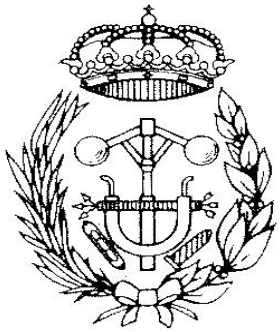


ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación :

INGENIERO TÉCNICO INDUSTRIAL ELÉCTRICO

Título del proyecto:

CATAMARAN RADIOCONTROLADO POR ORDENADOR

DOCUMENTO

Daniel Urtasun Cruz
Vicente Senosiáin
Pamplona, 15/11/2012

Computer radio controlled catamaran with night vision camera

INDEX

Objectives and general description:

Main:

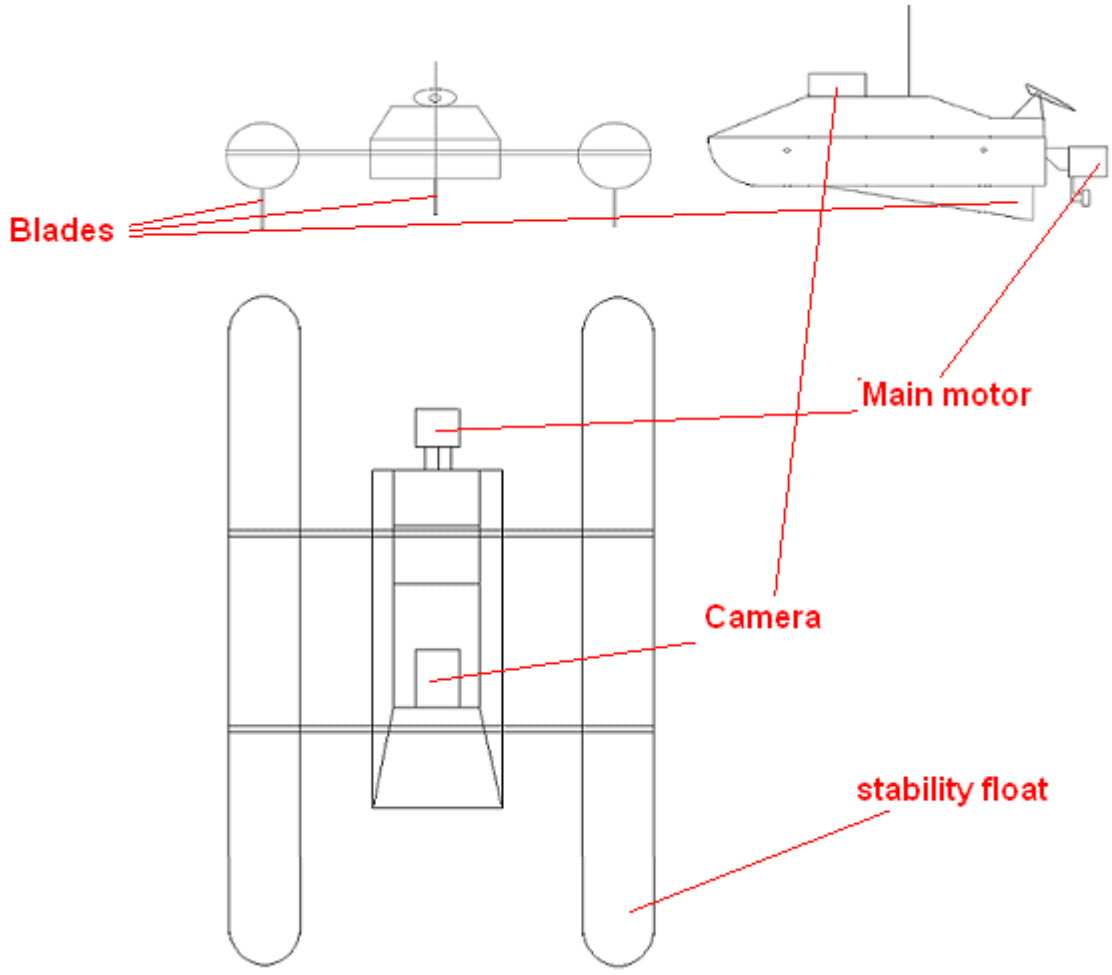
This project is meant to build a boat that is remotely controlled and is able to send video and audio in real time of the surroundings where it is located.

Secondary:

In case communication is lost, an automatic program is initialized making the boat move back to past positions where a communication link was established. This would require a GPS module.

This work is divided in several parts, once each of that parts have been tested and we are sure they work in the desired way, we can manage to make all of them work together and see if the whole system does work as we want.

First general idea gathering sketches:



First prototype design ideas:

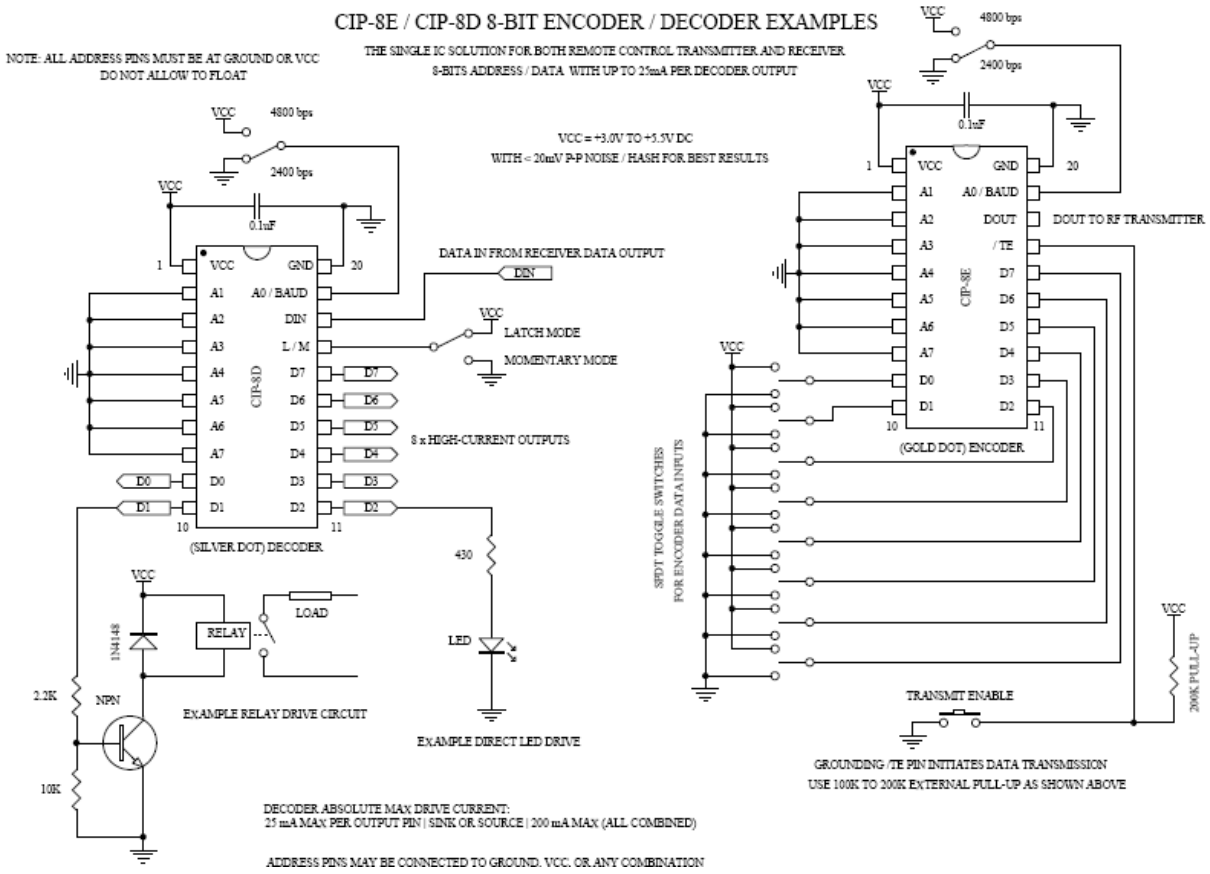
- To figure out where will the water level be, we use the physic law of flotation which states that the mass of water evacuated by the object equals the mass of that object.
- Boat's mass center should be located as low and centered as possible.
- With a given total mass, we will decide the appropriate hull dimensions to keep water level at a desired height.
- Catamaran style design is highly stable and must be considered.
- The chosen material is aluminum sheets, they are corrosion free, easy to cut and manipulate and heat conductors.
- Folding the sheets goes first, you can make holes to hold them in to flat surfaces and work comfortably. Finally make the cuts.
- The metallic joints are attached with silicone, it seals and isolates properly from moisture.

Components:
-One main motor.
-One electronic speed controller or ESC with BEC mode.
-A propeller, shaft, rudder and other items for the mechanical set up.
-A servo for the rudder position and thus direction control.
-A servo for the camera orientation
-The video/audio camera.
-An AV transmitter.
-Two 4200mA/h, 12v high discharge current battery packs for the whole system.

Remote control module testing:

A receiver and transmitter module pair have been purchased and we still need to put them to the test in a practical circuit to make an idea of the performance and limitations they show.

Basically, the most simple diagram of transmission is shown below:



It uses one encoder to codify binary data from the inputs (in this case we have 8 switches max) and this encoder puts this bytes into packets that are sent to the transmitter module in serial communication. The transmitter sends this info wireless through a 315MHz radio wave to the receiver and then the decoder unpacks it and shows the output again as a 8 bit (1byte) binary output.

Encoder/decoder:

Each packet has only 2 data bytes and another byte associated that identifies which decoder those data bytes belong to (direction byte).

A 10mS guard time is inserted between each encoded packet transmission to allow the decoder time to receive, decode, verify, and process each packet.

Encoder/decoder features:

- Latched or momentary outputs

- Low consumption mode (TE pin)
- Variable baud rates at 2400/4800 bits per second (consider that lower baud rates are more reliable).
- Supply voltage:..... 5V (min 3V)
- Supply current:..... 0,8mA (without active loads)
- Total power dissipation:..... 800mW

Transmitter and receiver modules:

LINX-TXM-315-LR
LINX-RXM-315-LR

Features in both modules:

- Range:..... 3,000 feet ≈ 1Km
- Supply voltage:..... 3V

Input output pins:.....3v "Please verify that the minimum voltage will meet the high threshold requirement of the device to which data is being sent."
→ decoder (threshold input at 0,8Vcc)

- Operating temperature:...70ºC
- Supply current:..... 3,4mA (transmitter)5,2mA (receiver)
- Max baud rate:..... 10000bits per second.

Transmitter and receiver must have a 3V supply thus we must adapt to 3v the encoder data output and to 5V the decoder data input being encoder and decoder connected to 5V.

1.Transmitter

PIN ASSIGNMENTS

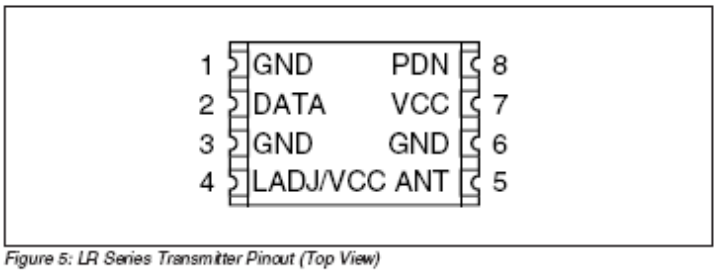


Figure 5: LR Series Transmitter Pinout (Top View)

PIN DESCRIPTIONS

Pin #	Name	Description
1	GND	Analog Ground
2	DATA	Digital Data Input
3	GND	Analog Ground
4	LADJ/V_{CC}	Level Adjust. This line can be used to adjust the output power level of the transmitter. Connecting to V_{CC} will give the highest output, while placing a resistor to V_{CC} will lower the output level (see Figure 4 on Page 3).
5	ANT	50-ohm RF Output
6	GND	Analog Ground
7	V_{CC}	Supply Voltage
8	PDN	Power Down. Pulling this line low will place the transmitter into a low-current state. The module will not be able to transmit a signal in this state.

ABSOLUTE MAXIMUM RATINGS

Supply Voltage V_{CC}	-0.3	to	+3.6	VDC
Any Input or Output Pin	-0.3	to	V_{CC} + 0.3	VDC
Operating Temperature	-40	to	+85	°C
Storage Temperature	-40	to	+90	°C
Soldering Temperature	+225°C for 10 seconds			

***NOTE* Exceeding any of the limits of this section may lead to permanent damage to the device. Furthermore, extended operation at these maximum ratings may reduce the life of this device.**

2.Receiver

PIN ASSIGNMENTS

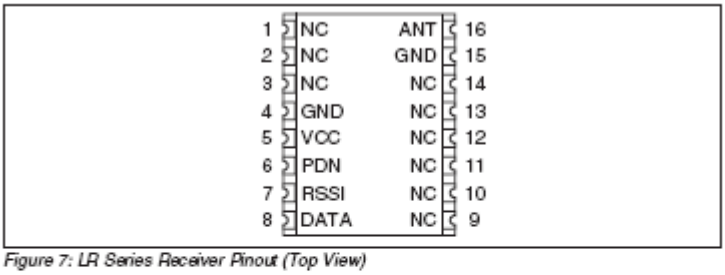


Figure 7: LR Series Receiver Pinout (Top View)

PIN DESCRIPTIONS

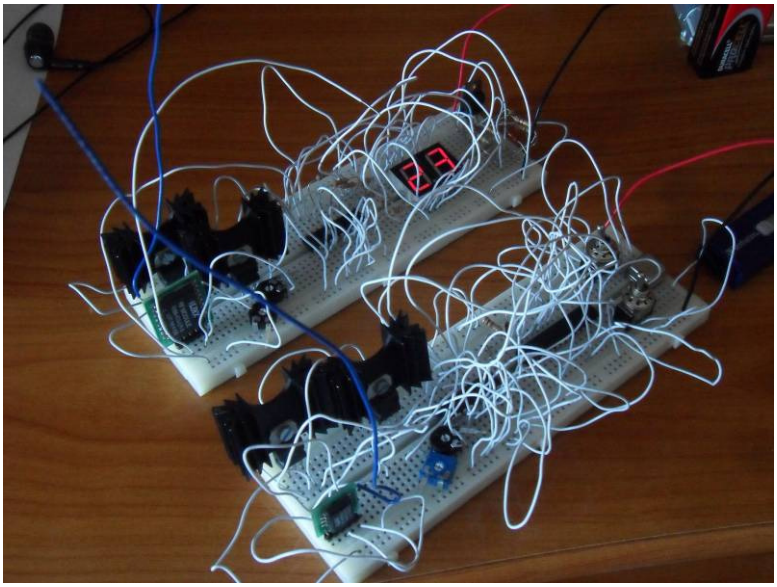
Pin #	Name	Description
1	NC	No Connection
2	NC	No Connection
3	NC	No Connection
4	GND	Analog Ground
5	V_{CC}	Supply Voltage
6	PDN	Power Down. Pulling this line low will place the receiver into a low-current state. The module will not be able to receive a signal in this state.
7	RSSI	Received Signal Strength Indicator. This line will supply an analog voltage that is proportional to the strength of the received signal.
8	DATA	Digital Data Output. This line will output the demodulated digital data.
9	NC	No Connection
10	NC	No Connection
11	NC	No Connection
12	NC	No Connection
13	NC	No Connection
14	NC	No Connection
15	GND	Analog Ground
16	RF IN	50-ohm RF Input

ABSOLUTE MAXIMUM RATINGS

Supply Voltage V_{CC}	-0.3	to	+3.6	VDC
Supply Voltage V_{CC}, Using Resistor	-0.3	to	+5.2	VDC
Any Input or Output Pin	-0.3	to	+3.6	VDC
RF Input		0		dBm
Operating Temperature	-40	to	+70	°C
Storage Temperature	-45	to	+85	°C
Soldering Temperature	+225°C for 10 seconds			

***NOTE* Exceeding any of the limits of this section may lead to permanent damage to the device. Furthermore, extended operation at these maximum ratings may reduce the life of this device.**

Circuit Testing.



Number transmitting RF circuit.

Observation:
The circuit does reach long distances with no obstacles in between but the signal weakens if a wall or any other obstacle appears.

After trying two small projects and gaining enough confidence working with this devices it's time for the next level.

Camera, AV transmitter and receiver:

An audio and video transmitter and receiver have been obtained from a Hong Kong store for a low cost making it quite suspicious of possible scam.

They have the remarkable feature of being able to transmit for up to 1km distances with few interferences (the same range the other RF devices have) and a radio channel frequency of 0,9GHz which is far beyond 315Mz and does not interfere with typical 2,4 wifi or Bluetooth frequencies found almost everywhere.

And what is more important: 2,4GHz frequencies are highly absorbed by water or moisture, this means that a boat would reach shorter distances than if it was transmitting the same power with a 0,9GHz antenna.

We also have a small camera that suits the required connections and is even able to let us see in the dark with IR (infra red) leds.

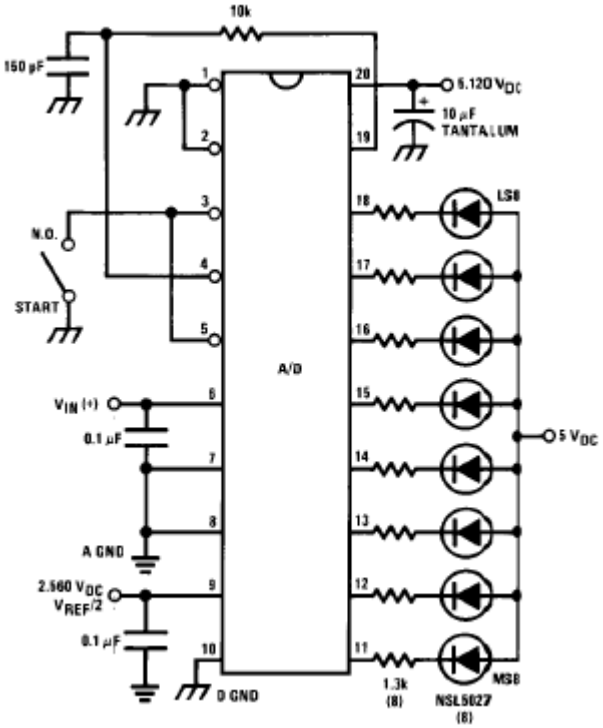
After setting all up, the behavior of the AV equipment has been studied and amazingly everything was working as it should, no fails at all, and the communication range does reach long distances too.
Just one problem: The transmitter's temperature rises too much, additional dissipation methods are needed, otherwise nearby circuits could be damaged.

This last problem could perfectly be solved using a metallic hull in order to have more heat dissipating surface in our model boat, something like aluminum sheets would be great.

First approach working with servos, micros and AV equipment:

A first integrated circuit, the ADC0804, takes an analog signal from pin 6, makes an analog-digital conversion and sends one byte output to the encoder, next, to the decoder and finally to the micro. This last pic 16f886 will take the binary input and use it to make a PWM that will finally control the servo attached to the camera.

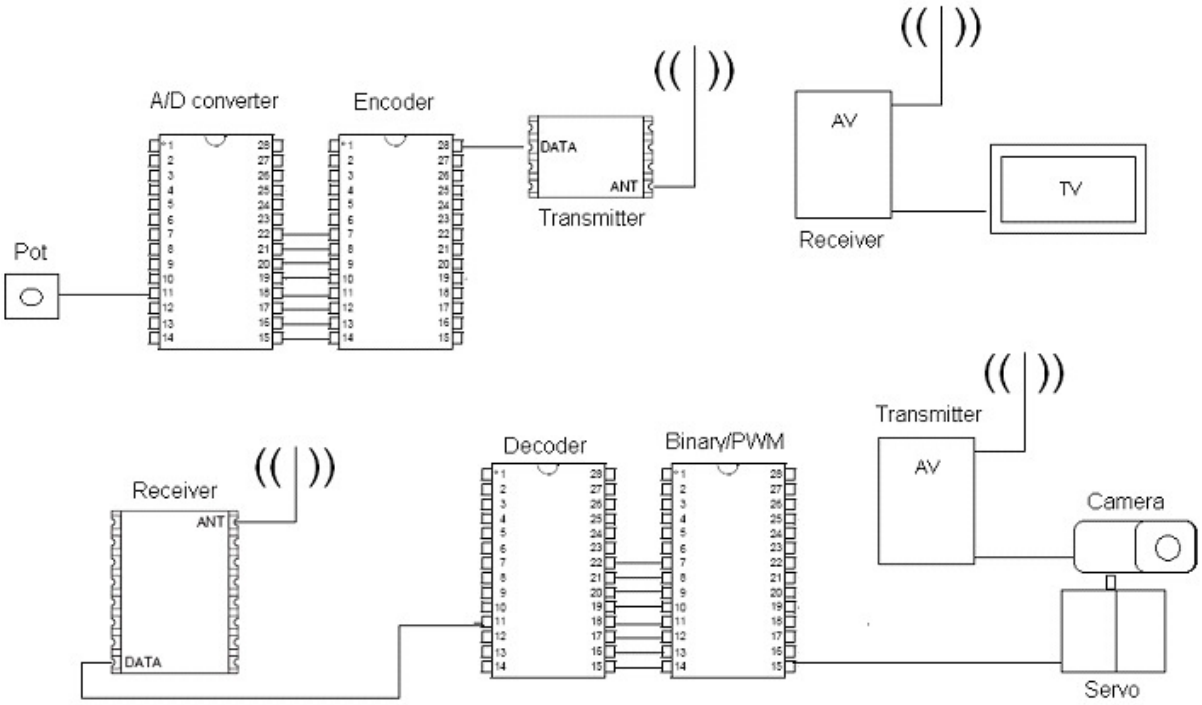
Below are shown the connections for the ADC0804 IC test (Analog to binary):



Assembly code for the pic 16f886 micro: (Binary to PWM conversion).

```
List      p=16F886                ;Microprocessor type
          include  "P16F886.INC"  ;Internal register definition

          __config          _CONFIG1,
_LVP_OFF&_PWRTE_ON&_WDT_OFF&_EC_OSC&_FCMEN_OFF      ;First configuration word
          __config          _CONFIG2, _WRT_OFF&_BOR40V          ;Second configuration word

Temporal          equ       0x20          ;Temporal variablePeriod          equ       .1250
        ;Period 20000uS (1250*16 Prescaler)

                  org       0x00          ;Reset vector
                  goto      Inicio
                  org       0x05

;Main program

Beginning         clrf      PORTC         ;Output cleaning
                  bsf       STATUS,RP0
                  bsf       STATUS,RP1    ;Banc 3
                  clrf      ANSEL
                  clrf      ANSELH        ;A and B digital ports
                  bcf       STATUS,RP1    ;Banc1
                  movlw     b'11111011'
                  movwf     TRISC         ;RC2 output
                  movlw     Periodo-1
                  movwf     PR2           ;period register loading
                  bcf       STATUS,RP0    ;Selects banc 0

;CCP1 module is working in the PWM mode with signal output through RC2/CCP1

                  movlw     b'00001100'
                  movwf     CCP1CON

;TMR2 Is working with a 1:16 preescaler so with a 4MHz frecuency evolves
;every 16uS ((4*Tosc)*16)

                  movlw     b'00000111'
                  movwf     T2CON         ;T2 On

;adjusts pulse width with the given input RA5:RA0
Loop      movf    PORTA,W               ;RA5:RA0 represent a n value

                  andlw     b'11111111'
                  movwf     CCPR1L        ;Pulse width adjust (n*16 Prescaler )
                  goto      Loop          ;Endless loop

                  end                     ;End of source program
```

First application:



Now we have taken the required steps to control a camera from the distance, this way we test how do camera and servos perform with two different wireless channels working simultaneously checking possible interferences and fails.

Everything its been done step by step in the most organized way, there have been a couple of problems setting up this circuit though, a servo was damaged, in other case the operating frequency needed was higher than the supplied, if not, we found voltages or currents that didn't belong, missing pins and all kind of obstacles.

But after checking datasheets, thinking thoroughly, taking care of voltages and making some other adjustments the small circuit was successfully working in no time and the camera was perfectly RF controlled being able to give us real time images too with zero interferences.

Boat control computer interface

This step has been once again challenging and difficult to solve in the beginning, I wanted full control over boat parameters, being able to see them on the PC screen, use the keyboard to make any changes and obviously everything in a practical and reliable basis.

After some research work I found AutoHotkey, a free, open-source utility for Windows that enables the user to automate almost anything, create macros and hotkeys for the keyboard and mouse by programming in a C related language.
As soon as I found it, I started learning from tutorials and some time later, worked in the design of the interface.

AVerTV is another software that allows the user to accomplish a variety of tasks and has the potential of converting an analogue output of a surveillance camera into a USB signal (remember that until now we needed a TV screen with AV input). In essence, is the way we can see what's going on through the camera in the computer instead of in a television, it's nothing but another homemade adaptation.

USB to Digital output software and hardware

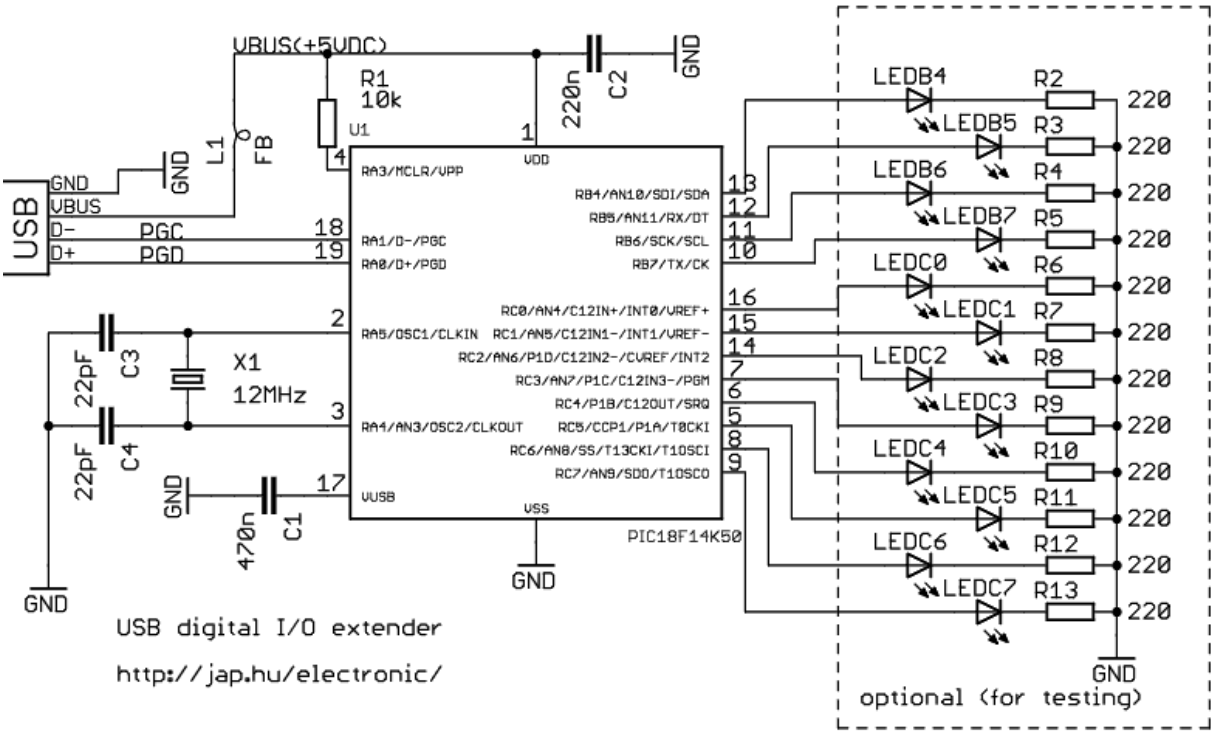
Some internet research over this topic leads to a small project that fits perfectly what we are looking for, you can find it at **http://jap.hu/electronic/usbio.html**
I'll just show the most interesting parts below :

description

Add general purpose input/output lines to your computer based projects. This project is a 12-bit digital I/O extender using the Microchip PIC18f14k50 microcontroller which connects to an USB host port. The microcontroller is available in through-hole DIP20 and SMD packages, too.

Circuit diagram

The device is powered by the USB bus. The ports RB4-7, RC0-7 can be set on a bit-by-bit basis to input or output direction. All LEDs on the schematic are optional, and are only shown for testing the device. You can find the USB connector pinouts at http://pinouts.ws/usb-pinout.html
I have an SMD version PCB for the circuit, which will be published here. If you don't want to make your own PCB, you can buy an MCP2200 breakout board. The MCP2200 is a PIC18F14K50 mcu pre-programmed with firmware from Microchip.



Controlling the I/O extender

A computer or an embedded RISC board with an USB host port can be used to control the I/O extender. The controlio example code uses libusb. This means that you don't need a kernel driver to interface with it.

The controlio example code can be invoked with these parameters:

command line	description
controlio getport	read the digital I/O port states
controlio setdir <n>	set the digital I/O port directions. Use bit 1 to set direction to input, bit 0 to set direction to output *<n> is a 16-bit decimal number between 0-65535*
controlio setport <n>	set the digital I/O port output states. Sets all output I/O pins. Use bit 1 to set output to HIGH, 0 to set output to LOW *<n> is a 16-bit decimal number between 0-65535*
controlio setbit <n>	set the selected digital I/O port output pins to HIGH. Use bit 1 to select I/O pins to set *<n> is a 16-bit decimal number between 0-65535*
controlio clearbit <n>	set the selected digital I/O port output pins to LOW. Use bit 1 to select I/O pins to clear

Compiling the host side code under windows

- download and install mingw and msys from mingw.org
- extract files from usbio.zip c/ directory to c:\mingw\msys\1.0\home\user
- download and install libusb-win32 from sourceforge
- copy include/usb.h from the libusb package to c:\mingw\msys\1.0\include
- copy lib/gcc/libusb.a from the libusb package to c:\mingw\msys\1.0\lib
- start the mingw shell from programs->mingw->mingw shell
- compile the code with

```
gcc -o controlio controlio.c /lib/libusb.a -I/include
```

- connect the USB device to the computer
- start bin/inf-wizard from the libusb package
- select the connected device from the list, then create and install a driver for it
- find the executable controlio.exe under c:\mingw\msys\1.0\home\user
- check operation with the command

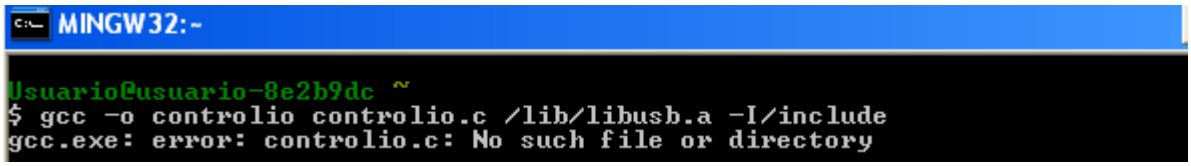
```
controlio getport
```

Download

Download usbio.zip, contents:

directory	description
c/	source code for the USB host, please check section "controlling the I/O extender"
pic18f14k50/	source code and compiled HEX file (firmware) for the Microchip pic18f14k50 microcontroller

Even if everything is so clear, there were problems while trying to compile:



Solution: Extract controlio.c instead of the given “c” folder in C:\mingw\msys\1.0\home\user

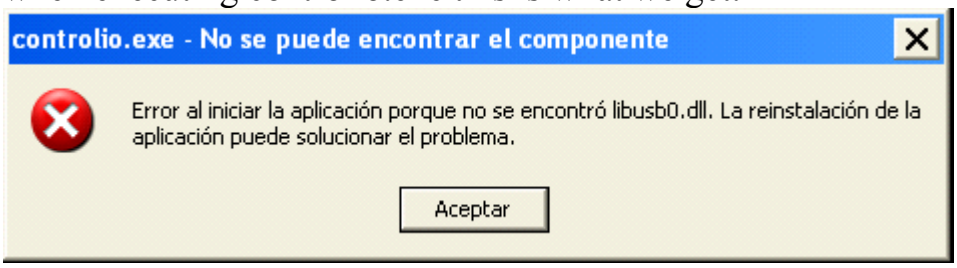
Solution: libusb package contains a file with different name from usb.h in the so called “include” folder, change it to usb.h or controlio.c program won’t compile properly.

This is how the output looks like when everything is fine:

```
Usuario@usuario-8e2b9dc ~
$ gcc -o controlio controlio.c /lib/libusb.a -I/include

Usuario@usuario-8e2b9dc ~
$
```

Now we have controlio.exe in C:\mingw\msys\1.0\home\user

when executing controlio.exe this is what we get:



In the libusb-win32-bin-README (inside "bin" folder) we find the following quote:

"ALL ARCHITECTURES:
  x86\libusb0_x86.dll: x86 32-bit library. Must be renamed to libusb0.dll"

So we just do that and extract that file where controlio.exe is. Now it's all right.

To build the required hardware, we program the PIC 18F14k50 with the supplied firmware and proceed reproducing the connections seen in the schematic (PIC and USB pins below). After everything is verified, its time to start inf-Wizard. It will ask us for a connected device which will be “jap.hu USB firmware” and will generate some drivers for it.
We are not finished yet, last thing we need to do is to open the hardware add assistant that will give us finally the chance of associating our hardware with those drivers created.

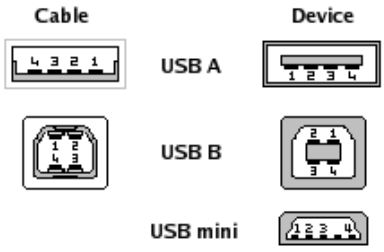
Hardware add assistant location:
Panel de control/Impresoras y otro hardware/Agregar hardware

PIC’s pin-bit assignment:

bit	0	1	2	3	4	5	6	7	8	9	10	11
pin	16	15	14	7	6	5	8	9	13	12	11	10

USB pinout:

Pin	Signal	Color	Description
1	VCC		+5V
2	D-		Data -
3	D+		Data +
4	GND		Ground



Things to point out about MinGW Shell:

Useful commands for this project:

•controlio setport N

N is a BCD (Binary coded in decimal) number where each binary digit represents a pin state: High (1) or Low (0).(it affects all the pins)

•controlio setbit N

In this case it only affects the pins that belong to "1" digits sending them to High state and ignoring the rest ("0" does not affect any pin)

•controlio clearbit N

In opposition to setbit, clearbit will only apply for pins linked to a "0" digit, in which case will set that pins to low state.("1" has no effect).

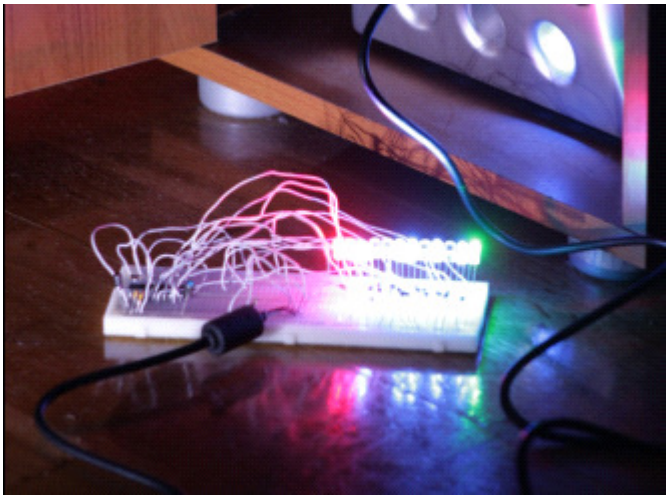
Decimal to binary conversion particularities

Decimal numbers range from 0 to 65535 and we have 12bits available giving us $2^{12} = 4096$ possible combinations ¿is this a nonsense? well, I think so.
We won't get rid of some testing again to figure this thing out:
Checking outputs we finally find that from numbers 0 to 255 we get the correct output (255 being the first 8bits to 1) but at 256 the ninth bit won't set to 1 and at 257 the count continues as if the ninth bit was indeed 1.

Solution to this apparent loss of useful pins: the ninth pin sets at 4096 and the count continues correctly up to 65535:

Decimal	0 to 255	4096 to 65535
Binary	000000000000 to 000011111111	000100000000 to 111111111111

Functioning hardware and software:

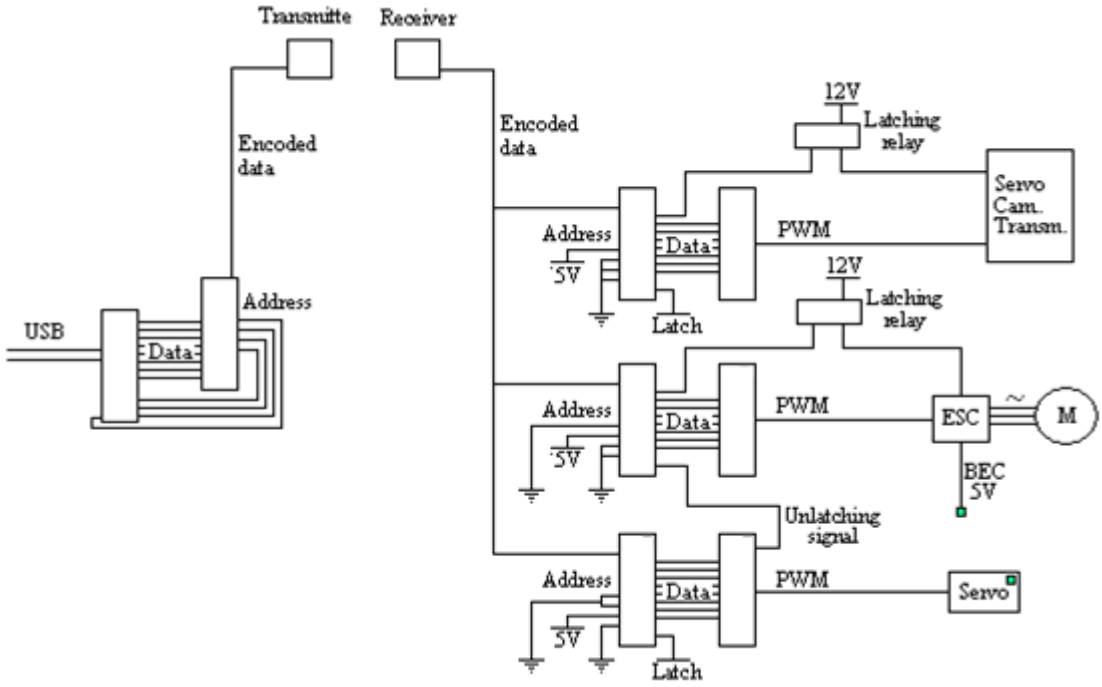


Final remarks:

We finally got it working under Win XP but when we tried to set everything all over under Win 7 (laptop computer) we had new problems that needed to be solved, it took some time to figure this out but finally we had to modify part of the controlio.c program in order to get it working in our laptop as well.

Electrical and electronic systems planning.

The following scratch diagram shows what my goal is, some things have been omitted to add more simplicity and make it understandable:



.

Relays used are latching type or biestable aiming at power saving and for their control, proper drivers are needed (not shown). Taking care of how much current coils take (f.e. using a capacitor) will avoid sudden voltage falls that could affect other components.

All decoders will be in latch mode at all times except the motor controller one, which as we said, must be variable.

The ESC (electronic speed controller) has a voltage regulator to 5v (BEC) and will feed the rudder servo, which is used when general power is in "ON" mode. The rest of the IC's will work under a 5v voltage regulator excluding the transmitter and receiver (3v), the USB powered micro and its surrounding friends.

Relays and its drivers

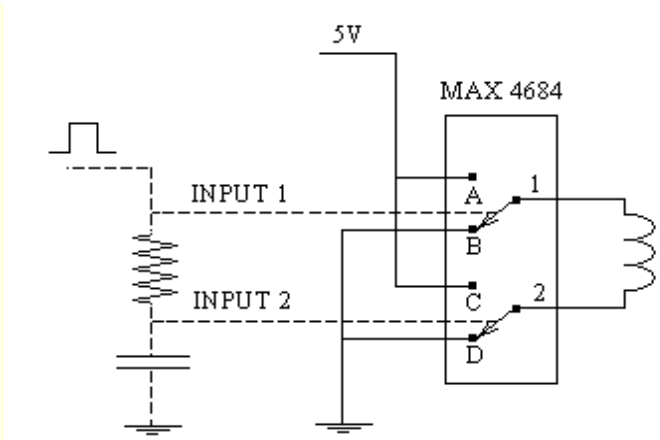
Following up, some soldering must be done: Relay drivers are provided in umax packaging so pins are not arranged in the commonly used dip style proportions. This means that we need to solder them to a umax-10 to dip-10 adapter board.

We have two different relays, here the most important features:
1.SPDT mini latching relay single coil: (For camera, servo and transmitter part)
- Coil nominal voltage: 5V
- Coil resistance : 125Ω
- Rated or nominal amperage: 5A

2.SPDT latching relay single coil: (For motor and direction servo)
- Coil nominal voltage: 5v
- Coil resistance: 42Ω
- Rated amperage: 12-20 A

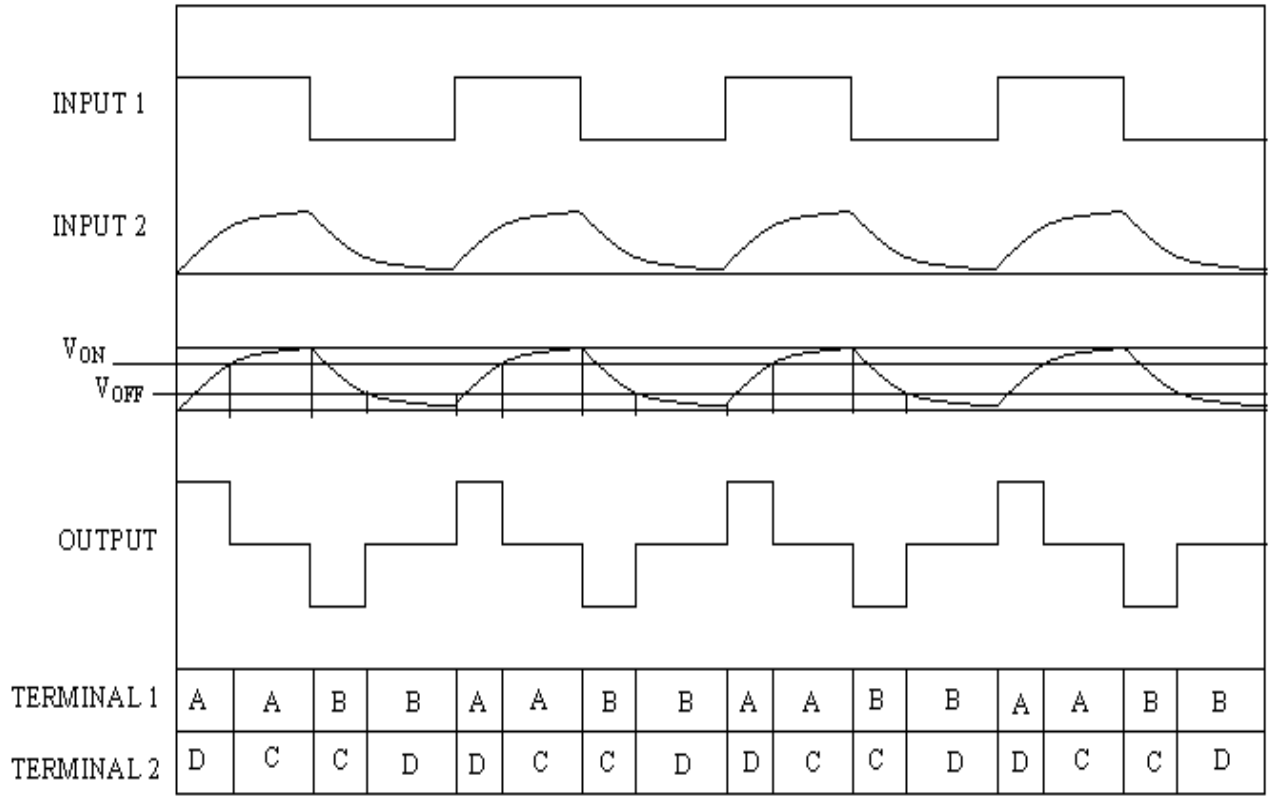
Drivers must be able to supply enough current (120mA at least) without suffering. Chosen ones are the Maxim 4684 for both relays, they can supply up to 3000mA continuously and are powered by a 5v source, which is adequate.

Single coiled relay driver setup:

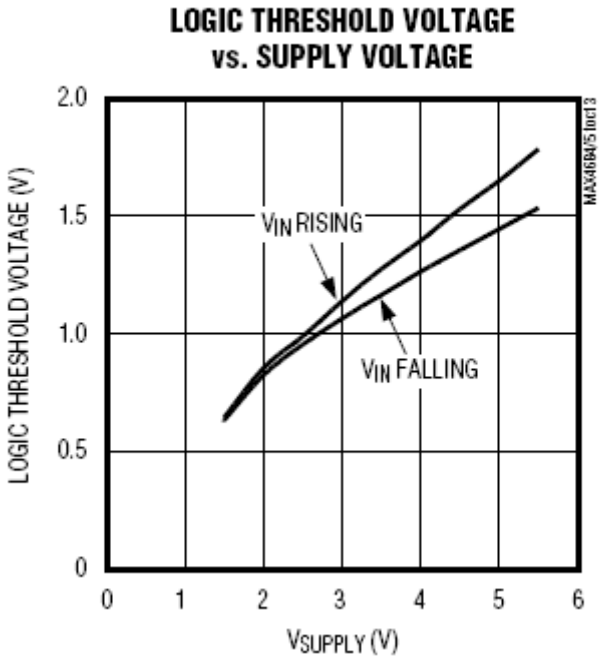


Depending on the relay switching time ($<T_{ON}$) we need to figure out R and C, This values will also be influenced by the threshold voltages of terminal 2 when switching on (V_{ON}) or off (V_{OFF}).

Waveform diagram representation for periodic switching:



From Maxim 4684 datasheet :



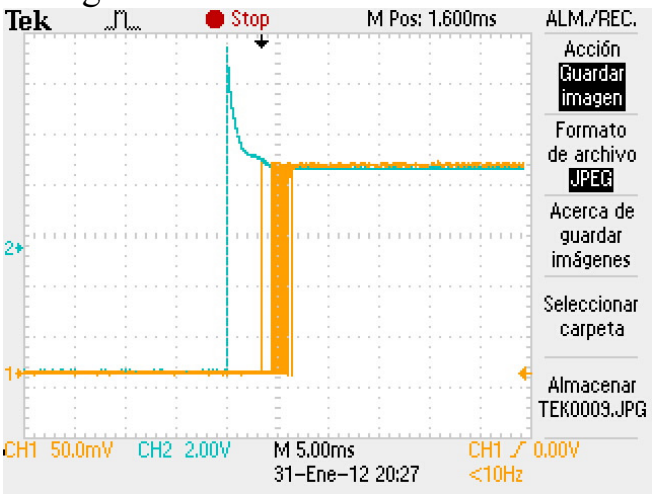
We take values for 5v supply and get: V_{ON}=1,7v V_{OFF}=1,

First relay driver RC value calculation:

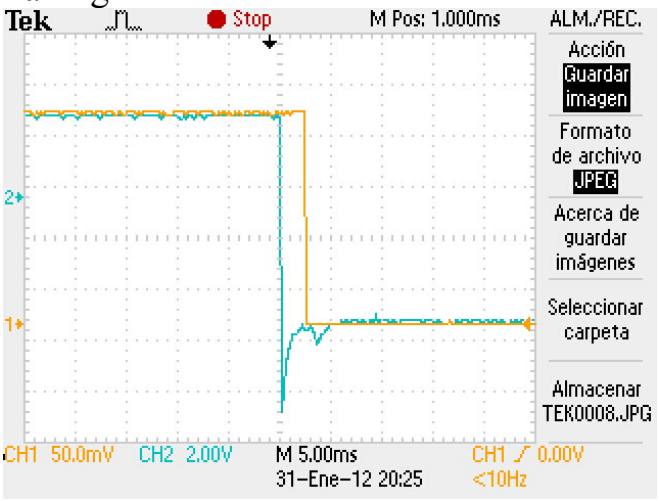
Coil data from relay datasheet:

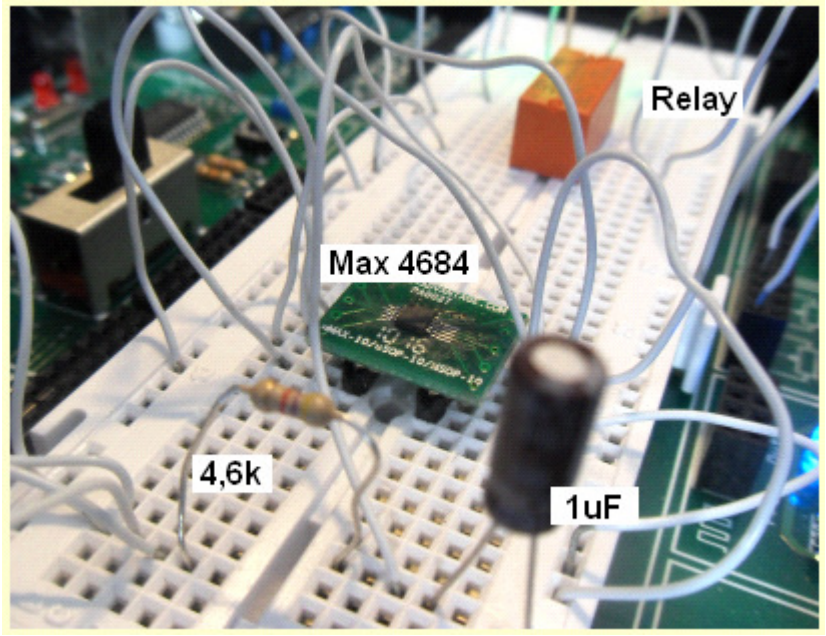
Minimum energization time	20 ms
Information on reduced pulse duration with higher energization voltages on demand	

This means that theoretically, we need at least τ=0,048s for coil energization, but some experimental testing shows the limit close to τ=0,002s.This happens because the 20ms given in the datasheet belongs to the higher coil voltage version (24v) (the slowest switching of all available).
Finally, as a precaution and reliability measure, next parameters are chosen: R=4.6KΩ, C=1μF (τ=0.0046s) and T_{ON} ,T_{OFF} >250ms (Sleep 250).
Oscilloscope analysis:
Rising



Falling





Relay 1 control setup test.

Second relay driver RC value calculation:

From datasheet:

Minimum energization duration	30 ms

Applying the same calculations as before we would need theoretically τ=0,072s at least but once again, this is for the slowest of all relays on the datasheet.

In practice we will use a 10uF capacitor and a 2,2KΩ resistor.

Contact information:

Rated current	16 A
Limiting continuous current	16 A, UL: 20 A

This means that the nominal current is 16A and the maximum breaking current capacity according to UL is 20A.

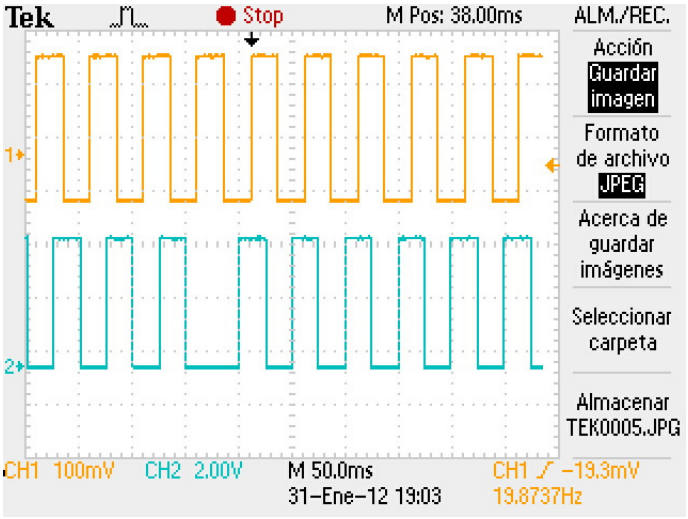
Final considerations:

The transient could affect other components if we don't take any care. This can be somehow counteracted by introducing another capacitor in the scheme to maintain the voltage constant and avoid sudden drops during the coil's magnetization process.

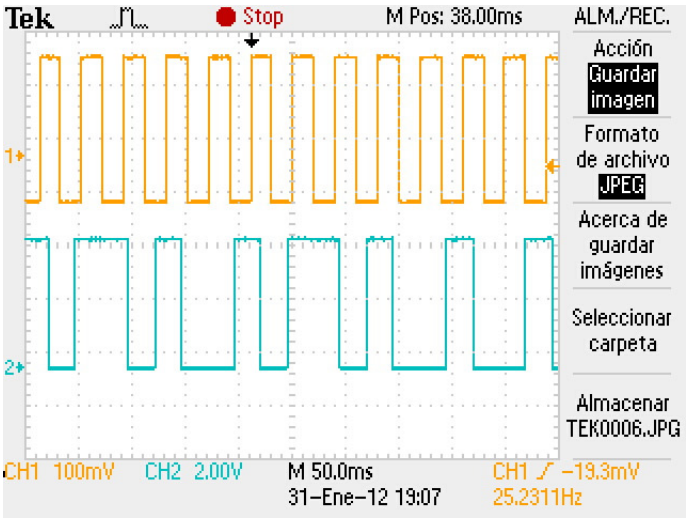
Encoding data rate characterization

Encoding operating limits:
CH1=Encoder input, CH2=Decoder output

Frequency limit



Over frequency limit



Conclusion: The maximum operating frequency equals 20Hz.

From this point we can establish the parallel data transmission rate according to our needs:

Standard ..... 1byte/25ms (fastest)
Reliable ..... 1byte/30ms (avoids byte "jumping"; see pictures above)
Very reliable ..... 1byte/≥60ms (avoids "jumping" and transmission errors)

When programming, sleep functions will allow important data to be securely sent.

Communication link check, error detection and correcting.

Even after we did almost everything possible in our hands to avoid any communication errors, they still occur and can't be overlooked.

Problem description
First of all, we test the camera servo part and find an error sent randomly in time that switches the relay off and puts the camera in 0 position (all the way to the right) with no apparent reason. Said in a different way, it sends zeroes to all data bits from time to time.

Problem fixing
The main idea is that the only valid data considered must meet at least one of this conditions to "filter" the pernicious data bursts:

1.The relay is ON
2.The relay is off but data is different from 0

This means that when we get an all-zero data burst, we will simply ignore it.

So we modify the assembly code in the loop and make some other modifications in ports as well (relay signal input and output in port C):

```
Loop    movf    PORTA,W             ;RA5:RA0 represent a n value
                BTFSS   PORTC,5     ;Pin 6 (bit 5) is the relay value input
                goto    Label1      ;When relay value is 0 it goes to label1
                andlw   b'11111111' ;When relay value is 1 it works normally
                movwf   CCPR1L      ;Pulse width adjust (n*16 Prescaler )
                bsf   PORTC,4       ;Relay signal output to high, Pin 5 (bit 4)
                goto    Loop
```

```
Label1
        movf    PORTA,0
        btfss   STATUS,Z        ;Checks 0 flag
        goto    notzero
        goto    Loop            ;If relay signal is 0 and PORT A is 0 too then ignore (error situation)
        notzero                 ;If relay signal is 0 and Port A is different from 0 then send relay signal
                bcf   PORTC,4   ;Clear relay output if data is different from 0 (no errors)
                goto    Loop
end                             ;End of source program
```

Everything is working just fine now.

Quote: The origin of this error has been described later, it is caused by the interrupt subroutine that sends pulses periodically for the communication link verification and a better way of solving this is making all threads critical so that no one can interfere until they are done. This is why we now use the “critical” command at the beginning of each routine.

Position and alarm lights

It seems convenient to install some external lighting that will fit the alarm requirements making them blink in case communication is lost, battery ran out or simply because we want to make the boat visible at night for example.

For this reason we chose some backlights that are powered by a 12v source. They are bright and large enough to be seen from the distance and compared to other models, the most efficient. In addition to this, with an internal resistance of 600Ω there is no need of additional resistors so they are out of any doubt, the best option.

Choosing to put all 6 lights in parallel we get a total resistance of 100Ω and a transistor will finally close all this circuit with ground, or leave it open.

White led Backlight

From transistor datasheet we realize that for $5V_{GS}$ and a current demand of ≈120mA, (six backlights in parallel) our operating point belongs to a drain to source voltage drop of less than 0,06v (0,5%). Furthermore, our transistor is mosfet type, which is voltage activated making it quite efficient as well.

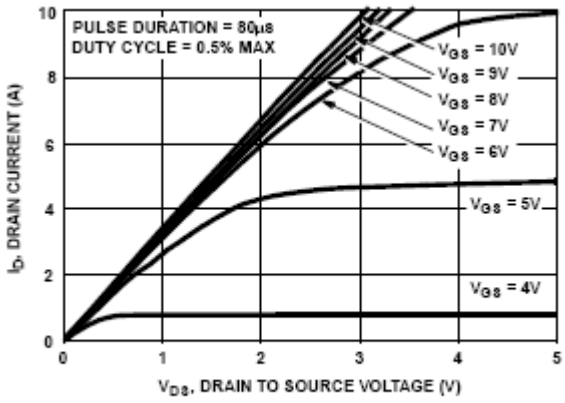


FIGURE 6. SATURATION CHARACTERISTICS

Voltage regulators

Switching regulator (12v to 5v):

We could choose a switching regulator as a main voltage supply for all circuitry, because it is more energy efficient. The side effect here is that our output is chopped and needs to be filtered. This would not be critical if it wasn't because this noise could affect significantly receiver sensitivity and as it's said in the datasheet, we need to keep this noise always below 20mV.

We don't want to take any risk with the possibility of ripple voltages affecting our electronics so we will use an independent linear power supply acting as the main 5v supply for all our circuitry and will use the switching one just for the higher power demanding device (camera's servo) with its proper heat sink attached.

Here the typical switching regulator's setup with the input capacitor, schotky diode, a 2 order passive filter (LC), the feedback line and we also have a ON/OFF signal:

Typical Application (Fixed Output Voltage Versions)

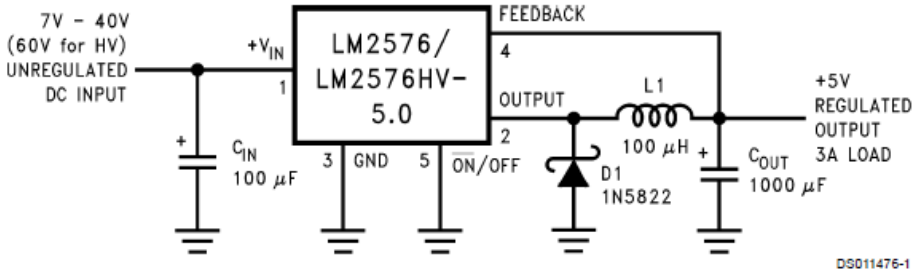


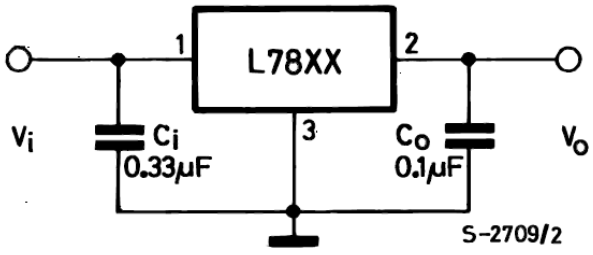
FIGURE 1.

A 470uF capacitor and a 100uH inductor with a 47uF Cin seems to work just fine with our servo so that's what we will use.

Linear regulator (12v to 5v):

We explained above the reason of using this regulator: It may not be very energy efficient but for low current consumption cases that doesn't really matter much and its clean output is preferred for our electronics (specially our receiver) as seen on the warnings given in its datasheet.

Another visible advantage is the design simplicity and polyvalence as we can appreciate below:



USB Part regulator (5v to 3,3v):

This is a LDO or low dropout regulator which will be linear to avoid noise issues. The transmitter requires a 3v supply voltage, the same as the receiver, in the first case we put this LDO regulator but in the second case we put a 330Ω resistor in series and we can connect it to 5v (This possibility is clearly indicated in its datasheet):

"The LR can utilize a 4.3 to 5.2VDC supply provided a 330-ohm resistor is placed in series with VCC."

Low battery charge indicator

What would happen if for any reason we ran out of battery in the middle of somewhere we can't get trough? well as we know that's not a desirable situation, that leads us to build a convenient circuit that can identify whether or not the battery pack needs to be refilled.

Working description:
The principle of operation is simple, we will take advantage of the fixed reverse voltage of some special components called Zener diodes (I assume you know what a Zener diode is, if not, please make your own research). When the battery is charged, its voltage will be over this particular Zener volts and a current will flow through the diode so we could just put two series resistors making a divider and take an analog signal out of them. This way, by using a comparator and a reference voltage, when the voltage diminishes below that reference, it means that the battery charge is low and when this happens, that comparator will change the output logic state.

New problem:

The power sources and function generators at the labs are not always reliable and may indicate Vpp (peak to peak voltage) when we are really dealing with the amplitude (1/2 Vpp).Or we could end up working with power sources that don’t necessarily make a “fine” adjust when we wanted and jump 2 or 3 volts at once.

That, and not being careful enough checking voltages and not being cautious, brings us to the perfect conditions where you can break all maximum limits, far beyond all datasheet recommendations and become the testing device into smoke.

What I want to say with all this is that I burnt the transmitter but it wasn’t my fault. Anyway I had to pay it again from my pocket and this is how I have learned to double-check from now on (No, I’m not meaning using a money check twice).

PCB layout description:

There are three different boards intended to work in different tasks, they are explained now:

The main control board

This one contains all the decoders and microcontrollers that will manage the flow of data and send the proper outputs to the peripherals, the relays that will decide on the one hand whether or not power our camera, receiver and servo and in the other hand our driving part (motor and rudder servo).

The transmitter board

In this case we get all our data from our USB port we convert it to binary, encode it into a serial communication output and send it wirelessly.

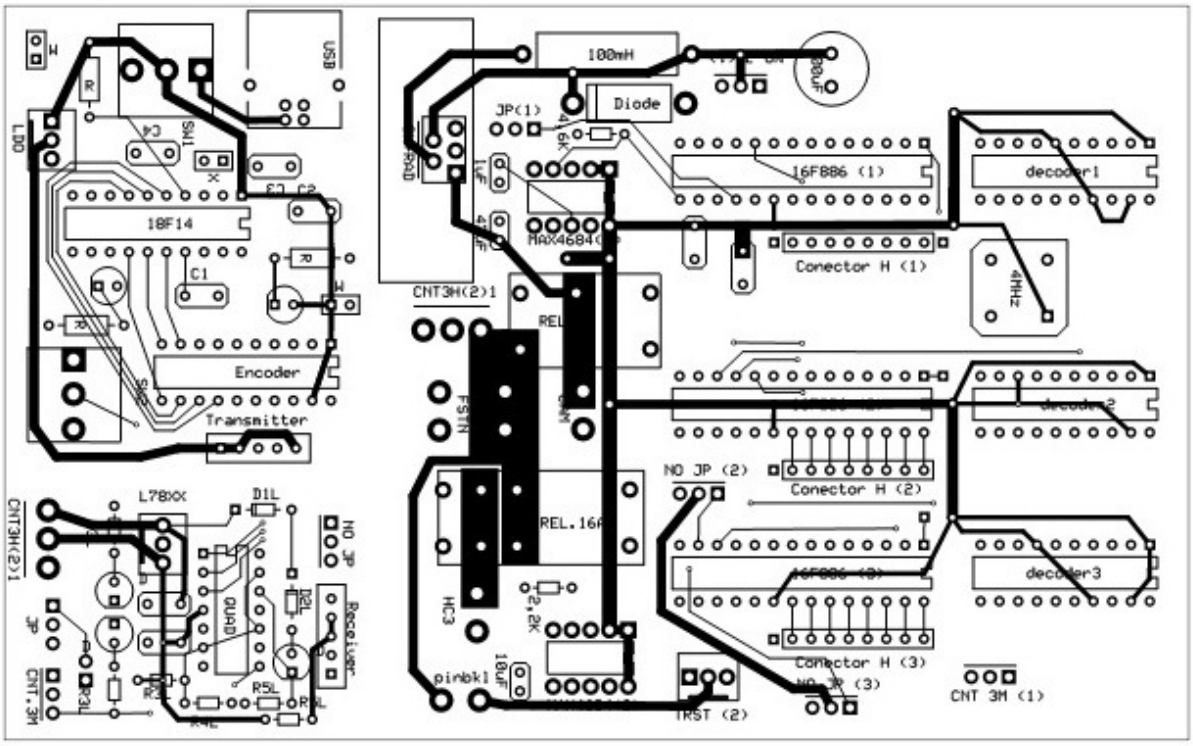
The receiver board

Even if this may be the smaller of all, it gets the encoded data from the receiver and converts it to TTL standard (0-5v) using a comparator, and uses a comparator too and some Zener diodes in order to determine the actual charge of the battery and alert us when a recharge is necessary.

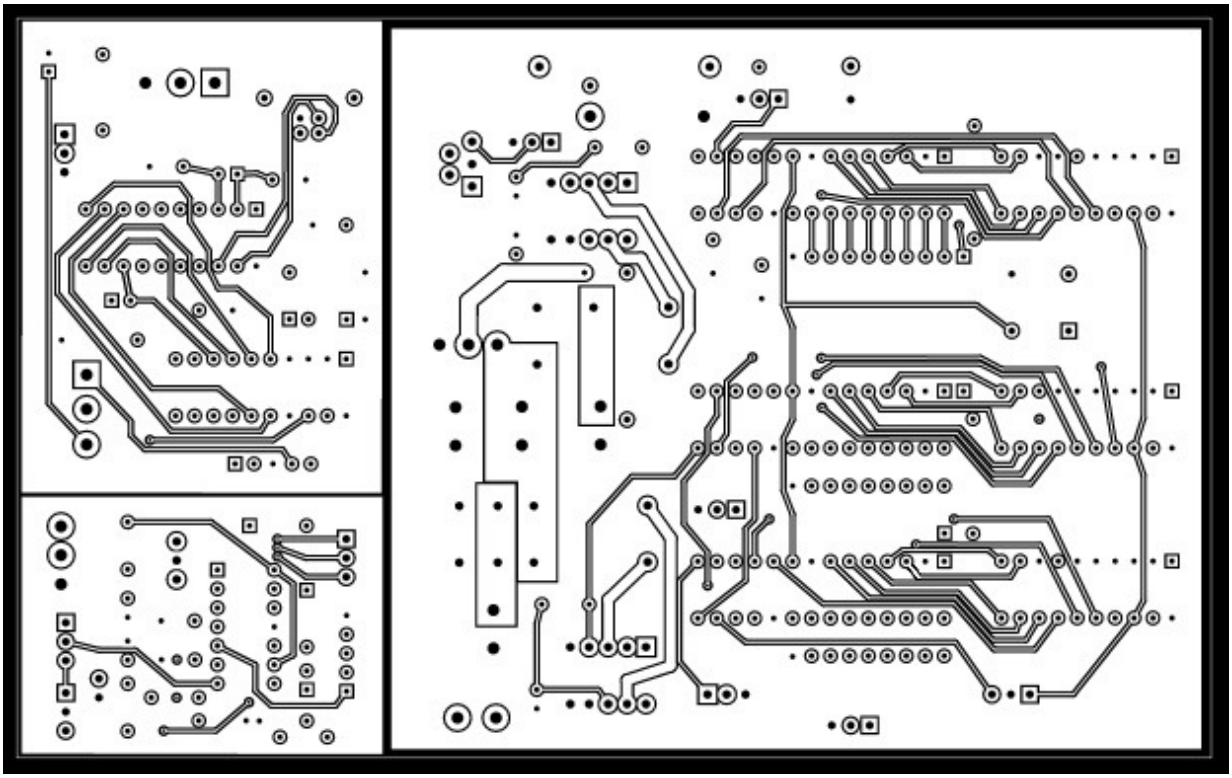
ORDERED BOARD DESIGN:

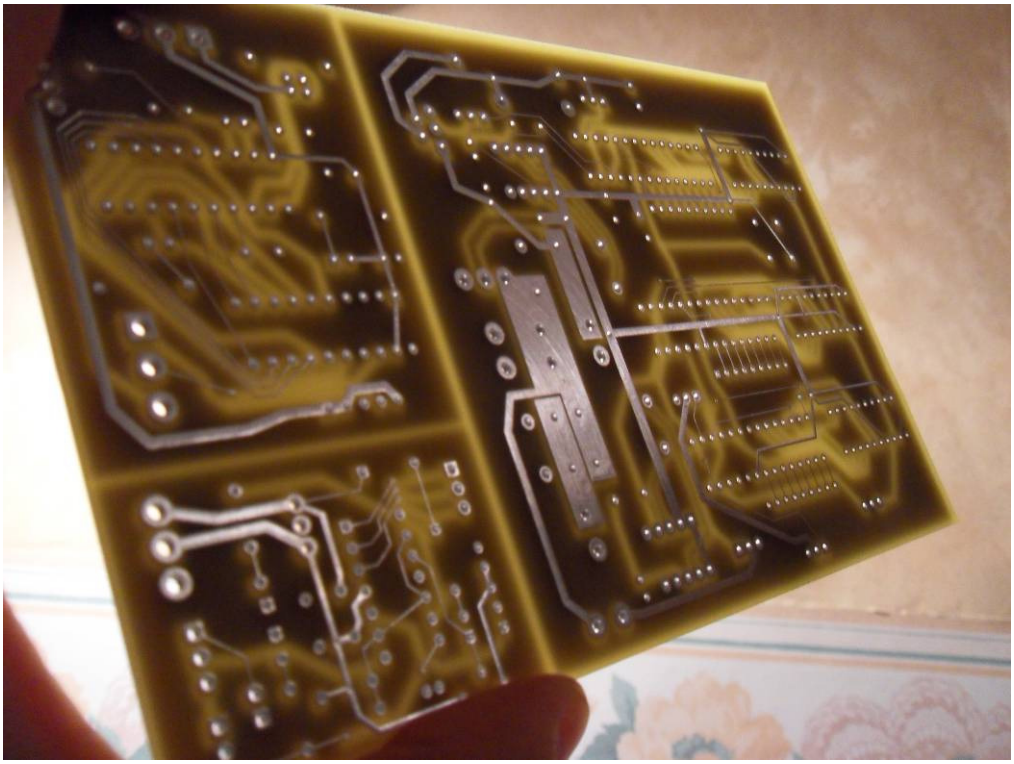
It contains the mentioned above three sections: The main control, the transmitter and the receiver circuits which have to be split off independently afterwards. The design has been done using ExpressPCB software which is free and makes it easy to order the desired product.

TOP PCB LAYER:



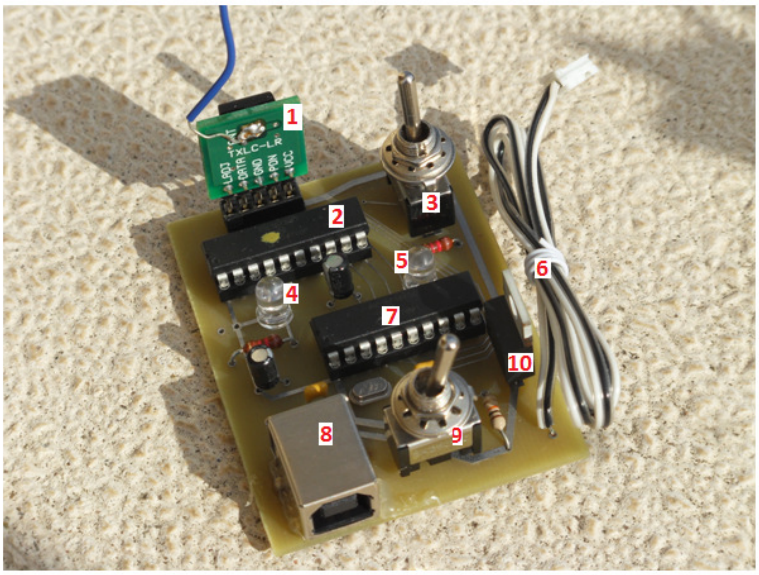
BOTTOM LAYER (with inverted colors for ground plane):





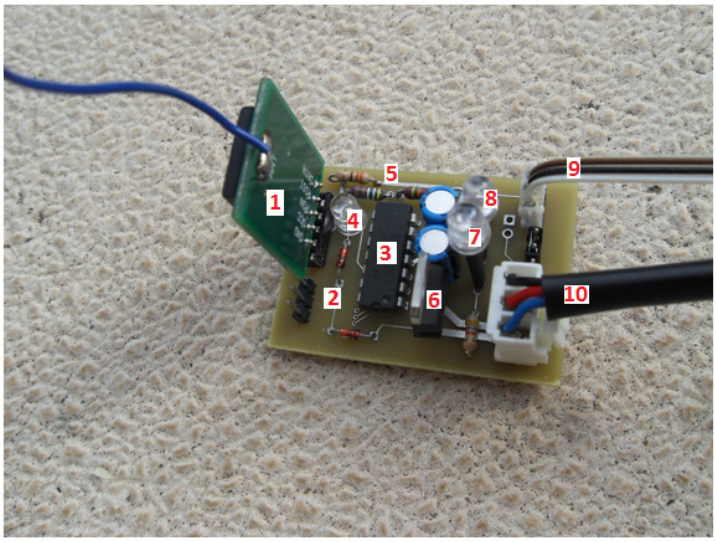
Double sided PCB's seen from behind a light source. It can be appreciated that no solder mask nor silkscreen layers have been applied.

Transmitter board completed:



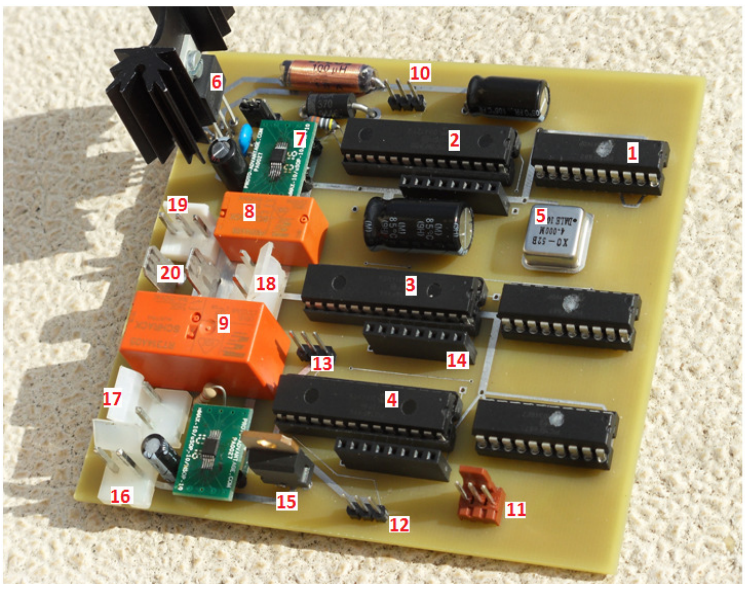
1) Transmitter
2)Encoder (data and directions)
3) wireless/wired communication sw.
4) 4 ON/OFF indicator LED
5) Communication LED
6) Wired communication cable
7) USB to digital converter IC
8) USB connector
9) ON/OFF switch
10) LDO 5v to 3,3v voltage regulator

Receiver board completed:



1) Receiver
2) Zenner diodes for Low battery voltage detection circuit
3) 4 Op.Amplifier IC for battery voltage comparators and for receiver output conditioning (3v to 5v)
4) LED brightness proportional to Battery pack's actual charge
5) Voltage divider at 2,5v
6) Main linear voltage regulator 12v to 5v
7) Green LED, main power indicator
8) Red LED, low power indicator
9) Data and battery state communication
10) 12v and 5v power link

Main control board completed:



1) Decoder 1
2) Pic microcontroller for camera servo and relay control
3) Pic microcontroller for motor control, relay and com. check
4) Pic microcontroller for rudder,lights and battery state.
5) 4MHz canned oscillator
6) Switching voltage regulator for 5v servomotor
7) Latching relay driver circuit
8) Camera, servo and AV transmitter ON/OFF latching relay
9) Main motor and rudder ON/OFF latching relay 20A max
10) Camera orientation servo connector
11) Encoded data and battery state signal connector
12) BEC and motor speed controller signal's connector
13) Rudder servo suply (BEC) and control connector
14) Microcontroller's Port B easy access connector
15) Backlight LEDS ON/OFF switching transistor
16) Backlight LED connector
17) Main motor supply connector (to electronic speed control)
18) Camera and AV transmitter supply connector
19) 12v and 5v connector
20) Main 12v battery pack connector

Peripherals

Manufacturer's specifications are given below for further details.

Motor & speed controller

MOTOR CHARACTERISTICS:

Graupner Inline 450 11.1V

New INLINE motors with low specific rpm rating
Ideal for direct drives in aircraft and boat models
Low outer diameter and cables routed toward the rear facilitate installation in narrow model fuselages and hulls
Precision-ground, twin-ballraced motor shaft

Specification
Connecting plug : one
Operating voltage range : 7,4 ... 14,8 V
No-load speed: 15873 U/min
All-up weight, approx. : 98 g
Free shaft length: 10 mm
Recommended controller: 7232
Output : 200 W
Number of poles: 2
Nominal voltage: 11,1 V
Case length: 45 mm
Shaft diameter: 2,3 mm
Case diameter: 24 mm
Revolutions/Volt: 1430

SPEED CONTROLLER CHARACTERISTICS:

Graupner Brushless Control 18 BEC (5.5-15V)

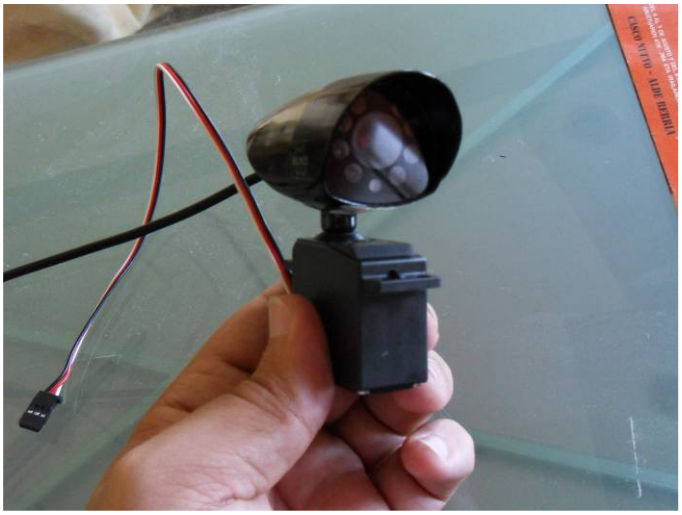
Highly efficient, high-power switching BEC system
SET button for simple programming of transmitter travels

Programming unit for simple speed controller programming
Important functions can also be programmed using the SET button

Operating voltage range 5.5 - 15V
Cell count, Ni-Cd, Ni-MH 6 to 12
Cell count, Li-Po, Li-Io 2 to 4
Cont. current (brushless motors) 18 A
Maximum motor current (10 sec.) 22 A

Camera & servo assembly

This attachment has been done by the technique of forcing a plastic plain part fit into a metallic threaded part by twisting it ignoring the fact that they don't belong.
This technique may seem primitive or stubbornness but now, they actually do belong perfectly. The joint is solid and robust.



Rudder and propeller assembly

Manufacturer's description:

Complete Surface Drive Flexi-shaft assembly with strut and rudder inline for Surface for dog drive 3/16 and 4mm bore props with 4mm flexishaft for 540/700 size motors and similar brushless and .12/.15 nitro. Suitable for Hydro's, Deep V's and Catamarans. Contains 4mm square ended Flexi-shaft with separate 4mm square drive steel stub-shaft, drive dog, PTFE teflon tube, strut with lead/teflon bushes, brackets and all nuts, bolts needed. Flexi and stub is 310mm long but can be cut down to length.Longer lengths can be supplied.

3 Bladed Propeller RH 40mm - Pitch 21mm

Other components

Plain Coupling Insert 2.3mm

2.3mm Plain Insert, Includes grub screw

Plain Coupling Insert 4mm

4mm Plain Insert, Includes grub screw

Universal Joint

Modular type universal joint, requires inserts to complete.

Pushrod Connector (10)

For 1.5-2.0 mm diameter wire pushrods and bowden cable inner tube (No. 3500.2). The pushrod connector is fixed to the servo output arm using two nuts locked against each other. Pack includes pushrod connector, metal M2 nuts, M3 grub screws and socket screw wrench SW 1.5 mm. Pack of 10

Rubber Bellows (2)

Graupner Rubber Pushrod Bellows
For waterlight linkages. 32 mm long, opening 1.5 and 5 mm diameter.

Clevis with Pin M3

Clevis with pin. Nickle-plated, with female M3 thread.
Spring steel shanks with 1.5 mm diameter. pin. Thread M3

Clevis Set M2 (2)

Clevis Set, nut and threaded bushing M2 thread (2 per pack)

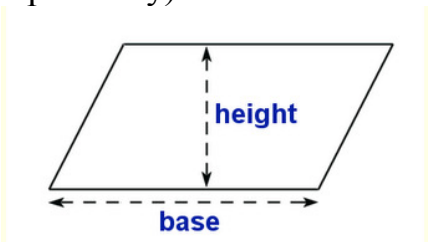
10in Threaded Rod (2)

10in M2 Steel rod for rudder connection. Threaded one end

Servo to rudder driving mechanism

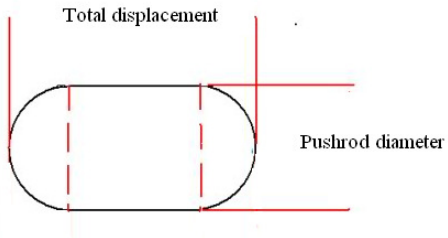
In essence this part seems to be easy to build but has a tricky part where some design changes needed to be done pursuing a proper mechanism performance.

The original idea was to connect the rudder driving arm to the servo by using nothing more than pushrods in a direct connection, this, in the geometrical point of view consists of a parallelogram with variable height (we consider the base length's parallel lines representing the pushrods and the other parallel lines the rudder and servo arms respectively).

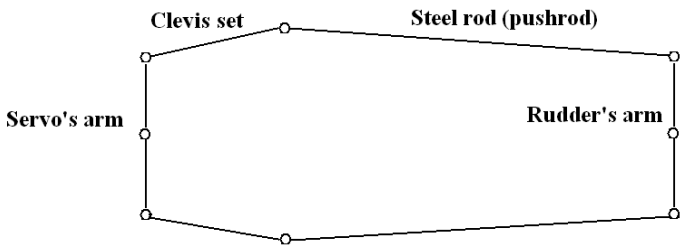


That variable height is precisely what causes problems: We find that this rods will have not only a longitudinal displacement as we predicted, but a transversal displacement that wasn't taken in to account too.

This lateral movement would require a special orifice in the hull, something like a elongated circle instead of the desired circular drill where the pushrod bellow cylinder would match:



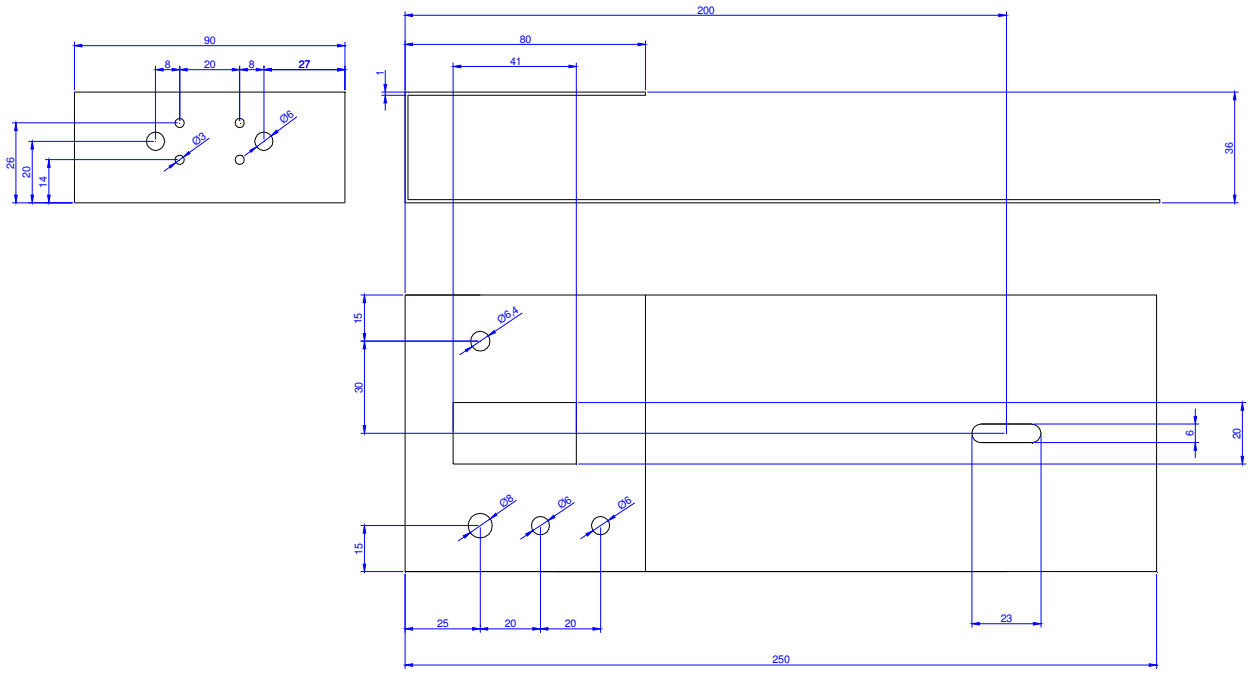
This inconvenience makes us consider a different geometric distribution from the original parallelogram, one that could be more adaptable to this height issue:

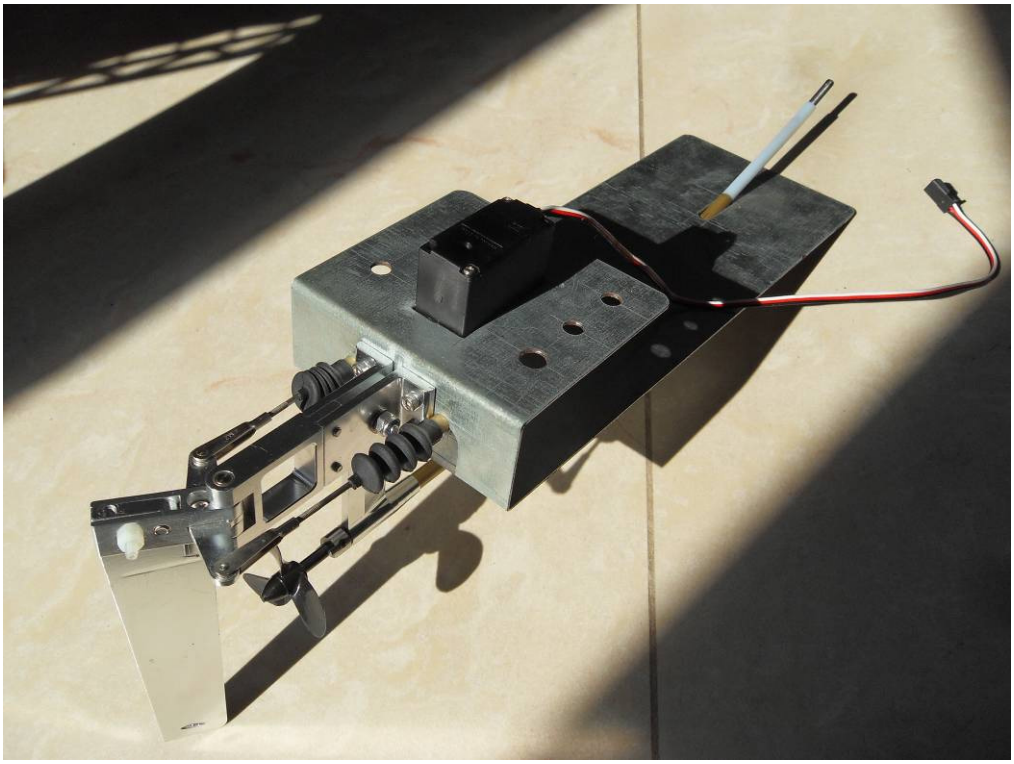


This one above demonstrates to be less rigid and does the intended function perfectly.

Mount prototype design

In this occasion we can't just see how our new rudder assembly works unless we find a way of putting it solidly fixed to some kind of mount where the corresponding driving servo is, so, this is where the need of this part comes from.
See the AutoCad drawings followed by the final result:

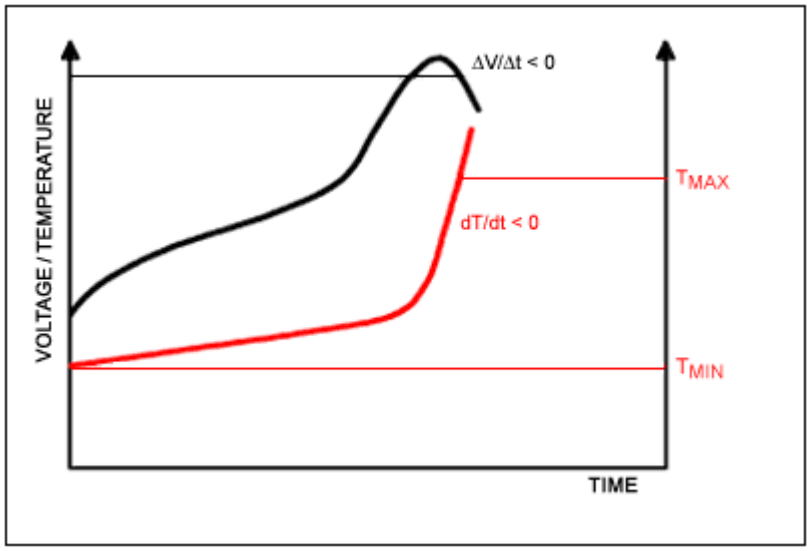




Battery pack charging characteristics

The two battery packs recently purchased are composed of 10 1,2v independent cells in series. According to manufacturer's specifications each of this cells has 4200mAh of capacity and that is something hard to believe considering the unknown brand and the fact that no other NiMh cells in the market have such capacity rating.

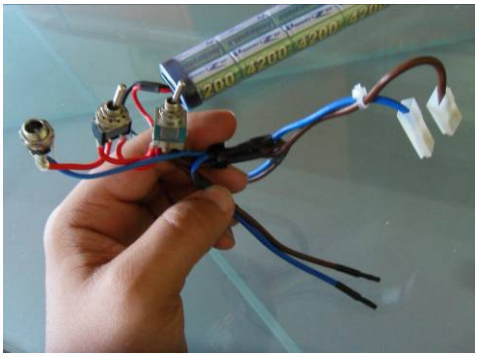
Then, I made a research over the NiMh cells and I found out many things I didn't know until then, one of them being the charging characteristic curve from which you can easily deduce that more than a constant voltage what we need is a constant current to charge it:

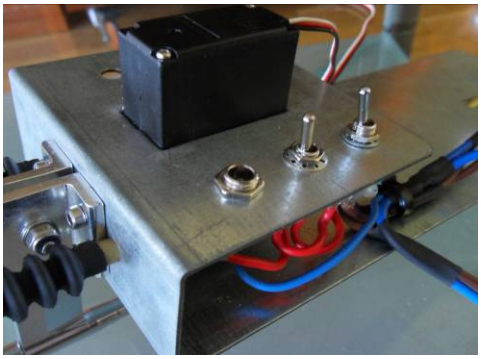


The moment the voltage peak is reached, we consider that the battery is full, once this point is left behind the voltage drops and all the current it draws starts to turn into heat. If the charger doesn't stop right before that peak, it would lead to a irreversible damage not only to your battery pack but possibly to your entire burning house. The final idea here is that we need a trustworthy adapter.

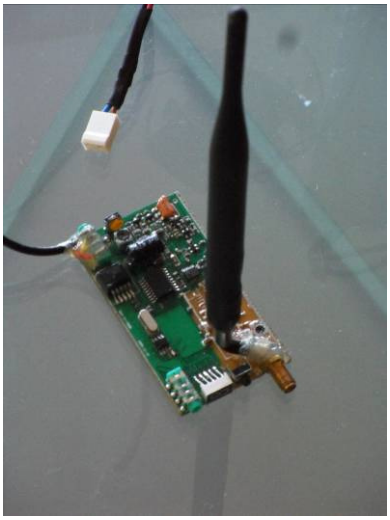
Another inconvenience would be that we can't charge this battery packs in parallel neither: we need to do it one by one because the peak won't occur at the same time for both packs and if the dv/dt change is not detected in one of them, it could lead to total destruction (different sources on internet confirm this).
Yet, we can connect them that way (parallel) in normal operation (discharging).

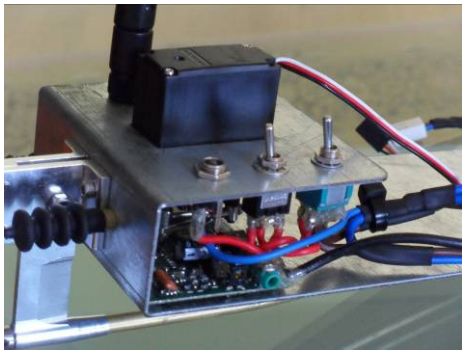
By making a circuit that charges the battery packs one by one when the main switch is off and puts them in parallel when we switch the circuit on, worries are gone.





AV transmitter placing





Removing the casing of the transmitter we get rid of a great amount of unused empty space and we can put this card wherever it fits best. In the other hand, the antenna has been changed and now it can come out of the deck as it can be seen on the pictures.

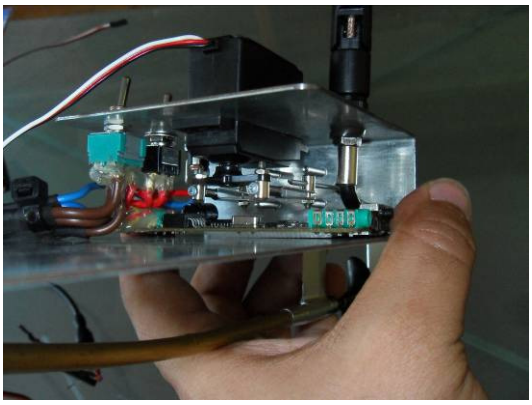
Refrigeration or cooling techniques

Heat, moisture or corrosion are probably our worst enemies, here we will talk about how to fight against the first one.

AV transmitter cooling

The act of removing the case, brings not only advantages but disadvantages too, the casing itself acted as a heat sink, and getting rid of it may be not a good idea at all, but I was aware of this problem before and this is why the placing of this object is crucial and more than a convenient place, is the place to be.

The metal piece in the bottom of this transmitter will be in contact with the metallic hull which obviously has cold water running in the outer side. This is finally the best justification for why this is the perfect location for it.



Motor cooling

Our main driving motor can provide up to 200w, we know that a brushless motor is efficient but losses are there, and with them, the so feared and destructive heat. The idea again is making use of water as coolant taking it from outside cold and returning it hotter as a result of refrigerating the motor. At this point, we will use the resources we have and create those we don't with the intention of making this water circuit work.

What we have is an orifice in the lower part of the rudder from where water will come in every time the boat is moving. Then, this water will come out of the upper part where we can attach a rubber tube.

Still, we don't have any way of cooling the motor itself. Making a cooling coil out of copper tube may seem easy but it was not, twisting the tube smashes it and closes the internal cavity, so introducing a cable we avoided that. But then, the cable got strongly trapped and we had to do it again with a greased cable, but…
Anyway what matters is that it's finally done.

Moisture protection

PCB protection is highly desirable against corrosion or short circuits created by accidental water drops. This is why conformal coating spray will be used, as its name suggests it covers all PCB connections and pads with an isolating substance. This is especially important because some of our PCB's don't have protection at all, not even the typical green solder mask that most of us are used to. I wanted to order the boards with this feature (solder mask) in the beginning but costs skyrocketed and I decided to forget about it. I'd say that the application of this cover in this circumstances is mandatory and giving more than one spray layer may be a good idea too.

Computer interface, user instructions and main features

The computer interface is initialized automatically just by double clicking on the executable file from AutoHotkey, this one will start the AverTV software and maximize the window where the images from the camera will be shown. Then, it will start the MinGW shell and immediately will hide that window, which will be still active, being the place where all the commands are printed from the main program so that they can be sent to the USB port with controlio.exe.

With the intention of making everything more visual and intuitive, every time any variable is changed from any hotkey (keyboard key press or mouse click), a tag on the screen pops showing the state of the variable in that moment, for example, spinning the mouse wheel up, boat speed increases, then, a tag on the screen will give us an approach of the actual speed in a scale of 0 (stop) to 15 (max). I give a list of the different variables that we can modify and the Hotkey that will actually do it:

Variable	Hotkey	Range
Speed faster/slower	Mouse wheel up/down	0 to 15
Camera/boat control	Mouse middle button	Camera or boat
Boat direction	Mouse click left/right (with boat control active)	- 32 to 32 in steps of 8 or 16 (variable).
Boat direction step	Control +/-	8 or 16
Camera direction	Mouse click left/right (with camera control active)	-32 to 32 in steps of 2,4,8 or 16 (variable).
Camera direction step	+/-	2,4,8 or 16
General supply	G	ON/OFF
Camera supply	C	ON/OFF
Illumination	L	ON/OFF
Exit program	X	-

Hydrostatic principles

Hydrostatics deals with the physics of fluids at rest, and is a sub field of fluid mechanics. The science of fluids in motion belongs to fluid dynamics.

STABILITY

Our catamaran is stable to inclining moments due to the double hull that characterizes it, still, the boat is prone to direction errors caused by wrong weight distributions, let's see some basic questions to understand what this moments mean in a monohull ship. The next parts have been taken from the text "buoyancy principles":

Figure 12-9 shows how an INCLINING MOMENT is produced when a weight is moved outboard from the centerline of the ship. If the object weighing 20 tons is moved 20 feet outboard from the centerline, the inclining moment will be equal to 400 foot-tons (F x d, or 20 x 20).

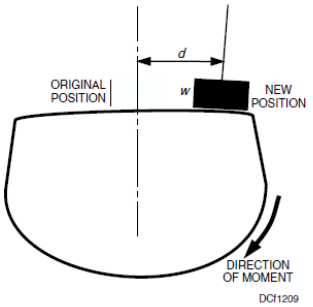


Figure 12-9. Inclining moment produced by moving a weight outboard.

Figure 12-10 shows how a forward (or aft) movement of weight produces a TRIMMING MOMENT. Let's assume that a 20-ton weight is moved 50 feet forward; the trimming moment produced is 20 x 50, or 1,000 foot-tons.

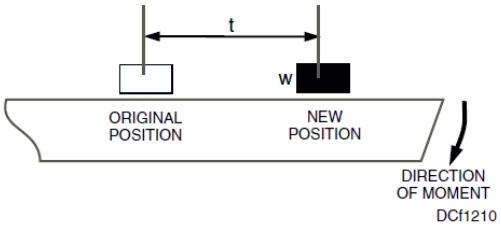


Figure 12-10. Trimming moment.

After checking that out we extract one simple conclusion: The average weight's gravity center* must be located in a centered position to avoid inclining or trimming moments that could affect stability.

*The gravity centre of a body is the imaginary point where the total weight of a material body may be thought to be concentrated.

Later on, we must consider stability due to buoyancy vs gravity:

Horizontal pressures on the sides of a ship cancel each other under normal conditions, as they are equal forces acting in opposite directions (fig. 12-17). The vertical pressure may be regarded as a single force—the force of buoyancy acting vertically upward through the CENTER OF BUOYANCY (B).

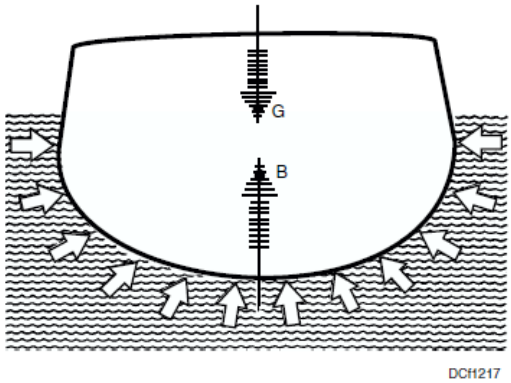


Figure 12-17. Relationship of the forces of buoyancy and gravity.

*The buoyancy centre of a body is the imaginary point where the total lifting pressure of a submerged body may thought to be concentrated.

If you study figure 12-20, you will notice that a RIGHTING or RESTORING MOMENT is present. This righting moment is caused by the two equal and opposite forces, each of W tons (displacement) magnitude, separated by a distance GZ, which constitutes the LEVER ARM OF MOMENT. Figure 12-20 shows that the ship is stable because the center of buoyancy (B) has shifted far enough to position the buoyant force where it tends to restore the ship to an even keel or an upright position.

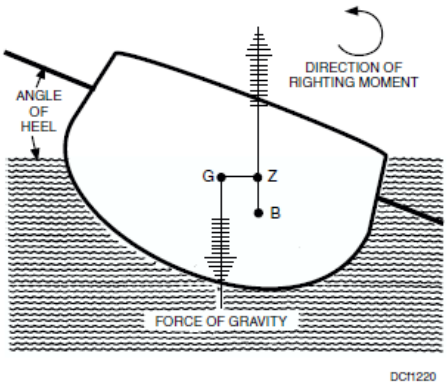


Figure 12-20. Development of righting moment when a stable ship inclines.

However, it is possible for conditions to exist which do not permit B to move far enough in the direction in which the ship rolls to place the buoyant force outboard of the force of gravity. The moment produced will tend to upset the ship, rendering it unstable. Figure 12-21 shows an unstable ship in which the relative positions of B and G produce an UPSETTING MOMENT. In this illustration it is obvious that the cause of the upsetting moment is the high position of G (center of gravity) and the GEOMETRIC CENTER OF THE UNDERWATER BODY (B—the center of buoyancy).

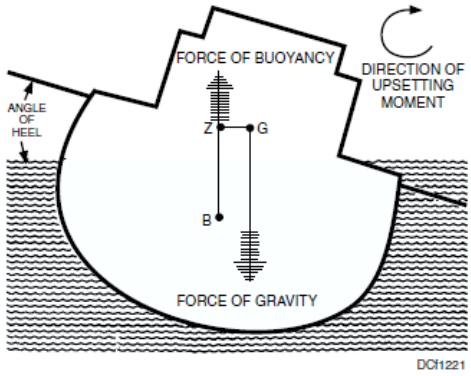


Figure 12-21. Development of an upsetting moment when an unstable ship inclines.

From these ideas we conclude that a lower gravity center will always bring more stability to the boat as well.

DRAFT

Water level may be calculated by just considering the Archimedes buoyancy law:

“The upward buoyant force exerted on a body immersed in a fluid is equal to the weight of the fluid the body displaces.”

Using this law with some additional physics, draft calculation will be easy:

In equilibrium, the upward buoyancy force equals the total weight of the boat, so, the displaced liquid’s weight is the boat’s total weight.
Known this liquid’s density and weight we can calculate the volume it occupies, which coincides with the volume that hull will occupy below the water surface.
From here, draft calculation comes from solving a single unknown equation, being this equation the function that represents the volume of the hull subordinate to the height (unknown factor).

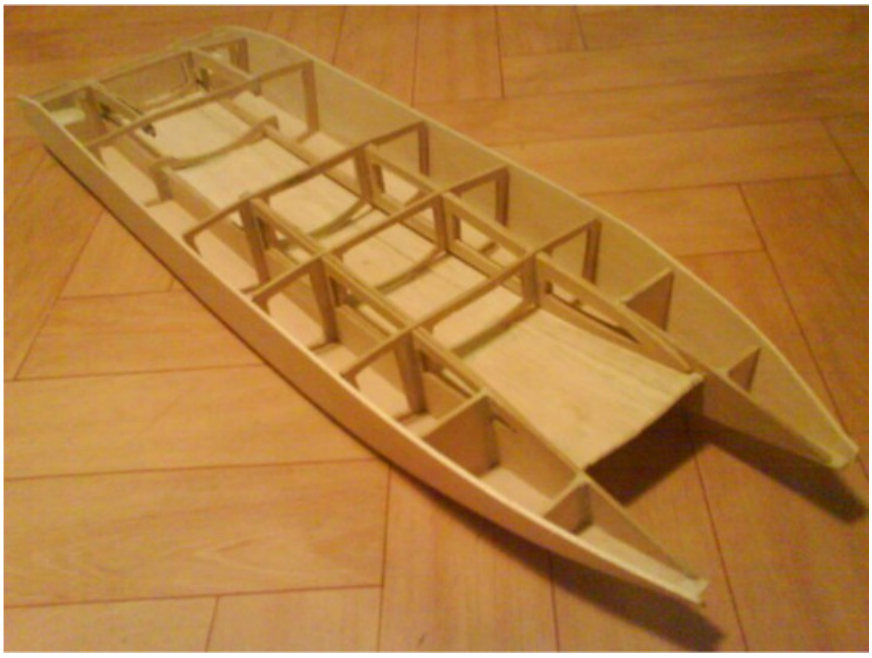
High gain antenna testing

I'm using a dish antenna even though it isn't designed for this kind of applications, parabolic antennas play an important role in high frequency microwave communications usually with satellites at 10GHz and my links use 1 GHz and 350 MHz so its gain will be obviously lesser than the typical. Anyway it is worth trying just to learn a little bit more from it by experimentation and see if there are any improvements.

Hull design

The design and manufacturing of this piece is yet to be done, but I gathered some ideas taken from other projects that may help. I also found a software specially created for this, it's called FreeShip and it can be used to create a hull considering all necessary calculations for it. I'll show some pictures that give an idea of the possibilities that we have:

Pictures from the project at www.miliamperios.com :: Catamaran RC



Deck design

This design is the next challenge and will have to hold the camera assembly and the hull-deck joint must be perfectly water tight. FreeShip software can be used to model this part again.

Final boat testing

Here is where the Titanic history begins lol. I hope everything works fine the first time if not, the second will, if not, the third… well, after all, this is what testing means; making the necessary improvements over and over until you reach the desired performance.

Final conclusions

All electrical and electronic systems work successfully and I'm happy with the results obtained, the project needs to be early exhibited due to the lack of time and personal economic problems that fade it into the background being replaced by new priorities.

APPENDIX

AutoHotkey programming examples

This an example code developed in AutoHotkey to show the potential of this tool in order to fulfill the objectives:

First part:

```
F1::  ;F1 is a hotkey, when pressed, it executes the code below.
Run,C:\Archivos de programa\AVerMedia\AVerTV\AVerTV.exe
;Executes AVerTV software:
Sleep,5000 ;Waits until the program is fully initialized.
WinMaximize, AVerTV-Vídeo ;Maximizes the popping window
Run,C:\Documents and Settings\Usuario\Mis documentos\ambition30.exe
Sleep, 100
```

Second part:

```
CoordMode, ToolTip, Screen     ;Coordinates screen referenced for the tooltips or window labels
y=0     ;Speed
x=0     ;Direction
z=0     ;Camera position
q=8     ;Acceleration rate
w=8     ;Camera servo acceleration
e=8     ;Direction servo acceleration
h=OFF      ;General power source
k=OFF       ;Camera power source
j=BARCO  ;Boat/camera control activated
Add(a, b)    ;Function sum of a and b
{
   return a+ b    ; "Return" expects an expression.
}
Multi(a, b)       ;Function multiply a by b
{
   return a*b
}
Div(a, b)         ;Function division of a over b
{
   return a/b
}
Label:
          *#CapsLock::        ;Win+Capslock keys takes control of camera system
                    if (j="BARCO")
                              j=CAMARA
ToolTip, Control  %j%,100, 200    ;Shows the result on PC screen
          return
```

```
*CapsLock::             ;Capslock key takes control of boat system
        if (j="CAMARA")
                j=BARCO
ToolTip, Control  %j%,100, 200    ;Shows the result on PC screen
        return
        *Enter::              ;Enter key centers the direction or the camera depending on which control is active
        if (j="BARCO")
        {       if (h="ON")
                        x:=0  ;DIRECTION POSITION L=x+32
        }
        else
                if (k="ON")
                        z:=0  ;CAMERA POSITION M=z+32
ToolTip, Velocidad:  %y%`n Direccion: %x%`n Pos.camara: %z%,100, 150
        return
        *NumpadAdd::          ;The + key increases the acceleration in boat or in camera movements
        if (j="BARCO")
        {       if (h="ON")
                        if (q=1)
                                q=4
                        else
                                if (q<16)
                                        q:=Multi(q,2)
ToolTip, INC.V %q%,100, 200
        }
        else
        {       if (k="ON")
                        if (w=1)
                                w=4
                        else
                                if (w<16)
                                        w:=Multi(w,2)
ToolTip, INC.CAM  %w%,100, 200
          }

        return
        *^NumpadAdd::    ;Control+ increases the acceleration in direction movements
                if (h="ON")
                        if (e=1)
                                e=4
                        else
                                if (e<16)
                                        e:=Multi(e,2)
ToolTip, INC.DIR  %e%,100, 200
        return
        *NumpadSub::          ;The - key decreases the acceleration in boat or camera movements
        if (j="BARCO")
        {       if (h="ON")
                        if (q=4)
                                q=1

                        else
                                if (q>4)
                                        q:=Ceil(Div(q,2))  ;Ceil function  rounds a number
ToolTip, DEC.V  %q%,100, 200
         }
        else
```

```
                {       if (k="ON")
                                if (w=4)
                                        w=1
                                else
                                        if (w>4)
                                                w:=Ceil(Div(w,2))
ToolTip, DEC.CAM  %w%,100, 200
                        }

        return
        *^NumpadSub::       ;Control- keys decrease acceleration in direction movements
                if (h="ON")
                        if (e=4)
                                e=1
                        else
                                if (e>4)
                                        e:=Ceil(Div(e,2))
ToolTip, DEC.DIR  %e%,100, 200
        return
        *up::                   ;Up arrow key increases the speed in a given acceleration rate
                if (h="ON")
                        if (y<64) and (Add(y,q)<=64)      ;(We suppose that the boat won't move backwards)
                         {
                                 y:=Add(Multi(q,Ceil(Div(y,q))),q )   ;All this stuff here is meant to avoid
; problems like overflowings or other numeric flaws when adding numbers that change increasing rate.
                                ;SPEED
                        }
                ToolTip, Velocidad:  %y%`n Direccion: %x%`n Pos.camara: %z%,100, 150

        return
        *Down::         ;Down arrow key will slow down the boat
                if (h="ON")
                        if (y>0) and (Add(y,-q)>=0)    ;(We suppose that the boat won't move backwards)
                        {       y:=Add(Multi(q,Ceil(Div(y,q))),-q)
                                ;SPEED
                        }
                ToolTip, Velocidad:  %y%`n Direccion: %x%`n Pos.camara: %z%,100, 150

        return
        *Left::             ;Left arrow key will change direction to the left in the boat or in the camera position
                if (j="BARCO")
                {       if (h="ON")
                                if (x>-32) and (Add(x,-e)>=-32)
                                {       x:=Add(Multi(e,Ceil(Div(x,e))),-e)
                                        ;DIRECTION POSITION L=x+32
                                }
                }
                else
                        if (k="ON")
                                if (z>-32) and (Add(z,-w)>=-32)
                                {       z:=Add(Multi(w,Ceil(Div(z,w))),-w)
                                        ;CAMERA POSITION M=z+32
                                }
                ToolTip, Velocidad:  %y%`n Direccion: %x%`n Pos.camara: %z%,100, 150
        return
        *Right::        ;Right arrow key will change direction to the right in boat or camera position
                if (j="BARCO")
```

```
		{	if (h="ON")
				if (x<32) and (Add(x,e)<=32)
				{	x:=Add(Multi(e,Ceil(Div(x,e))),e)
					;DIRECTION POSITION L=x+32
				}
		}
		else
			if (k="ON")
				if (z<32) and (Add(z,w)<=32)
				{	z:=Add(Multi(w,Ceil(Div(z,w))),w)
					;CAMERA POSITION M=z+32
				}
			ToolTip, Velocidad:  %y%`n Direccion: %x%`n Pos.camara: %z%,100, 150
	return
	*Space::	;Space key stops the boat an centers direction and camera position
			y=0  ;SPEED
			x=0  ;DIRECTION POSITION L=x+32
			z=0  ;CAMERA POSITIONM=z+32

			ToolTip, Velocidad:  %y%`n Direccion: %x%`n Pos.camara: %z%,100, 150
	return
	*Tab::		;Tab key activates or deactivate the general power source
			if (h="OFF")
			{	h=ON ;ACTIVATE BOAT
				y=0 ;SPEED
				x=0  ;DIRECTION POSITION L=x+32
			}
			else
				h=OFF  ;DEACTIVATE BOAT
			ToolTip,  Alimentación general: %h% `n Alimentación cámara: %k%,100, 450
	return
	*#Tab::		;Win+Tab keys activate or deactivate camera system
			if (k="OFF")
			{	k=ON  ;ACTIVATE CAMERA
				z=0  ;CAMERA POSITION M=z+32
			}
			else
				k=OFF  ;DEACTIVATE CAMERA
			ToolTip,  Alimentación general: %h% `n Alimentación cámara: %k%,100, 450
	return

Goto, Label
```

;COMMENT: BLUE REMARKS are flags where subroutine calls should be that send variables such us speed, direction etc. to the driver that controls USB communications and ultimately the encoder.

Set of instructions:

1. F1 for initialization
2. Tab for general power start up (motor and orientation)
3. Windows+Tab for camera switch
4. Capslock takes over boat's control

5. Windows+capslock takes over camera control
6. up/down arrow regulates speed
7. left/right arrow regulates orientation if operating in boat control, otherwise, it'll change camera position
8. Space clears all values, straightens camera and orientation and stops.
9. Enter straightens camera or direction depending on actual control.
10. + for an increase in acceleration on speed or camera movements
11. - for a decrease in acceleration on speed or camera movements
12. Ctrl+ increase in direction movements acceleration.
13. Ctrl- decrease in direction movements acceleration.

Comment:
The program could fail when executed in a different computer: files are located in different folders, screen resolution may vary...

Subroutine calls (BLUE REMARKS) may bring us more problems;
1. If the code is executed in such speed that it calls a subroutine every 2ms but the encoder is able to send data just every 10ms, that means we won't send the info 4 out of 5 times. This means inefficiency and slowness.
2. In the scenario of putting a sleep10ms function in each thread, the situation gets worse, the program could lose fluency and it would slow it down again.

So, what we need to figure out is a way of sending the most recent info every 10ms without affecting the speed of the main program, or not sending nothing at all if no new info has been generated.

This is tricky, but once again a solvable problem, lets put an easy example of how this can be done by multitasking:

```
Run C:\Archivos de programa\Microsoft Works\wkswp.exe  ;Executes works text editor
WinWait, Documento sin título - Procesador de textos de Microsoft Works,
WinActivate, Documento sin título - Procesador de textos de Microsoft Works,
WinWaitActive, Documento sin título - Procesador de textos de Microsoft Works,
SetTimer,subprograma,1000  ;Runs a synchronous timer that calls a subroutine every 1000ms (by synchronous meaning that it will run at the specified frequency (1000ms) even when the script is busy with another task)
CoordMode, ToolTip, Screen
a=0
b=0
r=0
l=1
m=2
o=3
Add(x, y)
```

```
{
   return x + y   ;
}
Subprograma:  ;This is the subroutine that will only run every 1000ms

   if a=1 ;If new data has been generated  by Juan, (a=1) then:
   {
     SendInput,"Juan: " %r%¨{SPACE}%l%{SPACE} ;Send the new values to the  text editor
     a=0
    }
    if b=1 ;If new data has been generated by Luis, (b=1) then:
    {
     SendInput,"Luis: "%m%¨{SPACE}%o%{SPACE} ;Send the new values to the text editor
     b=0
   }
return
Label:
*Left::  ;Left arrow key will do a "Juan" thread, (keep this key pressed and see what happens).
r:=Add(r, 1) ;Juan's operations.
l:=Add(l,1)
ToolTip, JUAN %r% %l%,100, 200 ;This will show Juan values in real time on the PC screen.
a=1
return
*Right::  ;Right arrow key will do a "Luis" thread, (keep this key pressed and see what happens).
m:=Add(m, 1)
o:=Add(o,1)
ToolTip, LUIS %m% %o%,100, 200
b=1
return
```

So, in this case, we only send info every 1000ms, it always is the most recent created and it only updates that info when it is outdated. Doing this kind of structure is far more effective than any other alternative when computer threads run faster than the encoding process or calls are a burden for threads in general.

However, that's not the situation, in fact, threads run slower than the encoding process, this can be demonstrated by just using the program above:

While running it, if Right arrow key is pressed continuously, a series of numbers will appear, notice that there is a difference between a number and the next. Let's say there is an average difference of 3 numbers when the timer is set to 100ms, it means the length of that thread is about 30ms which is 3 times the encoding time.

So finally, using this type of structures is discouraged, Instead, making subroutine calls at the end of each thread is the best solution.

Another way of commanding the boat instead of using the keyboard could be the mouse, here is an easy example of mouse control (Press x key to exit):

```
CoordMode, ToolTip, Screen
a=0
b=0
```

```
r=0
l=0
o=0
Add(x, y)
{
   return x + y   ;
}
*MButton::  ;Middle mouse key (by pressing the wheel) would switch from boat to camera control or vice versa
l:=Add(l,1)
ToolTip, BARCO O CAMARA %l%,100, 200
return
*RButton::   ;Right mouse key moves boat/camera to the right
o:=Add(o,1)
ToolTip,DERECHA %o%,100, 200
return
*LButton::   ;Left mouse key would do the same but to the left
r:=Add(r,1)
ToolTip, IZQUIERDA%r%,100, 200
return
*WheelUp::  ;Scrolling upwards would increase speed
a:=Add(a,1)
ToolTip, DEPRISA%a%,100, 200
return
*WheelDown:: ;Scrolling down would slow it down
b:=Add(b,1)
ToolTip, DESPACIO%b%,100, 200
return
*x::ExitApp  ;Very important: x key makes the zombie mouse be again like it was before.
```

Matching Main program with USB interface:

One overview of the possible pros and cons of how we arrange the program calls is interesting, because it may affect how reliable our communication is, how fast it is and it could determine the performance critically:

Consider that the encoder packs data and address inputs every 10ms and sends them, it does not mean that input was actually what we wanted to send, because when it was picked up maybe the process of setting the inputs wasn't finished yet.

If the address is wrongly sent, we could send info to the wrong device or if data is wrong, we could mess with our device instead of control it. Reliability can be achieved by doing the next operations:

1. Addresses can avoid "interferences" by just being simple and incompatible: If we consider addresses like 0001 or 0010 or 0100 there are not possible errors (there is no way we can get another address if setting "1" is uncompleted) yet, considering 0000 as a device address here would be a ticket to troubleworld.

2. Data uncertainties may be more sophisticated to give a solution, a robust solution is

to set data output first with address at 0000, then set the device address. After data is sent we put address pins to 0000 and start over again (Set data, set address, clear address):

```
SendInput,controlio setport %y%{ENTER}
SendInput,controlio setbit 4096{ENTER}
SendInput,controlio clearbit 4096{ENTER}
```

As it is visible here first solution is easy but the second adds more commands, hence it affects processing speed. Use or not this last method will depend on the performance shown by the system later, we will evaluate the convenience of adding more reliability while sacrificing speed or not.
Remember that we said if threads are slowed too much by burden calls or encoding waits, we can always use our multitasking structures.

Now we improve the main example program with mouse control instead of keyboard arrow keys, make calls in a reliable fashion and since all decoders are in latch mode, as a primary security measure, it's mandatory to include a subroutine that sends a pulse every second to let the boat unlatch the rest of the decoders case connection is lost. This way we know it won't go too far without RF connection. Here is a new edition (but not the last one):

```
Run,C:\Archivos de programa\AVerMedia\AVerTV\AVerTV.exe
Sleep,5000
WinMaximize, AVerTV-Vídeo
Run,C:\MinGW\msys\1.0\msys.bat
WinActivate,MINGW32:~  ;MinGW shell is where we will write commands for the USB transmission.
Sleep,500
WinHide,MINGW32:~ ;Hides the window but remains operable
SetTimer,subprograma,1000  ;Subroutine timer that will send a pulse every 1000ms
CoordMode, ToolTip, Screen
v=0 ;Local variable
c=0
y=0 ;Speed
x=0 ;Direction blade
z=0 ;Camera orientation
q=8
w=8
e=8
h=OFF
k=OFF
j=BARCO
Subprograma:
{               v:=Add(c,10112)    ;Overwrites the pulse bits in the decoder's actual bits  avoiding interferences.
                SendInput,controlio setport %v%{ENTER}
                SendInput,controlio clearbit 128{ENTER}
                SendInput,controlio clearbit 9984 {ENTER}
}
return
```

```
Add(a, b)
{
   return a+ b   ; "Return" expects an expression.
}
Multi(a, b)
{
   return a*b   ; "Return" expects an expression.
}
Div(a, b)
{
   return a/b   ; "Return" expects an expression.
}

            *^NumpadAdd::
                        if (h="ON")
                                    if (e=1)
                                                e=4
                                    else
                                                if (e<16)
                                                            e:=Multi(e,2)
ToolTip, INC.DIR  %e%,100, 200
            return
            *MButton::  ;Pressing  mouse middle button we will choose boat or camera control
                        if (j="BARCO")
                                    j=CAMARA

                        else
                                    j=BARCO


ToolTip, Control  %j%,100, 200
            return
            *^NumpadSub::
                        if (h="ON")
                                    if (e=4)
                                                e=1
                                    else
                                                if (e>4)
                                                            e:=Ceil(Div(e,2))
ToolTip, DEC.DIR  %e%,100, 200
            return
            *Enter::
                        if (j="BARCO")
                        {           if (h="ON")
                                                x:=0
                                                SendInput,controlio setport 32{ENTER}
                                                SendInput,controlio setbit 9984{ENTER}
                                                 SendInput,controlio clearbit 9984{ENTER}
                        }
                        else
                                    if (k="ON")
                                    {
                                                z:=0
                                                SendInput,controlio setport 32{ENTER}
                                                SendInput,controlio setbit 19968{ENTER}
                                                            SendInput,controlio clearbit 19968{ENTER}
                                    }
```

```
ToolTip, Velocidad:  %y%`n Direccion: %x%`n Pos.camara: %z%,100, 150

        return

        *NumpadAdd::
                if (j="BARCO")
                {       if (h="ON")
                                if (q=1)
                                        q=4
                                else
                                        if (q<16)
                                                q:=Multi(q,2)
ToolTip, INC.V %q%,100, 200
                }
                else
                {       if (k="ON")
                                if (w=1)
                                        w=4
                                else
                                        if (w<16)
                                                w:=Multi(w,2)
ToolTip, INC.CAM  %w%,100, 200
                  }
          return
        *NumpadSub::
                if (j="BARCO")
                {       if (h="ON")
                                if (q=4)
                                        q=1

                                else
                                        if (q>4)
                                                q:=Ceil(Div(q,2))
ToolTip, DEC.V  %q%,100, 200
                 }
                else
                {       if (k="ON")
                                if (w=4)
                                        w=1
                                else
                                        if (w>4)
                                                w:=Ceil(Div(w,2))
ToolTip, DEC.CAM  %w%,100, 200
                  }



        return

        *Wheelup::  ;Spinning the wheel forward or “up” the main motor will speed up.
                if (h="ON")
                        if (y<64) and (Add(y,q)<=64)      ;We consider the main motor has not reverse speed
                        {       y:=Add(Multi(q,Ceil(Div(y,q))),q )
                                SendInput,controlio setport %y%{ENTER}
                                SendInput,controlio setbit 4096{ENTER}
                                SendInput,controlio clearbit 4096{ENTER}
                        }
```

```
		ToolTip, Velocidad:  %y%`n Direccion: %x%`n Pos.camara: %z%,100, 150

	return

	*WheelDown::  ;Spinning the wheel backwards or "down" will slow it "down".
		if (h="ON")
			if (y>0) and (Add(y,-q)>=0)
			{	y:=Add(Multi(q,Ceil(Div(y,q))),-q)
				SendInput,controlio setport %y%{ENTER}
				SendInput,controlio setbit 4096{ENTER}
				SendInput,controlio clearbit 4096{ENTER}
			}
		ToolTip, Velocidad:  %y%`n Direccion: %x%`n Pos.camara: %z%,100, 150

	return
	*LButton:: ;Pressing the left button in boat or camera control provides boat or camera orientation control.
		if (j="BARCO")
		{	if (h="ON")
				if (x>-32) and (Add(x,-e)>=-32)
				{	x:=Add(Multi(e,Ceil(Div(x,e))),-e)
;We show a representative range from -32 to 32 on screen for camera or direction but actual span is 0 to 64 ;being 32 where they are centered not at 0.
					c:=Add(x,32)
				SendInput,controlio setport %c%{ENTER}
				SendInput,controlio setbit 9984{ENTER}
				SendInput,controlio clearbit 9984{ENTER}
				}
		}
		else
			if (k="ON")
				if (z>-32) and (Add(z,-w)>=-32)
				{	z:=Add(Multi(w,Ceil(Div(z,w))),-w)
					v:=Add(z,32)
				SendInput,controlio setport %v%{ENTER}
				SendInput,controlio setbit 19968{ENTER}
				SendInput,controlio clearbit 19968{ENTER}
				}
		ToolTip, Velocidad:  %y%`n Direccion: %x%`n Pos.camara: %z%,100, 150
	return
	*RButton::  ;Right button will give us again control over boat or camera orientation (obviously to the right).
		if (j="BARCO")
		{	if (h="ON")
				if (x<32) and (Add(x,e)<=32)
				{	x:=Add(Multi(e,Ceil(Div(x,e))),e)
				;lomo con pimientos (a-b)
					c:=Add(x,32)
				SendInput,controlio setport %c%{ENTER}
				SendInput,controlio setbit 9984{ENTER}
				SendInput,controlio clearbit 9984{ENTER}
				}
		}
		else
			if (k="ON")
				if (z<32) and (Add(z,w)<=32)
				{	z:=Add(Multi(w,Ceil(Div(z,w))),w)
					v:=Add(z,32)
				SendInput,controlio setport %v%{ENTER}
```

```
                                    SendInput,controlio setbit 19968{ENTER}
                                    SendInput,controlio clearbit 19968{ENTER}
                                }
        ToolTip, Velocidad:  %y%`n Direccion: %x%`n Pos.camara: %z%,100, 150
return
*Space::
        y=0
        x=0
        z=0
          SendInput,controlio setport 0{ENTER}
        SendInput,controlio setbit 4096{ENTER}
        SendInput,controlio clearbit 4096{ENTER}
        SendInput,controlio setport 32{ENTER}
        SendInput,controlio setbit 9984{ENTER}
        SendInput,controlio clearbit 9984{ENTER}
        SendInput,controlio setbit 19968{ENTER}
          SendInput,controlio clearbit 19968{ENTER}
 ToolTip, Velocidad:  %y%`n Direccion: %x%`n Pos.camara: %z%,100, 150
return
  *Alt::
        if (k="OFF")
        {       k=ON
                    SendInput,controlio setport 128 {ENTER}
                    SendInput,controlio setbit 19968{ENTER}
                Sleep,250 ;Step function of 250ms? duration for latching relay switch.
                    SendInput,controlio clearbit 128{ENTER}
                SendInput,controlio clearbit 19968{ENTER}
                z=0
        }
        else
        {       k=OFF
                SendInput,controlio setport 128{ENTER}
                    SendInput,controlio setbit 19968{ENTER}
                    Sleep,250 ;Step function of 250ms? duration for latching relay switch.
                    SendInput,controlio clearbit 128{ENTER}
                SendInput,controlio clearbit 19968{ENTER}

        }
        ToolTip,  Alimentación general: %h% `n Alimentación cámara: %k%,100, 450
return
*Tab::
        if (h="OFF")
        {
                y=0
          SendInput,controlio setport 0 {ENTER}
        SendInput,controlio setbit 4096{ENTER}
        SendInput,controlio clearbit 4096{ENTER}
                x=0
        SendInput,controlio setport 32{ENTER}
        SendInput,controlio setbit 9984{ENTER}
        SendInput,controlio clearbit 9984{ENTER}
                h=ON
        SendInput,controlio setport 128{ENTER}
          SendInput,controlio setbit 4096{ENTER}
          Sleep,100
          SendInput,controlio clearbit 128{ENTER}
        SendInput,controlio clearbit 4096{ENTER}
```

```
}
else
{       h=OFF
  SendInput,controlio setport 128{ENTER}
SendInput,controlio setbit 4096{ENTER}
  Sleep,100
  SendInput, controlio clearbit 128{ENTER}
SendInput,controlio clearbit 4096{ENTER}
}
ToolTip,  Alimentación general: %h% `n Alimentación cámara: %k%,100, 450
return

*x::ExitApp ;exits and enables the mouse to operate normally again.
```

Assembly codes

Assembly code for the camera control microcontroller

This micro has to control the servo that orientates the camera and control the relay that powers all camera systems.

```
        List    p=16F886                ;Microprocessor type
                include  "P16F886.INC"  ;Internal register definition

                __config        _CONFIG1,
_LVP_OFF&_PWRTE_ON&_WDT_OFF&_EC_OSC&_FCMEN_OFF      ;First configuration word
                __config        _CONFIG2, _WRT_OFF&_BOR40V          ;Second configuration word

Temporal            equ     0x20            ;Temporal variable

Period              equ     .1500           ;Period 20000uS (1250*16 Prescaler)

                            org     0x00            ;Reset vector
                            goto    Beginning
                            org     0x05

;Main program

Beginning       clrf    PORTC           ;Output cleaning
                        bsf             STATUS,RP0
                        bsf             STATUS,RP1      ;Banc 3
                        clrf    ANSEL
                        clrf    ANSELH                  ;A and B digital ports
                        bcf             STATUS,RP1      ;Banc 1
                        movlw   b'1101011'
                        movwf   TRISC           ;Pin 3 and pin 5 outputs, the rest, inputs.
                        movlw   Period-1
                        movwf   PR2                     ;period register loading
                        bcf             STATUS,RP0      ;Selects banc 0
;CCP1 module is working in the PWM mode with signal output through RC2/CCP1

                        movlw   b'00001100'
```

```
                movwf   CCP1CON

;TMR2 Is working with a 1:16 preescaler so with a 4MHz frecuency evolves
;every 16uS ((4*Tosc)*16)

                movlw   b'00000111'
                movwf   T2CON                   ;T2 On

;adjusts pulse width with the given input RA5:RA0
Loop    movf    PORTA,W                         ;RA5:RA0 represent a n value
                BTFSS   PORTC,5                         ;Pin 6 (bit 5) is the relay value input
                goto    Label1  ;When relay value is 0 it goes to label1
                andlw   b'11111111' ;When relay value is 1 it works normally
                movwf   CCPR1L                  ;Pulse width adjust (n*16 Prescaler )
                bsf   PORTC,4   ;Relay signal output to high,Pin 5 (bit 4)
                goto    Loop                    ;Endless loop
                Label1
                        movf    PORTA,0
                        btfss   STATUS,Z        ;Checks 0 flag
                        goto    notzero
                        goto    Loop    ;If relay signal is 0 and PORT A is 0 too then ignore (error situation)
                        notzero ;If relay signal is 0 and Port A is different from 0 then send relay signal
                                bcf   PORTC,4   ;Clear relay output if data is different from 0 (no errors)
                                goto    Loop
end
```

Assembly code for the motor control microcontroller

The firmware of this microcontroller deals with different tasks, first of all, it controls the electronic speed controller with a PWM signal. Second, it switches on and off the motor and rudder servo power relay. Third, it has to check the communication link for possible communication errors in which case, the motor will be shut down and a light alarm will also be activated until communication link is recovered.

```
                List    p=16F886                ;Microprocessor type
                include         "P16F886.INC"   ;Internal register definition

;Ajusta los valores de las palabras de configuración durante el ensamblado.Los bits no empleados
;adquieren el valor por defecto.Estos y otros valores se pueden modificar según las necesidades

                __config        _CONFIG1,
_LVP_OFF&_PWRTE_ON&_WDT_OFF&_EC_OSC&_FCMEN_OFF          ;First configuration word
                __config        _CONFIG2, _WRT_OFF&_BOR40V
        ;Second configuration word

                cblock  0x20                            ;Aplication variables init
                        Byte_L                          ;Lower part of the counter var
                        Delay                           ;Variable para la temporización
                        MinCount                        ;Constant for minimum counting of signals to compare
```

```
endc
Period          equ     .1250           ;Period 20000uS (1250*16 Prescaler)


        org     0x00            ;Reset vector
        goto    Beginning
        org     0x04                    ;Interrupt vector
        goto    Inter
        org     0x05
```

;Programa de tratamiento de la interrupción que se provoca cuando el TMR0 temporice 10mS.
;Trabajando a 4MHz el TMR0 evoluciona cada 1uS. Con un preescaler de 256, hay que cargar
;el valor 39 para provocar una interrupción cada 10 mS. Esta se repite 100 veces para obtener una
;temporización total de 1 seg.

```
Inter           decfsz  Delay,F                 ;Ha pasado 1000mS (1") ??
                goto    No_1000_mS              ;No
Si_1000_mS      bcf             T1CON,0                 ;TMR1 en Off, cuenta de pulsos externos
detenida
                bcf             STATUS,C
                movf    TMR1L,W
                movwf   Byte_L                  ;Salva parte baja del contador
                movlw   ~.39
                movwf   TMR0                    ;Repone el TMR0 para temporizar 10 ms
                movlw   .100
                movwf   Delay                   ;Repone variable para temporizar otro segundo
                movlw   b'0000001'
                movwf   MinCount                ;Counter min value is 3 pulses per second to consider
communication
                bcf             INTCON,2        ;Repone flag del TMR0
                clrf    TMR1L
                clrf    TMR1H                   ;Borra el TMR1
                bsf             T1CON,0                 ;TMR1 en On, se inicia la nueva cuenta de
pulsos externos
                retfie

No_1000_mS      movlw   ~.39
                movwf   TMR0                    ;Repone para temporizar otros 10mS
                bcf             INTCON,2                ;Repone el flag del TMR0
                retfie

;Main program


Beginning               clrf    PORTB                   ;B Port clear
                clrf    PORTA                   ;A Port clear
                movlw   b'0000010'
                movwf   PORTC                   ;C Port clear except decoder latching bit
                bsf             STATUS,RP0
                bsf             STATUS,RP1              ;Banc 3
                clrf    ANSEL
                clrf    ANSELH                  ;A and B digital ports
                bcf             STATUS,RP1              ;Banc 1
```

```
                clrf    TRISB                   ;Port B output
                movlw   b'11111111'
                movwf   TRISA                   ;port A input
                movlw   b'01101001'
                movwf   TRISC                   ;Pin 2,3,5 and 8 of port C (RC1,RC2,RC4 and RC7)
outputs, the rest, inputs.
                movlw   b'11000111'
                movwf   OPTION_REG              ;Preescaler de 256 asociado al TMR0
                movlw   Period-1
                movwf   PR2                     ;period register loading
                bcf             STATUS,RP0      ;Selects banc 0


;El TMR1 actúa como contador externo asíncrono y con un preescaler de 1:1
                movlw   b'00000010'
                movwf   T1CON                   ;TMR1 Off
                clrf    TMR1L
                clrf    TMR1H                   ;Puesta a 0 del TMR1

;El TMR0 interrumpe cada 10mS que se repetirá 100 veces para conseguir 1 segundo.
                movlw   .100
                movwf   Delay                   ;Prepara temporización total de 1000mS (1")
                movlw   ~.39
                movwf   TMR0                    ;TMR0 comienza a temporizar 10 ms
                bsf             T1CON,0                 ;TMR1 en On, comienza la cuenta de
pulsos externos
                movlw   b'10100000'
                movwf   INTCON                          ;Habilita interrupción del TMR0

;CCP1 module is working in the PWM mode with signal output through RC2/CCP1

                movlw   b'00001100'
                movwf   CCP1CON

;TMR2 Is working with a 1:16 preescaler so with a 4MHz frecuency evolves
;every 16uS ((4*Tosc)*16)

                movlw   b'00000111'
                movwf   T2CON                   ;T2 On

;Bucle principal del programa

Loop    movf    PORTA,W                         ;RA5:RA0 represent a n value
                andlw   b'11111111' ;When relay value is 1 it works normally
                movwf   CCPR1L                  ;Pulse width adjust (n*16 Prescaler )
                BTFSS   PORTC,5                 ;Pin 6 (bit 5) is the relay value input
                goto    Label
                bsf     PORTC,4 ;Relay signal output to high,Pin 5 (bit 4)
                goto    Label2
Label   bcf     PORTC,4 ;Clear relay output
                goto    Label2
Label2          movf    MinCount,0
                subwf   Byte_L,0
                btfss   STATUS,C        ;If c=0 then Byte_L is lesser than MinCount
                goto    lesser
                goto    bigger
lesser          bcf     PORTC,1
```

```
                bsf PORTC,7
        goto    Loop
bigger  bsf     PORTC,1
                bcf     PORTC,7
        goto    Loop
end                                     ;End of program
```

Assembly code for the rudder control microcontroller

The next assembly code controls the boat's rudder servo with a PWM signal and controls the lights of the boat in different ways: If the communication is lost a fast blinking light alarm will be set, in case battery is low, a slow blinking light alarm and finally, whenever we want it and while no other blinking alarm is active, we can even switch them on or off if we wish.

```
        List    p=16F886                ;Microprocessor type
                include "P16F886.INC"   ;Internal register definition

                __config        _CONFIG1,
_LVP_OFF&_PWRTE_ON&_WDT_OFF&_EC_OSC&_FCMEN_OFF      ;First configuration word
                __config        _CONFIG2, _WRT_OFF&_BOR40V          ;Second configuration word
cblock  0x20                            ;Aplication variables init
                Delay                           ;Variable para la temporización
                Temporal                                ;Temporal variable
        endc

Period          equ     .1250           ;Period 20000uS (1250*16 Prescaler)

                org     0x00            ;Reset vector
                goto    Beginning
                org     0x04                            ;Interrupt vector
                goto    Inter
                org     0x05
;interrupcion

Inter   decfsz  Delay,F                 ;Ha pasado el tiempo delay?
                goto    No_1000_mS              ;No
Si_1000_mS
                movlw   ~.39
                movwf   TMR0                    ;Repone el TMR0 para temporizar
                BTFSC   PORTC,4
                goto    com_perdida
                BTFSC   PORTC,5
                goto    bateria_gastada
                BTFSC   PORTC,0
                goto    luz
                bcf             PORTC,3
                movlw   .10
                movwf   Delay                   ;Repone variable para temporizar otro segundo
                goto    continuar
com_perdida
```

```
                movlw   .25
                movwf   Delay                   ;Repone variable para temporizar otro segundo
                BTFSC   PORTC,3
                goto    cambio1
                bsf             PORTC,3
                goto    continuar
cambio1         bcf             PORTC,3
                goto    continuar
bateria_gastada
                movlw   .50
                movwf   Delay                   ;Repone variable para temporizar otro segundo
                BTFSC   PORTC,3
                goto    cambio2
                bsf             PORTC,3
                goto    continuar
cambio2         bcf             PORTC,3
                goto    continuar
luz
                movlw   .10
                movwf   Delay                   ;Repone variable para temporizar otro segundo
                bsf             PORTC,3
                goto    continuar

continuar
                bcf             INTCON,2        ;Repone flag del TMR0
                retfie

No_1000_mS      movlw   ~.39
                movwf   TMR0                    ;Repone para temporizar otros 10mS
                bcf             INTCON,2                ;Repone el flag del TMR0
                retfie


;Main program

Beginning               clrf    PORTC           ;Output cleaning
                                clrf    PORTB           ;Output cleaning
                bsf             STATUS,RP0
                bsf             STATUS,RP1      ;Banc 3
                clrf    ANSEL
                clrf    ANSELH                  ;A and B digital ports
                bcf             STATUS,RP1      ;Banc 1
                movlw   b'11111111'
                movwf   TRISA           ;RA ports all INPUTS.
                movlw   b'00000000'
                movwf   TRISB           ;Rb ports all outputs.
                movlw   b'00110011'
                movwf   TRISC           ;RC0,RC1,rc4 and RC5 inputs, the rest, outputs.
                movlw   Period-1
                movwf   PR2                     ;period register loading
                movlw   b'11000111'
                movwf   OPTION_REG              ;Preescaler de 256 asociado al TMR0
                bcf             STATUS,RP0      ;Selects banc 0
;El TMR0 interrumpe cada 10mS que se repetirá 100 veces para conseguir 1 segundo.
                movlw   .10
                movwf   Delay                   ;Prepara temporización total de 1000mS (1")
                movlw   ~.39
```

```
                        movwf   TMR0                    ;TMR0 comienza a temporizar 10 ms
                        movlw   b'10100000'
                        movwf   INTCON                  ;Habilita interrupción del TMR0

;CCP1 module is working in the PWM mode with signal output through RC2/CCP1

                        movlw   b'00001100'
                        movwf   CCP1CON

;TMR2 Is working with a 1:16 preescaler so with a 4MHz frecuency evolves
;every 16uS ((4*Tosc)*16)

                movlw   b'00000111'
                movwf   T2CON                   ;T2 On

;adjusts pulse width with the given input RA5:RA0
Loop    movf    PORTA,W                         ;RA5:RA0 represent a n value
                andlw   b'11111111'
                movwf   CCPR1L                  ;Pulse width adjust (n*16 Prescaler )
                goto    Loop
end                                             ;End of source program
```

Encoder and decoder microcontrollers and RF modules

These integrated circuits have been purchased at Reynolds electronics, a RF specialized store located in Daytona Beach (Florida), they are provided with enough documentation and are relatively cheap. Anyway when I purchased them they were not working so I asked them for the firmwares in order to program them again which at the beginning they refused to provide me, but after I could convince them of their fault, they sent them to me and I could program and make them work. Due to the confidentiality of those firmwares I can't show them here.

CATAMARAN RADIOCONTROLADO POR ORDENADOR

DANIEL URTASUN CRUZ
VICENTE SENOSIAIN

Objetivos

- Diseño y construcción de un barco radiocontrolado por ordenador
- Posibilidad de visión en tiempo real de imágenes desde cubierta
- Posibilidad de visión nocturna y control de orientación de la cámara.
- Control de velocidad del barco y control de encendido y apagado de sistemas.

Transmisión radio frecuencia

- Estudio y experimentación de diferentes módulos RF.
- Estudio y prueba de codificadores y decodificadores de datos.
- Estudio de transmisores y receptores de AV (Audio y video) para la cámara
- Prueba simultánea de dos canales de datos a 315MHz y de AV a 950MHz

Conversión de señal AV a USB

- Los PC’s no disponen de entrada AV
- AVerTV es un software acompañado de un dispositivo AV-USB.
- Podemos ver las imágenes en el PC y también grabar, editar y otras tantas opciones.

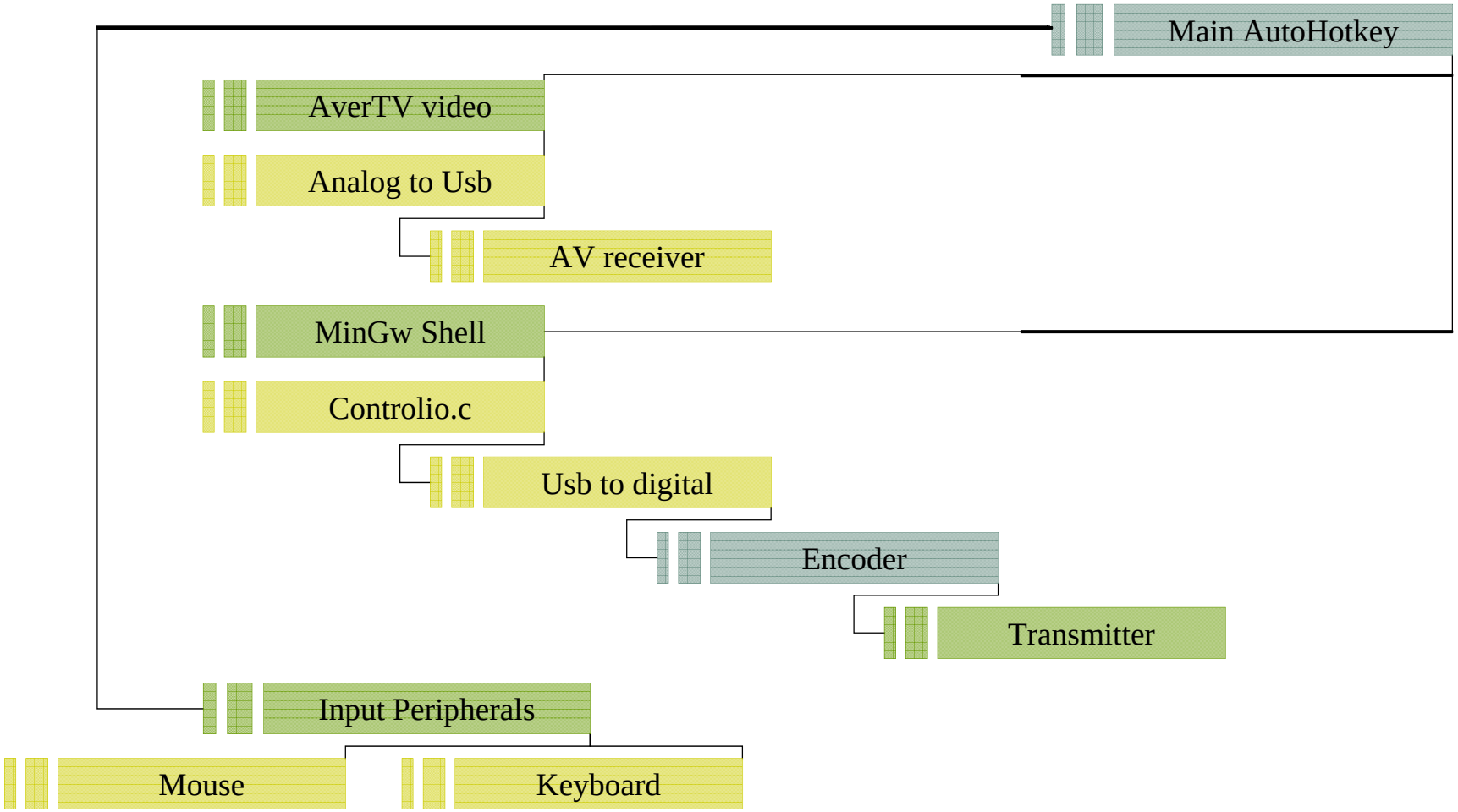
Diseño del interfaz de control

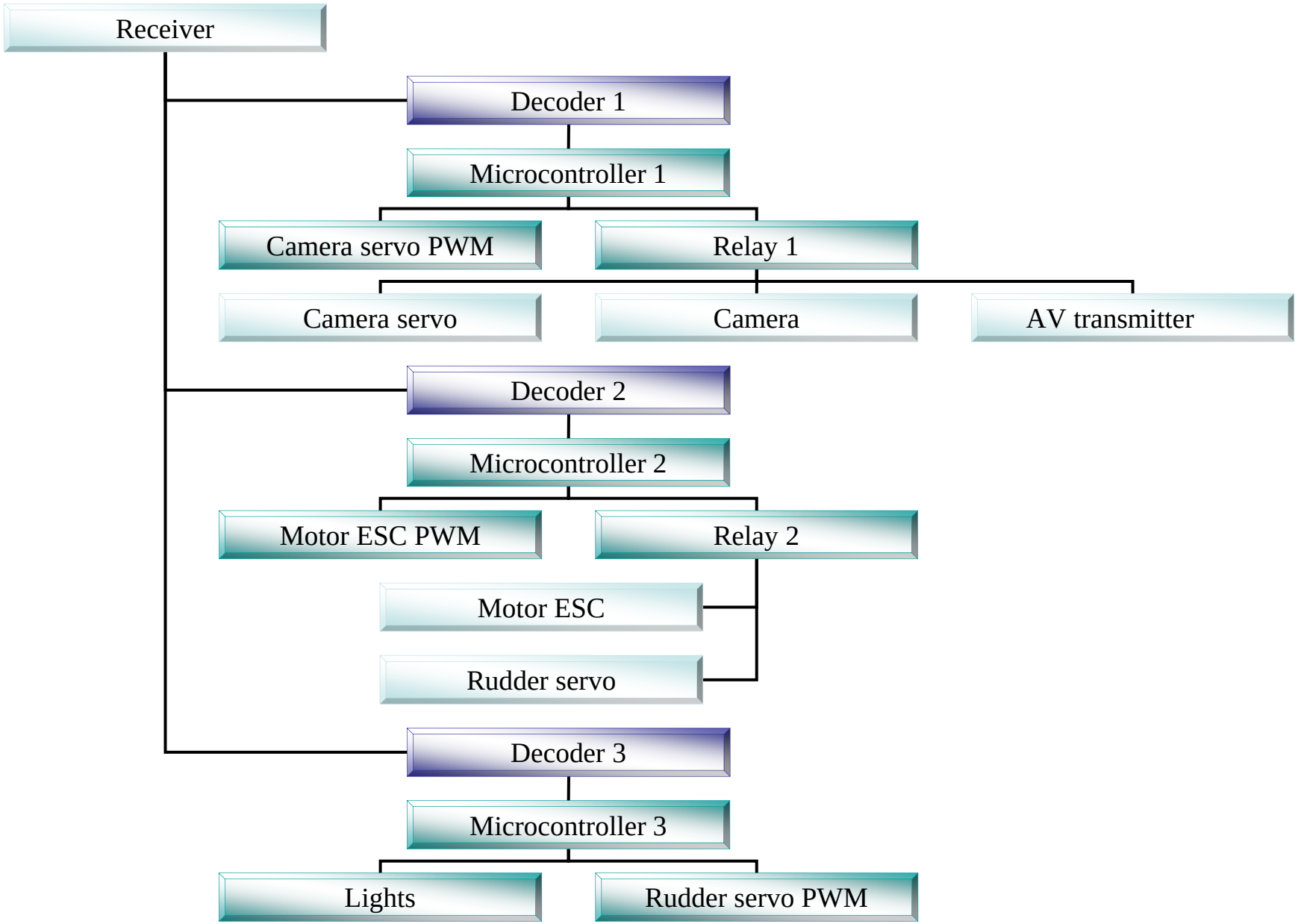
- AutoHotkey es una utilidad de código abierto para Windows.
- Permite automatizar procesos y crear hotkeys haciendo que el teclado o el ratón puedan interactuar con el programa creado.
- Perfecto para realizar el programa principal de control para nuestro PC.

USB to digital I/O extender

- A través de un Shell enviamos comandos al Driver que los envía por USB.
- Estos comandos incluyen números en decimal que se traducen en las conexiones a binario de 12 bits.
- El USB envía los datos a un micro que contiene el firmware que actúa sobre los 12 pines (bits) según el comando enviado.

Diagramas de flujo explicativos



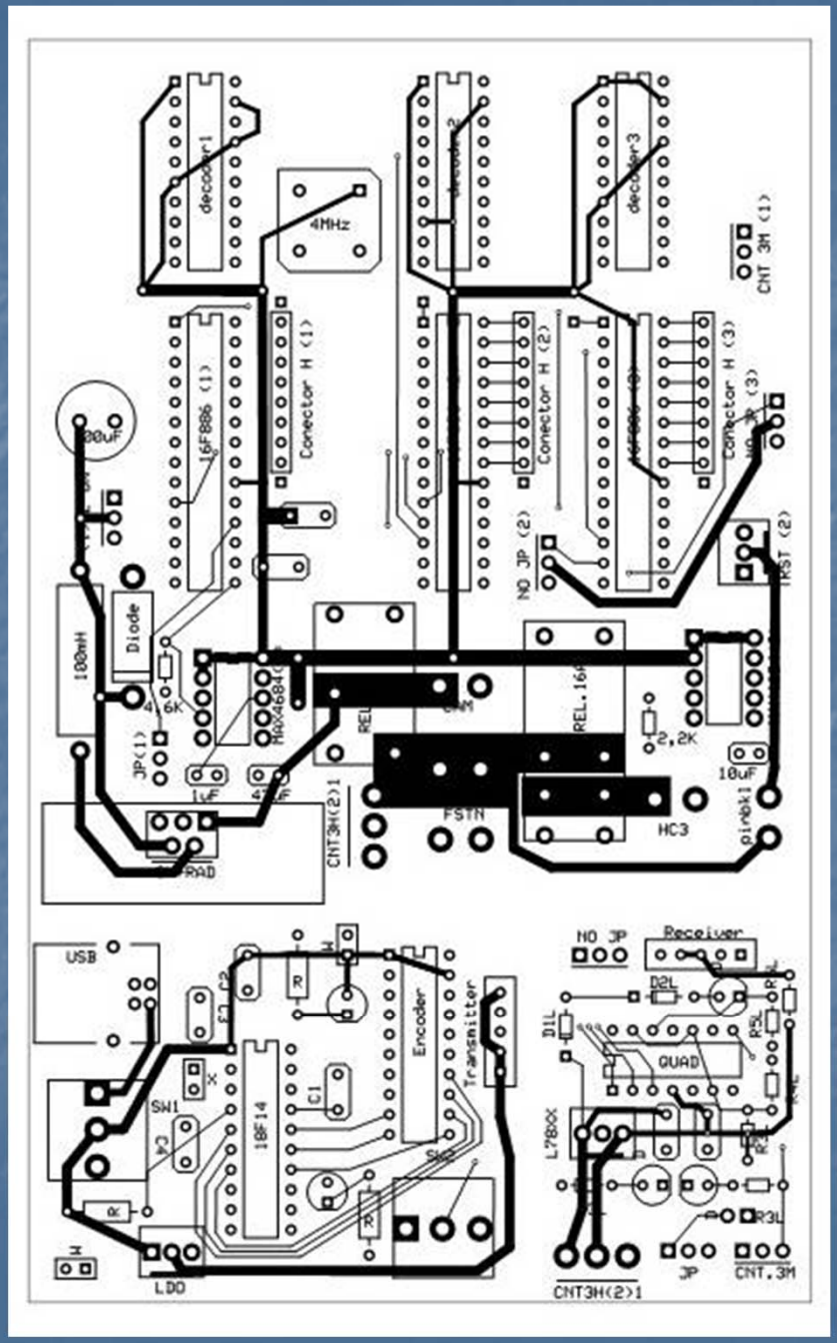


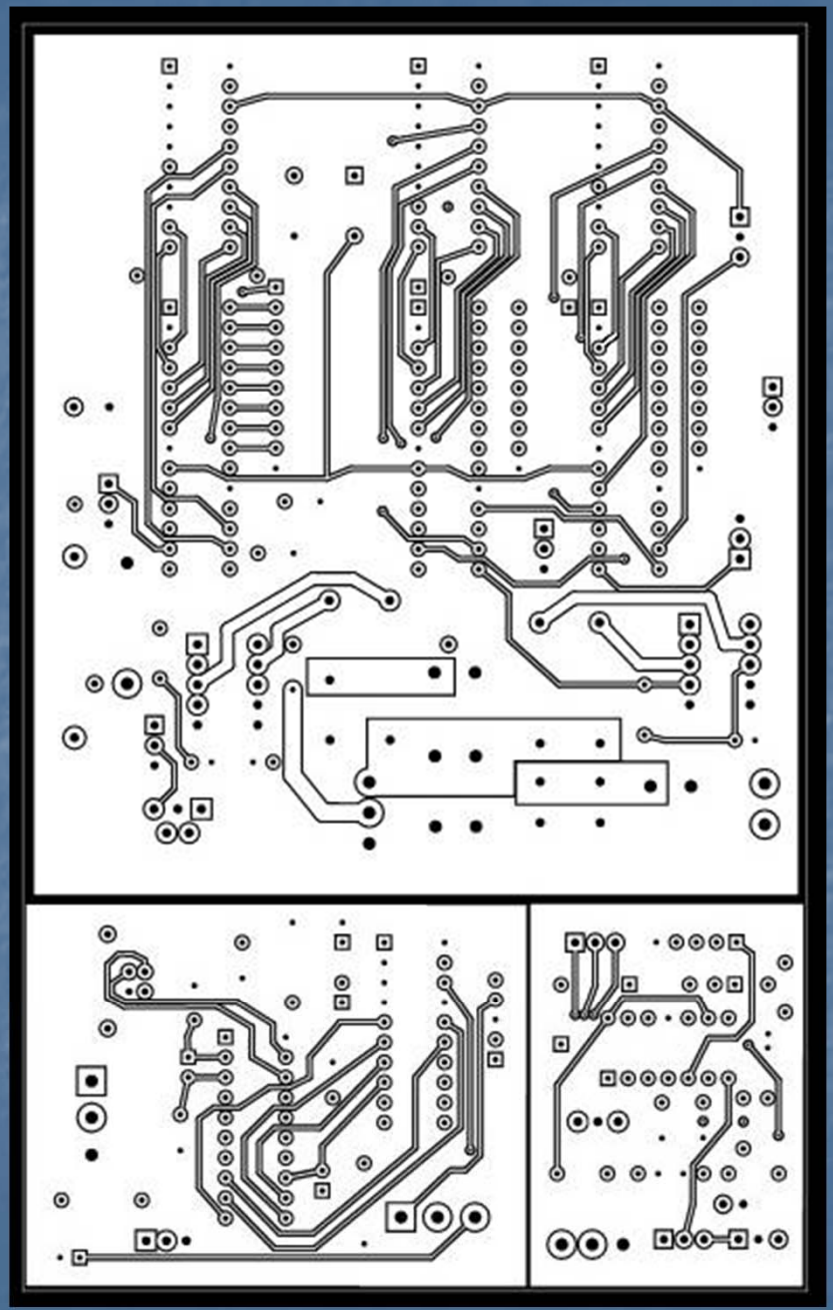
Consideraciones para el diseño de los circuitos

- Detección de fallo de comunicación para evitar perdida de control del barco.
- Detección de batería baja para volver a la orilla con tiempo suficiente.
- Iluminación Led de alarma y de posición.
- Revisión de hojas de características de los componentes y pruebas en placas proto.

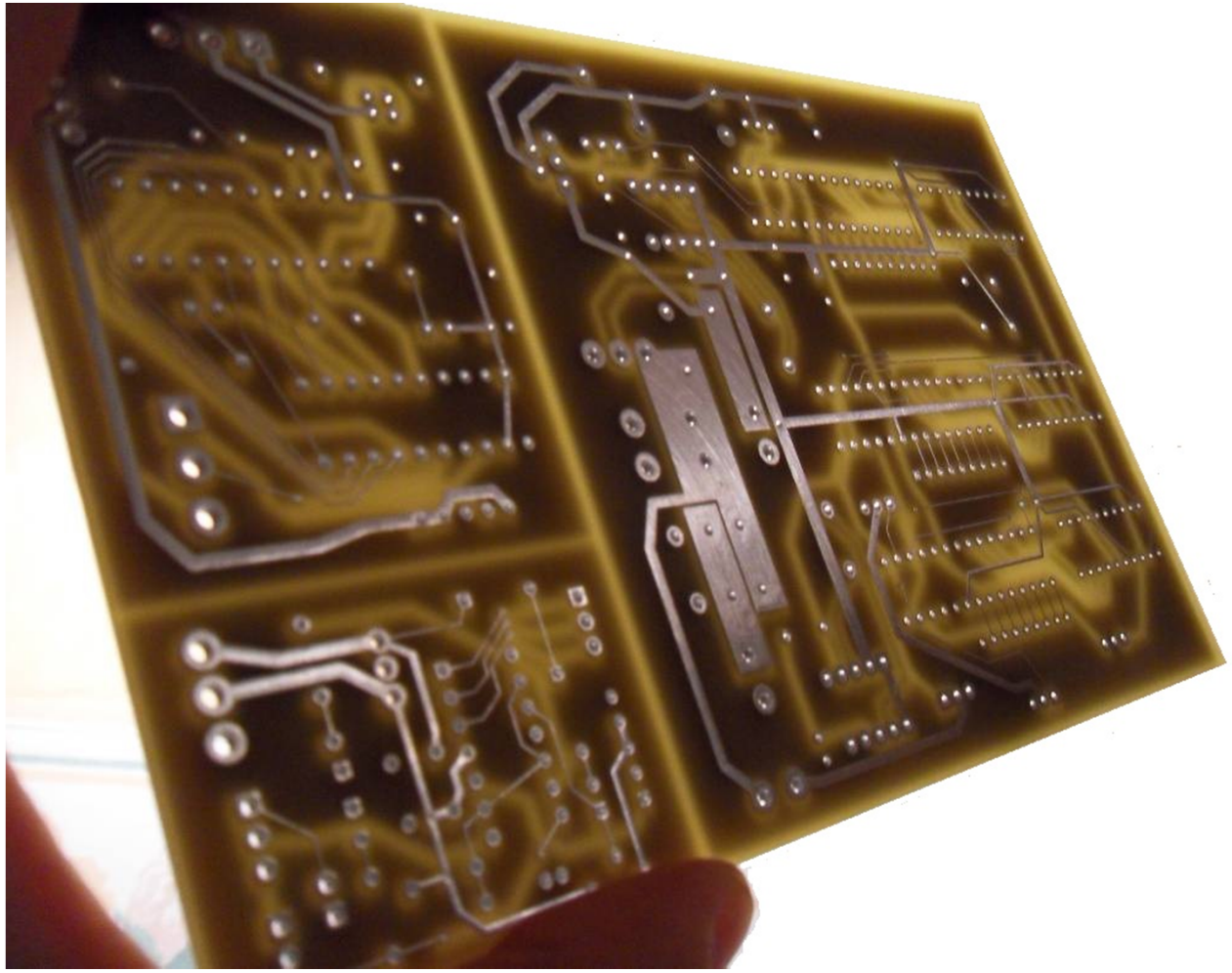
Diseño de circuitos

- Tarjeta principal: Con decodificadores y PIC's Control de relés para el control de barco y cámara control de velocidad, dirección, posición cámara, luces de alarma activadas por batería baja, incomunicación o a voluntad y apagado del motor en caso de incomunicación…
- Tarjeta de transmisión: Envío de datos codificados de forma inalámbrica a largo alcance.
- Tarjeta de recepción: Recibe y acondiciona los datos y sistema de detección de batería baja.

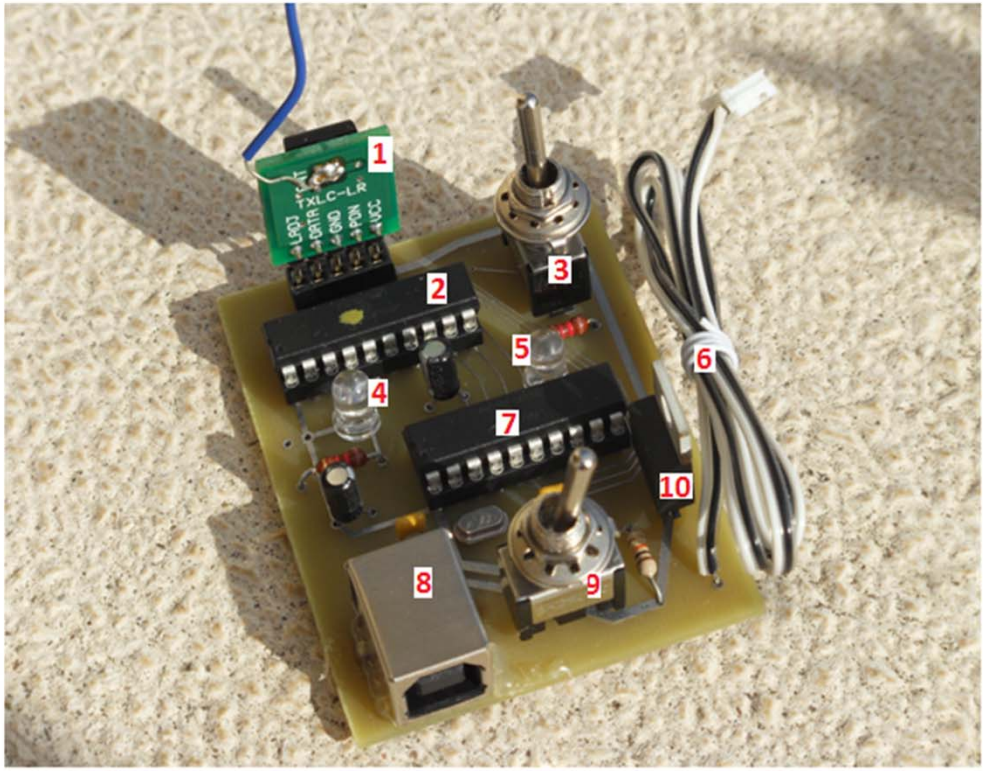




PCB encargada

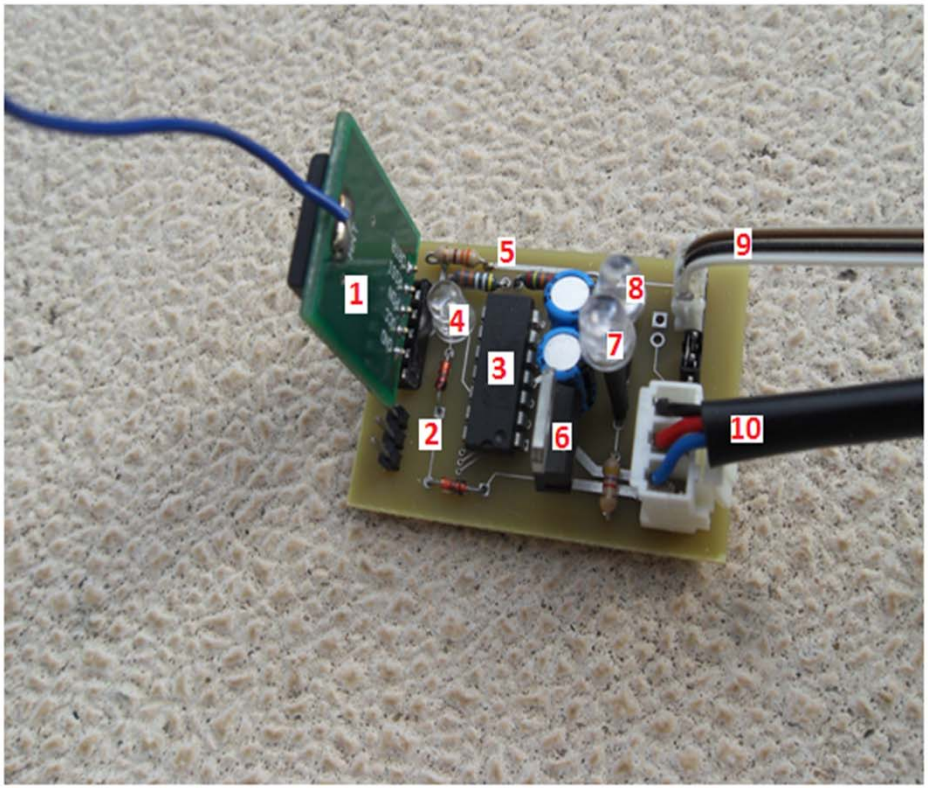


Tarjeta de transmisión



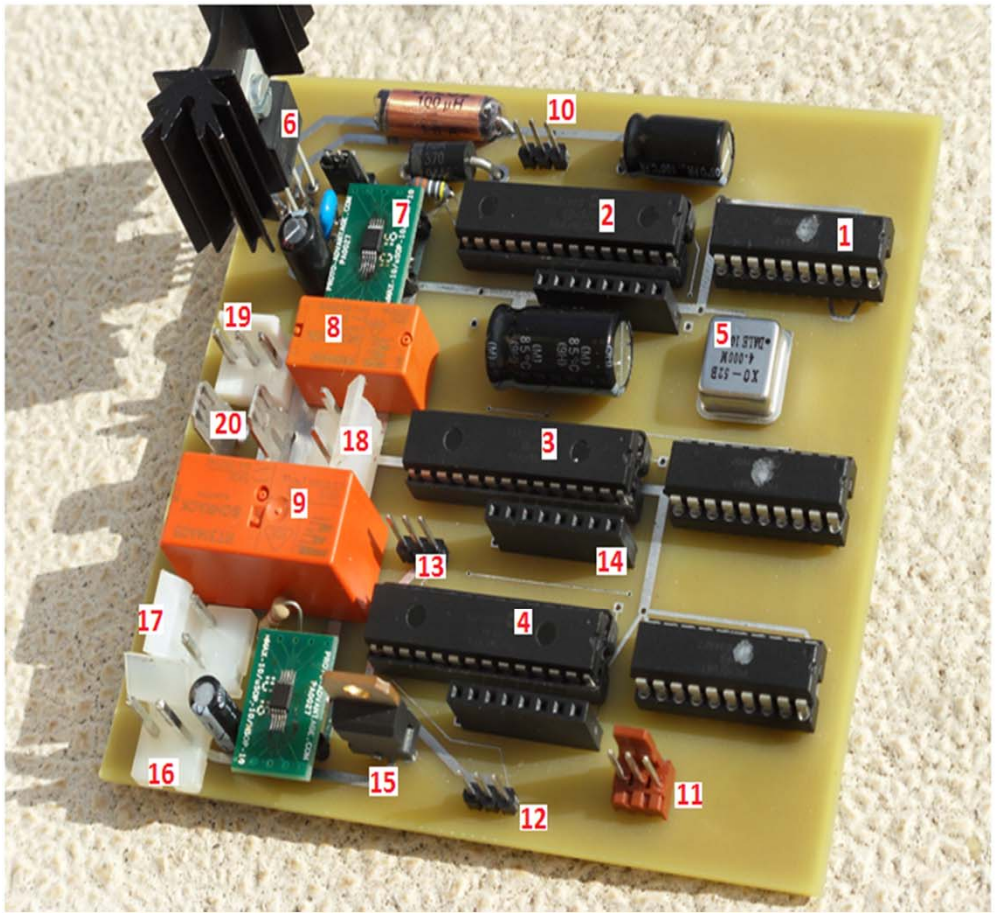
1) Transmitter
2)Encoder (data and directions)
3) wireless/wired communication sw.
4) 4 ON/OFF indicator LED
5) Communication LED
6) Wired communication cable
7) USB to digital converter IC
8) USB connector
9) ON/OFF switch
10) LDO 5v to 3,3v voltage regulator

Tarjeta de recepción



1) Receiver
2) Zenner diodes for Low battery voltage detection circuit
3) 4 Op.Amplifier IC for battery voltage comparators and for receiver output conditioning (3v to 5v)
4) LED brightness proportional to Battery pack's actual charge
5) Voltage divider at 2,5v
6) Main linear voltage regulator 12v to 5v
7) Green LED, main power indicator
8) Red LED, low power indicator
9) Data and battery state communication
10) 12v and 5v power link

Tarjeta de control



1) Decoder 1
2) Pic microcontroller for camera servo and relay control
3) Pic microcontroller for motor control, relay and com. check
4) Pic microcontroller for rudder,lights and battery state.
5) 4MHz canned oscillator
6) Switching voltage regulator for 5v servomotor
7) Latching relay driver circuit
8) Camera, servo and AV transmitter ON/OFF latching relay
9) Main motor and rudder ON/OFF latching relay 20A max
10) Camera orientation servo connector
11) Encoded data and battery state signal connector
12) BEC and motor speed controller signal's connector
13) Rudder servo suply (BEC) and control connector
14) Microcontroller's Port B easy access connector
15) Backlight LEDS ON/OFF switching transistor
16) Backlight LED connector
17) Main motor supply connector (to electronic speed control)
18) Camera and AV transmitter supply connector
19) 12v and 5v connector
20) Main 12v battery pack connector

Conjunto cámara-servomotor

•Cámara de vigilancia resistente al agua y con detección de baja luminosidad y leds infrarrojos para visión nocturna.

•Servomotor de alta velocidad y par capaz de direccionar la cámara en cualquier ángulo desde 0º a 180º estando centrado a 90º.

Conjunto motor-variador

1. Motor brushless potencia 200w y bajas rpm

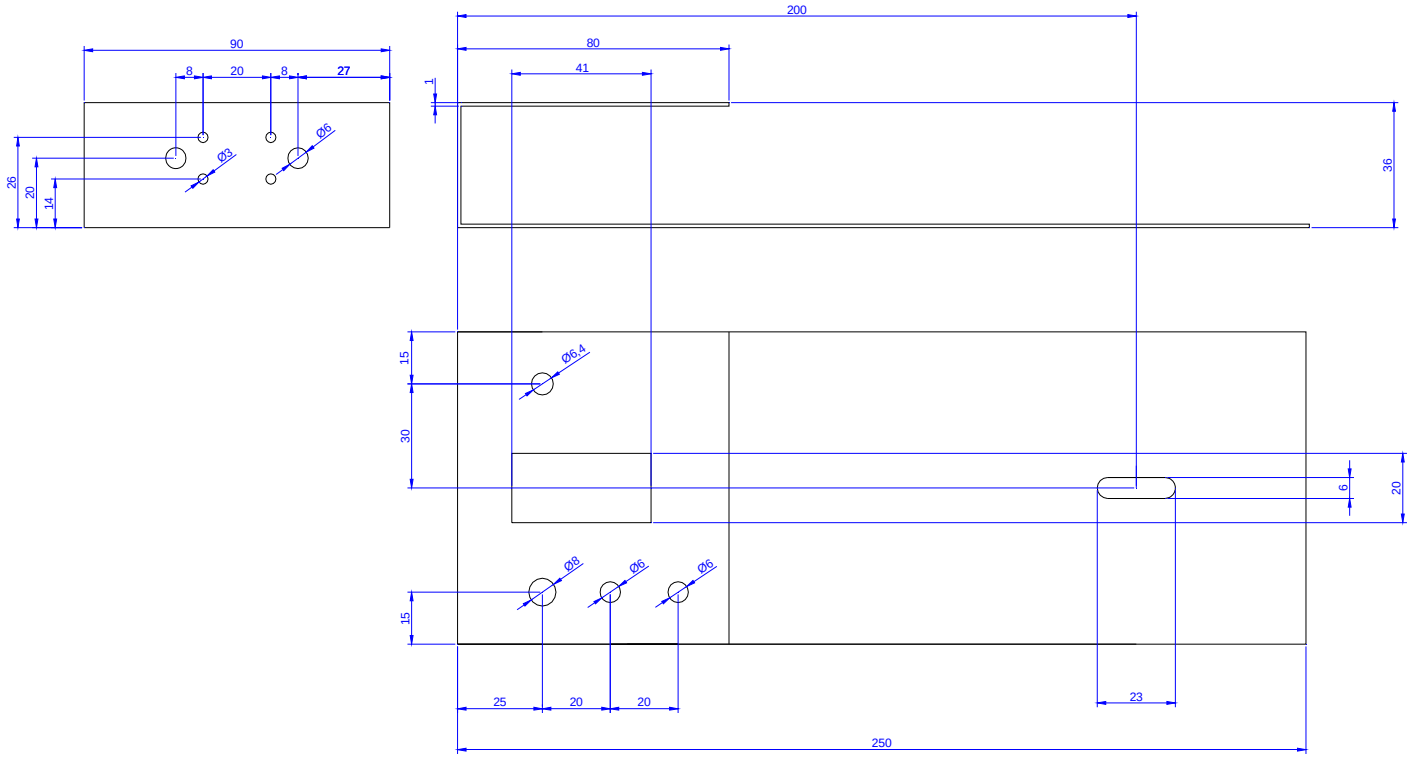
2. Variador de frecuencia trifásico de hasta 18A controlado por PWM con sistema BEC capaz de actuar como regulador de 5v para alimentar servo de timón.

Conjunto timón-hélice

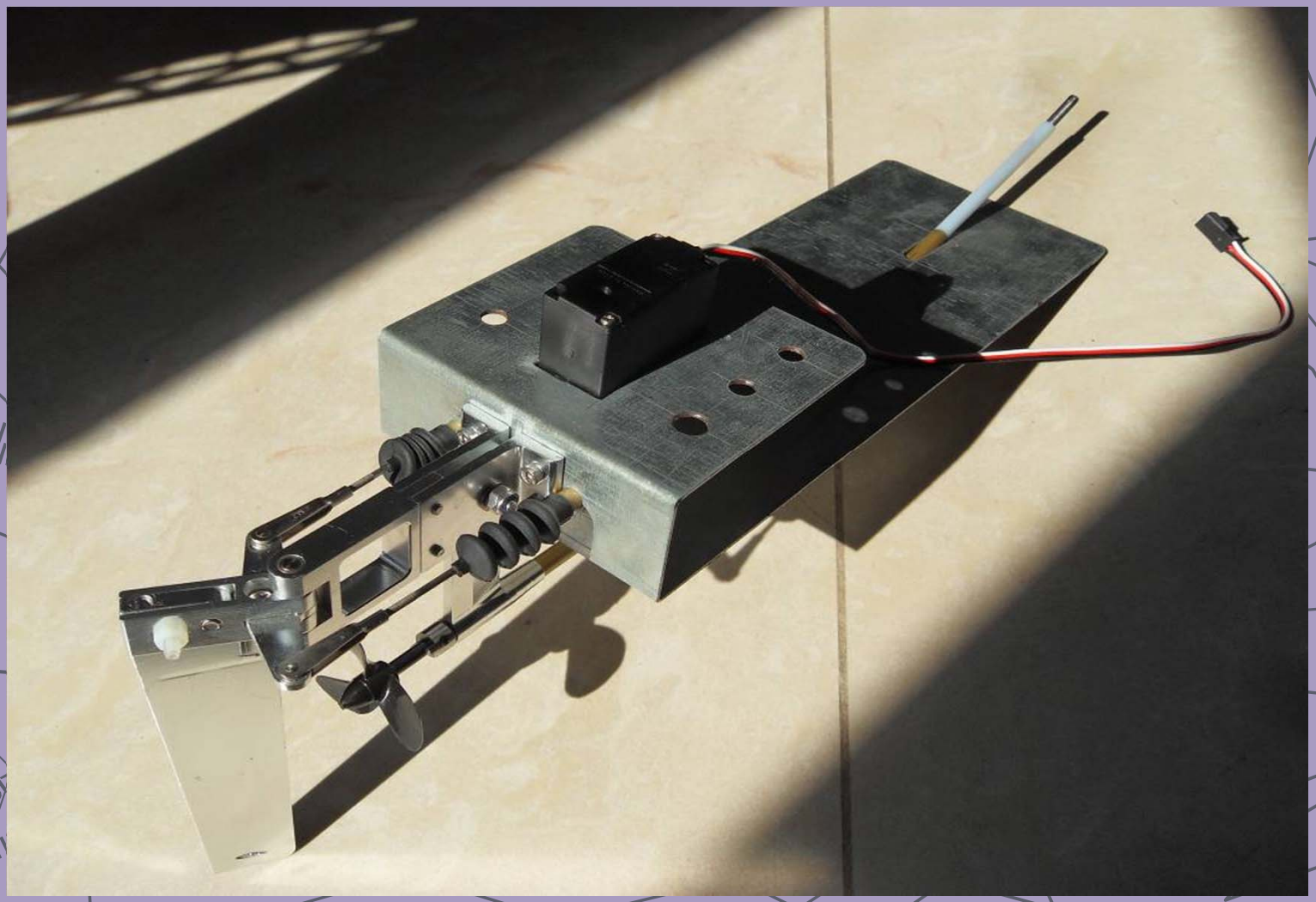
•Conjunto timón compuesto de varias piezas entre las que destaca el eje flexible, cubiertas del eje impermeables, timón, hélice y soporte.

Diseño de montura

- Es necesario crear una montura sobre la cual queden fijados el conjunto timón-hélice y el servo motriz del timón. Plano en AutoCad:

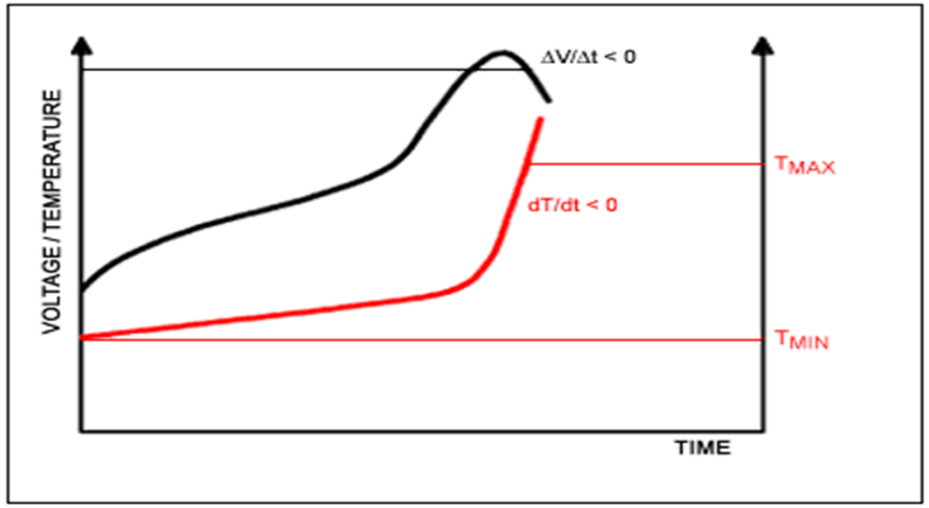


Creación de la montura



Características de la batería

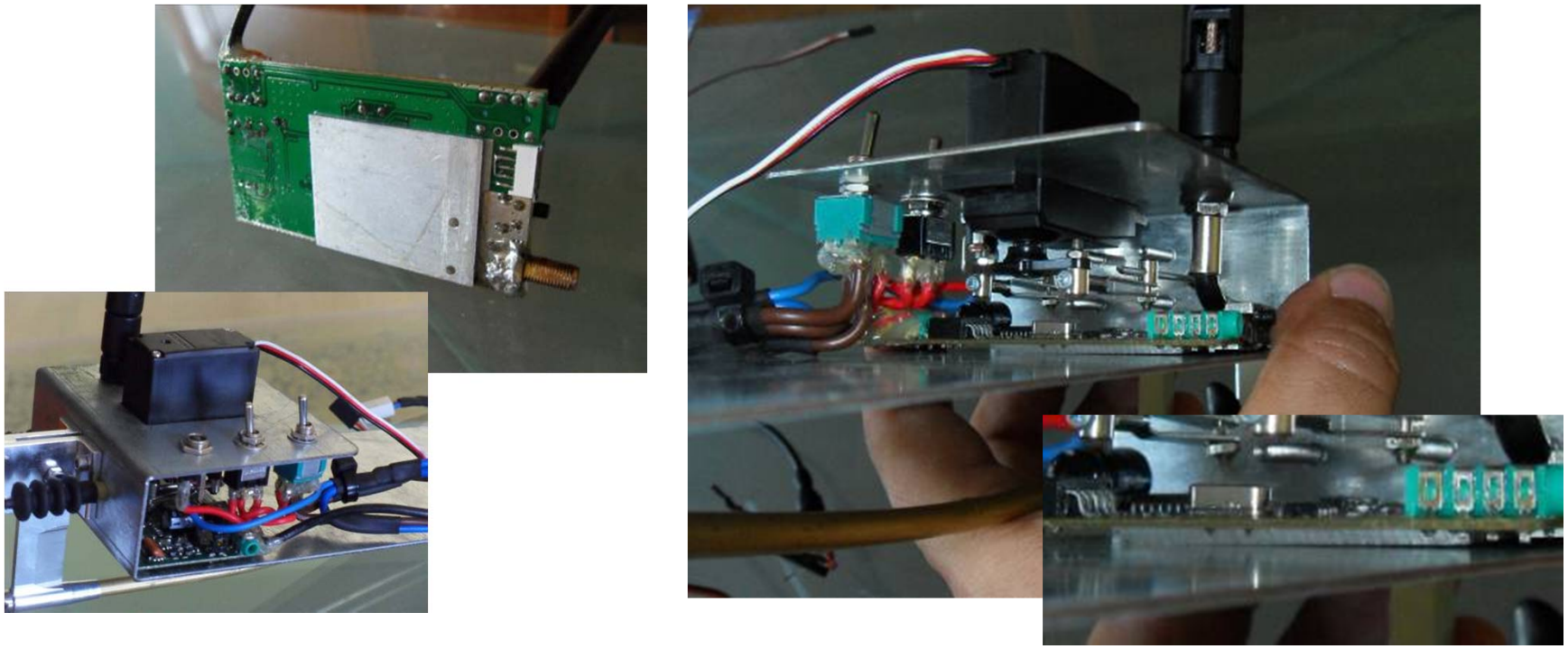
- 2 baterías de 12v, 4200mA/h, alta corriente de descarga tipo NiMh.



- Característica de carga particular, es necesario cargar las baterías de forma independiente para evitar daños.

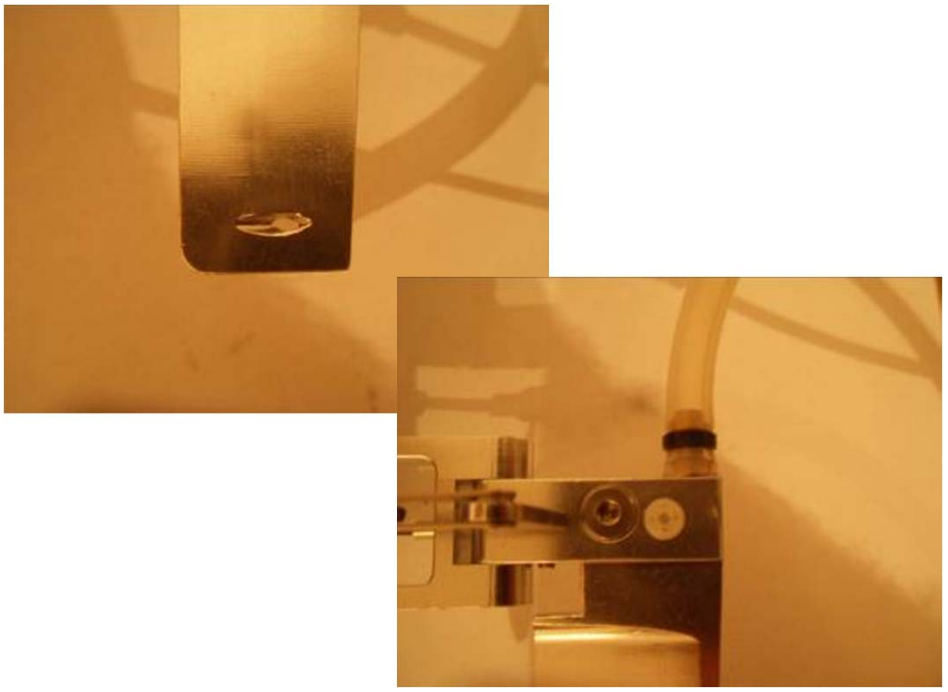
Sistemas de refrigeración

- La tarjeta del transmisor ha sido extraída de su cubierta metálica que hacía las veces de radiador. Es necesario refrigerar.



Sistemas de refrigeración II

- Refrigeración del motor brushless por circulación de agua.



Estabilidad por distribución de masas

- Por ser catamarán es muy estable a momentos de inclinación lateral.
- Veamos los momentos en un monocasco:

Figure 12-9 shows how an INCLINING MOMENT is produced when a weight is moved outboard from the centerline of the ship. If the object weighing 20 tons is moved 20 feet outboard from the centerline, the inclining moment will be equal to 400 foot-tons (F x d, or 20 x 20).

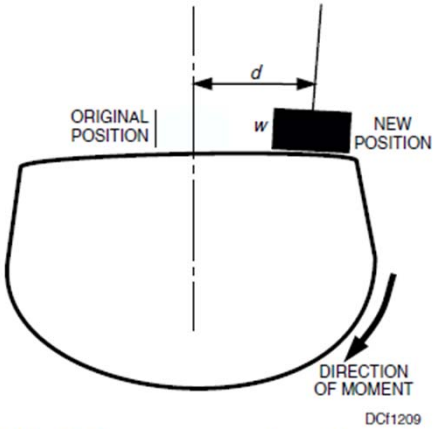


Figure 12-9. Inclining moment produced by moving a weight outboard.

Figure 12-10 shows how a forward (or aft) movement of weight produces a TRIMMING MOMENT. Let's assume that a 20-ton weight is moved 50 feet forward; the trimming moment produced is 20 x 50, or 1,000 foot-tons.

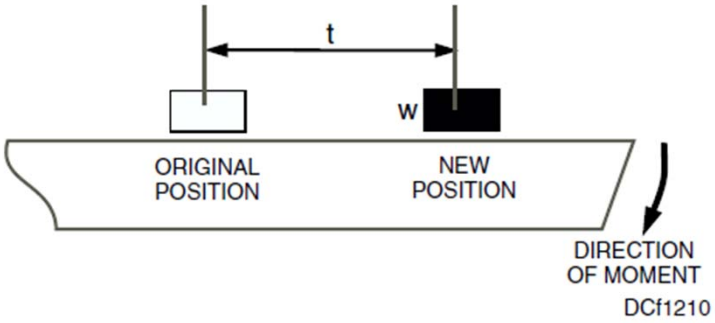


Figure 12-10. Trimming moment.

Estabilidad, gravedad vs. flotabilidad

- En equilibrio estos vectores fuerza son de igual módulo, dirección opuesta y sobre la misma directriz, cancelándose mutuamente.
- Los vectores fuerza debido al peso y presión no necesariamente coinciden en la misma directriz todo el tiempo por lo que se producen momentos que pueden inestabilizar y volcar el barco: Esto sucede concretamente cuando el centro de gravedad está alto respecto al de flotabilidad.

Estabilidad gravedad vs. flotabilidad

If you study figure 12-20, you will notice that a RIGHTING or RESTORING MOMENT is present. This righting moment is caused by the two equal and opposite forces, each of W tons (displacement) magnitude, separated by a distance GZ, which constitutes the LEVER ARM OF MOMENT. Figure 12-20 shows that the ship is stable because the center of buoyancy (B) has shifted far enough to position the buoyant force where it tends to restore the ship to an even keel or an upright position.

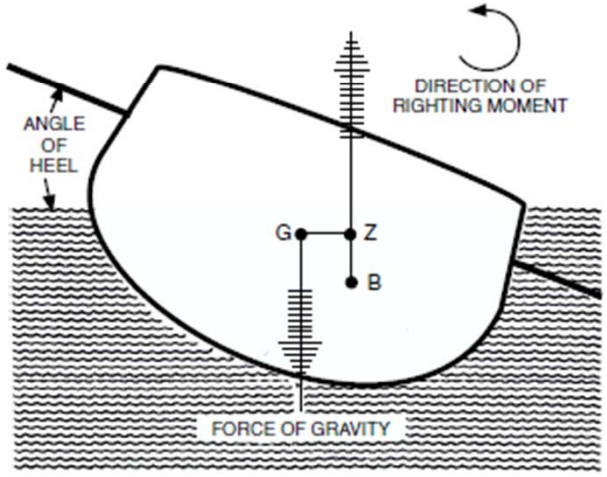


Figure 12-20. Development of righting moment when a stable ship inclines.

However, it is possible for conditions to exist which do not permit B to move far enough in the direction in which the ship rolls to place the buoyant force outboard of the force of gravity. The moment produced will tend to upset the ship, rendering it unstable. Figure 12-21 shows an unstable ship in which the relative positions of B and G produce an UPSETTING MOMENT. In this illustration it is obvious that the cause of the upsetting moment is the high position of G (center of gravity) and the GEOMETRIC CENTER OF THE UNDERWATER BODY (B—the center of buoyancy).

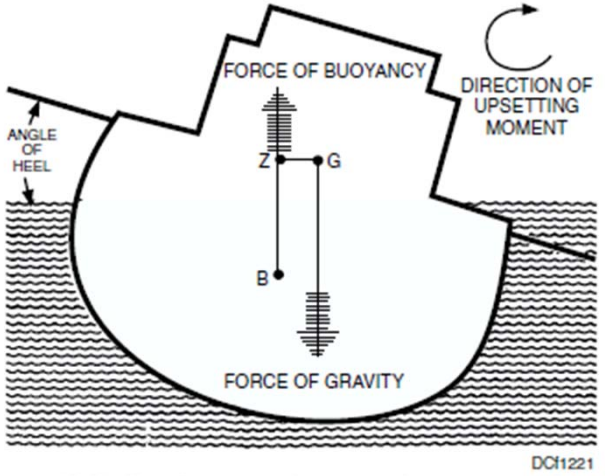
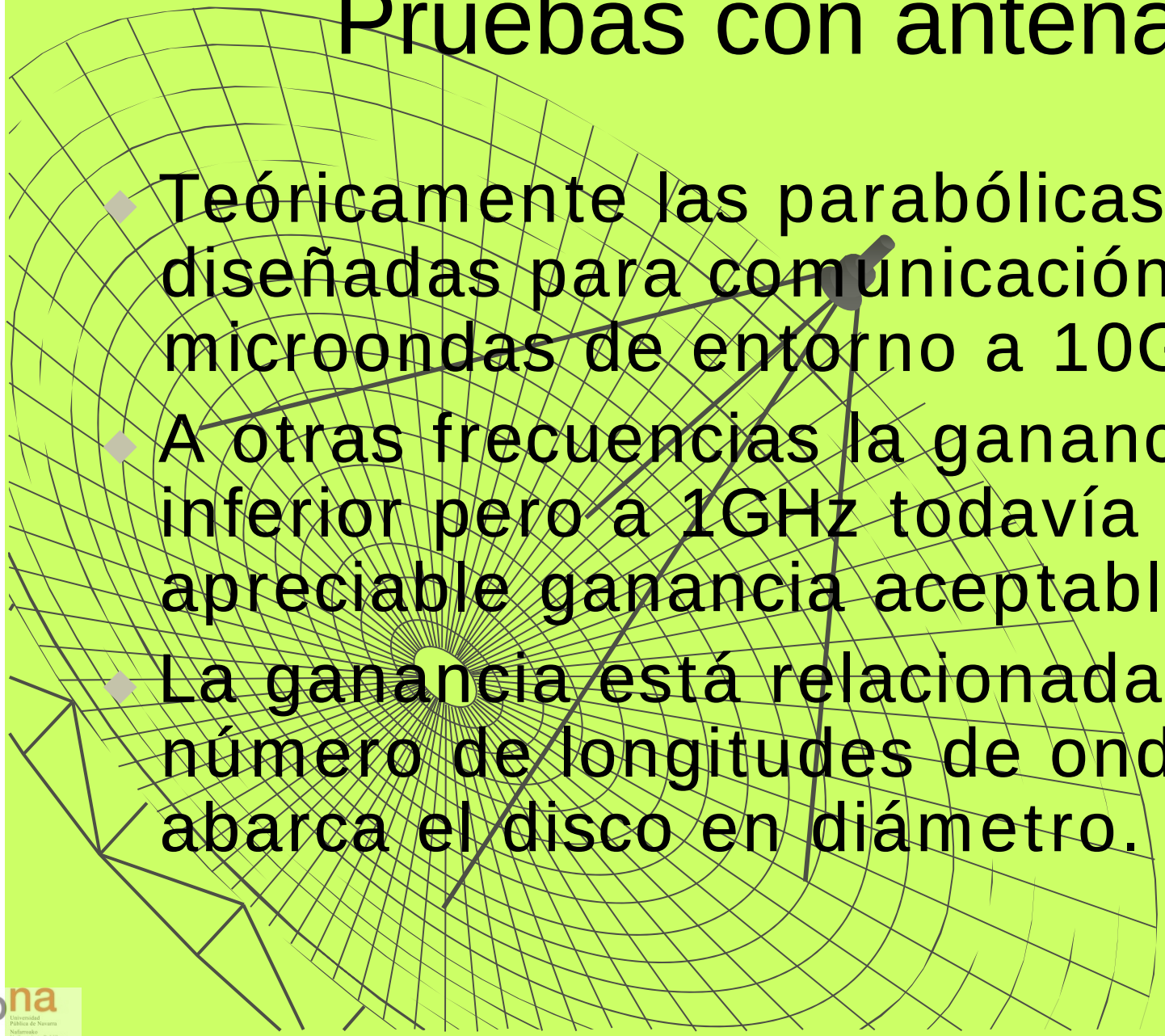


Figure 12-21. Development of an upsetting moment when an unstable ship inclines.

Nivel de la superficie de agua

- El peso del agua desplazada por el barco es igual al peso total del barco
- Con la densidad del agua y conocido el peso sabemos el volumen del agua desplazada o volumen de casco bajo la superficie.
- A partir de una función del volumen del casco en función de la altura, conocido el volumen de agua desplazada podemos despejar donde está el nivel del agua (en equilibrio).

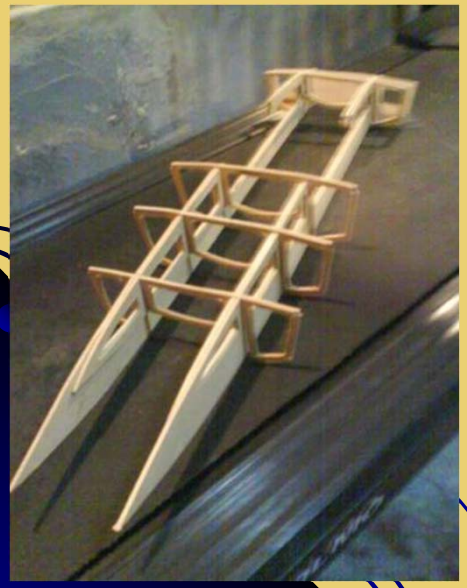
Pruebas con antenas



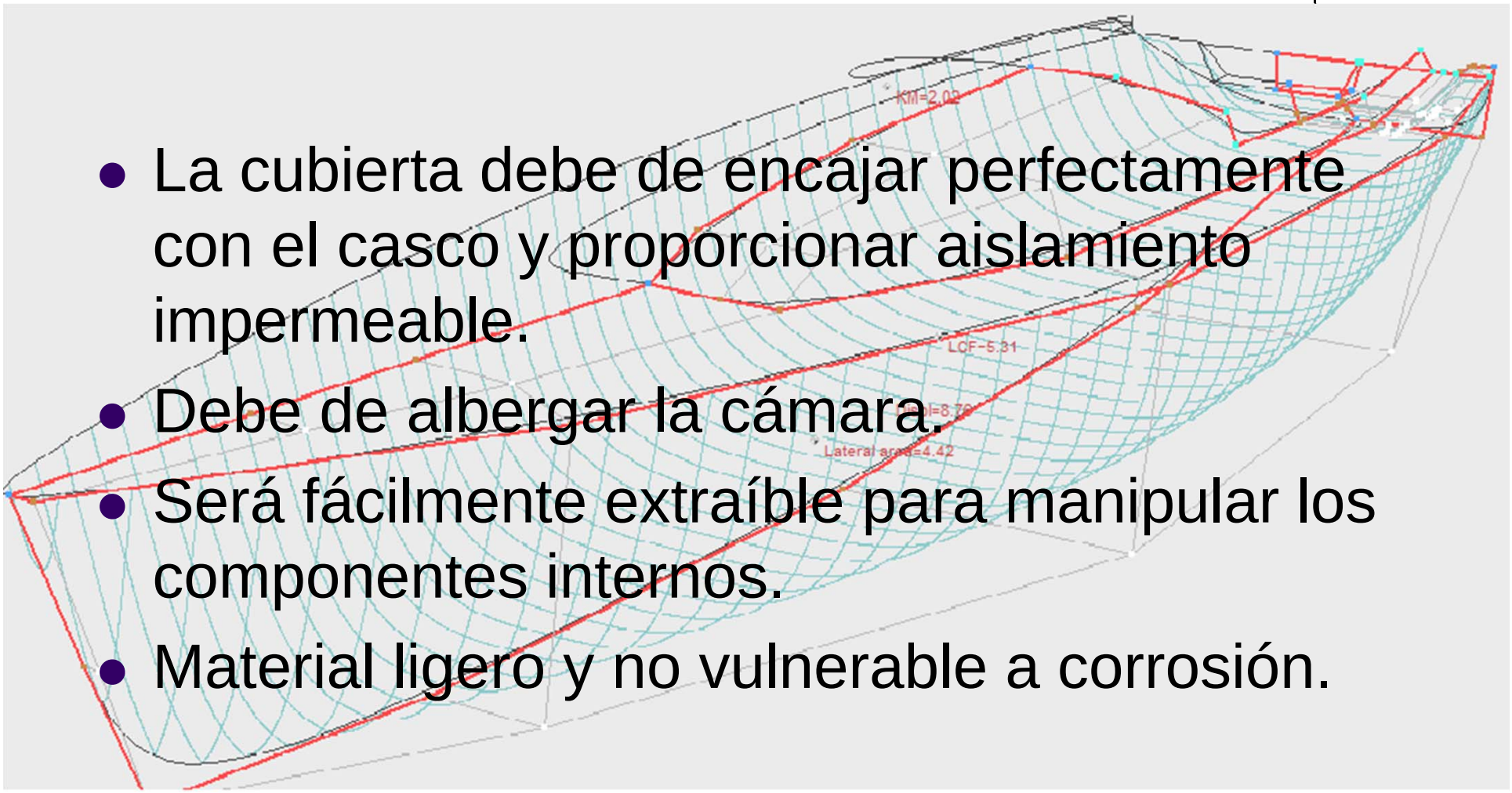
- Teóricamente las parabólicas están diseñadas para comunicación de microondas de entorno a 10GHz
- A otras frecuencias la ganancia es inferior pero a 1GHz todavía es apreciable ganancia aceptable.
- La ganancia está relacionada con el número de longitudes de onda que abarca el disco en diámetro.

Diseño del casco

- Esta parte no esta acabada pero se han tomado ideas de otros proyectos en la red:



Diseño de la cubierta



- La cubierta debe de encajar perfectamente con el casco y proporcionar aislamiento impermeable.
- Debe de albergar la cámara.
- Será fácilmente extraíble para manipular los componentes internos.
- Material ligero y no vulnerable a corrosión.

Control del catamarán

Variables manejadas por el usuario:

Variable	Hotkey	Range
Speed faster/slower	Mouse wheel up/down	0 to 15
Camera/boat control	Mouse middle button	Camera or boat
Boat direction	Mouse click left/right (with boat control active)	- 32 to 32 in steps of 8 or 16 (variable).
Boat direction step	Control +/-	8 or 16
Camera direction	Mouse click left/right (with camera control active)	-32 to 32 in steps of 2,4,8 or 16 (variable).
Camera direction step	+/-	2,4,8 or 16
General supply	G	ON/OFF
Camera supply	C	ON/OFF
Illumination	L	ON/OFF
Exit program	X	-

Pruebas de funcionamiento

- Finalmente se realizarán las pruebas necesarias en un lago.
- VIDEO DEMOSTRATIVO
- ACTIVACION DEL SOFTWARE
- PROYECTO EN INGLES

Conclusiones

- Todos los sistemas funcionan correctamente.
- El diseño del casco y cubierta son los únicos que quedan por diseñar y terminar.
- Necesaria mayor financiación pero la parte electrónica y eléctrica no va a cambiar.