Anexo I planos

LISTA DE MATERIALES

CANTIDAD	DENOMINACIÓN	NORMA	MARCA	PLANO
1	ABRAZADERA		1	3
1	ALOJAMIENTO RODAMIENTOS		2	4
1	CENTRADOR		3	5
1	EJE		4	7
1	ESTATOR		5	6
1	ROTOR		6	3
1	TAPA SUPERIOR		7	5
1	TAPA INFERIOR		8	7
1	BASTIDOR		9	7
1	RUEDA FÓNICA		10	4
3	CASQUILLO 20mm		11	4
1	CASQUILLO ROTOR		12	4
1	CASQUILLO RODAMIENTOS		13	4
2	RODAMIENTO W_63800		14	-
3	TORNILLO CABEZA AVELLANADA M4X16	ISO 10642	15	-
8	TORNILLO CABEZA HEXAGONAL M4X8	ISO 4017	16	-
2	TORNILLO CABEZA HEXAGONAL M4X12	ISO 4017	17	-
1	TORNILLO CABEZA HEXAGONAL M4X20	ISO 4017	18	-
1	TUERCA M4	ISO 4032	19	-
4	TUERCA M10	ISO 4032	20	-

ESCUELA DE INGENIERIAS INDUSTRIALES
DEPARTAMENTO DE INGENIERÍA ENERGÉTICA Y FLUIDOMECÁNICA

DISEÑADO POR: FECHA

ADRIÁN MARTIN PINTOS 17/07/2020

TUTOR:

ANDRÉS MELGAR BACHILLER

UNIVERSIDAD DE VALLADOLID

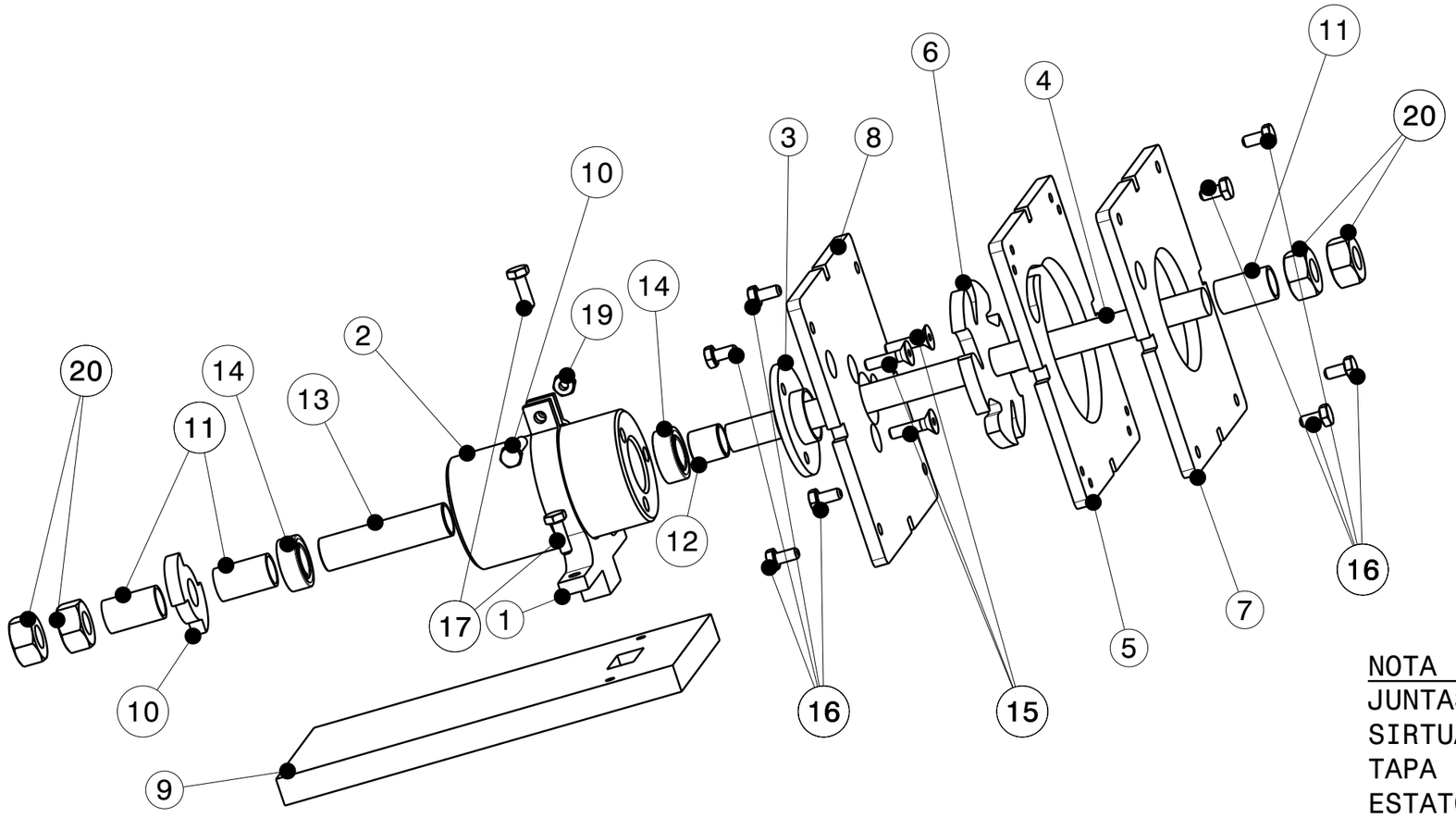
TÍTULO

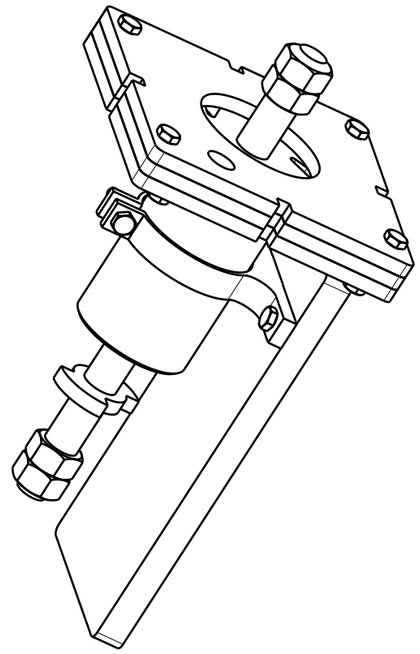
BANCO DE ENSAYO PARA TURBINAS

	NUMERO DE PLANO	REV
A4	1	X

ESCALA 1:1

HOJA 1/7





CANTIDAD	DENOMINACIÓN	MARCA
1	ABRAZADERA	1
1	ALOJAMIENTO RODAMIENTOS	2
1	CENTRADOR	3
1	EJE	4
1	ESTATOR	5
1	ROTOR	6
1	TAPA SUPERIOR	7
1	TAPA INFERIOR	8
1	BASTIDOR	9
1	RUEDA FÓNICA	10

CANTIDAD	DENOMINACIÓN	MARCA
3	CASQUILLO 20mm	11
1	CASQUILLO ROTOR	12
1	CASQUILLO RODAMIENTOS	13
2	RODAMIENTO w_63800	14
3	TORNILLO CABEZA AVELLANADA M4X16	15
8	TORNILLO CABEZA HEXAGONAL M4X18	16
2	TORNILLO CABEZA HEXAGONAL M4X12	17
1	TORNILLO CABEZA HEXAGONAL M4X20	18
1	TUERCA M4	19
4	TUERCA M10	20

NOTA 1: ES NACESARIA LA ELABORACIÓN DE JUNTAS DE PAPEL DE 0,25mm QUE DEBEN SIRTUARSE ENTRE EL ESTATOR (MARCA 5) Y LA TAPA INFERIOR (MARCA 8); Y ENTRE EL ESTATOR Y LA TAPA SUPERIOR (MARCA 7).

NOTA 2: EL MONTAJE SE DEBERÁ SITUAR EN UN EMPLAZAMIENTO QUE NO PERMITA EL MOVIMIENTO DEL BASTIDOR (MARCA 9).

NOTA 3: LOS SOPORTES DE LOS SENSORES DEBERÁN SER DEFINIDOS EN FUNCIÓN DE SUS DIMENSIONES Y POSICIONAMIENTO, ASGURANDO SU CORRECTO FUNCIONAMIENTO, SIN AFECTAR AL DESEMPEÑO DEL CONJUNTO.

ESCUELA DE INGENIERIAS INDUSTRIALES
DEPARTAMENTO DE INGENIERÍA ENERGÉTICA Y FLUIDOMECÁNICA

DISEÑADO POR: ADRIÁN MARTIN PINTOS FECHA 17/07/2020

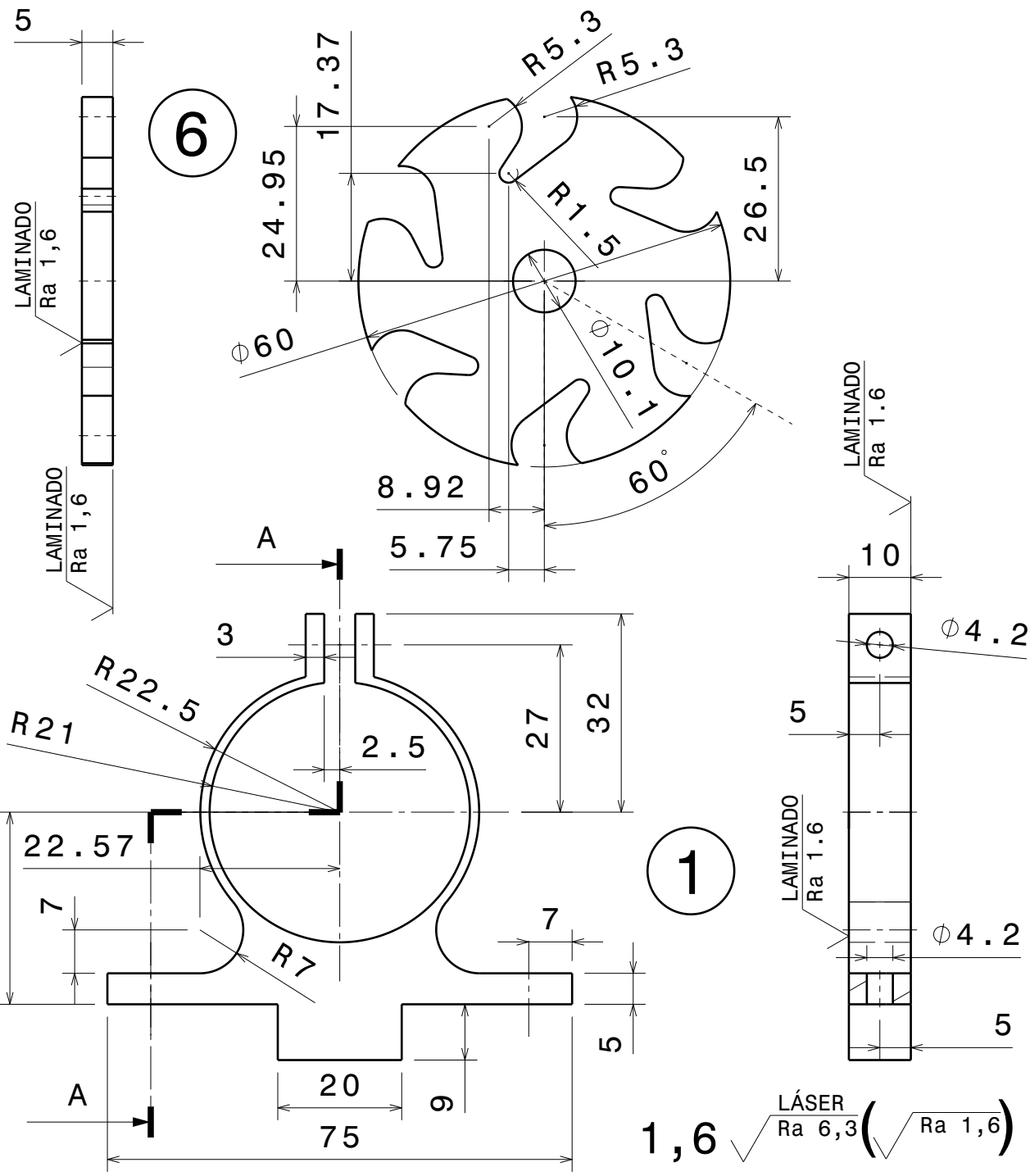
TUTOR: ANDRÉS MELGAR BACHILLER

UNIVERSIDAD DE VALLADOLID

TÍTULO: BANCO DE ENSAYOS PARA TURBINAS

	NÚMERO DE PLANO	REV
A3	2	X

ESCALA 1:2 | HOJA 2/7



MARCA	DENOMINACIÓN	MATERIAL
1	ABRAZADERA	F111
6	RODETE	F111

RADIOS DE REDONDEO 1mm

Tol. Gen. ISO 2768-mK

ESCUELA DE INGENIERIAS INDUSTRIALES
DEPARTAMENTO DE INGENIERÍA ENERGÉTICA Y FLUIDOMECÁNICA

DISEÑADO POR: FECHA

ADRIÁN MARTIN PINTOS 17/07/2020

TUTOR:

ANDRÉS MELGAR BACHILLER

UNIVERSIDAD DE VALLADOLID

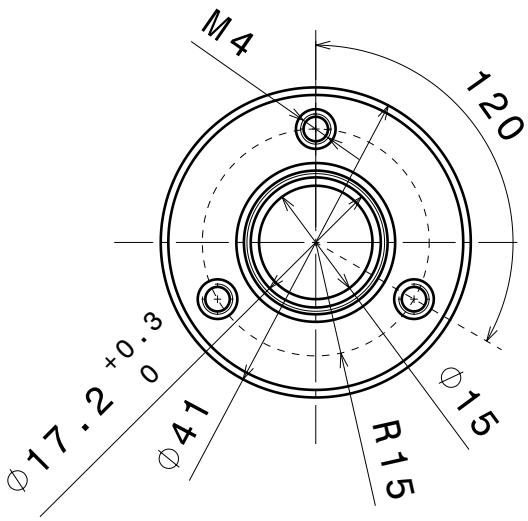
TÍTULO

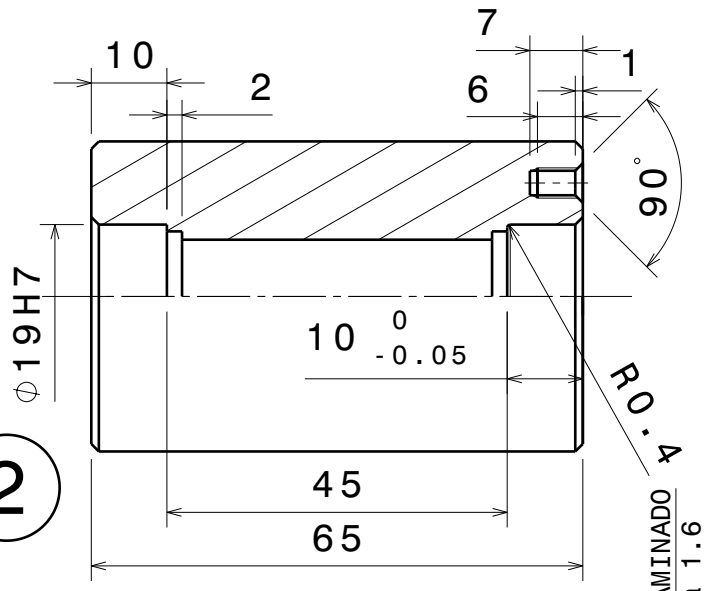
BANCO DE ENSAYO PARA TURBINAS

	NÚMERO DE PLANO	REV
A4	3	X

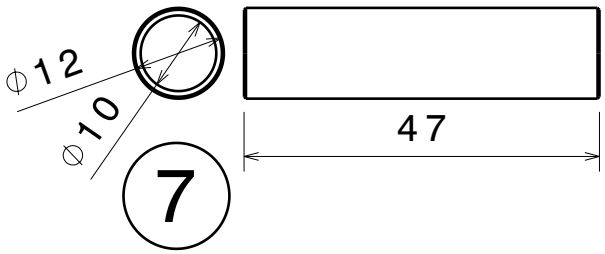
ESCALA 1:1

HOJA 3/7

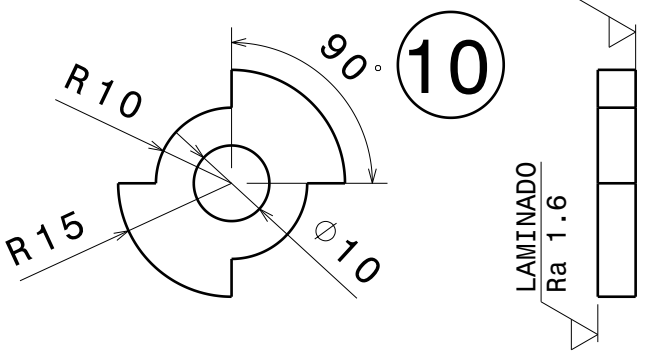




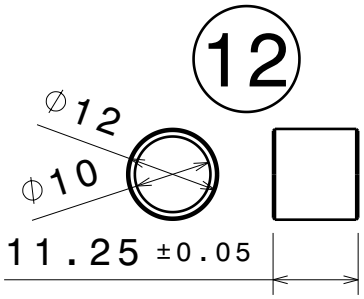
2



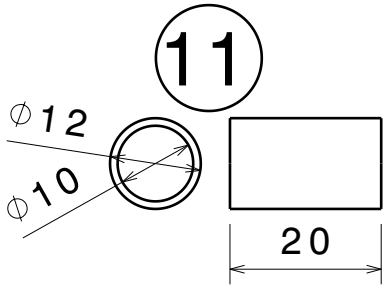
7



10



12



11

10 LÁSER Ra 6,3 (Ra 1,6)

2,7,12,11 Ra 1.6

CHAFLANES GENERALES 1X45º
RADIOS DE REDONDEO 1mm
CHAFLANES CASQUILLOS 0.25X45º

MARCA	DENOMINACIÓN	MATERIAL
2	ALOJAMIENTO RODAMIENTOS	F111
10	RUEDA FÓNICA	F111
11	CASQUILLO 20mm	F111
12	CASQUILLO ROTOR	F111
13	CASQUILLO RODAMIENTOS	F111

Tol. Gen. ISO 2768-mK

ESCUELA DE INGENIERIAS INDUSTRIALES
DEPARTAMENTO DE INGENIERÍA ENERGÉTICA Y FLUIDOMECÁNICA

UNIVERSIDAD DE VALLADOLID

TÍTULO

BANCO DE ENSAYOS PARA TURBINAS

DISEÑADO POR: FECHA

ADRIÁN MARTIN PINTOS 17/07/2020

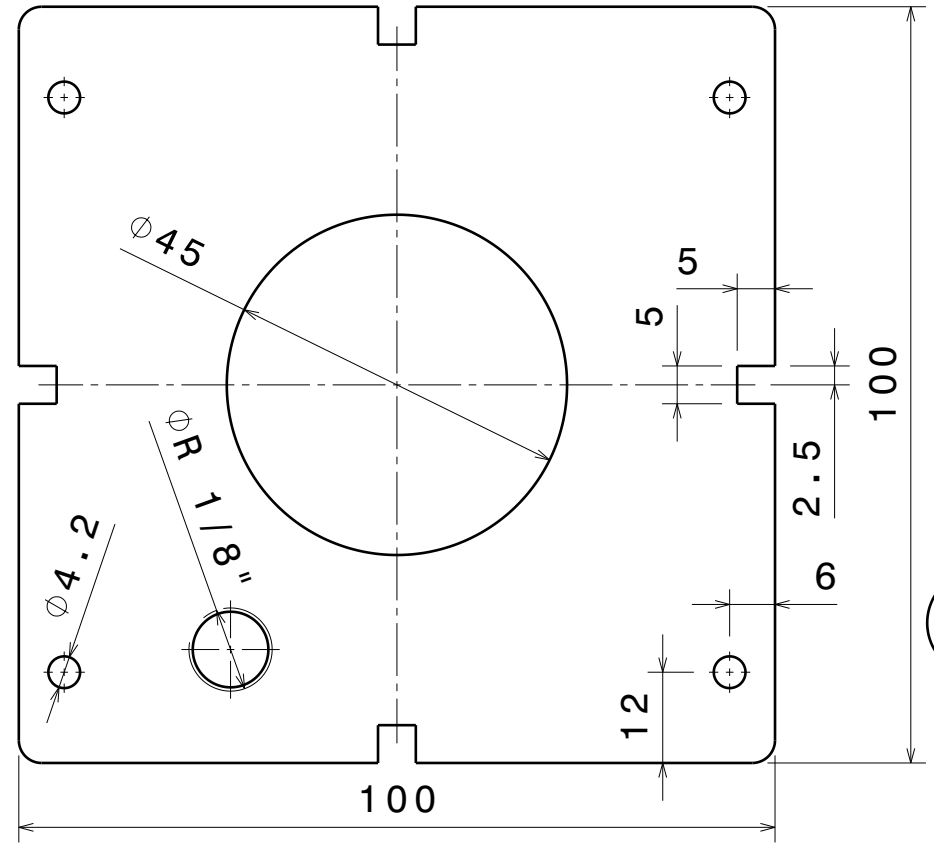
TUTOR:

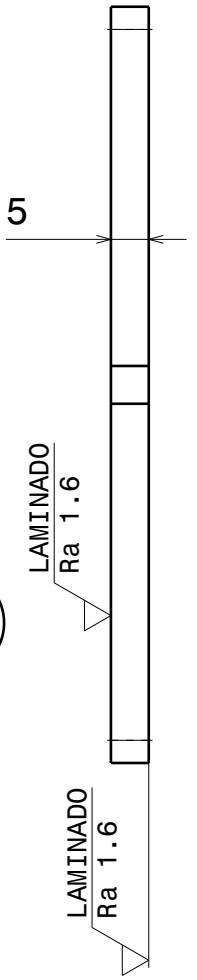
ANDRÉS MELGAR BACHILLER

	NÚMERO DE PLANO	REV
A4	4	X

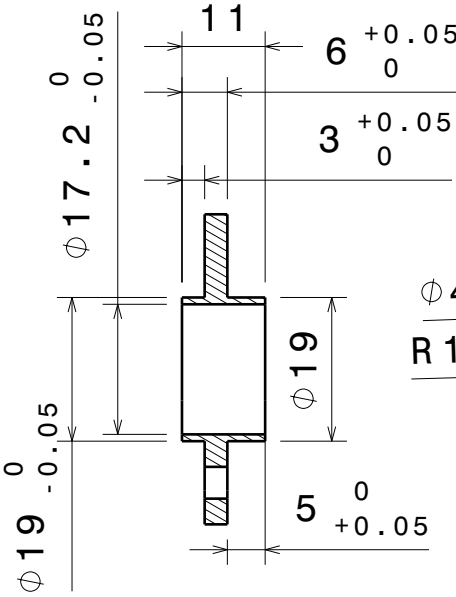
ESCALA 1:1

HOJA 4/7

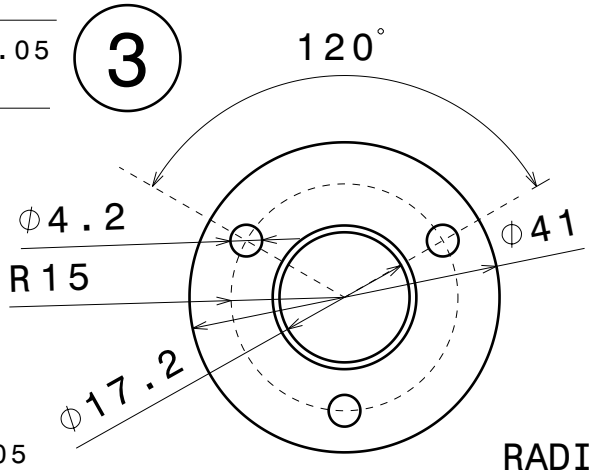




7



3



3 Ra 1,6

7 LÁSER Ra 6,3 (Ra 1,6)

CHAFLANES 0,5X45º

RADIOS DE REDONDEO 1mm

MARCA	DENOMINACIÓN	MATERIAL
7	TAPA SUPERIOR	F111
3	CENTRADOR	F111

Tol. Gen. ISO 2768-mK

ESCUELA DE INGENIERIAS INDUSTRIALES
DEPARTAMENTO DE INGENIERÍA ENERGÉTICA Y FLUIDOMECÁNICA

UNIVERSIDAD DE VALLADOLID

TÍTULO

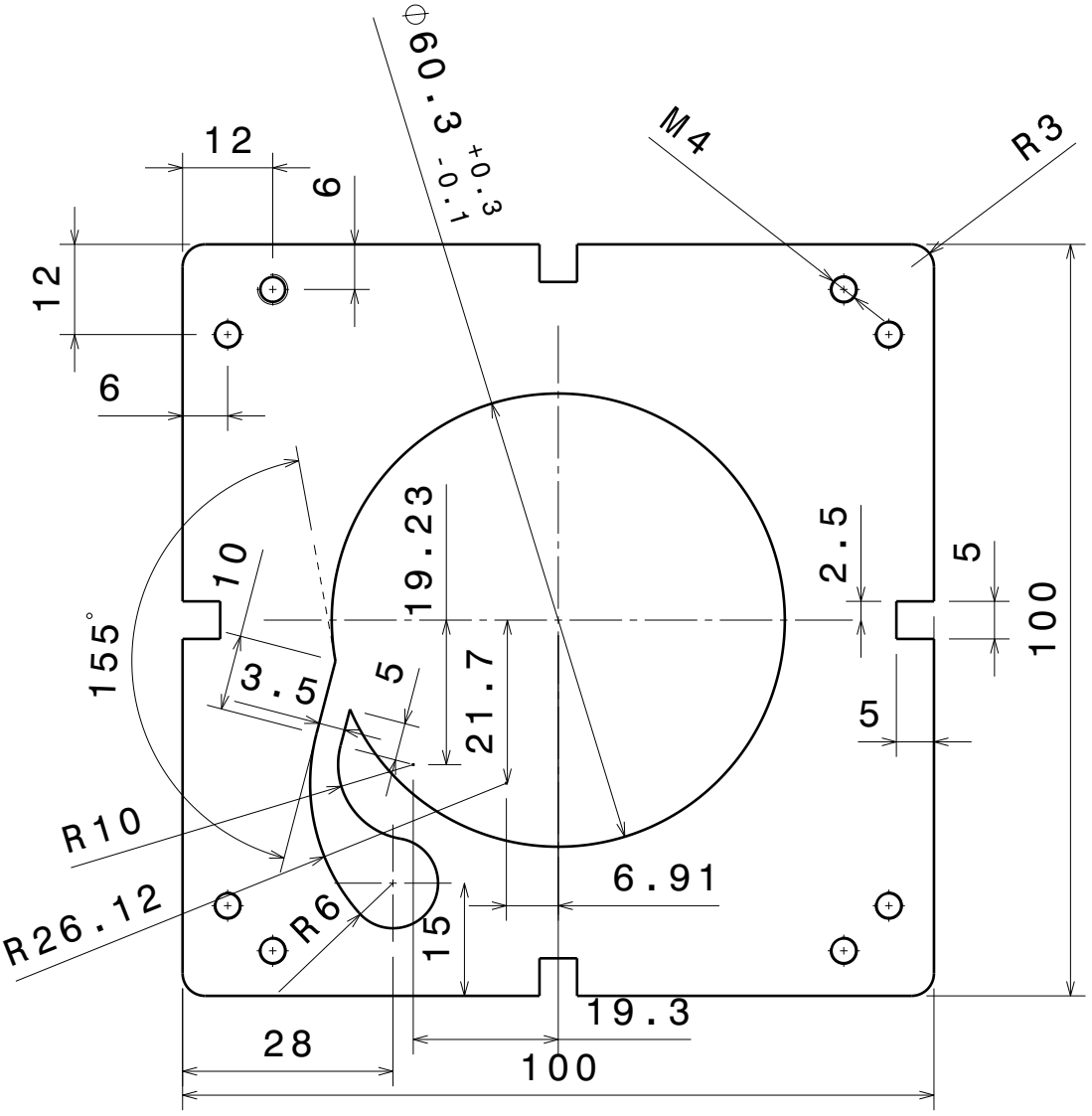
BANCO DE ENSAYOS PARA TURBINAS

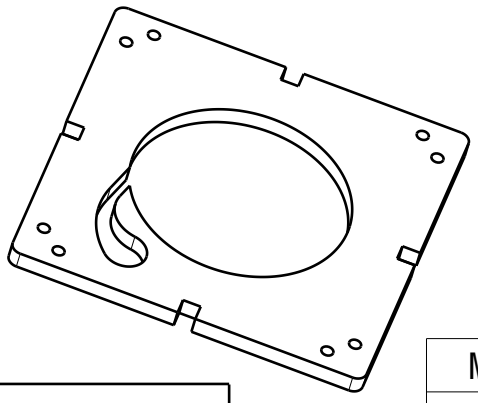
DISEÑADO POR: ADRIÁN MARTIN PINTOS FECHA 17/07/2020

TUTOR: ANDRÉS MELGAR BACHILLER

A4 | NÚMERO DE PLANO 5 | REV X

ESCALA 1:1 | HOJA 5/7





LÁSER
Ra 6,3 (Ra 1,6)

RADIOS DE REDONDEO 0.5 mm

MARCA	DENOMINACIÓN	MATERIAL
5	ESTÁTOR	F111

Tol. Gen. ISO 2768-mK

ESCUELA DE INGENIERIAS INDUSTRIALES
DEPARTAMENTO DE INGENIERÍA ENERGÉTICA Y FLUIDOMECÁNICA

UNIVERSIDAD DE VALLADOLID

TÍTULO

BANCO DE ENSAYOS PARA TURBINAS

DISEÑADO POR: FECHA

ADRIÁN MARTIN PINTOS 17/07/2020

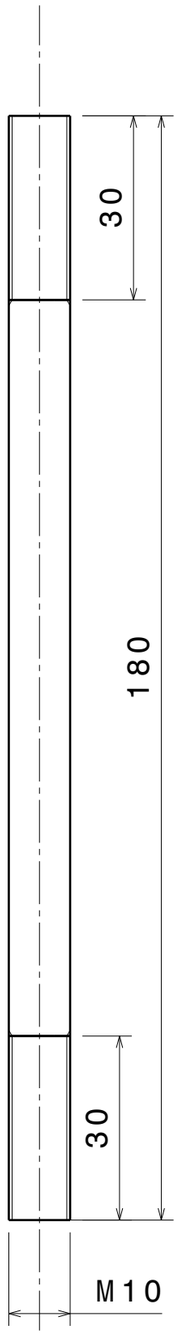
TUTOR:

ANDRÉS MELGAR BACHILLER

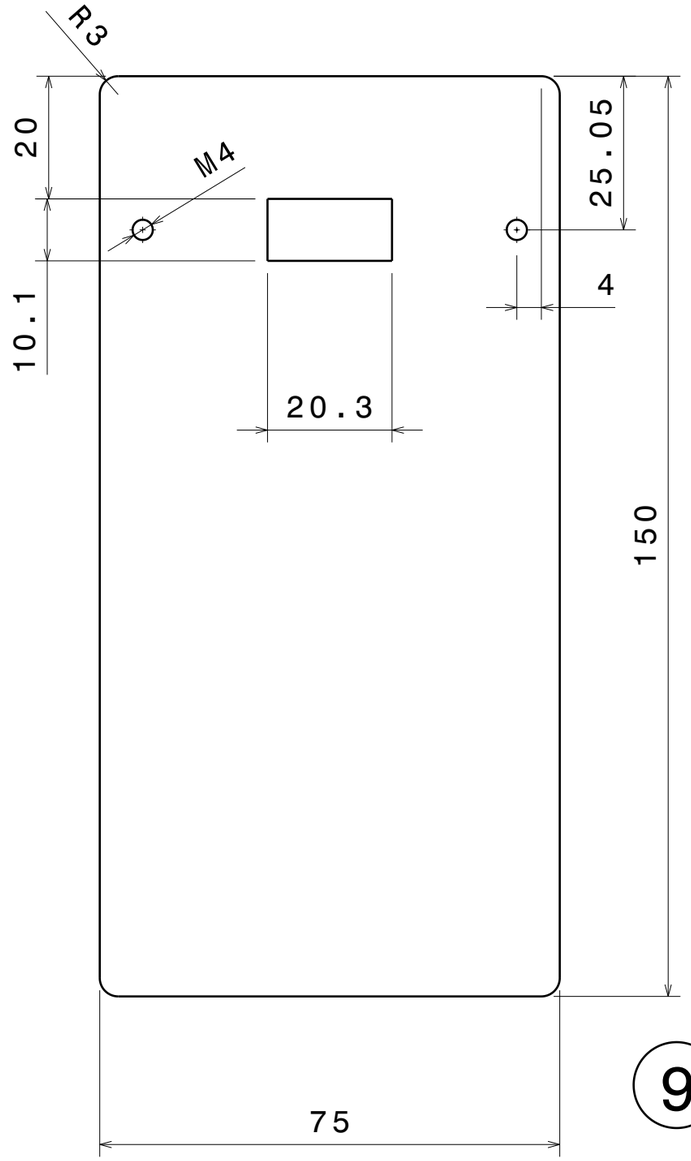
A4	NÚMERO DE PLANO 6	X

ESCALA 1:1

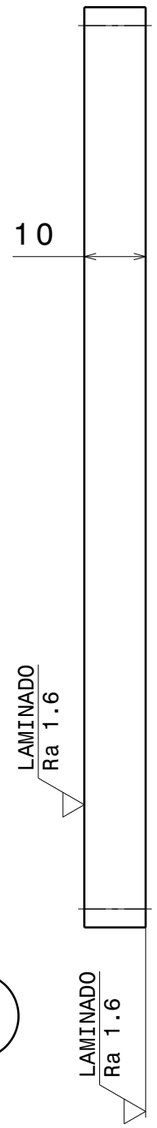
HOJA 6/7

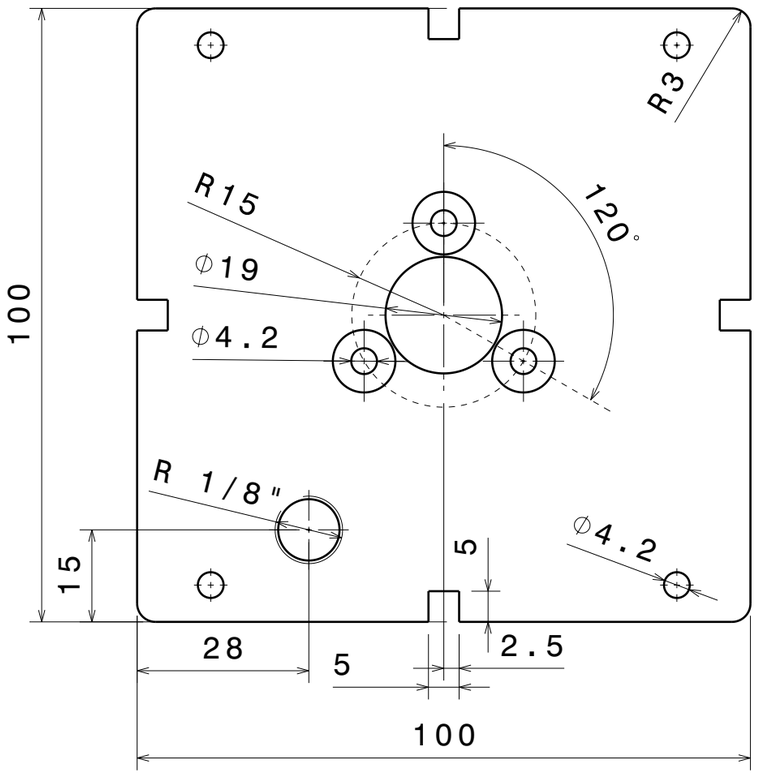


4

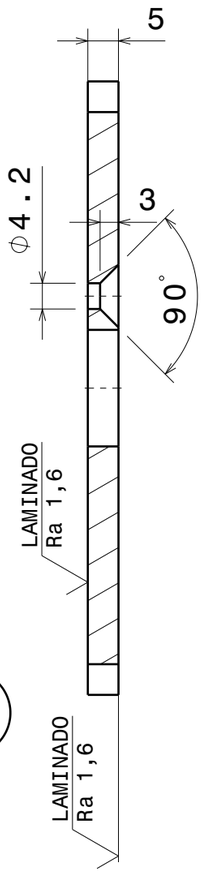


9





8



MARCA	DENOMINACIÓN	MATERIAL
9	BASTIDOR	F111
8	TAPA INFERIOR	F111
4	EJE	F111

$4\left(\sqrt{\text{Ra 1,6}}\right)$

$8{,}9\sqrt{\substack{\text{LÁSER}\\\text{Ra 6,3}}}\left(\sqrt{\text{Ra 1,6}}\right)$

Tol. Gen. ISO 2768-mK

CHAFLANES 1x45º RADIOS DE REDONDEO 1mm

ESCUELA DE INGENIERIAS INDUSTRIALES
DEPARTAMENTO DE INGENIERÍA ENERGÉTICA Y FLUIDOMECÁNICA

UNIVERSIDAD DE VALLDOLID

TÍTULO

BANCO DE ENSAYOS PARA TURBINAS

DISEÑADO POR: FECHA

ADRIÁN MARTIN PINTOS 17/07/2020

TUTOR:

ANDRÉS MELGAR BACHILLER

A3	NÚMERO DE PLANO 7	REV X
ESCALA 1:1		HOJA 7/7

Anexo II código

Código del controlador

Principal

```
#include <SPI.h>

#include "pin_out.h"

#include "EEPROM.h"

#include "Espera.h"


//**********************Media Movil

#define Muestras_MediaMovil 7

#include "Media_Movil.h"


// ******************Termopares

#include "MAX31855K.h"

MAX31855k  Temp1(CS1,0),Temp2(CS3,0);


bool leer=1; //bandera para proceso de lectura

Media_Movil TM1(Muestras_MediaMovil),TM2(Muestras_MediaMovil);  //valores medios de cada medida

TEspera F_Muestreo,F_Display;  //variable para cuenta de tiempo de muestreo

char version[]="Turbina V1.0.";

const byte interruptPin = rpm;

unsigned long oldt=0;

int i=0,n=0;


boolean par=false;

void leer_sondas()

{float T;
```

```
   T=Temp1.readTempC();
   if(T>0) TM1.Media(T);
   T=Temp2.readTempC();
   if(T>0) TM2.Media(T);
}
void IRAM_ATTR isr()
{
if(par){
  oldt=micros();
}
  par^=1;
 // digitalWrite(ledPin,par);
}
void setup() {
  Serial.begin(1000000);
  pinMode(interruptPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(interruptPin),isr, RISING);
    oldt=0;
   F_Muestreo.set(100);  //muestreo cada 200ms
   F_Display.set(500); //frecuencia de display 500ms
}
void loop() {
//Actualización sondas
  if(F_Muestreo.done()){
          F_Muestreo.reset();
          leer_sondas();}
   if(oldt!=0){
      //Serial.print(i);
      //Serial.print(",");
      Serial.print(oldt);
```

```
    if(F_Display.done()){F_Display.reset();
                    Serial.print(",");
                    Serial.print(String(TM1.get()));
                    Serial.print(",");
                    Serial.print(String(TM2.get()));
                    Serial.print(",");
                    Serial.print(String(1.1134*analogRead(A2) + 171.89));
                    Serial.print(",");
                    Serial.println(String(1.1134*analogRead(A1) + 171.89));

    Serial.println();
    oldt=0;}
}
```

Biblioteca Espera.cpp

```
/*   Espera.c                                        */
/*   (c) 2019 Jose Angel Moneo                              */
/*   v1.3                                             */
/*   Clase Espera                                     */
/* ========================================================================
*/

#include "Espera.h"

//asigna el valor tiempo de espera a lavariable tsec
void TEspera::set(int espera)
{ Tsec=millis()+espera;
Tespera=espera; //alamacena el tiempo fijado
  fdone=false;
}
```

```
//espera a que se cumpla el tiempo de espera asignado
//if op es true realiza un reset automático al cumplirse el tiempo
boolean TEspera::done(bool op)
{  if (!fdone) if(Tsec<millis()) {fdone=true; if (op) reset();}
  return fdone;
}
void TEspera::reset(void)
{   Tsec=millis()+Tespera;
  fdone=false;
}
```

Biblioteca Espera.h

```
//  Clase Espera.h
// (c) Jose Angel Moneo (2016)
//  Sistema de espera sin usar timer
//
 // (c)  José Angel Moneo (2016)
// V1.2



#ifndef _Espera_class_H
#define _Espera_class_H
#if ARDUINO >= 100
 #include "Arduino.h"
#else
 #include "WProgram.h"
#endif
class TEspera
{
```

```
 private:
      unsigned long Tsec;
      int Tespera; //tiempo almacenado
      boolean fdone; //marca tiempo finalizado, para no tener que comprobar el tiempo
despues de que se cumpla
public:
   TEspera(){}; //constructor para servo. indicamos el las salidas del servo
   boolean done(bool op=false);
   void set(int espera);
   void reset(); //Reinicia el tiempo preficado anteriormente
            };
  #endif
```

Biblioteca MAX31855K.cpp

```
#include "MAX31855k.h"
//////////////////////////////////////////////////////////////////////////////
// Description  : This constructor does the required setup
// Input        : uint8_t _cs: The Arduino pin number of the chip select line
//              for this instance
//              uint8_t _sck: The Arduino pin number of the SPI clock
//              uint8_t _so:  The Arduino pin number of the SPI MISO
//              uint8_t _vcc: The Arduino pin number to source the power
//              uint8_t _gnd: The Arduino pin number to sink the power
// Output       : Instance of this class with pins configured
// Return       : None
// Usage        : SparkFunMAX31855k <name>(<pinNumber>);
//////////////////////////////////////////////////////////////////////////////
MAX31855k::MAX31855k(const uint8_t _cs, const bool _debug) : cs(_cs), debug(_debug)
{
  // Redundant with SPI library if using default SS
```

```
  pinMode(cs, OUTPUT);
  // SCK & MOSI set in SPI library, MISO autoconfigures
  SPI.begin();
  digitalWrite(cs, HIGH);
}
////////////////////////////////////////////////////////////////////////////
// Change the cs pin
////////////////////////////////////////////////////////////////////////////
void MAX31855k::setCS(int pin)
{
  digitalWrite(cs, HIGH); // Make sure to set old cs high to disable the chip
  cs = pin;             //change to new cs pin
  pinMode(cs, OUTPUT);
  digitalWrite(cs, HIGH);
}
////////////////////////////////////////////////////////////////////////////
// Deconstructor does nothing.  It's up to the user to re-assign
// chip select pin if they want to use it for something else.  We don't call
// SPI.end() in case there is another SPI device we don't want to kill.
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
// Description  : This function reads the cold junction temperature
// Input        : None
// Output       : Loads member variables with data from the IC
// Return:      : None
// Usage        : <objectName>.readBytes();
// MAX31855K Memory Map:
//  D[31:18] Signed 14-bit thermocouple temperature data
//  D17      Reserved: Always reads 0
//  D16      Fault: 1 when any of the SCV, SCG, or OC faults are active, else 0
```

```
// D[15:4]  Signed 12-bit internal temperature
// D3       Reserved: Always reads 0
// D2       SCV fault: Reads 1 when thermocouple is shorted to V_CC, else 0
// D1       SCG fault: Reads 1 when thermocouple is shorted to gnd, else 0
// D0       OC  fault: Reads 1 when thermocouple is open-circuit, else 0
//////////////////////////////////////////////////////////////////////////////
void MAX31855k::readBytes(void)
{
  digitalWrite(cs, LOW);

  SPI.beginTransaction(SPISettings(4000000, MSBFIRST, SPI_MODE0));  // Defaults
  data.bytes[3] = SPI.transfer(0x00);
  data.bytes[2] = SPI.transfer(0x00);
  data.bytes[1] = SPI.transfer(0x00);
  data.bytes[0] = SPI.transfer(0x00);
  SPI.endTransaction();
  digitalWrite(cs, HIGH);
  return;
}

//////////////////////////////////////////////////////////////////////////////
// Description  : This function reads the current temperature
// Input        : SparkFunMAX31855k::units _u: The units of temperature to
//                return. C (default), K, R, or F
// Output       : Error messages before freezing
// Return:      : float: The temperature in requested units.
// Usage        : float tempC = <objectName>.read_temp();
//              float tempC = <objectName>.read_temp(SparkFunMAX31855k::C);
//              float tempF = <objectName>.read_temp(SparkFunMAX31855k::F);
//              float tempK = <objectName>.read_temp(SparkFunMAX31855k::K);
```

```
//              float tempR = <objectName>.read_temp(SparkFunMAX31855k::R);
////////////////////////////////////////////////////////////////////////////
float MAX31855k::readTemp(MAX31855k::units _u)
{
  int16_t value;
  float temp;

  readBytes();
  if (checkHasFault()) {
    return NAN;
  } else {
    // Bits D[31:18] are the signed 14-bit thermocouple temperature value
    if (data.uint32 & ((uint32_t)1 << 31)) { // Sign extend negative numbers
      value = 0xC000 | ((data.uint32 >> 18) & 0x3FFF);
    } else {
      value = data.uint32 >> 18; // Shift off all but the temperature data
    }
  }
  temp = value/4.0;
  switch (_u) {
  case F:
    temp = (temp * 9.0 / 5.0) + 32.0;
    break;
  case K:
    temp += 273.15;
    break;
  case R:
    temp = (temp + 273.15) * 9.0 / 5.0;
  case C:
  default:
```

```
    break;
  }
  return temp;
}
////////////////////////////////////////////////////////////////////////////
// Description  : This function reads the cold junction temperature
// Input        : None
// Output       : None
// Return:      : float: The temperature in ˚C
// Usage        : float tempC = <objectName>.readCJT();
////////////////////////////////////////////////////////////////////////////
float MAX31855k::readCJT(void)
{
  float ret;
  readBytes();
  if (checkHasFault())
  {
    return NAN;
  }
  if (data.uint32 & ((int32_t)1 << 15))
  { // Sign extend negative numbers
    ret = 0xF000 | ((data.uint32 >> 4) & 0xFFF);
  }
  else
  {
    ret = (data.uint32 >> 4) & 0xFFF;
  }
  return ret/16;
}
////////////////////////////////////////////////////////////////////////////
```

```
// Description  : This function checks the fault bits from the MAX31855K IC
// Input        : None
// Output       : Serial prints debug mesages if debug == true
// Return:      : Fault bits that were high, or 8 for unknow & 0 for no faults
// Usage        : checkHasFault();
////////////////////////////////////////////////////////////////////////////
uint8_t MAX31855k::checkHasFault(void)
{
  if (!data.uint32) {
    // If all bits are low, either it's not wired right, or we actually measured
    // 0˚.  There is no way to tell. With debug turned on this will warn.
    if (debug)
      Serial.println(F("\nMAX31855K::All bits were zero.  Fishy..."));
  }
  if (data.uint32 & ((uint32_t)1<<16)) { // Bit D16 is high => fault
    if (data.uint32 & 1) {
      if (debug)
        Serial.println(F("\nMAX31855K::OC Fault: No Probe"));
      return 0b1;
    } else if (data.uint32 & (1<<1)) {
      if (debug)
        Serial.println(F("\nMAX31855K::SCG Fault: Thermocouple is shorted to GND"));
      return 0b10;
    } else if (data.uint32 & (1<<2)) {
      if (debug)
        Serial.println(F("\nMAX31855K::SCV Fault: Thermocouple is shorted to VCC"));
      return 0b100;
    } else {
      if (debug)
        Serial.println(F("\nMAX31855K::Unknown Fault"));
```

```
    return 0b1000;
   }
  } else {
   return 0; // No fault
  }
}
```

Biblioteca MAX31855K.h

```
/**************************************************************************
* Copyright (c) Jose Angel moneo Fernandez
*
**************************************************************************
/
/**************************************************************************
*
* @file                    MAX31855K.h
*
            Dependencies: SPI library for ESP32
* @version                 0.1.0
* @date                    19/10/03
 *                                                *
 * Library to read temperature from a MAX6675 type K thermocouple digitizer    *
 *                                                *
 * SO output format:                                        *
 * 16 bits                                            *
 * The first bit, D15, is a dummy sign bit and is always zero.              *
 * Bits D14:D3 contain the converted temperature in the order of MSB to LSB.   *
 * Bit D2 is normally low and goes high when the thermocouple input is open.   *
 * D1 is low to provide a device ID for the MAX6675 and bit D0 is tri-state.   *
 *                                                *
 * The method read_temp(max31855k::unit) takes a parameter which determines    *
```

```
 * the unit of temperature returned.                                *
 * max31855k::C will return degrees Celsius                         *
 * max31855k::K will return degrees Kelvin                          *
 * max31855k::F will return degrees Fahrenheit                      *
 * max31855k::R will return degrees Rankine                         *

*************************************************************************
**/

#ifndef _MAX31855k_h_
#define _MAX31855k_h_
#include <Arduino.h>
#include <SPI.h> // Have to include this in the main sketch too... (Using SPI)
const uint8_t NONE = 255; // This is used to indicate VCC or GND pin isn't used
class MAX31855k
{ public:
  // Simple Arduino API style guide functions
  inline float readTempC() { return readTemp(MAX31855k::C); }
  inline float readTempF() { return readTemp(MAX31855k::F); }
  inline float readTempR() { return readTemp(MAX31855k::R); }
  inline float readTempK() { return readTemp(MAX31855k::K); }
  // More advanced code concepts used below
  enum units {
    F, C, K, R
  };
  // If non-zero will turn on serial debugging messages
  uint8_t debug;
  // Returns the temperature in degrees F, K, R, or C (default if unspecified)
  float readTemp(MAX31855k::units _u=C);
  // Returns the cold junction temperature in ˚C
  float readCJT(void);
```

```
  // Pass a pin number to set as CS
  void setCS(int pin);
  MAX31855k(const uint8_t, const bool _debug=0);
  ~MAX31855k() {} // User responsible 4 reassigning pins & stopping SPI
protected:
  union { // Union makes conversion from 4 bytes to an unsigned 32-bit int easy
    uint8_t bytes[4];
    uint32_t uint32;
  } data;
  uint8_t cs;
  void readBytes(void);
  uint8_t checkHasFault(void);
};
#endif /* __max6675_H_ */
```

Biblioteca Media_Movil.cpp

```
/* ============================================================================
*/
/*                                                                            */
/*   Media_Movil.c                                                            */
/*   (c) 2019 Jose Angel Moneo                                                */
/*                                                                            */
/*   Clase Media Movil                                                        */
/* ============================================================================
*/
#include "Media_Movil.h"
Media_Movil::Media_Movil(int muestras){
   Muestras=muestras;
   for(int h=0;h<20;h++)     Medidas[h]=0; //borra tabla de medias
    S_Medidas=0; //Borra la suma de valores
```

```
}
float Media_Movil::Media(float valor)
{ S_Medidas-=Medidas[0]; //Resta el valor que sale de la tabla
  //Hace hueco moviendo los datos hacia el fondo y suma los datos para la media
  for(int i=0;i<Muestras;i++)
      Medidas[i]=Medidas[i+1];
    S_Medidas+=Medidas[Muestras-1]=valor;  //añade el valor al fina de la tabla y lo suma a la suma
  return get();
}
//toma la media movil actual
float Media_Movil::get()
{  return S_Medidas/Muestras;}
```

Biblioteca Media_Movil.h

```
//  Clase Media_Movil.h
// (c) Jose Angel Moneo (2019)
//
// devuelve su media movil
//  v1.1
 // (c)  José Angel Moneo (2019)
#ifndef _Media_Movil_class_H
#define _Media_Movil_class_H
#if ARDUINO >= 100
 #include "Arduino.h"
#else
 #include "WProgram.h"
#endif
class Media_Movil
{ private:
```

```
  float  Medidas[20];  //Tabla de valores de las últimas medidas para media para 20 variables con una media de 20 maxima
  float S_Medidas;  //Suma de las medidas
  int Muestras;
public:
  Media_Movil(int muestras); //constructor para servo. indicamos el las salidas del servo
  float  Media(float valor);  //almacena un nuevo valor y devuelve la media movil actual
  float get();  //devuelve la media movil actual
          };
 #endif
```

Biblioteca pin_out.h

```
//Analogicas
#define A1 26
#define A2 25
#define A3 33
#define A4 32
//Reles
#define R1 2
#define R2 15
#define R3 13
#define R4 27
//cs SPI
#define CS1 5
#define CS2 17
#define CS3 16
#define CS4 4
#define DRDY 0
//#define Wire18b20 34
#define rpm 35
```

//SCLK = 18, MISO = 19, MOSI = 23,

Código software

```
from tkinter import *
import numpy as np
import serial
import threading
import matplotlib.pylab as plt
import math
from matplotlib.backends.backend_tkagg import (
    FigureCanvasTkAgg, NavigationToolbar2Tk)
# Implement the default Matplotlib key bindings.
from matplotlib.backend_bases import key_press_handler
from matplotlib.figure import Figure
import serial
import time
import sys


global a,b, esp32
a=0
b=0

def graficar():
    global selx_fila,sely_fila,etx,ety
    v_datfich2 = []
    v_datarray2 = []
    datos2 = []
    fichero2 = open("datos2.txt", 'r')
    v_datfich2 = (fichero2.readlines())
    fichero2.close()
    n = 0
    mat_graficar=[]
    while n < (len(v_datfich2)):
        temp2 = (v_datfich2[n])
        temp2 = str(temp2.strip("b \ n \ r '"))
        v_temp2 = []
        v_temp2 = (np.array(temp2.split()))
        v_temp2 = v_temp2.astype(str)
        mat_graficar.append(v_temp2)
        n += 1

    fig = Figure(figsize=(7, 4), dpi=100)
    v_x=[]
    v_y=[]
    n=0
    while n<len(mat_graficar):
        v_x.append(float(mat_graficar[n][selx_fila]))
        v_y.append(float(mat_graficar[n][sely_fila]))
        n+=1
    grafica = fig.add_subplot(111)
    grafica.plot(v_x, v_y)
    grafica.set_xlabel(etx)
    grafica.set_ylabel(ety)
    canvas = FigureCanvasTkAgg(fig, master=marco_graf)
    canvas.draw()
```

```
    canvas.get_tk_widget().grid(row=1, column=0, padx=25, pady=25)

def calculos():
    global term_leer
    v_datfich=[]
    v_datarray=[]
    datos=[]
    fichero = open("datos1.txt", 'r')
    v_datfich=(fichero.readlines())
    fichero.close()
    n=0
    while n<(len(v_datfich)):
        temp=(v_datfich[n])
        temp=str(temp.strip("b \ n \ r '"))
        v_temp=[]
        v_temp=(np.array(temp.split()))
        v_temp=v_temp.astype(np.float)
        if len(v_temp)>1:
            datos.append(list(v_temp))
        if len(v_temp)==1:
            datos.append(list([v_temp[0],"a","a","a","a"]))
        n+=1
    #hasta aqui lee el fichero y prepara la matriz
    #ahora vamos a rellenar los datos que sean necesarios antes de
transformar las magnitudes
    n=0
    while n<len(datos):
        datos[n][4]=float(ent_Pext.get())
        n+=1
    #se interpola los valores de presión y temperaatura
    n=0
    i=0
    j=0
    k=0
    while (datos[n][1] or datos[n][2] or datos[n][3] or
datos[n][4])=="a":
        n+=1
    j=n
    n=0
    while n<j:
        datos[n][1] = datos[j][1]
        datos[n][2] = datos[j][2]
        datos[n][3] = datos[j][3]
        datos[n][4] = datos[j][4]
        n += 1
    n=len(datos)-1
    while (datos[n][1] or datos[n][2] or datos[n][3] or
datos[n][4])=="a":
        n-=1
    i=n
    n=len(datos)-1
    while i<n:
        datos[n][1] = datos[i][1]
        datos[n][2] = datos[i][2]
        datos[n][3] = datos[i][3]
        datos[n][4] = datos[i][4]
        n -= 1
    n=0
    while n<len(datos):
        if (datos[n][1] or datos[n][2] or datos[n][3] or
datos[n][4])=="a":
```

```
            k=n
            while (k<len(datos)) and ((datos[k][1] or datos[k][2] or
datos[k][3] or datos[k][4])=="a") :
                    k+=1
            pendientet0=(datos[k][1]-datos[j][1])/(datos[k][0]-
datos[j][0])
            indt0=datos[j][1]-(pendientet0*datos[j][0])
            datos[n][1]=pendientet0*datos[n][0]+indt0

            pendientet2 = (datos[k][2] - datos[j][2]) / (datos[k][0] -
datos[j][0])
            indt2 = datos[j][2] - (pendientet2 * datos[j][0])
            datos[n][2] = pendientet2 * datos[n][0] + indt2

            pendientep0 = (datos[k][3] - datos[j][3]) / (datos[k][0] -
datos[j][0])
            indp0 = datos[j][3] - (pendientep0 * datos[j][0])
            datos[n][3] = pendientep0 * datos[n][0] + indp0

            pendientep2 = (datos[k][4] - datos[j][4]) / (datos[k][0] -
datos[j][0])
            indp2 = datos[j][4] - (pendientep2 * datos[j][0])
            datos[n][4] = pendientep2 * datos[n][0] + indp2
        else:
            j=k
        n+=1
    n=0

    #una vez preparados los datos se extraen las magnitudes.
    print("calculo de magnitudes")
    magnitudes=[]
    n=0
    while n<(len(datos)-1):
        magnitudes.append([])

magnitudes[n].append((((float(datos[n][0])+float(datos[n+1][0]))/2)-
float(datos[0][0]))/1000000)

        magnitudes[n].append(60000000/(float(datos[n+1][0])-
float(datos[n][0])))

        magnitudes[n].append(float((datos[n][3]-500)*0.0017225)*1e5)

        magnitudes[n].append(float(datos[n][1])+273.15)

        magnitudes[n].append(float(datos[n][4])) #si se añade el
sensor de presion a la salida es necesario modificar esta linea

        magnitudes[n].append(float(datos[n][2]) + 273.15)
        n+=1
    #Limpiamos las revoluciones
    n=0
    while n<(len(magnitudes)-1):
        if
(float(magnitudes[n][1])>float(1.5*float(magnitudes[n+1][1])))and
magnitudes[n][1]>1000:
            magnitudes[n][1]=((magnitudes[n-
1][1]+magnitudes[n+1][1])/2)
        n+=1
    #calculamos la aceleración
    print("calculo aceleracion")
```

```
    num_puntos = 11
    if(len(magnitudes)>num_puntos):
        n=int((num_puntos-1)/2)
        i=n-(int((num_puntos-1)/2))
        print("i vale:",i)
        print("n vale:",n)
        while n < (len(magnitudes) - ((num_puntos-1)/2)):
            sum_x=0
            sum_y=0
            sum_xy=0
            sum_x2=0
            aceleracion=0
            while i<((n+((int(num_puntos)-1)/2))+1):
                sum_x=sum_x+float(magnitudes[i][0])
                sum_y=sum_y+float(magnitudes[i][1])

sum_xy=sum_xy+float(magnitudes[i][0])*float(magnitudes[i][1])
                sum_x2=sum_x2+float(magnitudes[i][0]**2)
                i+=1
            aceleracion=(2*np.pi/60)*(((num_puntos)*sum_xy)-
(sum_y*sum_x))/(((num_puntos)*sum_x2)-(sum_x**2))
            magnitudes[n].append(aceleracion)
            i=n-(int((num_puntos-1)/2)-1)
            n += 1
        n=0
        while n<len(magnitudes):
            if (n<(int((num_puntos-1)/2))):
                acel=(magnitudes[int((num_puntos-
1)/2)][6]/magnitudes[int((num_puntos-1)/2)][0])*magnitudes[n][0]
                magnitudes[n].append(acel)
            n+=1
        n = 0
        pend=(magnitudes[len(magnitudes) - (int((num_puntos - 1) /
2))-1][6])/(magnitudes[len(magnitudes) - (int((num_puntos - 1) / 2))-
1][0]-magnitudes[len(magnitudes)-1][0])
        indpen=-magnitudes[len(magnitudes)-1][0]*pend
        while n < len(magnitudes):
            if (n >= (len(magnitudes) - (int((num_puntos - 1) / 2)))):
                acel=pend*magnitudes[n][0]+indpen
                magnitudes[n].append(acel)
            n += 1
    else:
        n=0
        while n<len(magnitudes):
            magnitudes[n].append(0)


    #calculamos potencia efectiva, potencia de perdidas y rendimiento
isoentrópico
    print("calculo potencia")
    n=0
    while n<len(magnitudes):

        if magnitudes[n][6]>=0:

magnitudes[n].append(magnitudes[n][6]*magnitudes[n][1]*float(ent_I.get
()))
            magnitudes[n].append("a")

T2_teorica=magnitudes[n][3]*((magnitudes[n][4]/(magnitudes[n][2]+magni
tudes[n][4]))**((1.4-1)/1.4))
```

```
            if(T2_teorica-magnitudes[n][3]!=0):
                rend_iso=((magnitudes[n][5]-
magnitudes[n][3])/(T2_teorica-float(magnitudes[n][3])))

            else:
                rend_iso=(-1)
            try:
                if rend_iso > 0 and rend_iso<1:
                    magnitudes[n].append(rend_iso)
                else:
                    magnitudes[n].append("a")
            except:
                magnitudes[n].append("a")
        else:
            magnitudes[n].append("a")
            magnitudes[n].append(abs(magnitudes[n][6] *
magnitudes[n][1] * float(ent_I.get())))
            magnitudes[n].append("a")
        n+=1
    #ordenamos las magnitudes por velocidad angular en una matriz
auxiliar
    print("ordenar")
    n=0
    matriz_aux=[]
    while n<len(magnitudes):
        matriz_aux.append(magnitudes[n])
        n+=1

    n=0
    i=0
    k=0
    while n<len(magnitudes):
        num_menor = matriz_aux[0][1]
        while i<len(matriz_aux):
            if matriz_aux[i][1]<=num_menor:
                num_menor=matriz_aux[i][1]
                k=i
            i+=1
        i=0
        magnitudes[n].append(matriz_aux[k][1])
        magnitudes[n].append(matriz_aux[k][7])
        magnitudes[n].append(matriz_aux[k][8])
        magnitudes[n].append(matriz_aux[k][9])
        matriz_aux.pop(k)
        n+=1
    #reordenada la matriz con w,Pe,pp, rend iso, se pasa al cálculo de
las falta de datos de Pe, Pp y rend iso
    print("reordenando potencia")

    n=0
    i=0
    magnitudes[0][11] = 0
    magnitudes[len(magnitudes)-1][11] = 0
    while n<len(magnitudes):
        if magnitudes[n][11]=="a":
            i=n
            while magnitudes[i][11]=="a":
                i+=1
            if ((magnitudes[i][10]-magnitudes[n-1][10])!= 0):
                magnitudes[n][11]=(((magnitudes[i][11]-magnitudes[n-
```

```
1][11])/(magnitudes[i][10]-magnitudes[n-
1][10]))*magnitudes[n][10])+(magnitudes[n-1][11]-(magnitudes[n-
1][10]*(magnitudes[i][11]-magnitudes[n-1][11])/(magnitudes[i][10]-
magnitudes[n-1][10])))
            else:
                magnitudes[n][11] =magnitudes[n-1][11]
        n+=1
    n=0
    i=0
    print("reordenando potencia perdidas")
    magnitudes[n][12] = 0
    magnitudes[len(magnitudes) - 1][12] = 0
    while n < len(magnitudes):
        if magnitudes[n][12] == "a":
            i = n
            while magnitudes[i][12] == "a":
                i += 1
            if (magnitudes[i][10] - magnitudes[n - 1][10] != 0):
                magnitudes[n][12] = (((magnitudes[i][12] -
magnitudes[n - 1][12]) / (
                                     magnitudes[i][10] - magnitudes[n -
1][10])) * magnitudes[n][10]) + (
                                                          magnitudes[n -
1][12] - (magnitudes[n-1][10]*(magnitudes[i][12] - magnitudes[n -
1][12]) / (

magnitudes[i][10] - magnitudes[n - 1][10])))
            else:
                try:
                    magnitudes[n][12] =magnitudes[n-1][12]
                except:
                    magnitudes[n][12]=0
        n += 1
    print("reordenando rendimiento")
    n=0
    i=0
    magnitudes[n][13] = 0
    magnitudes[len(magnitudes) - 1][13] = 0
    while n < len(magnitudes):
        if magnitudes[n][13] == "a":
            i = n
            while magnitudes[i][13] == "a":
                i += 1
            if (magnitudes[i][10] - magnitudes[n - 1][10] != 0):
                magnitudes[n][13] = (((magnitudes[i][13] -
magnitudes[n - 1][13]) / (
                                magnitudes[i][10] - magnitudes[n - 1][10])) *
magnitudes[n][10]) + (
                                                      magnitudes[n - 1][13] -
(magnitudes[n-1][10]*(magnitudes[i][13] - magnitudes[n - 1][13]) / (
                                                      magnitudes[i][10] -
magnitudes[n - 1][10])))
            else:
                magnitudes[n][13] =magnitudes[n-1][13]

        n += 1
    #con estas magnitudes calculamos el resto, Pi, Mi, rend mec y rend
total
    n=0
    while n<len(magnitudes):
        magnitudes[n].append(magnitudes[n][11]+magnitudes[n][12])
```

```
            if magnitudes[n][10]!=0:
                magnitudes[n].append(magnitudes[n][14]/magnitudes[n][10])
            else:
                magnitudes[n].append(0)
            if magnitudes[n][14]!=0:
                magnitudes[n].append(magnitudes[n][11]/magnitudes[n][14])
            else:
                magnitudes[n].append(0)
            magnitudes[n].append(magnitudes[n][13]*magnitudes[n][16])

            n+=1

        fichero2 = open("datos2.txt", 'w')
        n=0
        i=0
        while n<len(magnitudes):
            while i<len(magnitudes[n]):
                fichero2.write(str(magnitudes[n][i]))
                fichero2.write(" ")
                i+=1
            n+=1
            i=0
            fichero2.write("\n")
        fichero2.close()
        #calculo de máximos
        n=0
        var_maxrpm = 0
        while n<len(magnitudes):
            if magnitudes[n][10]>var_maxrpm:
                var_maxrpm=magnitudes[n][10]
            n+=1

        n=0
        var_maxpot = 0
        while n<len(magnitudes):
            if magnitudes[n][11]>var_maxpot:
                var_maxpot=magnitudes[n][11]
            n+=1

        n=0
        var_maxrend = 0
        while n<len(magnitudes):
            if magnitudes[n][17]>var_maxrend:
                var_maxrend=magnitudes[n][17]
            n+=1
        sa_rpmmax['text']=round(var_maxrpm,2)
        sa_Pemax['text'] =round(var_maxpot,2)
        sa_renmax['text'] =round(var_maxrend,2)
        graficar()


    def lecturas():
        global a,b, esp32, term_leer
        t0 = 0
        while b==0:
            try:
                esp32 = serial.Serial(ent_Puer.get(), 1000000,timeout=1)
                et_estadoPuer['text'] = "Conectado"
                d_inst = str(esp32.readline())     #elimino la primera
    linea para que no de problemas
                while a==0:
```

```
                try:
                    d_inst =str(esp32.readline())
                    d_inst = str(d_inst.strip("b \ n \ r '"))
                    d_inst=str([d_inst.replace(',',' ')])
                    d_inst =str(d_inst.strip("]['"))
                    v_inst=[]
                    v_inst = np.array(d_inst.split())
                    v_inst = v_inst.astype(np.float)


                    t1=v_inst[0]
                    if t0!=0:

                        sa_rpminst['text']=60000000/(t1-t0)
                    t0=t1
                    if len(v_inst)>1:
                        Pinst=v_inst[3]*222.24 -103421.38
                        sa_Pinst['text']=str(Pinst)
                        sa_T0inst['text'] =str(v_inst[1])
                        sa_T1inst['text'] =str(v_inst[2])
                except:
                    print("No puedo imprimir porque no tengo datos")



            d_inst = str(esp32.readline())  # elimino la primera linea
para que no de problemas

            fichero=open("datos1.txt", 'w')
            term_leer=False
            while a==1:
                d_inst = str(esp32.readline())
                d_inst = str(d_inst.strip("b \ n \ r '"))
                d_inst = str([d_inst.replace(',', ' ')])
                d_inst = str(d_inst.strip("]['"))
                fichero.write((d_inst))
                fichero.write("\n")
            fichero.close()
            esp32.close()
            term_leer = True
        except:
            et_estadoPuer['text']="Sin conexión"

def seleccion():
    global sel_visual, selx_fila,sely_fila,etx,ety
    sel_visual.get()

    if sel_visual.get()==1:
        selx_fila=0
        sely_fila=1
        etx="Tiempo (s)"
        ety="Velocidad angular (rpm)"
    if sel_visual.get()==2:
        selx_fila=0
        sely_fila=6
        etx="Tiempo (s)"
        ety="Aceleración angular (rad/s^2)"
    if sel_visual.get()==3:
        selx_fila=0
        sely_fila=2
```

```
            etx="Tiempo (s)"
            ety="Presión (Pa)"
        if sel_visual.get()==4:
            selx_fila=0
            sely_fila=3
            etx="Tiempo (s)"
            ety="Temperatura 0 (k)"
        if sel_visual.get()==5:
            selx_fila=0
            sely_fila=5
            etx="Tiempo (s)"
            ety="Temperatura 2 (K)"
        if sel_visual.get()==6:
            selx_fila=10
            sely_fila=11
            etx="Velocidad angular (rpm)"
            ety="Potencia efectiva (W)"
        if sel_visual.get()==7:
            selx_fila=10
            sely_fila=14
            etx="Velocidad angular (rpm)"
            ety="Potencia indicada (W)"
        if sel_visual.get()==8:
            selx_fila=10
            sely_fila=15
            etx="Velocidad angular (rpm)"
            ety="Par indicado (Nm)"
        if sel_visual.get()==9:
            selx_fila=10
            sely_fila=13
            etx="Velocidad angular (rpm)"
            ety="Rendimiento isoentrópico"
        if sel_visual.get()==10:
            selx_fila=10
            sely_fila=16
            etx="Velocidad angular (rpm)"
            ety="Rendimiento mecánico"
        if sel_visual.get()==11:
            selx_fila=10
            sely_fila=17
            etx="Velocidad angular (rpm)"
            ety="Rendimiento total"
        graficar()
    def var_usuario():
        print(ent_Pext.get())
        print(ent_I.get())
    def ejecutar():
        global a
        a=1
        sa_est['text']="en curso"
    def parar():
        global a,term_leer
        a=0
        sa_est['text']="concluida"
        while term_leer != True:
            print("Espero a true=",term_leer)
        calculos()
```

```
mutex = threading.Lock()

ventana=Tk()
ventana.title("BETM")
ventana.geometry("1700x600")
ventana.configure(background='mint cream')

global sel_visual,selx_fila,sely_fila,etx,ety,term_leer,esp32
term_leer = True
sel_visual=IntVar()
selx_fila=0
sely_fila=1
etx="Tiempo (s)"
ety="Velocidad angular (rpm)"
ent_Puer="'COM3'"

c_fondo="mint cream"
c_tex="mint cream"
c_entrada="mint cream"
f_titulo="ms reference sans serif",12,
f_texto="ms reference sans serif",10,
f_entradas='arial'
tam_tex=1
tam_marpadx=5
tam_marpady=5
tam_texpadx=10
tam_texpady=0

########################################################################
########################
marco_var=Frame(ventana)
marco_var.configure(background=c_fondo,
padx=tam_marpadx,pady=tam_marpady,bd=1,relief='flat')
marco_var.grid(row=0,column=0, sticky="WENS")

marco_pru=Frame(ventana)
marco_pru.configure(background=c_fondo,
padx=tam_marpadx,pady=tam_marpady,bd=2,relief='flat')
marco_pru.grid(row=1,column=1, sticky="WENS")

marco_graf=Frame(ventana)
marco_graf.configure(background='white',
padx=tam_marpadx,pady=tam_marpady,bd=1,relief='flat')
marco_graf.grid(row=1,column=0, sticky="WE")

marco_t=Frame(ventana)
marco_t.configure(background=c_fondo,
padx=tam_marpadx,pady=tam_marpady,bd=1,relief='flat')
marco_t.grid(row=2,column=0, sticky="WE")

marco_rpm=Frame(ventana)
marco_rpm.configure(background=c_fondo,
padx=tam_marpadx,pady=tam_marpady,bd=0.5,relief='flat')
marco_rpm.grid(row=3,column=0, sticky="WE")
########################################################################
#################################



et_param=Label(marco_var, text="Parámetros del ensayo          ",
```

```
font=f_titulo)
et_param.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
et_param.grid(row=0,column=0)

et_paramP=Label(marco_var, text="Presión exterior (Pa): ",
font=f_texto)
et_paramP.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
et_paramP.grid(row=0,column=1)

ent_Pext=Entry(marco_var,justify="center")
ent_Pext.grid(row=0 ,column=2)
ent_Pext.insert(0, "101325")

et_paramI=Label(marco_var, text="Inercia (Kg m2): ", font=f_texto)
et_paramI.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
et_paramI.grid(row=0,column=3)

ent_I=Entry(marco_var,justify="center")
ent_I.grid(row=0 ,column=4)
ent_I.insert(0, "1e-5")

et_paramPuer=Label(marco_var, text="Puerto: ", font=f_texto)
et_paramPuer.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
et_paramPuer.grid(row=0,column=5)

ent_Puer=Entry(marco_var,justify="center")
ent_Puer.grid(row=0 ,column=6)
ent_Puer.insert(0, 'COM3')

et_estadoPuer=Label(marco_var, font=f_texto)
et_estadoPuer.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
et_estadoPuer.grid(row=0,column=7)

########################################################################
#########################


et_t=Label(marco_t, text="Gráficas tiempo", font=f_titulo)
et_t.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
et_t.grid(row=0,column=0)

bt_rpm=Radiobutton(marco_t, text="w", font=f_texto)
bt_rpm.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex, variable=sel_visual, command=seleccion, value=1)
bt_rpm.grid(row=0,column=1)
bt_rpm.select()

bt_ac=Radiobutton(marco_t, text="a", font=f_texto)
bt_ac.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex, variable=sel_visual, command=seleccion, value=2)
bt_ac.grid(row=0,column=2)

bt_P=Radiobutton(marco_t, text="P", font=f_texto)
bt_P.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex, variable=sel_visual, command=seleccion, value=3)
```

```
bt_P.grid(row=0,column=3)

bt_T0=Radiobutton(marco_t, text="T0", font=f_texto)
bt_T0.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex, variable=sel_visual, command=seleccion, value=4)
bt_T0.grid(row=0,column=4)

bt_T2=Radiobutton(marco_t, text="T2", font=f_texto)
bt_T2.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex, variable=sel_visual, command=seleccion, value=5)
bt_T2.grid(row=0,column=5)

#########################################################################
#########################

et_rpm=Label(marco_rpm, text="Gráficas rpm", font=f_titulo)
et_rpm.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
et_rpm.grid(row=0,column=0)

bt_Pe=Radiobutton(marco_rpm, text="Pe", font=f_texto)
bt_Pe.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex, variable=sel_visual, command=seleccion, value=6)
bt_Pe.grid(row=0,column=1)

bt_Me=Radiobutton(marco_rpm, text="Pi", font=f_texto)
bt_Me.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex, variable=sel_visual, command=seleccion, value=7)
bt_Me.grid(row=0,column=2)

bt_Pi=Radiobutton(marco_rpm, text="Mi", font=f_texto)
bt_Pi.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex, variable=sel_visual, command=seleccion, value=8)
bt_Pi.grid(row=0,column=3)

bt_iso=Radiobutton(marco_rpm, text="Rend. iso.", font=f_texto)
bt_iso.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex, variable=sel_visual, command=seleccion, value=9)
bt_iso.grid(row=0,column=4)

bt_mec=Radiobutton(marco_rpm, text="rend. mec.", font=f_texto)
bt_mec.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex, variable=sel_visual, command=seleccion, value=10)
bt_mec.grid(row=0,column=5)

bt_tot=Radiobutton(marco_rpm, text="rend. total", font=f_texto)
bt_tot.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex, variable=sel_visual, command=seleccion, value=11)
bt_tot.grid(row=0,column=6)

#########################################################################
#########################

et_pru=Label(marco_pru, text="Valores instantáneos", font=f_titulo)
et_pru.configure(padx=tam_texpadx, pady=tam_texpady+15, bg=c_tex,
height=tam_tex)
et_pru.grid(row=0,column=0)

et_rpminst=Label(marco_pru, text="Velocidad angular (rpm)",
font=f_texto)
et_rpminst.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
```

```
height=tam_tex)
et_rpminst.grid(row=1,column=0)

sa_rpminst=Label(marco_pru, font=f_texto)
sa_rpminst.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
sa_rpminst.grid(row=1,column=1)

et_Pinst=Label(marco_pru, text="Presión (Pa)", font=f_texto)
et_Pinst.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
et_Pinst.grid(row=2,column=0)

sa_Pinst=Label(marco_pru, font=f_texto)
sa_Pinst.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
sa_Pinst.grid(row=2,column=1)

et_T0inst=Label(marco_pru, text="Temperatura entrada (°C)",
font=f_texto)
et_T0inst.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
et_T0inst.grid(row=3,column=0)

sa_T0inst=Label(marco_pru, font=f_texto)
sa_T0inst.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
sa_T0inst.grid(row=3,column=1)

et_T1inst=Label(marco_pru, text="Temperatura salida (°C)",
font=f_texto)
et_T1inst.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
et_T1inst.grid(row=4,column=0)

sa_T1inst=Label(marco_pru, font=f_texto)
sa_T1inst.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
sa_T1inst.grid(row=4,column=1)

et_p=Label(marco_pru, text="Prueba", font=f_titulo)
et_p.configure(padx=tam_texpadx, pady=tam_texpady+15, bg=c_tex,
height=tam_tex)
et_p.grid(row=5,column=0)

bot_ini=Button(marco_pru, text=("INICIAR"), font=f_texto,
command=ejecutar)
bot_ini.configure(padx=tam_texpadx+10, pady=tam_texpady+2, bg="green",
height=tam_tex, bd=1)
bot_ini.grid(row=6, column=0)

et_est=Label(marco_pru, text="Estado de la prueba", font=f_texto)
et_est.configure(padx=tam_texpadx, pady=tam_texpady+10, bg=c_tex,
height=tam_tex)
et_est.grid(row=7,column=0)

sa_est=Label(marco_pru, font=f_texto)
sa_est.configure(padx=tam_texpadx, pady=tam_texpady+10, bg=c_tex,
height=tam_tex)
sa_est.grid(row=7,column=1)
```

```
bot_stop=Button(marco_pru, text=("PARAR"), font=f_texto,
command=parar)
bot_stop.configure(padx=tam_texpadx+14, pady=tam_texpady+2, bg="red",
height=tam_tex, bd=1)
bot_stop.grid(row=8, column=0)

et_max=Label(marco_pru, text="Valores máximos", font=f_titulo)
et_max.configure(padx=tam_texpadx, pady=tam_texpady+15, bg=c_tex,
height=tam_tex)
et_max.grid(row=9,column=0)

et_rpmmax=Label(marco_pru, text="velocidad angular (rpm):",
font=f_texto)
et_rpmmax.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
et_rpmmax.grid(row=10,column=0)

sa_rpmmax=Label(marco_pru, font=f_texto)
sa_rpmmax.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
sa_rpmmax.grid(row=10,column=1)

et_Pemax=Label(marco_pru, text="Potencia efctiva (W):", font=f_texto)
et_Pemax.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
et_Pemax.grid(row=11,column=0)

sa_Pemax=Label(marco_pru, font=f_texto)
sa_Pemax.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
sa_Pemax.grid(row=11,column=1)

et_renmax=Label(marco_pru, text="Rendimiento total:", font=f_texto)
et_renmax.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
et_renmax.grid(row=12,column=0)

sa_renmax=Label(marco_pru, font=f_texto)
sa_renmax.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
sa_renmax.grid(row=12,column=1)



t = threading.Thread(target=lecturas)
t.setDaemon(True)
t.start()



ventana.mainloop()
b=1
a=1
try:
    esp32.close()
except:
    print("no hay puerto")
sys.exit()
```