3. Esquema elèctric genèric de la instal·lació de 12 V d'una embarcació.

Per als vaixells que enlloc de tenir un subministrament de 12 V aquest és de 24 V es necessiten un mínim de dues bateries en sèrie de 12 V cada una.

2.1.3.2 Sistema de corrent Altern (220 V AC)

El quadre elèctric de 220 V no és obligatori i no tots els vaixells disposen d'aquest. L'esquema de la Il·lustració 4 és només un exemple genèric i de l'embarcació on s'han fet les proves.

Aquesta, disposa d'un endoll que connecta a terra (pantalà del amarratge) i mentre estigui al club es disposarà de tot el sistema de 220 V, es a dir, es podran connectar els aires condicionats, es podrà veure la TV, es podrà cuinar, etc. Si es surt a navegar i es vol disposar de tots aquests serveis s'ha d'engegar un generador que funciona amb el gasoil dels tancs de gasoil dels motors principals. Aquest sistema també manté les bateries sempre carregant i evita que quan es va a engegar l'embarcació no es pugui per baix voltatge de bateries. És per això que és molt important saber si en algun moment les bateries es deixen de carregar ja sigui perquè han saltat els diferencials de la torreta del pantalà, perquè s'ha espatllat el carregador, perquè s'ha anat la llum, etc.



Il·lustració 4. Esquema elèctric genèric de la instal·lació de 220 V d'una embarcació.

2.1.4 Hèlix de proa/popa

Les hèlix del motor o motors principals són les que donen la propulsió al vaixell i, per tant, la potència. Tot i això, molts d'aquests també disposen d'alguna o algunes hèlix complementaries que es solen posar a proa, generalment, i a popa.

Aquestes hèlix no funcionen amb motor de combustió sinó que ho fan en un motor elèctric que pot funcionar a 12 o 24 V de corrent contínua. Tenen una potència molt menor i només serveixen per al moment de la maniobra.

Els vaixells molt grans (>50 metres d'eslora) degut a que la potència proporcionada per les bateries no és suficient per a moure tanta massa, a més de que disposen de més espai i de que la diferència de preu no sol afectar tant, aquestes hèlix funcionen amb motors de combustió interna.

Il·lustració 5. Hélix de proa d'una embarcació (Wikipedia, Hélice de maniobra, 2023)

En la Il·lustració 5 es pot veure un vaixell de la mercant amb tres hèlix de proa. Com aquest vaixell és de entre 200 i 300 metres d'eslora, no només necessita tres hèlix enlloc d'una sinó que aquestes sí que son propulsades en motors de combustió per a una major potència de maniobra.

2.2 Arquitectura del sistema

En general, tots els sistemes intel·ligents poden dividir la seva arquitectura en tres grans blocs: un microcontrolador central, els sensors i actuadors i la interfície d'usuari. A continuació els analitzarem:

2.2.1 ESP32

2.2.1.1 Característiques tècniques

El microcontrolador central es un microcontrolador ESP32. Les avantatges d'aquest microcontrolador son (Espressif, 2024) la seva capacitat de processament, una connectivitat WiFi i Bluetooth i les diferents entrades i sortides per a poder connectar els diferents sensors i actuadors, així com el seu baix cost i alta fiabilitat. També, i en relació al projecte, s'ha tingut en compte la seva compatibilitat amb Blynk IoT i la seva resistència a entorns hostils degut a la alta concentració de sal i humitat.

L'ESP32 és un microcontrolador de doble nucli que inclou un processador Tensilica Xtensa LX6, amb una freqüència de fins a 240 MHz, cosa que li permet manejar tasques complexes i realitzar processament en temps real. A més, l'ESP32 compta amb:

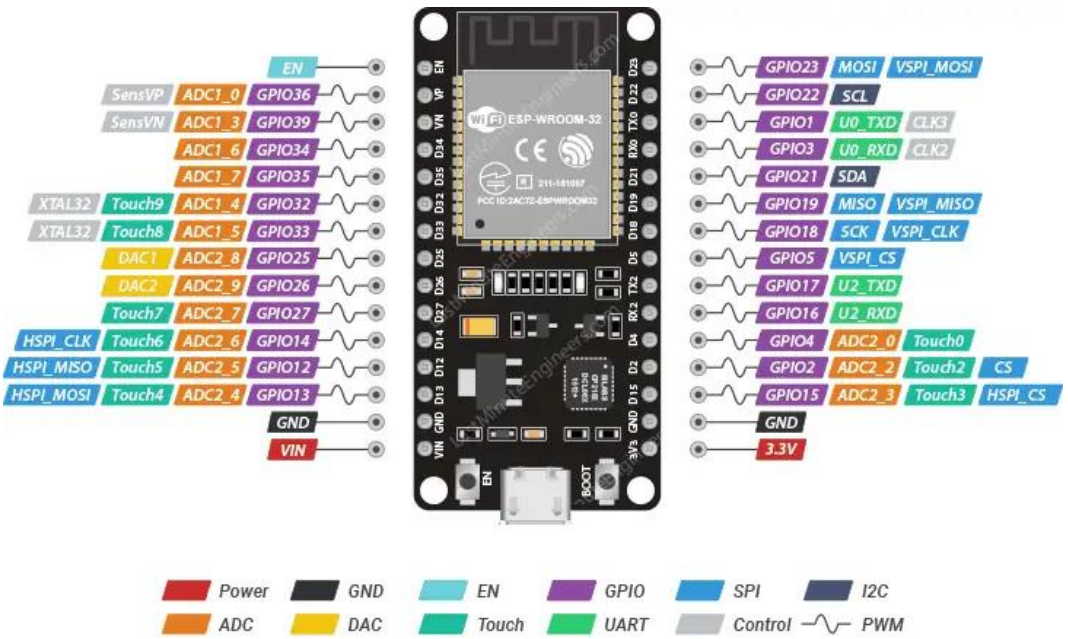
- **Connectivitat Sense fil**: Integra capacitats de Wi-Fi i Bluetooth, incloent Bluetooth Low Energy (BLE), la qual cosa el converteix en una opció ideal per a aplicacions IoT que requereixen comunicació sense fil fiable i de baix consum.
- **Múltiples interfícies**: L'ESP32 suporta una àmplia gamma d'interfícies, com UART, SPI, I2C, i PWM, cosa que facilita la connexió amb diversos sensors i actuadors, i permet la seva integració en sistemes més complexos.
- **Capacitat de Processament**: Amb una memòria RAM de 520 KB i opcions d'emmagatzematge extern a través de SPI Flash, l'ESP32 pot gestionar aplicacions que requereixen processament intensiu de dades i emmagatzematge.

- **Eficiència Energètica**: L'ESP32 és conegut per la seva capacitat d'operar en modes de baixa potència, cosa que és crucial en aplicacions on l'eficiència energètica és prioritària, com en dispositius alimentats per bateria.
- **Seguretat Integrada**: El microcontrolador inclou característiques de seguretat avançades, com ara encriptació de maquinari, arrencada segura, i gestió de claus, cosa que garanteix la protecció de les dades i la integritat del sistema.

2.2.1.2 Pinout

El microcontrolador ESP32 disposa d'una àmplia gama de pins funcionals que permeten la connexió amb una gran varietat de perifèrics i sensors. Alguns dels més importants son:

- **Alimentació**: Disposa de 2 pins de GND per utilitzar com a voltatge de referència (terra); 1 pin 3,3V que dona una sortida estable de 3,3 V per alimentar sensors i mòduls externs; 1 pin VIN que s'utilitza per alimentar la placa amb una bateria externa amb un voltatge de 5 V a 12 V.
- **39 pins GPIO**: Aquests pins poden configurar-se com a entrades o sortides digitals, tot i que del pin GPIO34 al pin GPIO39 només poden utilitzar-se com a entrada.
- **ADC**: Aquests pins permeten la conversió de senyals analògiques (sensors de temperatura, potenciòmetres, sensors de bateries, etc.) en digitals. El rang de voltatge va de 0 V a 3,3 V. La resolució és de 12 bits, el que significa que 2^{12} = 4096 i, per tant, el valor digital pot variar entre 0 i 4095. Aquest rang dona una mesura molt precisa. Encara que es conta en dos canals ADC, és important saber que el canal ADC2 no es pot utilitzar mentre es fa ús del WiFi.



Il·lustració 6. Pinout de la placa ESP32 utilitzada (Engeneers, 2024)

2.2.2 Sensors i Actuadors

L'objectiu d'aquest projecte es que sigui molt adaptable a diferents embarcacions i necessitats. És per això que, a diferencia dels altres, aquest bloc podrà variar molt de components depenent de la finalitat del projecte.
En general, el sistema inclourà una varietat de sensors i actuadors que permetran la monitorització i el control de diferents aspectes de la embarcació:

- **Sensors de temperatura i humitat**: Claus per a la supervisió de les condicions ambientals i poder controlar els sistemes com els aires condicionats, sobreescalfament dels motors o d'algunes peces susceptibles d'espatllar-se amb altes temperatures generalment ubicades a la sala de màquines.
- **Sensors de nivells de bateries**: Una embarcació, sobretot a l'hivern, tendeix a estar molts de mesos parada. Tot i això, les bateries solen estar connectades amb un cable a les torres de llum dels pantalans i aquesta llum les carrega i manté el 220 V funcionant com les neveres, aires condicionats, etc. Saber en tot moment l'estat de les bateries permet una actuació ràpida i no arribar el dia que vols sortir a navegar i no pots engegar els motors.
- **Sensors d'aigua a les sentines**: Les sentines és el punt més delicat d'una embarcació. Es el punt més baix que hi ha i, per tant, el que es troba a baix de l'aigua. A més, és on hi ha els motors i els eixos que foraden el casc per arribar a les hèlix. És per això que saber en tot moment que les sentines estan seques o tenir una notificació instantània de quan no ho estan pot ser la diferència entre que la embarcació s'enfonsi o no.
- **Sensors de fum, gasos, CO2, etc.**: Aquests sensors són molt útils per a la seguretat. Poder tenir una alarma per si hi ha una mala combustió i reaccionar a temps pot ser vital en una embarcació on hi ha, de mitja, 500 litres de combustible inflamable.
- **Actuadors de relé**: Permetran utilitzar les sortides de molt baixa potència del microcontrolador i activar motors i bombes de més alta potència.

2.2.3 Interfície d'usuari remota

La interfície d'usuari remota està implementada amb l'aplicació Blynk, que permet als usuaris controlar tots els aspectes nombrats anteriorment en temps real des de qualsevol lloc del mon. L'aplicació proporciona una interfície gràfica intuïtiva i permet rebre alertes i notificacions.

Blynk IoT es una plataforma d'internet de les coses on es pot crear aplicacions per a controlar i monitoritzar hardware des de un telèfon mòbil o un panel de control web. Blynk IoT funciona en una àmplia gama de plataformes de hardware, com Arduino, Raspberry Pi, ESP8266, ESP32, entre d'altres, el que permet ser molt versàtil i adaptar-se a diferents projectes.

La plataforma permet als usuaris arrossegar i deixar anar "widgets" per crear interfícies personalitzades, establir regles d'automatització i rebre notificacions en temps real i anàlisi de dades. A més, Blynk IoT ofereix connectivitat al núvol, cosa que permet accedir i gestionar els dispositius de forma remota des de qualsevol lloc del món.

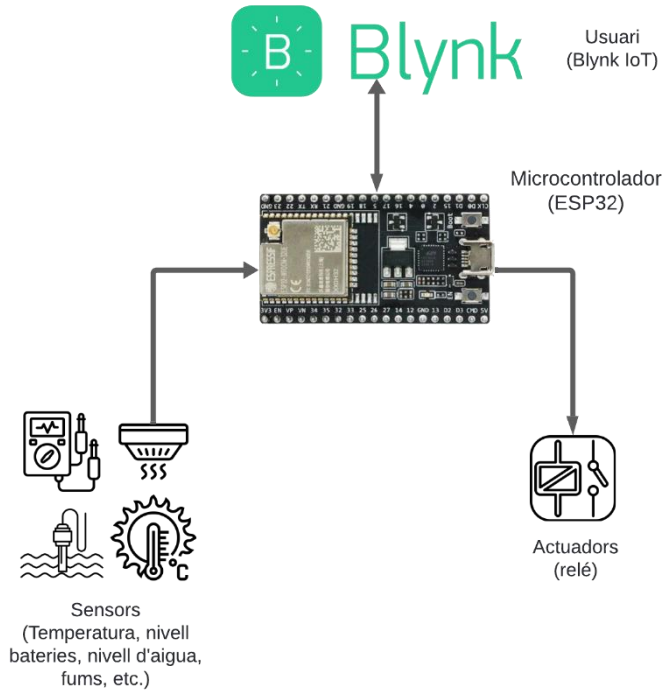
Actualment Blynk IoT disposa de quatre opcions de pagament:

Pla	Preu/mes	Dispositus Suportats	Funcionalitats Destacades
Gratis	0 $	2	"Widgets" bàsics, notificacions limitades
Maker	6.99 $	10	Tots els "widgets", mes notificacions
Pro	49.99 % / 99.99 $	40 / 100	Panell personalitzable, històric de dades
Enterprise	+400 $	il·limitats	Integracions avançades, suport dedicat, reports detallats

Taula 1. Comparativa plans de subscripció Blynk. (Blynk, 2024)

De moment s'utilitza l'opció gratuïta ja que com només es treballarà en un client només es necessitarà un dispositiu, i amb els "widgets" bàsics és més que suficient per poder cobrir totes les necessitats del projecte.

2.2.4 Diagrama de l'arquitectura



Il·lustració 7. Diagrama final

El diagrama de l'arquitectura es centra en el microcontrolador ESP32. Aquest llegeix totes les entrades dels sensors i actua amb els actuadors. Mentrestant, va enviant i rebent les dades mitjançant l'aplicació de Blynk.

3 IMPLEMENTACIÓ

3.1 Configuració del hardware

L'objectiu que es busca en tot moment és trobar la màxima funcionalitat al mínim preu possible. També s'ha de buscar que es pugui implementar tot el que s'ha pensat teòricament a la pràctica i que no hi hagi cap incompatibilitat dels materials o dificultats per passar el cablejat.

La selecció de components i el seu posterior cablejat és un pas molt important, sobretot a nivell econòmic ja que és el punt que si no es pensa bé es poden espatllar components alguns d'ells relativament cars. Aquest procés implica escollir adequadament cada element per a garantir la seva compatibilitat amb l'ESP32, la seva durabilitat en un entorn humit i amb sal com és la mar i la capacitat de que compleixi els requisits que es demanen.

Per al microcontrolador es va començar escollint una placa Arduino MKR WiFi 1010, aquesta es va sobreescalfar a causa d'un codi poc eficient i es va comprar una altra de nova millorant el codi i aplicant un producte per a protegir la electrònica. Després d'un temps d'un funcionament prou correcte es va decidir canviar per la placa actual, un ESP32. Aquesta decisió es va dur a terme degut a l'alt cost de la placa Arduino (uns 40 € aproximadament) en relació a la ESP32 (uns 3 €) i de les característiques tècniques. A continuació es mostrarà una taula comparativa amb aquestes característiques.

	Arduino MKR WIFI 1010	**ESP32**
Microcontrolador [bits]	32	32
Velocitat CPU [MHz]	48	**240**
RAM [kB]	32	**520**
Flash [kB]	256	**4000**
WiFI [b/g/n]	IEEE 802.11	IEEE 802.11
Voltatge [V]	3,3	3,3
Nombre de pins digitals	8	**36**
nombre d'ADC	7	**18** (9 amb WiFi)
nombre de DAC	1	**2**
Preu [€]	40	**3**

Taula 2. Taula comparativa entre Arduino MKR WiFi 1010 i ESP32. (Blynk, 2024) (Arduino, 2024)

Com s'observa a la taula anterior, es pot veure un clar guanyador entre el Arduino i el ESP32 ja que o bé tenen les mateixes característiques en punts com els bits del microcontrolador, la tecnologia WiFi o el voltatge d'operació o l'ESP32 té molt millors característiques en tota la resta.

3.1.1 Sensor de temperatura

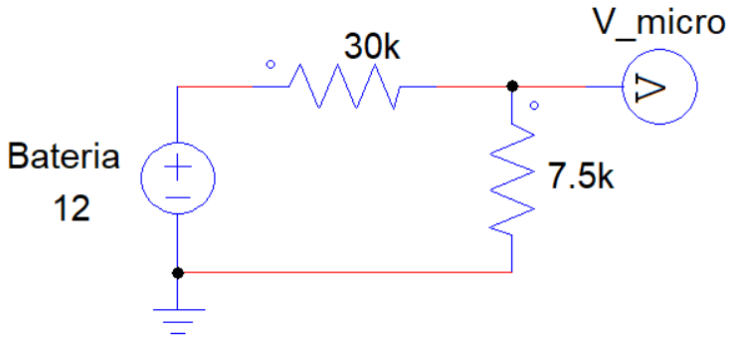
Els sensors de temperatura son uns sensors molt versàtils ja que permeten tenir un control de la temperatura de diferents espais per a diferents propòsits. Per exemple, es pot col·locar un sensor a les habitacions per al control dels aires condicionats, a la sala de màquines per al control de la temperatura dels motors, al mateix microcontrolador per controlar els ventiladors de refrigeració, etc.

Per controlar els aires condicionats del iot es van seleccionar sensors de temperatura com l'LM35 i dos relé per a cada AC: un de maniobra i un de potència. Això es degut a que l'únic relé que es va trobar que funcionés a 3,3 V tenia una intensitat màxima de 10 A. Es va mirar el consum dels AC i es va veure que en règim nominal aquests tenien un consum de 8 A amb pics de fins a 15 A. Per seguretat, es va decidir a que aquests relés controlessin a uns altres de fins a 25 A.

3.1.2 Sensor de nivell de bateries

Per controlar l'estat de les bateries es va decidir comprar un sensor de voltatge. Aquest sensor és, simplement, dos resistències que funcionen com a divisor de tensió com el que s'ha descrit en l'Esquema 1 i que, per tant, es podria haver fet sense la necessitat de comprar-ho, però degut al baix cost del sensor (0,60 € aproximadament) es va decidir tenir un sistema més robust.
A més, si es té en compte el preu d'hora d'un tècnic que es dediqui a soldar les resistències i els cables en el cas de haver-ho fet hagués sortit més car en temps que el fet de comprar el sensor i cargolar els cables.
És cert que en el cas d'arribar a enviar a fabricar una placa específica per a aquest propòsit ja es pensaria en afegir aquestes dues resistències sense la necessitat de comprar el sensor.

Sabem que les bateries son de 12 V. Una bateria de 12 V pot donar un voltatge de sortida màxim d'uns 15 V quan està carregada i mentre està carregant-se. Com l'entrada analògica del microcontrolador va de 0V a 3,3V, s'ha de fer un divisor de tensió que per a 16 V doni un voltatge de 3,3 V així hi ha aquest volt de marge de seguretat.



Esquema 1. Disseny del sensor de voltatge.

El circuit de Esquema 1 mostra un divisor de tensió tal que:

$$V_{micro} = \frac{7{,}5\ k}{7{,}5\ k + 30\ k} \cdot V_{Bateria}$$

Amb aquest divisor per a un voltatge de la bateria de 12 V es tindrà un voltatge d'entrada al microcontrolador de 2,4 V. De la mateixa forma, per a que tinguem una tensió de 3,3 V a l'entrada del microcontrolador la bateria hauria de donar un voltatge de 16,5 V.

Per a trobar la resolució del sistema és necessari saber que la entrada analògica del microcontrolador funciona a 12 bits (2^{12} = 4096) i que, per tant prendrà valors de entre 0 per a 0 V de la bateria i 4095 per a 16,5 V de la bateria.
Sabent això, podem trobar la resolució tal que:

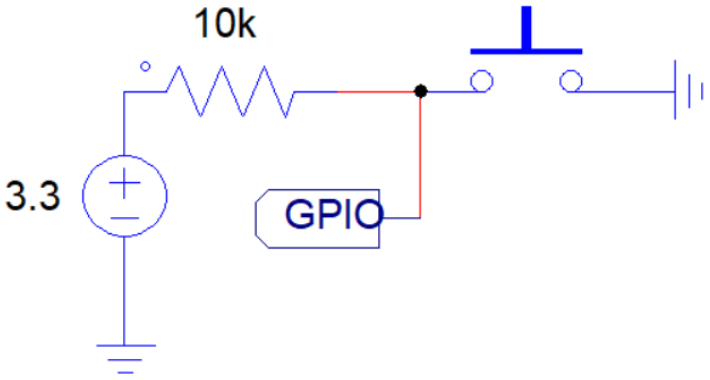
$$Resolució = \frac{V}{2^n - 1} = \frac{16{,}5}{4095} \approx \mathbf{4\ mV}$$

Per tant, l'error màxim que hi haurà serà de 4 mV, un valor molt acceptable ja que el que es busca es tenir uns valors de l'ordre dels volts.

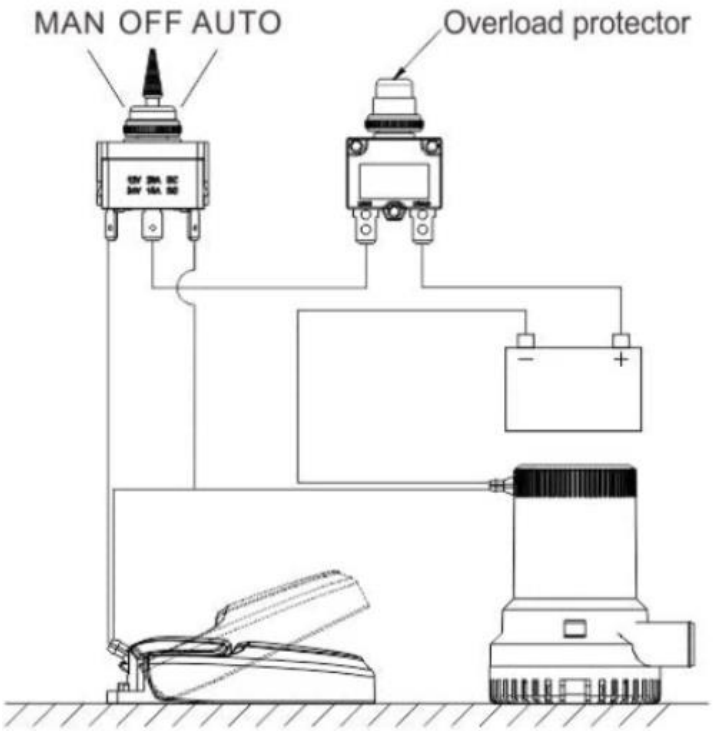
3.1.3 Sensors d'aigua a les sentines

A la sentina d'una embarcació sol anar una bomba de 12 V que expulsa l'aigua que es pugui acumular a la mar. Com es obvi, aquesta bomba no funciona de forma continua sinó que només ho fa quan hi detecta aigua per a que no es cremi i augmentar la seva vida útil.

Com es veu al esquema, aquesta bomba porta també una boia que actua com a interruptor. D'aquesta forma, per saber si hi ha aigua a la sentina s'ha connectat l'interruptor de la boia a una entrada del microcontrolador amb una resistència pull-up.



Esquema 2. Resistència Pull-up.



Esquema 3. Cablejat i funcionament d'una bomba de buidatge. (Mardefondo, 2024)

Amb aquest sistema es poden saber dues coses: si es rep una notificació de forma esporàdica es pot avisar a algú o anar si tenim la possibilitat de donar una ullada però no hi te per que ser res urgent. Si, en canvi, es reben notificacions cada cert temps de forma periòdica s'ha d'anar de forma urgent ja que o bé s'ha espatllat la bomba i per tant l'aigua que hi ha no pot ser expulsada o bé hi entra tanta aigua que la bomba no te prou cabal per expulsar-la.

3.1.4 Comptador d'hores de motor

Una funció també interessant és poder saber les hores dels motors. Tots els motors nàutics tenen uns protocols de manteniment oficials on es mira tant el temps des de la última revisió com les hores dels motors. A continuació es mostrarà un *Service Protocol* de Volvo Penta per a motors D12-A,E,F,G,H. Es mostrarà només una part representativa.

B

Every 1000 Hours / at Least Every 12 Months	
Fuel fine filter	R
Fuel pre-filter, filter insert	R
Reverse gear, oil and filter (Twin Disc)	R
Reverse gear, oil strainer (Twin Disc)	C
Air filter insert	R
Coolant Filter	R
Engine and revers gear, inspection for leakage	I
Engine and reverse gear, inspect hoses and cable clamping	I
Engine and revers gear, cleaning / painting	I
Flush pump / Bilge pump, impeller	I

C

Every 2000 Hours	
Valve clearance	I

D

Every 2000 Hours / at Least Every 24th Month	
Turbo	I
Drive Belts[4)]	I
Seawater pump impeller	R

Il·lustració 8. Service Protocol de Volvo Penta per al manteniment periòdic oficial dels motors. (Volvo, 2024)

Com es veu, el manteniment de la taula B, per exemple, s'ha de fer abans de les 1000 hores de funcionament o als 12 mesos, el que primer arribi. D'igual forma, el manteniment de la taula D s'ha de fer abans de les 2000 hores o abans de 24 mesos.
Això és important tenir-ho controlat ja que si el client es passa d'aquests valors el motor quedaria fora de garantia i si hi hagués algun problema s'hauria de fer càrrec. A part, si es passa d'aquests valors també incrementa el risc d'avariar.

Dit això, no és difícil veure que si tenim la possibilitat de que arribi una notificació al mòbil cada cop que s'arriba a unes hores de motor o un temps en mesos pot assegurar el bon manteniment dels motors, allargar la seua vida útil i evitar quedar fora garantia.

3.1.5 Actuadors relé

Per poder controlar motors elèctrics de 220 Vac i uns 10 A de consum amb un microcontrolador de molt baixa potència i de 3,3 Vdc és necessària la utilització de relés. Aquests relés funcionen com a contactors que al rebre una senyal d'entrada aquests exciten una bobina que tanca el circuit de sortida el qual és totalment independent al de l'entrada.

Per al control d'equips de menys de 7 ampers s'ha utilitzat relé de la marca Bestep model JQC3F-03VDC-C. Aquests funcionen amb 3,3V i permeten que hi circuli per la sortida un corrent de fins a 10 A.
Per al control d'equips de 10 A o més, en canvi, s'ha utilitzat els relés nombrats anteriorment com a maniobra i s'ha connectat en sèrie un altre relé de la marca Songle model SLA-24VDC-SL-C. Aquests s'han de connectar als 24 V de les bateries de serveis de l'embarcació a l'entrada i permeten que hi circuli un corrent a la sortida de fins a 30 A.

3.1.6 Convertidor 12 Vdc a 5 Vdc

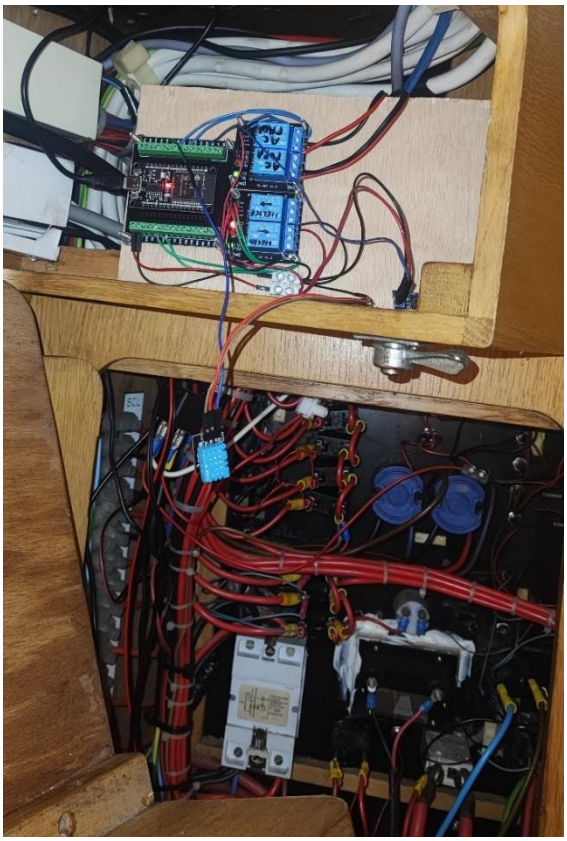
Al principi la placa estava alimentada amb un transformador de 220 VAC a 5 VDC. Això, a part de que el material feia pujar molt el preu final del producte com el transformador i el cable, ocupava molt d'espai i, per tant, no era pràctic. A més, queda la placa inutilitzada en el moment que el iot surt de port (es desconnecta de la toma de terra del pantalà i no engega el generador). Per tal de solucionar això es va buscar un LM7805 per convertir els 12V de les bateries a 5V per poder alimentar la placa per l'entrada Vin.

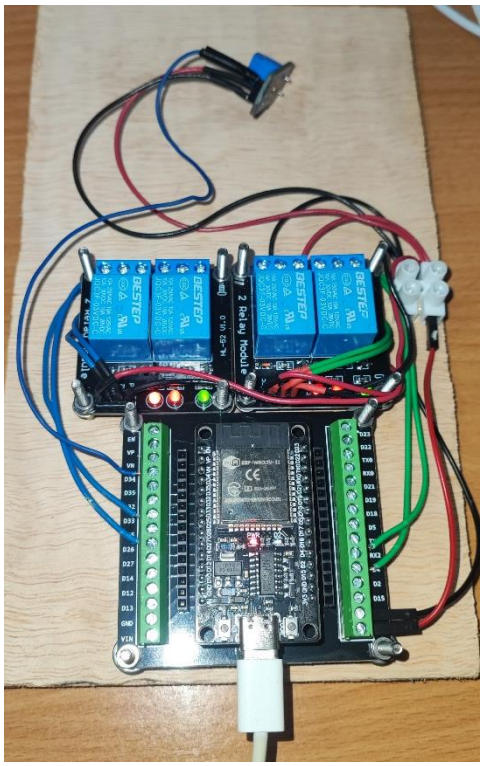
3.1.7 Ventilació

Un dels problemes que mes es va tenir va ser el fet de que hi havia moments en els que es perdia la connexió, moments en els que tornava fins que en un punt determinat ja deixava de tenir connexió de forma permanent. Després d'investigar-ho, es va arribar a la conclusió que quan la placa es sobreescalfava aquesta es posava en mode d'estalvi d'energia per a baixar la temperatura i evitar que es cremi. Per solucionar això, es van instal·lar uns ventiladors quadrats de 40cm. Aquests funcionen amb 12 V i a 6000rpm amb 25 dB de soroll. Amb aquesta instal·lació sembla que el problema ha desaparegut.

3.1.8 Prototip final

Finalment, i per aquest iot en concret, es va dissenyar un prototip que pogués unificar tots els elements comentats anteriorment i implementar-ho en el millor lloc de l'embarcació (que no molesti, que no ocupi altres llocs ja ocupats i que estigui el més prop possible del quadre elèctric per tenir que passar el mínim nombre de cables.





3.2 Configuració del software

A diferència del hardware, el software te la característica que la majoria de la feina només està al principi i que després, en cas de tenir més clients, només s'hauria de copiar el codi ja fet.

3.2.1 Programació del ESP32

En aquest apartat es descriurà el procés de programació del microcontrolador ESP32, que és essencial per implementar les funcionalitats de monitorització i control remot al iot. La programació de l'ESP32 es realitza utilitzant l'entorn de desenvolupament Arduino IDE, a causa de la facilitat d'ús i la compatibilitat amb l'ESP32. Per començar, cal instal·lar el suport per a ESP32 a Arduino IDE, la qual cosa s'aconsegueix instal·lant la llibreria "BlynkSimpleEsp32.h".

Un cop configurat l'entorn de desenvolupament, es procedeix a escriure el codi que controlarà els sensors i els actuadors connectats a l'ESP32. El codi base inclou la inicialització dels pins GPIO, la configuració del WiFi, i la definició de les funcions per llegir dades dels sensors i controlar els actuadors. També s'integra el codi necessari per establir la connexió amb la plataforma Blynk, utilitzant l'Auth Token proporcionat per Blynk per autenticar la comunicació entre l'ESP32 i l'aplicació.

A la programació de l'ESP32, cal controlar els pins d'entrada i sortida de manera eficient. Per exemple, els pins analògics es configuren per llegir valors de sensors com ara temperatura o voltatge de bateries, mentre que els pins digitals s'utilitzen per controlar dispositius com ara els relés dels aires condicionats o les bombes d'aigua. S'utilitzen funcions de lectura i escriptura digital per interactuar amb aquests pins, i s'implementen condicions lògiques al codi per realitzar accions automàtiques basades en les dades rebudes dels sensors. Per exemple, si un sensor detecta un nivell d'aigua alt a la sentina, el codi enviarà automàticament una notificació al usuari.

A més, s'inclou la programació per gestionar la connexió Wi-Fi, cosa que permet que l'ESP32 es connecti a la xarxa i enviï dades a l'aplicació Blynk. La connexió Wi-Fi es fa mitjançant biblioteques específiques que faciliten l'establiment de la connexió i el maneig de possibles errors, assegurant que el sistema es mantingui operatiu fins i tot si es perd temporalment la connexió. A més, també s'utilitzen algunes funcions per a poder tenir un control de les connexions WiFi que faci que l'usuari pugui decidir a quina xarxa es vol connectar. És crucial manejar correctament els esdeveniments de connexió i reconnexió per mantenir la comunicació constant amb la plataforma Blynk.

Un altre aspecte important en la programació de l'ESP32 és optimitzar l'ús de memòria i recursos. Atès que l'ESP32 té recursos limitats, cal assegurar-se que el codi sigui tan eficient com sigui possible, utilitzant tècniques com l'estalvi de memòria, l'optimització del consum d'energia, i la minimització de la latència en la lectura de sensors i el control actuadors. A més, s'implementen rutines de maneig d'errors per assegurar que el sistema sigui robust i es pugui recuperar de situacions inesperades sense interrupcions significatives.

Finalment, una vegada que el codi està escrit i provat a l'entorn de desenvolupament, es carrega a l'ESP32 utilitzant la connexió USB. Després de carregar el codi, es fan proves exhaustives per assegurar que l'ESP32 funcioni correctament i que totes les funcions de

monitorització i control operin com s'espera. Durant aquestes proves, es verifiquen tant la funcionalitat del maquinari connectat com la capacitat de l'ESP32 per interactuar amb l'aplicació Blynk a temps real. Aquestes proves permeten identificar i corregir qualsevol problema abans de la implementació final del sistema al iot.

En resum, la programació de l'ESP32 és un procés integral que inclou des de la configuració de l'entorn de desenvolupament fins a l'escriptura i la prova del codi que controlarà tots els aspectes del sistema de monitorització i control remot del iot. Una programació acurada i eficient garanteix que el sistema sigui fiable, respongui a temps als esdeveniments, i es mantingui operatiu fins i tot en condicions desafiadores.

3.2.2 Configuració de la app de Blynk

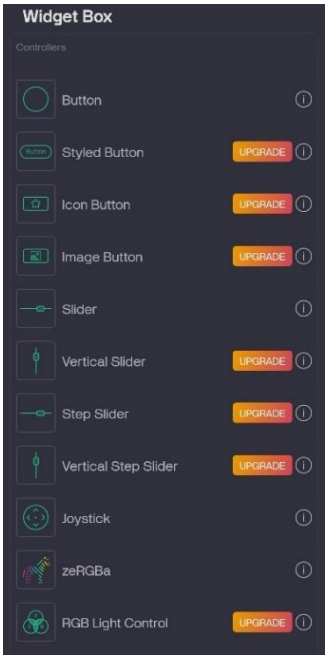
Aquest apartat detalla la configuració de l'aplicació Blynk. Aquesta proporciona una interfície intuïtiva, fàcil i minimalista que facilita tant la interacció amb els sensors i actuadors com l'experiència d'usuari.

Després d'haver creat el conte, es pot crear un projecte de forma totalment gratuïta. La creació del projecte genera automàticament un "Auth Token" únic per al dispositiu. Aquest token és la base per a la comunicació entre el ESP32 i la plataforma.

3.2.2.1 Interfície d'usuari

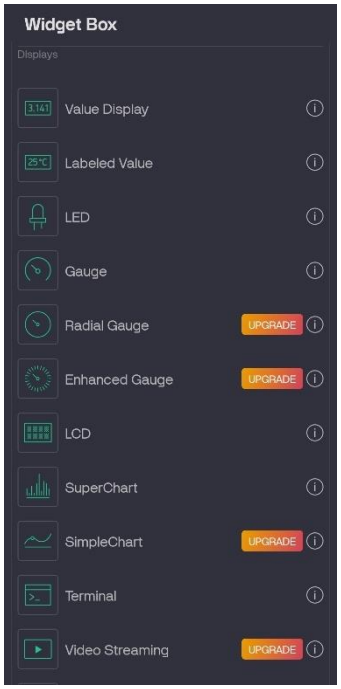
La interfície d'usuari pot ser creada mitjançant un sistema de *widgets* que Blynk ja porta incorporats. Aquest es divideixen en quatre grans blocs:

- ***Widgets* de control**: En aquest apartat hi ha una oferta de 17 opcions diferents, de les quals 4 són gratuïtes, dedicades a facilitar la interacció amb els actuadors. Hi ha, per exemple, polsadors, interruptors, controls lliscants o *joysticks* que es poden utilitzar per engegar o apagar llums, activar bombes d'aigua, ajustar la temperatura, engegar o apagar els aires condicionats, etc.
  Les opcions gratuïtes son un polsador bàsic, un control lliscant, un *joystick* i un control per als LED's RGB.



Il·lustració 9. Oferta de widgets per al control de l'aplicació Blynk.

- ***Widgets* de visualització**: En aquest apartat l'oferta és de 18 opcions i s'utilitzen per mostrar dades a temps real, com la temperatura, humitat, voltatge de bateries, etc.
  Les opcions gratuïtes son: indicador numèric, indicador numèric amb unitats, LED, indicador radial de 270 graus que mostra valors entre un mínim i un màxim, pantalla LCD de 16x2, un gràfic per visualitzar les dades històriques i un terminal.



Il·lustració 10. Oferta de widgets per a la visualització de l'aplicació Blynk.

- ***Widgets* d'interfície**: Aquí hi ha 10 opcions per escollir i permeten crear una interfície d'usuari molt més agradable per al usuari. Aquestes estan enfocades a

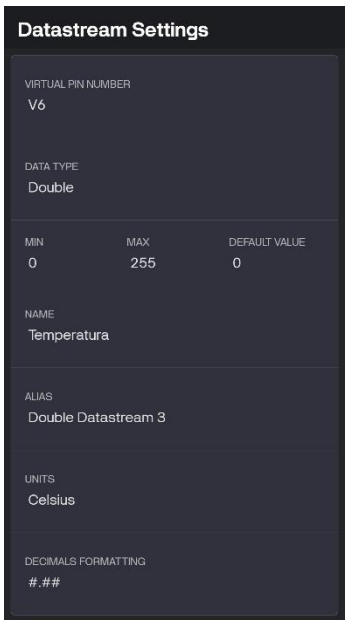
l'entrada de text i valors numèrics, utilitzar mapes, menús, text, i separadors dinàmics.
L'única opció disponible en la versió gratuïta és l'entrada de temps.

- **Altres**: L'oferta que ens dona Blynk aquí és de 4 opcions: un controlador de música, un polsador que et porta a una pagina web i l'opció de canviar el nom dels *datastreams*.

Per poder enllaçar aquests *widgets* amb el software del ESP32 s'ha d'utilitzar els *datastreams*. Cada sensor o actuador pot estar vinculat a un *datastream* específic. Aquests poden ser monitoritzats en temps real o emmagatzemats per a anàlisi posteriors.

3.2.2.2 Datastreams

Els *Datastreams* son utilitzats per l'entorn de Blynk com a variables virtuals per a la comunicació entre l'aplicació i el microcontrolador. Aquestes variables van fins de V0 fins a VN, sent N el nombre màxim de variables del projecte. Cadascuna d'aquestes variables es pot configurar amb el tipus de dades (integer, double o string); el valor màxim, mínim i el valor per defecte; el nom de la variable; l'alies; les unitats i un selector per activar l'històric.



Il·lustració 11. Exemple de la configuració d'un datastream a Blynk.

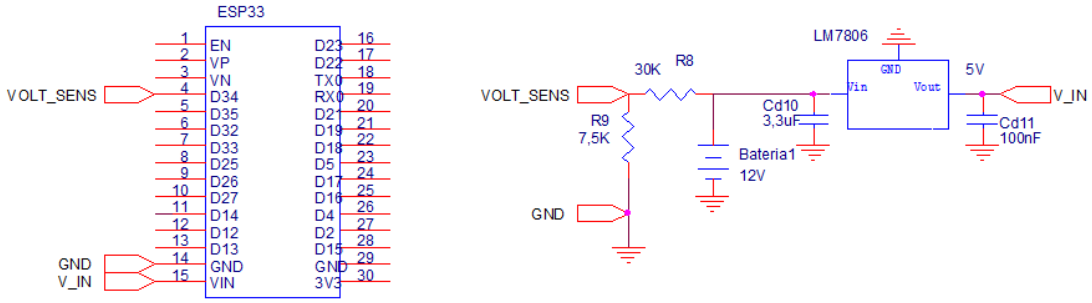
4 RESULTATS FINALS

4.1 Proves i calibració

4.1.1 Alimentació i sensor de voltatge

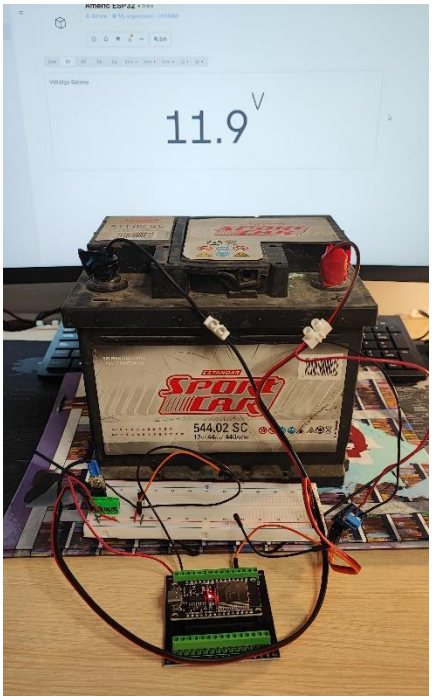
Per poder garantir la funcionalitat de tot el sistema s'ha hagut de anar fent proves intermèdies i així poder tenir la certesa que abans d'avançar funcioni el que ja s'ha fet. El primer que s'ha mirat ha estat l'alimentació de la placa amb el sensor de voltatge.

S'ha connectat el circuit integrat LM7805 a la bateria de 12 V i aquesta al sensor de voltatge. L'esquema utilitzat ha estat el següent:



Esquema 4. Connexió de l'alimentació del ESP32 i lectura de voltatge.

Amb aquesta prova s'ha comprovat que hi ha una bona connexió WiFi, que hi ha una bona connexió amb Blynk, que hi ha una bona alimentació i que, a més hi ha una bona lectura del voltatge de la bateria.



Il·lustració 12. Comprovació de la connexió del ESP32, la comunicació amb Blynk i la lectura del voltatge.

Com és una bateria vella és per això que dona un voltatge prou baix ja que aquestes bateries solen estar entorn dels 13 V. Es pot observar també com la placa està alimentada (LED vermell) sense tenir el cable USB-C connectat.

El codi utilitzat per a fer la lectura ha estat el següent:

```
#define BLYNK_TEMPLATE_ID "ESCRIURE TEMPLATE_ID"
#define BLYNK_TEMPLATE_NAME "ESCRIURE TEMPLATE_NAME"
#define BLYNK_AUTH_TOKEN "ESCRIURE AUTH_TOKEN"
//#define BLYNK_PRINT Serial

#include <BlynkSimpleEsp32.h>
#include <WiFi.h>

char ssid[] = "ESCRIURE EL NOM DEL WIFI";
char pass[] = "ESCRIURE LA CONTRASENYA DEL WIFI";

//DECLARACIÓ DE VARIBLES
const int sensorPin = 34; // Connectem el sensor al pin 34
const float volt_ref = 3.3; // Voltatge de referencia del ESP32

void setup() {
  Serial.begin(115200); // Inicialitzem comunicació sèrie

  //FEM LA CONNEXIÓ AMB BLYNK
  Blynk.begin(BLYNK_AUTH_TOKEN,ssid,pass);
}

void loop() {
  //LLEGIM L'ENTRADA DEL ADC (valor de 0 a 4095)
  int sensorValue = analogRead(sensorPin);

  //CONVERSIÓ DEL ADC A V (Ens donarà el voltatge al pin 34)
  float adc_voltage  = (sensorValue * volt_ref) / 4095.0;

  //CONVERSIÓ DE Vpin A Vbat (Ens donarà el voltatge a la bateria)
  float in_voltage = adc_voltage /0.2; //Paral·lel de 7,5 kohm i 30
kohm

  //MOSTREM EL VALOR A L'APLICACIÓ
  Blynk.virtualWrite(V1,in_voltage);

  //MOSTREM EL VALOR CADA SEGON
  delay(1000);
  Blynk.run();
}
```

Codi 1. Codi per a connectar el ESP32 a Blynk i llegir la lectura del voltatge.

4.1.2 Aires condicionats

A continuació es va provar de activar i desactivar els relés de l'AC amb l'aplicació amb les següents funcions:

```
void popa_on() { //FUNCIÓ PER A ENGEGAR L'AIRE DE POPA
  if (!aire_popa) { //Si l'aire està apagat
    digitalWrite(AIRE_POPA, LOW); //S'engega l'aire
    Blynk.setProperty(V4, "color", "#1CFF27"); //Es posa el botó de
Blynk a color verd
  }
  aire_popa = true; //Ens asegurem de no tornar a entar fins que no
l'apaguem
}

void popa_off() {//FUNCIÓ PER A APAGAR L'AIRE DE POPA
  if (aire_popa) {//Si l'aire està engegat
    digitalWrite(AIRE_POPA, HIGH); //S'apaga l'aire
    Blynk.setProperty(V4, "color", "#EE1010"); //Es posa el botó de
Blynk a color vermell
  }
  aire_popa = false; //Ens asegurem de no tornar a entar si no esta
engegat
}

void proa_on() {
  if (!aire_proa) {
    digitalWrite(AIRE_PROA, LOW);
    Blynk.setProperty(V3, "color", "#1CFF27");
  }
  aire_proa = true;
}

void proa_off() {
  if (aire_proa) {
    digitalWrite(AIRE_PROA, HIGH);
    Blynk.setProperty(V3, "color", "#EE1010");
  }
  aire_proa = false;
}
```

Codi 2. Funcions per al control dels AC's.

Aquestes funcions asseguren que només es doni l'ordre d'activar els aires si prèviament estaven desactivats i al revés. A més, un cop s'han engegat canvien els polsadors a color verd i quan s'han apagat a color roig. D'aquesta forma l'usuari pot saber a simple vista si la comanda ha arribat correctament ja que si canvia de color és perquè prèviament ha passat per la línia de codi que fa canviar l'estat dels aires.

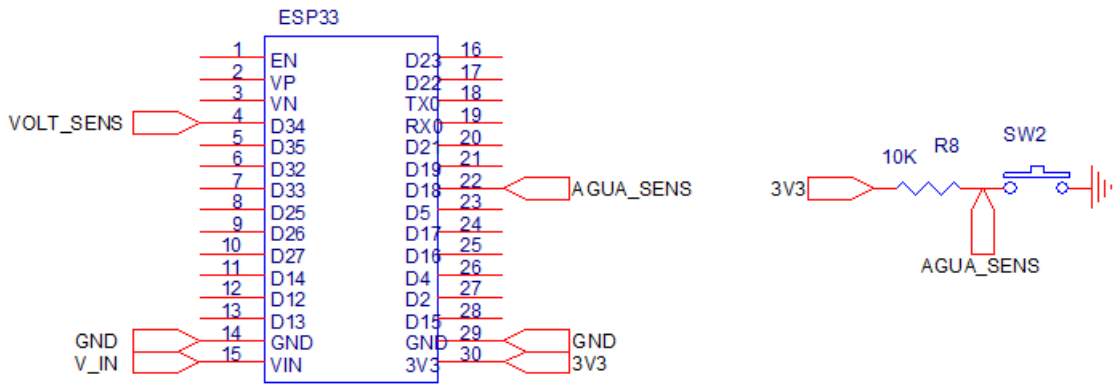
4.1.3 Control d'aigua a les sentines

Per poder saber si tenim aigua a les sentines s'envia una notificació la primera vegada que el sensor detecta aigua i després l'envia cada 1 minut si aquest segueix detectant. A continuació es mostra el codi:

```
void manageBombaAchique(){

  //Si fa més d'un minut des de que s'ha enviat la notificació
  if(!ACHIQUE_ESPERA){

  //Llegim el pin que connecta amb el sensor.
  BOMBA_ACHIQUE = digitalRead(18);

  //Com s'ha connectat amb pull-up, si detecta aigua curtcircuitarà
amb          massa.
  if (BOMBA_ACHIQUE == LOW){ //HI HA AIGUA A LES SENTINES

    //Es notifica a l'usuari
    Blynk.notify("S'ha detectat aigua a les sentines");

    //Es comença a contar el temps
    TempsUltim = millis();

    //Es canvia l'estat de la variable que controla el temps
    ACHIQUE_ESPERA == true;

    //Fins que no faci més d'un minut no s'actualitza la variable
    } else if(millis() - TempsUltim >= 60000) {
      ACHIQUE_ESPERA == false;
    }
  }
}
```

Codi 3. Codi per al control d'aigua a les sentines.

L'esquema que s'ha seguit ha estat el següent:



Esquema 5.Connexió del sensor d'aigua a les sentines.

4.1.4 Control d'hores de motor

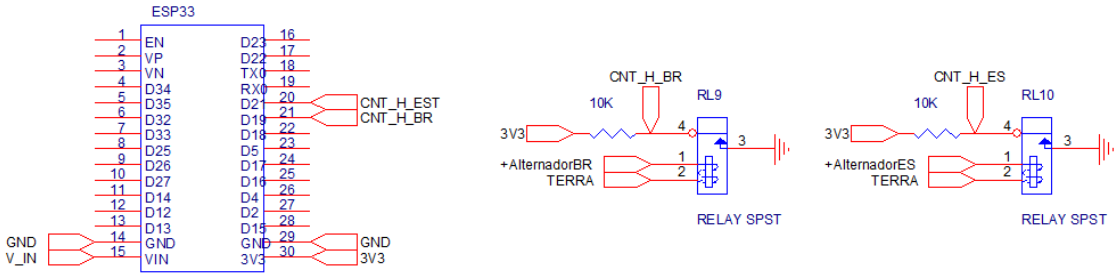
Per saber de forma elèctrica quan un motor està engegat es te dues opcions: es pot treure un cable del contacte de la clau o del alternador. L'opció de la clau normalment sol ser la més fàcil ja que els instruments de navegació solen estar prop de l'electrònica i l'alternador, en canvi, de vegades no te tant fàcil accés. Tot i això, tenir un comptador d'hores de motor al contacte de la clau pot fer que si s'està en el contacte posat, ja sigui perquè es vol veure el nivell de combustible, perquè s'està reparant un aparell que necessita d'estar connectat o perquè algú es pugui deixar el contacte donat, les hores que es contarien serien lleugerament superiors a les hores reals. En canvi, si ho connectem a l'alternador podem estar segurs que aquest donarà tensió només quan girin les corretges del motor i, per tant, quan aquest estigui engegat. A continuació es mostra el codi:

```
void manageControlHores(){

//MOTOR DE BABOR

//Donar la opció de posar el contador a 0 per a que cada cop que es fa
el manteniment comenci a contar per al següent.
  if(RESET_h_br){
    Hores_Babor = 0;
  }

//Si l'alternador dona tensió (Motor engegat)
  if(digitalRead(RELE_BABOR) == LOW){
    if(!ContantBR){
      startTimeBR = millis(); //Es comença a contar un cop s'engega el
motor
      ContantBR = true;
    }
  } else if(ContantBR){ //Es va actualitzant el temps un cop engegat
    TempsTranscorregutBR += millis() - TempsIniciBR;
    ContantBR = false;
  }
  if(ContantBR){
    Hores_Babor = TempsTranscorregutBR / 3600000;//Es pasa el temps a
h
    Blynk.virtualWrite(V_h_br, Hores_Babor);//S'envia a Blynk
    preferences.putFloat("HoresBabor",Hores_Babor);//Es guarda a la
Flash

    if(Hores_Babor >= 90.0){//Avisa de les 90 h de funcionament
      Blynk.notify("El motor de babor ha arribat a les 90 hores de
funcionament. Hora de manteniment!");
    }
  }

  if(RESET_h_es){
    Hores_Estribor = 0;
  }

  if(digitalRead(RELE_ESTRIBOR) == LOW){
    if(!ContantES){
      startTimeES = millis();
      ContantES = true;
    }
  } else if(ContantES){
    TempsTranscorregutES += millis() - TempsIniciES;
```

```
    ContantES = false;
  }
  if(ContantES){
    Hores_Estribor = TempsTranscorregutES / 3600000;
    Blynk.virtualWrite(V_h_es, Hores_Estribor);
    preferences.putFloat("HoresEstribor",Hores_Estribor);

    if(Hores_Estribor >= 90.0){
      Blynk.notify("El motor d'estribor ha arribat a les 90 hores de
funcionament. Hora de manteniment!");
    }
  }
}
```

Codi 4. Control de les hores de funcionament dels motors.

L'esquema ha sigut el següent:



Esquema 6. Connexió per al control d'hores dels motors.

4.1.5 Llegir temperatura

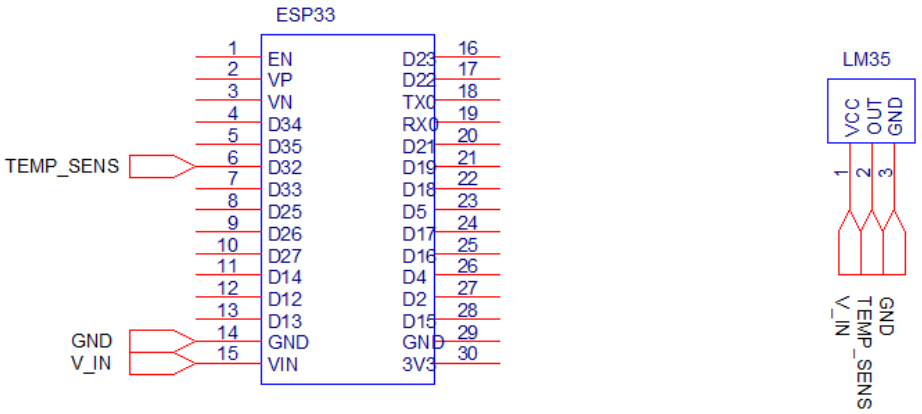
La lectura de temperatura s'ha fet amb els sensors LM35. Aquests sensors proporcionen una lectura de 10 mV/ºC. El codi utilitzat ha estat el següent:

```
const int sensorPin = 14;
int i=0;
float temperatura_mitja = 0;

[...]

void loop() {
  [...]
  int sensorTemp = analogRead(sensorPin);  // Llegeix el valor del pin
analògic
  float voltatge = sensorTemp* (3.3 / 4095.0);  // Conversió de 0-4095
a 0-3,3V
  float temperatura = voltatge * 100.0;  // Convertir el voltatge a
temperatura en graus Celsius

  if(i<5){ //Es fa la mitja de les ultimes 5 lectures
    temperatura_mitja = temperatura_mitja + temperatura;
    i++;
  } else {
    i=0;
    temperatura = temperatura_mitja/5;
    temperatura_mitja = 0;
```

```
    Serial.print("Temperatura: ");
    Serial.print(temperatura);
    Serial.println(" °C");
  }
  delay(500);

  [...]
}
```

Codi 5. Codi per a llegir la temperatura amb un sensor LM35 i mostrar-ho pel monitor Serial.

L'esquema ha sigut el següent:



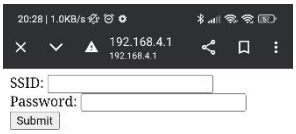
Esquema 7. Circuit per a connectar el sensor de temperatura LM35 a la placa ESP32.

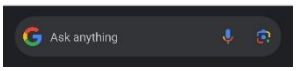
4.1.6 WiFi

Amb totes les proves fetes, es pot pensar en optimitzar i millorar la connexió WiFi, fent que l'usuari pugui decidir a quina xarxa connectar-se en qualsevol moment. Per això, s'ha pensat un sistema que utilitzant un servidor i amb llenguatge HTTP es pugui controlar el microcontrolador.

La gestió del WiFi funciona de la següent manera:

1. Connectem la placa a la corrent.
2. Ens connectem al WiFi de la placa **ESP32_AP**.
3. Introduïm la contrasenya preestablerta.
4. Obrim el navegador web.
5. Desconnectem les dades mòbils o qualsevol altra connexió a internet.
6. Introduïm al navegador la IP **192.168.4.1**.
7. S'obrirà la següent pantalla:





Il·lustració 13. Pantalla per a la connexió WiFi.

8. Introduir les credencials WiFi on es vulgui connectar.
9. Ja s'està connectat.

Il·lustració 14. Connexió WiFi establerta.

En el cas, per exemple, que s'estigui connectat al WiFi de la marina i es vol sortir a navegar i, per tant, es vol connectar al WiFi personal, s'ha de fer el següent:

1. Estant connectats al WiFi de la placa, introduïm al navegador la comanda **192.168.4.1/reset**.
2. Continuem amb el pas número 7 de la llista anterior.

Per fer aquesta connexió, s'ha utilitzat la llibreria "DNSServer.h". Aquesta llibreria serveix per a crear un servidor DNS en el dispositiu. Un servidor DNS el que fa es redirigir qualsevol intent d'accés a algun lloc web a una pàgina de configuració del ESP32.

Amb aquest servidor s'ha creat una pagina web per a definir la SSID i la contrasenya del WiFi i, a més, s'ha instal·lat tres botons per a tenir un accés directe als tres WiFi més utilitzats en el cas del projecte en concret que es el meu, el del client i el de la marina on es troba el iot. A continuació es mostra el codi:

```
// Configuració de l'AP
const char* ap_ssid = "ESP32_AP"; //S'introdueix el nom que es vol que
aparegui del WiFi
const char* ap_password = "12345678"; // S'introdueix la contrasenya
que es vol que es tingui del WiFI
IPAddress apIP(192, 168, 4, 1); //S'introdueix la IP per accedir a la
configuració
IPAddress netMsk(255, 255, 255, 0);

// Función per controlar el servidor Web
void handleRoot() { //Funció que s'executarà quan s'accedeixi a la IP
anterior
  String html = "<form action=\"/configure\" method=\"POST\">";
//Iniciem la variable HTML
  html += "SSID: <input type=\"text\" name=\"ssid\"><br>"; //Introduim
un camp d'entrada de text per a la SSID
  html += "Password: <input type=\"password\" name=\"password\"><br>";
//Introduim un camp per a l'entrada de text per a la contrasenya
  html += "<input type=\"submit\" value=\"Submit\">"; //Boró per a
l'enviament del formulari
  html += "</form>"; //Tancament del formulari

  html += "<h2>Redes Preconfiguradas</h2>";//Titol
  html += "<button
onclick=\"window.location.href='/connectCasa'\">Marina</button>";//Bot
ó per a connectar a la xarxa de la Marina
  html += "<button
onclick=\"window.location.href='/connectTrabajo'\">Javier</button>";
//Botó per a connecar a la xarxa del client
  html += "<button
onclick=\"window.location.href='/connectOficina'\">Sergi</button>";
//Botó per a connectar a la meva xarxa
  html += "<br><br><a href=\"/reset\">Resetear Credenciales</a>";
//Botó per a resetejar les credencials.

  server.send(200, "text/html", html); //Enviem tot el text HTML al
client (pàgina web)
}

// Función para manejar la configuración de WiFi
void handleConfigure() {
  String ssid = server.arg("ssid"); //S'obté el valor d'entrada del
camp de text SSID
  String password = server.arg("password");//S'obté el valor d'entrada
del camp de text Password

  saveCredentials(ssid.c_str(), password.c_str()); //Es guarden les
credencials a la memòria no volàtil
```

```
  String response = "Credentials saved. Restarting..."; //S'envia un
text conforme tot correcte
  server.send(200, "text/plain", response);
  delay(1000);
  ESP.restart();//Resetejem la placa
}


// Funció per a resetejar les credencials del WiFi
void handleReset() {
  preferences.begin("wifi", false); //Resetejem la SSID i la
contrasenya de la memoria no volàtil
  //preferences.clear();
  preferences.end();
  server.send(200, "text/plain", "Credentials cleared.
Restarting...");
  delay(1000);
  ESP.restart();
}

// Funció per a establir la conexió al WiFi "Marina"
void handleConnectMarina() {
  const char* ssid = "XX";
  const char* password = "AA";
  saveCredentials(ssid, password);
  server.send(200, "text/plain", "Conectando a WiFi Marina.
Reiniciando...");
  delay(1000);
  ESP.restart();
}

// Funció per a establir la conexió al WiFi "Javier"
void handleConnectJavier() {
  const char* ssid = "YY";
  const char* password = "BB";
  saveCredentials(ssid, password);
  server.send(200, "text/plain", "Conectando a WiFi Javier.
Reiniciando...");
  delay(1000);
  ESP.restart();
}

// Funció per a establir la conexió al WiFi "Sergi"
void handleConnectSergi() {
  const char* ssid = "ZZ";
  const char* password = "CC";
  saveCredentials(ssid, password);
  server.send(200, "text/plain", "Conectando a WiFi Sergi.
Reiniciando...");
  delay(1000);
  ESP.restart();
}

void setup() {
  Serial.begin(115200);

  // Configurar AP
  WiFi.mode(WIFI_AP_STA); //Configuració del mode dual (punt d'accés
```

```
AP i client de red STA)
  WiFi.softAP(ap_ssid, ap_password); //Es crea un punt d'accés amb la
SSID i la contrasenya preestablertes
  WiFi.softAPConfig(apIP, apIP, netMsk); //Es configura la IP i la
màscara de subred preestablertes

  // Iniciar servidor DNS
  dnsServer.start(53, "*", apIP); //Servidor DNS per al port estàndard
de DNS (53)

  // Configurar servidor web
  server.on("/", HTTP_GET, handleRoot);
  server.on("/configure", HTTP_POST, handleConfigure);
  server.on("/reset", HTTP_GET, handleReset);
  server.on("/connectMarina", HTTP_GET, handleConnectMarina);
  server.on("/connectJavier", HTTP_GET, handleConnectJavier);
  server.on("/connectSergi", HTTP_GET, handleConnectSergi);
  server.begin();
}
```

Codi 6. Codi per a la configuració WiFI.

4.1.7 Emmagatzematge en memòria no volàtil

Una altra opció que s'ha de tenir en compte és el fet de que per algun motiu la placa pot tenir alguna desconnexió de l'alimentació o algú pot polsar el botó de *reset*. És per això que s'ha hagut de pensar una forma de poder emmagatzemar la informació que no es vulgui perdre a una part de la memòria que, inclús sense alimentació, no es perdi la informació.

Per fer aquesta implementació, s'ha utilitzat la llibreria "Preferences.h". Aquesta llibreria s'utilitza en els microcontroladors ESP32 per emmagatzemar dades de forma permanent, inclús després de que el dispositiu s'apagui o es reinicïi. Per fer-ho, emmagatzema aquestes dades a la memòria no volàtil (Non-Volatile Storage).

L'ús que se li ha donat ha estat per a emmagatzemar les dades del WiFI (SSID i contrasenya), els valors de temperatura predefinits per l'usuari, els valors de histèresis predefinits per l'usuari i l'estat de tots els polsadors. A continuació es mostra el codi:

```
//Declarem els objectes Preferences
Preferences preferences;
Preferences temperatura;
Preferences histeresi;
Preferences selector_popa;
Preferences selector_proa;
Preferences modo;
Preferences selector_frio_calor;

// Funció per a guardar les credencials WiFi a la memòria no volàtil
(NVS)
void saveCredentials(const char* ssid, const char* password) {
  preferences.begin("wifi", false); //Iniciem l'accés amb el nom wifi
i amb mòde lectura/escriptura (false)
  preferences.putString("ssid", ssid); //Guardem la variable ssid a
memoria no volatil
  preferences.putString("password", password); //Guardem la variable
```

```
password a memoria no volatil
  preferences.end(); //Finalitzem l'accés i alliberem recursos
}

// Funció per a extreure les dades de la memoria no volàtil (NVS)
void loadCredentials(String& ssid, String& password) {
  preferences.begin("wifi", true); //Iniciem l'accés amb el nom wifi i
amb mòde lectura (true)
  ssid = preferences.getString("ssid", ""); //Recuperem les dades de
ssid. Si no hi ha, retorna un valor buit ""
  password = preferences.getString("password", ""); //Recuperem les
dades de password. Si no hi ha, retorna un valor buit ""
  preferences.end();//Finalitzem l'accés i alliberem recursos
}


void setup() {
  // Intentar conectar a la xarxa WiFi guardada
  String ssid, password;
  loadCredentials(ssid, password);

  //Inicialitzem l'accés a NVS per a diferents variables amb mode
lectura/escriptura (false)
  temperatura.begin("blynk", false);
  histeresi.begin("blynk-1", false);
  selector_popa.begin("blynk-2",false);
  selector_proa.begin("blynk-3",false);
  modo.begin("blink-4",false);
  selector_frio_calor.begin("blynk-5",false);

  //Es recuperen tots els valors emmagatzemats a la memòria no
volàtil. Si no hi ha, retorna un valor per defecte
  TEMP_CONSIGNA = temperatura.getFloat("temperatura", 20.0); // Valor
per defecte es 20.0
  HISTERESIS = histeresi.getFloat("histeresi", 0.1); // Valor per
defecte es 0.1
  SEL_PROA = selector_proa.getBool("selector_proa", false); // Valor
per defecte es OFF
  SEL_POPA = selector_popa.getBool("selector_popa", false); // Valor
per defecte es OFF
  SEL_MAN_AUTO = modo.getBool("modo",false); // Valor per defecte es
MANUAL
  SEL_FRIO_CALOR =
selector_frio_calor.getBool("selector_frio_calor",false); // Valor per
defecte es FRIO
}

[...]//Codi que mostra com es van carregant les variables

Blynk.virtualWrite(V_h_es, Hores_Estribor);//Escrivim les hores
actuals del motor d'estribor
preferences.putFloat("HoresEstribor",Hores_Estribor); //Guardem
aquestes hores a la memòria no volàtil

[...]

BLYNK_WRITE(V2) {
  SEL_MAN_AUTO = param.asInt(); //cada com que es polsa el selector de
mode Manual o Automàtic
```

```
  modo.putBool("modo", SEL_MAN_AUTO); //S'emmagatzema l'estat a la
memòria no volàtil
}

[...]
```

Codi 7. Còdi per a emmagatzemar variables a la memòria no volàtil.

4.2 Blynk

Per la part de Blynk finalment ha quedat un *dashboard* de la següent forma:

Il·lustració 15. Resultat del dashboard final.

A la part superior es pot veure com es pot seleccionar si es vol utilitzar els aires condicionats de forma manual (engegar-los i parar-los manualment i quan es vulgui) o de forma automàtica (que s'engeguin o es parin automàticament per mantenir la temperatura de consigna ± la histèresis). També es pot seleccionar entre el fred i la calor. Com els AC disposen de bomba de calor, si es volen utilitzar al hivern en mode automàtic voldrà dir que es voldrà tenir la embarcació més calenta que l'exterior, mentre que si es fa a l'estiu es voldrà tenir més freda.

Seguidament hi ha dos selectors per seleccionar si es vol engegar els aires de proa i/o popa. Si s'engeguen, aquests es tornaran de color verd advertint a l'usuari de que estan engegats.

Mes a baix hi ha els *sliders*, aquests permeten a l'usuari introduir els valors de histèresis i temperatura de consigna que es vol si es vol mantenir la temperatura de la embarcació constant sense la necessitat de anar engegant i apagat els aires de forma manual.

La part de la hèlix de proa és un *joystick* per a poder controlar aquesta quan s'atraca al port, es va sol i no hi ha ningú que t'ajudi a moure la proa del vaixell.

Finalment, a la part inferior veiem els valors de les variables que es van actualitzant de forma periòdica. Aquests son el WiFi on s'està connectat, la temperatura del iot actual, la intensitat de la senyal que es te del WiFi en negatiu, es a dir, 48 dB simbolitzen -48 dB i finalment el voltatge de les bateries.

Degut a les limitacions de Blynk per utilitzar el pla gratuït només s'ha disposat de 10 variables per a poder-se comunicar amb l'ESP32. Aquestes han estat:

V0: Selector de fred o calent
V1: Lliure
V2: Selector mode manual o automàtic
V3: Selector aire proa
V4 Selector aire popa
V5: Slider temperatura consigna
V6: Lliure
V7: Slider per seleccionar la histèresis
V8: Joystick hèlix de proa
V9: Terminal

S'han deixat 2 variables lliures per a poder ampliar en un futur noves funcionalitats i s'ha utilitzat un terminal per a poder visualitzar més d'una variable utilitzant-ne només una de Blynk.

5 CONCLUSIONS I PROPOSTES DE MILLORA

5.1 Materials utilitzats i viabilitat econòmica

Un dels objectius principals era que el projecte fos el més econòmic possible. Es creu que és un handicap afegit haver de tenir present la part econòmica ja que normalment et dificulta el no poder tenir accés a material o software de mes qualitat. A més, també s'ha fet tenint present la economia personal. A continuació es mostrarà una llista dels materials utilitzats i el cost:

Materials utilitzats	Quantitat	Preu unitari	Cost total
ESP32	1	2,98 €	2,98 €
Sensor Voltatge	1	0,63 €	0,63 €
Tira de terminals	2	0,07 €	0,14 €
2x Relé 3,3 V	1	2,10 €	2,10 €
Mòdul placa	1	1,58 €	1,58 €
Sensor LM35	1	1,70 €	1,70 €
Regulador L7805	1	0,21 €	0,21 €
Relé potència 230V	2	2,28 €	4,56 €
Ventilador	1	2,52 €	2,52 €
Material vari	1	2,00 €	2,00 €
Total			**18,43 €**

Taula 3. Llista dels materials utilitzats i el seu cost.

Per a calcular aquesta viabilitat també es te que tenir en compte les hores invertides. Es començarà mostrant les hores invertides per a aquest projecte:

Descripció	Temps invertit	Hores
Investigació	2 dies	8
Planificació	2 dies	8
Selecció del components	1 dies	4
Disseny del hardware	1 setmana	20
Desenvolupament del software	1 mes	80
Configuració de Blynk	3 dies	12
Proves	2 setmanes	40
Muntatge	3 dies	12
Documentació	1 mes i mig	120
Revisió i correcció	2 dies	8
Total		**312**

Taula 4. Hores invertides per a aquest projecte.

Aquestes hores son una aproximació i s'ha calculat tenint en compte una jornada de dilluns a divendres de 4 hores al dia.

Finalment, també s'ha de tenir en compte una aproximació de les hores necessàries per a poder implementar i vendre el producte a partir de la primera venta.

Descripció	Temps invertit	Hores
Configuració de Blynk	1 dia	4
Muntatge	2 dies	8
Proves	3 dies	12
Documentació	1 dia	4
Total		**28**

Taula 5. Hores invertides per a la venta a partir del primer producte.

Com gran part de la feina ja estaria feta, només s'ha contat la feina de muntatge i proves sent aquesta una feina que s'ha de fer específica per a cada client. Com és una feina que ja s'ha fet, aquestes hores serien inferiors ja que es tardaria menys en fer el mateix.

Segons el mercat, aquest es un producte que podria tenir un preu de venta de entre 500 € i 700 €. Aquest preu es calcula pensant que el producte es troba en unes bones condicions de venta, es a dir, un bon màrqueting, empaquetatge, muntatge, etc. Per això ho valorarem amb un cost de materials de 100 € per producte. Calculant el preu d'hora per a un PVP final mig de 600 €:

$$Preu\ d'hora = \frac{Preu\ venta - preu\ materials}{hores\ invertides} = \frac{600-100}{28} = 17{,}85\ {}^{€}/_{bruts}$$

Tenint en compte el valor de les hores d'un tècnic actualment, aquest preu no és un preu raonable ni factible. Per a que ho sigui, s'hauria de calcular les hores mínimes de muntatge per a que sortiges un preu d'hora brut d'uns 50 € tenint en compte el preu d'hora actual dels tècnics.

$$Hores = \frac{Preu\ venta - preu\ materials}{Preu\ d'hora} = \frac{600-100}{50} = 10\ hores$$

Si mantenint el cos dels materials i el preu de venta es pogués reduir el temps invertit per cada producte a 10 h, aquest seria un producte econòmicament viable.

5.2 Objectius

En aquest punt crec que és important recapitular i veure els objectius que es van posar en un principi i valorar com s'han resolt si s'ha pogut resoldre i perquè no si no:

1. Aconseguir una **connexió WiFi estable, còmoda per a l'usuari i funcional** per a poder garantir el correcte funcionament de tot el sistema.

S'ha pogut establir una connexió de WiFi estable gràcies a que el codi torna a connectar automàticament la connexió quan detecta que s'ha perdut i gràcies també a l'ús de ventilació forçada per evitar el sobreescalfament de la CPU.

2. Crear **un *dashboard* amb el software de Blynk** on l'usuari pugui tenir la millor experiència possible.

Tot i les limitacions del pla gratuït, s'ha creat un *dashboard* completament funcional i intuïtiu.

3. Crear un **sistema modulable** on es pugui adaptar a diferents necessitats de possibles clients i models d'embarcacions.

Gràcies a l'ús de funcions, s'ha ideat un codi on es poden afegir o treure aplicacions simplement afegint o traient funcions.

4. Poder **dissenyar, programar, muntar i provar** diferents aspectes com:
    a. Control de l'estat de les **bateries**.

S'ha pogut controlar l'estat de les bateries, tot i que no és un sensor extremadament precís (hi té un error d'aproximadament ±0,5V), en el cas que ens ocupa es necessita veure si les bateries estan carregades (14 V) o descarregades (menys de 12 V).

    b. Control de la **temperatura** per poder utilitzar-ho en temperatura ambient, sala de màquines, equip electrònic, etc.

El sensor LM35 ha permès poder fer una lectura de la temperatura bastant fiable. Tot i que és possible que no sigui el sensor més precís del mercat, està pensat per a aplicacions on l'error pot ser de fins a més de 1 grau.

    c. Control de les **hèlix de proa i/o popa**.

Les hèlix de proa son dispositius molt cars (més de 5000 €) i, tot i que s'ha dissenyat el sistema per fer-ho, no s'ha provat amb una hèlix de veritat per evitar danyar-la.

    d. Comptatge de les **hores dels motors** amb avisos cada certes hores preestablertes per millorar el manteniment.

Tot i que s'ha dissenyat i programat el comptatge d'hores, no s'ha provat per la dificultat de haver de passar els cables de la sala de màquines a el punt on es troba el ESP32.

    e. Control de presència d'**aigua a les sentines** amb avisos per poder fer actuacions preventives.

Tot i que s'ha dissenyat i programat el sensor d'aigua a la sentina, no s'ha provat per la dificultat de haver de passar els cables de la sala de màquines a el punt on es troba el ESP32.

    f. Control dels **aires condicionats** de bord, amb control automàtic i manual.

S'ha dissenyat i programat un sistema per al control dels aires condicionats tant manual com automàtic, tant per fred com per a calor i amb avisos de quan els aires es troben engegats i apagats.

5. Fer el tot el sistema **el més econòmic possible** degut als escassos recursos econòmics dels que es disposen i pensant en que sigui el més rentable possible per a una possible venta.

Amb tot el material, el cost total del sistema puja aproximadament a 30 €. Es creu que s'ha complert l'objectiu de fer el sistema el més econòmic possible (no es conta el temps empleat).

5.3 Conclusions

El desenvolupament d'un sistema de control remot per a les embarcacions ha resultat ser una solució eficaç, econòmica i intuïtiva per a l'usuari final, fet que permet millorar la gestió dels elements crítics a bord de la mateixa manera que permet controlar els elements de confort d'una forma molt més còmoda.

El món de la nàutica és un món on hi ha molta economia. El fet de poder aprofitar això en un poble on hi ha més de 300 amarratges per a iots de més de 10 metres d'eslora és una gran oportunitat.

Tot i que encara hi hagi un ampli marge de millora i perfeccionament, com poder utilitzar el software propi del ESP32 *Espressif* per poder generar un codi molt més controlable i eficient, a més de poder demanar a fabricar unes plaques electròniques personalitzades, crear una caixa per a poder posar tots els components amb la possibilitat de poder connectar els cables per fora i, en definitiva, més inversió, el que hi ha ara és un sistema completament funcional i que, de fet, ja està instal·lat i utilitzant-se.
El que m'agradaria fer en un futur també es poder provar el comptatge d'hores dels motors, el nivell de sentines o el control de la hèlix de proa.

Vull destacar també la gran oportunitat que he tingut de poder disposar d'un iot de 15 metres d'eslora per a fer les proves de camp. Al final poder tenir una embarcació d'aquestes característiques suposa un cost econòmic que molt poca gent pot assumir, jo el primer. A més, he pogut conèixer totes les parts que hi ha i, per tant, totes les parts on hi hauria de haver un control més exhaust.

Durant aquest estiu que ja ha estat muntat he anat tenint contacte amb el propietari per saber les seves sensacions i possibles millores que al final no veus fins que no ho proves. Moltes de les funcions que hi ha ara com el poder canviar el WiFi o el que es quedin emmagatzemades les dades a la memòria no volàtil són propostes que m'ha anat dient ell. Tot i això, i de moment, els inputs que he anat tenint d'ell han estat molt positius. Al final el sistema que tenia abans amb els aires condicionats, per exemple, era un sistema de ON/OFF que no permetia el poder regular la temperatura. Això suposava que si hi tenien que passar una nit o havien de dormir amb molt de fred o amb molta calor. A més, el poder engegar-los abans d'arribar sobretot si vas en família o amics i poder arribar al iot i gaudir d'una bona temperatura és un valor afegit molt important.

5.4 Propostes de millora

Tot i que ja s'han anat comentant, en aquest apartat es descriuran les propostes de millora que penso que podrien donar un valor afegit si es fessin en un futur:

- **APLICACIÓ MÒVIL:** Tot i que Blynk dona unes prestacions molt bones per a ser un programa gratuït i inclús el pla de subscripció és un pla bastant complert, crec que el més adequat seria poder tenir una aplicació pròpia o, almenys, alguna que donés més opcions de personalització.
- **PRESENTACIÓ:** Tot i que no afectaria al rendiment i a la funcionalitat del sistema, la presentació del producte final és molt important per mostrar serietat i professionalitat. Poder tenir una caixa de plàstic o metall amb tots els components ordenats a dintre faria que el producte donés una molt millor imatge.
- **PLACA ELECTRÒNICA:** Si només s'ha de fer un producte no val la pena, però si s'han de fer un numero relativament elevat, potser valdria la pena demanar a imprimir unes plaques electròniques a alguna empresa externa reduint molt el numero d'hores invertides per a cada producte i fent també un producte molt més professional.
- **SOFTWARE:** L'ús d'un software de més baix nivell podria donar una controlabilitat molt més elevada a canvi d'una inversió de temps molt més alta. És per això que per a un producte o prototip potser no val la pena però que si la intenció és fer-ne més aquesta es pot repartir entre més clients i fer que sigui més rentable la inversió.

6 REFERÈNCIES

Aliexpress. (2024). *Módulo de placa de expansión ESP32*. Obtenido de https://es.aliexpress.com/item/1005006249254586.html?spm=a2g0o.order_list.order_list_main.35.12a8194dy0cNkp&gatewayAdapt=glo2esp

Aliexpress. (2024). *Módulo de relé de 3V, 3,3 V*. Obtenido de https://es.aliexpress.com/item/1005003750654499.html?spm=a2g0o.order_list.order_list_main.40.12a8194dy0cNkp&gatewayAdapt=glo2esp

Aliexpress. (2024). *Placa de desarrollo ESP32 WiFi*. Obtenido de https://es.aliexpress.com/item/1005004230550198.html?spm=a2g0o.order_list.order_list_main.81.12a8194dy0cNkp&gatewayAdapt=glo2esp

Aliexpress. (2024). *Sensor de Voltaje*. Obtenido de https://es.aliexpress.com/item/1005006942950889.html?spm=a2g0o.order_list.order_list_main.66.12a8194dy0cNkp&gatewayAdapt=glo2esp

Arduino. (2024). *Arduino MKR WiFi 1010*. Obtenido de https://store.arduino.cc/en-es/products/arduino-mkr-wifi-1010

Blynk. (2024). *Simple plans that fit your needs*. Obtenido de https://blynk.io/pricing

Engeneers, L. M. (30 de 08 de 2024). *ESP32 Pinout Reference*. Obtenido de https://lastminuteengineers.com/esp32-pinout-reference/?utm_content=cmp-true

Espressif. (2024). *ESP32 Series Datasheet Version 4.6.* Shanghai.

Mardefondo. (2024). *Flotador bomba de achique interruptor sentina*. Obtenido de https://mardefondo.shop/producto/flotador-para-bomba-chique-sentina-barco/

Obarti, J. (24 / 02 / 2020). *Oceanica*. Recollit de https://www.oceanicanautica.es/cy/partes-barco-conceptos-basicos/

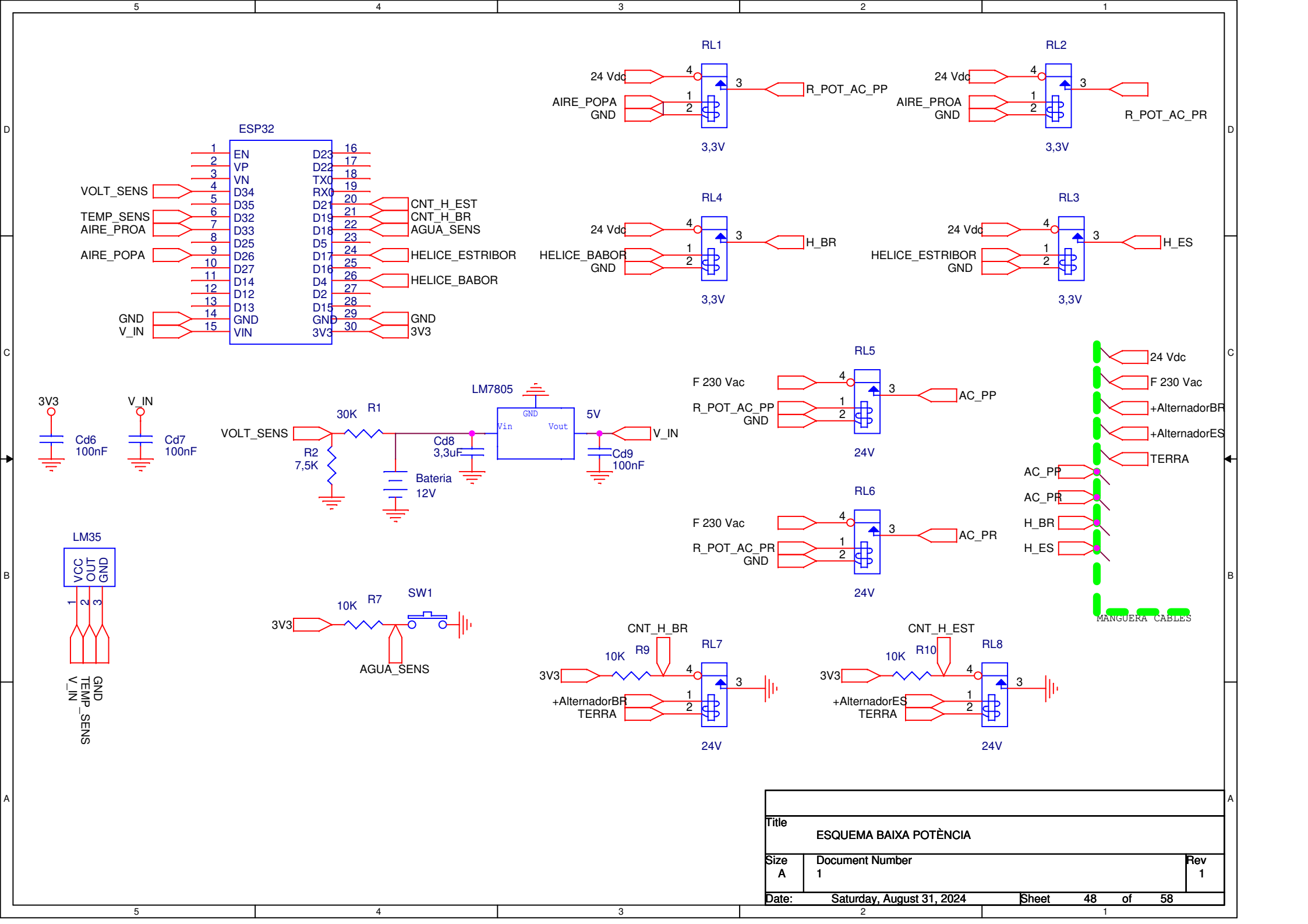
Volvo. (2024). *Protocolo Volvo.*

Wikipedia. (24 de 02 de 2021). *Sentina*. Obtenido de https://es.wikipedia.org/wiki/Sentina

Wikipedia. (10 de 10 de 2023). *Hélice de maniobra*. Obtenido de https://es.wikipedia.org/wiki/H%C3%A9lice_de_maniobra

7 ANNEXES

7.1 Esquema de tota la part electrònica



Title	ESQUEMA BAIXA POTÈNCIA	
Size A	Document Number 1	Rev 1
Date: Saturday, August 31, 2024	Sheet 48 of 58	

7.2 Codi complert

```
#define BLYNK_TEMPLATE_ID "RR" //Escriure el template_ID
#define BLYNK_TEMPLATE_NAME "SS" //Escriure el template name
#define BLYNK_AUTH_TOKEN "TT" //Escriure el auth token
#define BLYNK_PRINT Serial
#define DHTTYPE DHT11   // DHT 11
#define V_frio_calor V0
#define V_h_br V1
#define V_manual_auto V2
#define V_ac_proa V3
#define V_ac_popa V4
#define V_temp_consigna V5
#define V_h_es V6
#define V_histeresis V7
#define V_helice V8
#define V_terminal V9


#include "DHT.h"
#include <WiFi.h>
#include <WebServer.h>
#include <SimpleTimer.h>
#include <Preferences.h>
#include <WiFiClient.h>
#include <DNSServer.h>
#include <BlynkSimpleEsp32.h>
//#include <BlynkSimpleEsp8266_SSL.h>

Preferences preferences;
Preferences temperatura;
Preferences histeresi;
Preferences selector_popa;
Preferences selector_proa;
Preferences modo;
Preferences selector_frio_calor;
SimpleTimer timer;
WebServer server(80);
DNSServer dnsServer;

//PINES DE I/O
const int VOLT_SENS = 34;
const int TEMP_SENS = 32;
const int AIRE_PROA = 33;
const int AIRE_POPA = 26;
const int DHTPin = 14;
const int CNT_H_EST = 0;
const int CNT_H_BR = 0;
const int HELICE_ESTRIBOR = 17;
const int HELICE_BABOR = 4;


// Blynk virtual variables
  //AC
bool SEL_MAN_AUTO; // Manual by default
bool SEL_PROA;
bool SEL_POPA;
bool SEL_FRIO_CALOR;
float TEMP_CONSIGNA;
```

```
float TEMP = 0;
float HISTERESIS;
  //HELICE
int HELICE = 0;

//Variables software
  //AC
float Temp_Mitja_Alta = 0;
float Temp_Mitja_Baixa = 0;
boolean aire_popa = true;
boolean aire_proa = true;
float t;
  //VOLTAJE
int offset = -136.5;
int adc_value = 0;
float volt_ref = 3.3;
float adc_voltage = 0.0;
float in_voltage = 0.0;
int i_temp = 0;
int count_temp = 720;
int i_volt = 0;
int count_volt = 720;
  //CONTROL HORES
float Hores_Babor = 0;
float Hores_Estribor = 0;
bool RESET_h_br = false;
bool RESET_h_es = false;
bool ContantBR = false;
bool ContantES = false;
int TempsTranscorregutBR = 0;
int TempsTranscorregutES = 0;
int TempsIniciBR = 0;
int TempsIniciES = 0;

DHT dht(DHTPin, DHTTYPE);
WidgetTerminal terminal(V_terminal);

// Configuración del AP
const char* ap_ssid = "ESP32_AP";
const char* ap_password = "12345678";
IPAddress apIP(192, 168, 4, 1);
IPAddress netMsk(255, 255, 255, 0);

// Función para manejar la raíz del servidor web
void handleRoot() {
  String html = "<form action=\"/configure\" method=\"POST\">";
  html += "SSID: <input type=\"text\" name=\"ssid\"><br>";
  html += "Password: <input type=\"password\" name=\"password\"><br>";
  html += "<input type=\"submit\" value=\"Submit\">";
  html += "</form>";

  html += "<h2>Redes Preconfiguradas</h2>";
  html += "<button
onclick=\"window.location.href='/connectMarina'\">Marina</button>";
  html += "<button
onclick=\"window.location.href='/connectJavier'\">Javier</button>";
  html += "<button
onclick=\"window.location.href='/connectSergi'\">Sergi</button>";
  html += "<br><br><a href=\"/reset\">Resetear Credenciales</a>";
```

```
  server.send(200, "text/html", html);
}

// Función para manejar la configuración de WiFi
void handleConfigure() {
  String ssid = server.arg("ssid");
  String password = server.arg("password");

  // Guardar las credenciales en memoria no volátil (NVS)
  saveCredentials(ssid.c_str(), password.c_str());

  String response = "Credentials saved. Restarting...";
  server.send(200, "text/plain", response);
  delay(1000);
  ESP.restart();
}

// Función para guardar las credenciales en memoria no volátil (NVS)
void saveCredentials(const char* ssid, const char* password) {
  preferences.begin("wifi", false);
  preferences.putString("ssid", ssid);
  preferences.putString("password", password);
  preferences.end();
}

// Función para cargar las credenciales desde la memoria no volátil
(NVS)
void loadCredentials(String& ssid, String& password) {
  preferences.begin("wifi", true);
  ssid = preferences.getString("ssid", "");
  password = preferences.getString("password", "");
  preferences.end();
}

// Función para resetear las credenciales de WiFi
void handleReset() {
  preferences.begin("wifi", false);
  //preferences.clear();
  preferences.end();
  server.send(200, "text/plain", "Credentials cleared.
Restarting...");
  delay(1000);
  ESP.restart();
}

// Función para manejar la conexión a WiFi preconfigurada "Marina"
void handleConnectMarina() {
  const char* ssid = "XX";
  const char* password = "AA";
  saveCredentials(ssid, password);
  server.send(200, "text/plain", "Conectando a WiFi Marina.
Reiniciando...");
  delay(1000);
  ESP.restart();
}

// Función para manejar la conexión a WiFi preconfigurada "Javier"
void handleConnectJavier() {
  const char* ssid = "YY";
```

```
  const char* password = "BB";
  saveCredentials(ssid, password);
  server.send(200, "text/plain", "Conectando a WiFi Javier.
Reiniciando...");
  delay(1000);
  ESP.restart();
}

// Función para manejar la conexión a WiFi preconfigurada "Sergi"
void handleConnectSergi() {
  const char* ssid = "ZZ";
  const char* password = "CC";
  saveCredentials(ssid, password);
  server.send(200, "text/plain", "Conectando a WiFi Sergi.
Reiniciando...");
  delay(1000);
  ESP.restart();
}

void setup() {
  Serial.begin(115200);

  // Configurar AP
  WiFi.mode(WIFI_AP_STA);
  WiFi.softAP(ap_ssid, ap_password);
  WiFi.softAPConfig(apIP, apIP, netMsk);

  // Iniciar servidor DNS
  dnsServer.start(53, "*", apIP);

  // Configurar servidor web
  server.on("/", HTTP_GET, handleRoot);
  server.on("/configure", HTTP_POST, handleConfigure);
  server.on("/reset", HTTP_GET, handleReset);
  server.on("/connectMarina", HTTP_GET, handleConnectMarina);
  server.on("/connectJavier", HTTP_GET, handleConnectJavier);
  server.on("/connectSergi", HTTP_GET, handleConnectSergi);
  server.begin();

  Serial.print("AP IP address: ");
  Serial.println(WiFi.softAPIP());

  // Intentar conectar a la red WiFi guardada
  String ssid, password;
  loadCredentials(ssid, password);

  if (ssid != "") {
    WiFi.begin(ssid.c_str(), password.c_str());
    if (WiFi.waitForConnectResult() != WL_CONNECTED) {
      Serial.println("Connection Failed! Activating AP mode...");
    } else {
      Serial.println("Connected to WiFi!");
    }
  }

  Blynk.config(BLYNK_AUTH_TOKEN);
  Blynk.connect();

  temperatura.begin("blynk", false);
```

```
  histeresi.begin("blynk-1", false);
  selector_popa.begin("blynk-2",false);
  selector_proa.begin("blynk-3",false);
  modo.begin("blink-4",false);
  selector_frio_calor.begin("blynk-5",false);

  TEMP_CONSIGNA = temperatura.getFloat("temperatura", 20.0); // Valor
por defecto es 20.0
  HISTERESIS = histeresi.getFloat("histeresi", 0.1); // Valor por
defecto es 0.1
  SEL_PROA = selector_proa.getBool("selector_proa", false); // Valor
por defecto es OFF
  SEL_POPA = selector_popa.getBool("selector_popa", false); // Valor
por defecto es OFF
  SEL_MAN_AUTO = modo.getBool("modo",false); // Valor por defecto es
MANUAL
  SEL_FRIO_CALOR =
selector_frio_calor.getBool("selector_frio_calor",false); // Valor por
defecto es FRIO

  timer.setInterval(1000L, onesec);
  timer.setInterval(500L,manageHelice);
  timer.setInterval(5000L, fivesec);

  Blynk.setProperty(V_terminal, "color", "#646464");
  digitalWrite(HELICE_BABOR, HIGH);
  digitalWrite(HELICE_ESTRIBOR, HIGH);

  pinMode(AIRE_POPA, OUTPUT);
  pinMode(AIRE_PROA, OUTPUT);
  pinMode(HELICE_BABOR, OUTPUT);
  pinMode(HELICE_ESTRIBOR, OUTPUT);

  if(SEL_PROA){
    aire_proa = true;
    Blynk.virtualWrite(V_ac_proa,1);
    Blynk.setProperty(V_ac_proa, "color", "#1CFF27");
    digitalWrite(AIRE_PROA, LOW);
    Serial.println("SEL_PROA");
  }else if (!SEL_PROA){
    aire_proa = false;
    Blynk.virtualWrite(V_ac_proa,0);
    Blynk.setProperty(V_ac_proa, "color", "#EE1010");
    digitalWrite(AIRE_PROA, HIGH);
    Serial.println("!SEL_PROA");
  }
  if(SEL_POPA){
    aire_popa = true;
    Blynk.virtualWrite(V_ac_popa,1);
    Blynk.setProperty(V_ac_popa, "color", "#1CFF27");
    digitalWrite(AIRE_POPA, LOW);
    Serial.println("SEL_POPA");
  }else if (!SEL_POPA){
    aire_popa = false;
    Blynk.virtualWrite(V_ac_popa,0);
    Blynk.setProperty(V_ac_popa, "color", "#EE1010");
    digitalWrite(AIRE_POPA, HIGH);
    Serial.println("!SEL_POPA");
  }
```

```
  preferences.end();

  //WiFi.setSleep(true);
}

void loop() {
  dnsServer.processNextRequest();
  server.handleClient();
  //manageHelice();
  //manageControlHoras();
  //showWiFiSignalStrength();
  //manageControlHores();
  //manageBombaAchique();
  if (!SEL_MAN_AUTO) {
    manageManualMode();
  } else {
    manageAutoMode();
  }
  delay(50);
  if (WiFi.status() == WL_CONNECTED) {
    Blynk.run();
  }
  timer.run();
  delay(50);
}

void onesec(){
  if (WiFi.status() != WL_CONNECTED){
    //Serial.println("WiFi desconnectat");
  } else {
    //Serial.println("WiFi connectat");
  }
  if(!Blynk.connected()){
    //Serial.println("Blynk desconnectat");
  } else {
    //Serial.println("Blynk connectat");
  }
}

void manageControlHores(){
  if(RESET_h_br){
    Hores_Babor = 0;
  }
  if(RESET_h_es){
    Hores_Estribor = 0;
  }
  if(digitalRead(CNT_H_BR) == LOW){
    if(!ContantBR){
      int startTimeBR = millis();
      ContantBR = true;
    }
  } else if(ContantBR){
    TempsTranscorregutBR += millis() - TempsIniciBR;
    ContantBR = false;
  }
  if(ContantBR){
    Hores_Babor = TempsTranscorregutBR / 3600000;
    Blynk.virtualWrite(V_h_br, Hores_Babor);
```

```
    preferences.putFloat("HoresBabor",Hores_Babor);

    if(Hores_Babor >= 90.0){
      Blynk.logEvent("El motor de babor ha arribat a les 90 hores de
funcionament. Hora de manteniment!");
    }
  }

  if(digitalRead(CNT_H_EST) == LOW){
    if(!ContantES){
      int startTimeES = millis();
      ContantES = true;
    }
  } else if(ContantES){
    TempsTranscorregutES += millis() - TempsIniciES;
    ContantES = false;
  }
  if(ContantES){
    Hores_Estribor = TempsTranscorregutES / 3600000;
    Blynk.virtualWrite(V_h_es, Hores_Estribor);
    preferences.putFloat("HoresEstribor",Hores_Estribor);

    if(Hores_Estribor >= 90.0){
      Blynk.logEvent("El motor d'estribor ha arribat a les 90 hores de
funcionament. Hora de manteniment!");
    }
  }
}

void manageBombaAchique(){
  if(!ACHIQUE_ESPERA){
    BOMBA_ACHIQUE = digitalRead(18);
    if (BOMBA_ACHIQUE == LOW){
      //HI HA AIGUA A LES SENTINES
      Blynk.logEvent("S'ha detectat aigua a les sentines");

      TempsUltim = millis();
      ACHIQUE_ESPERA == true;
    } else if(millis() - TempsUltim == 600000) {
      ACHIQUE_ESPERA == false;
    }
  }
}

void manageManualMode() {
  if (SEL_PROA) {
    proa_on();
  } else {
    proa_off();
  }
  if (SEL_POPA) {
    popa_on();
  } else {
    popa_off();
  }
}

void manageAutoMode() {
  t = dht.readTemperature();
```

```
  Temp_Mitja_Alta = t - HISTERESIS;
  Temp_Mitja_Baixa = t + HISTERESIS;

  if (TEMP_CONSIGNA >= 10 && TEMP_CONSIGNA <= 40) {
    if (Temp_Mitja_Alta >= TEMP_CONSIGNA) {
      if (SEL_FRIO_CALOR){
        if (SEL_PROA) proa_off();
        if (SEL_POPA) popa_off();
      } else if (!SEL_FRIO_CALOR) {
        if (SEL_PROA) proa_on();
        if (SEL_POPA) popa_on();
      }
    } else if (Temp_Mitja_Baixa <= TEMP_CONSIGNA) {
      if (SEL_FRIO_CALOR) {
        if (SEL_PROA) proa_on();
        if (SEL_POPA) popa_on();
      } else if (!SEL_FRIO_CALOR){
        if (SEL_PROA) proa_off();
        if (SEL_POPA) popa_off();
      }
    }
  }
}

void manageHelice(){
  if (HELICE < 10){
    //estribor
    digitalWrite(HELICE_BABOR, HIGH); //Desconnecto babor
    delay(10);
    digitalWrite(HELICE_ESTRIBOR, LOW); //Connecto estribor
    Blynk.setProperty(V_helice,"color","#7D7777");
  } else if (HELICE > 245){
    //babor
    digitalWrite(HELICE_ESTRIBOR, HIGH); //Desconnecto estribor
    delay(10);
    digitalWrite(HELICE_BABOR, LOW); //Connecto babor
  } else {
    digitalWrite(HELICE_BABOR, HIGH); //Desconnecto babor
    digitalWrite(HELICE_ESTRIBOR, HIGH); //Desconnecto estribor
  }
}

void fivesec() {
  /*****************************
            TERMINAL
  *****************************/
  terminal.clear();
    //SHOW WIFI
  terminal.print("Conectado a: ");
  terminal.println(WiFi.SSID());
    //SHOW TEMPERATURA
  t = dht.readTemperature();
  terminal.print("Temperatura: ");
  terminal.print(t);
  terminal.println(" °C");
    //SHOW WIFI SIGNAL STRENGTH
  int rssi = -WiFi.RSSI();
  terminal.print("Señal WiFi: ");
  terminal.print(rssi);
```

```
  terminal.println(" dB");
    //SHOW VOLTAJE
  adc_value = analogRead(VOLT_SENS);
  Serial.println(adc_value);
  adc_voltage  = (adc_value * volt_ref) / 4095.0;
  in_voltage = adc_voltage /0.2;
  terminal.print("Voltaje Baterias: ");
  terminal.print(in_voltage);
  terminal.println(" V");
  /*****************************
            NOTIFICACIONS
  *****************************/
    //NOTIFICAR TEMPERATURA
  if(t>=30){
    if(count_temp < 720){
      count_temp ++;
    } else {
      count_temp = 0;
      if (i_temp<=12){
      i_temp++;
    } else {
      Blynk.logEvent("Temperatura del Americ de 30 °C. ¿Encender los
aires?");
      i_temp = 0;
    }
    }
  }
    //NOTIFICAR VOLTAJE
  if(in_voltage < 12){
    if(count_volt < 720){
      count_volt ++;
    } else {
      count_volt = 0;
      if (i_volt<=12){
      i_volt++;
    } else {
      Blynk.logEvent("Voltaje de las baterias de 12 V.");
      i_volt= 0;
    }
    }
  }

}

void popa_on() {
  if (!aire_popa) {
    digitalWrite(AIRE_POPA, LOW);
    Blynk.setProperty(V_ac_popa, "color", "#1CFF27");
    selector_popa.putBool("selector_popa", true);
  }
  aire_popa = true;
}

void popa_off() {
  if (aire_popa) {
    digitalWrite(AIRE_POPA, HIGH);
    Blynk.setProperty(V_ac_popa, "color", "#EE1010");
    selector_popa.putBool("selector_popa", false);
  }
```

```
  aire_popa = false;
}

void proa_on() {
  if (!aire_proa) {
    digitalWrite(AIRE_PROA, LOW);
    Blynk.setProperty(V_ac_proa, "color", "#1CFF27");
    selector_proa.putBool("selector_proa", true);
  }
  aire_proa = true;
}

void proa_off() {
  if (aire_proa) {
    digitalWrite(AIRE_PROA, HIGH);
    Blynk.setProperty(V_ac_proa, "color", "#EE1010");
    selector_proa.putBool("selector_proa", false);
  }
  aire_proa = false;
}

BLYNK_WRITE(V_frio_calor) {
  SEL_FRIO_CALOR = param.asInt();
  selector_frio_calor.putBool("selector_frio_calor", SEL_FRIO_CALOR);
}

BLYNK_WRITE(V_manual_auto) {
  SEL_MAN_AUTO = param.asInt();
  modo.putBool("modo", SEL_MAN_AUTO);
}

BLYNK_WRITE(V_ac_proa) {
  SEL_PROA = param.asInt();
  selector_proa.putBool("selector_proa", SEL_PROA);
}

BLYNK_WRITE(V_ac_popa) {
  SEL_POPA = param.asInt();
  selector_popa.putBool("selector_popa", SEL_POPA);
}

BLYNK_WRITE(V_temp_consigna) {
  TEMP_CONSIGNA = param.asFloat();
  temperatura.putFloat("temperatura", TEMP_CONSIGNA);
}

BLYNK_WRITE(V6) {
  TEMP = param.asFloat();
}

BLYNK_WRITE(V_histeresis) {
  HISTERESIS = param.asFloat();
  histeresi.putFloat("histeresi", HISTERESIS);
}

BLYNK_WRITE(V_helice) {
  HELICE = param.asInt();
}
```